EXASCALE COMPUTING PROJECT

ECP-U-AD-RPT_2021_00208

Map Applications to Target Exascale Architecture with Machine-Specific Performance Analysis, Including Challenges and Projections

WBS 2.2, Milestone PM-AD-1110

Andrew Siegel[1], Erik W. Draeger[2], Jack Deslippe[3], Tom Evans[4], Marianne M. Francois[5], Tim Germann[5], Dan Martin[3], and William Hart[6]

[1]Argonne National Laboratory
[2]Lawrence Livermore National Laboratory
[3]Lawrence Berkeley National Laboratory
[4]Oak Ridge National Laboratory
[5]Los Alamos National Laboratory
[6]Sandia National Laboratories

March 31, 2021

ORNL/TM-2021/2103

U.S. DEPARTMENT OF ENERGY | Office of Science

NNSA National Nuclear Security Administration

ECP-U-AD-RPT_2021_00208

ECP Milestone Report
Map Applications to Target Exascale Architecture with Machine-Specific Performance Analysis, Including Challenges and Projections
WBS 2.2, Milestone PM-AD-1110

Office of Advanced Scientific Computing Research
Office of Science
US Department of Energy

Office of Advanced Simulation and Computing
National Nuclear Security Administration
US Department of Energy

March 31, 2021

ORNL/TM-2021/2103

REVISION LOG

Version	Creation Date	Description	Approval Date
1.0	July 8, 2021	Original	

EXECUTIVE SUMMARY

This Exascale Computing Project (ECP) milestone report summarizes the status of all 30 ECP Applications Development (AD) subprojects at the end of FY20. In October and November of 2020, a comprehensive assessment of AD projects was conducted by the ECP leadership. Reviews occurred virtually between October 27, 2020 and November 12, 2020. The review committee—consisting of the AD lead, deputy, and L3—was tasked with evaluating each subproject's progress in porting their codes to early exascale architectures considered precursors to the planned exascale machines. This includes characterizing which modules have been ported to multi-accelerator nodes, initial performance analyses, the status of software integration, and a current vision of successes, obstacles, and next steps. As such, this report contains not only an accurate snapshot of each subproject's current status but also represents an unprecedentedly broad account of experiences in porting large scientific applications to next-generation high-performance computing architectures.

A high-level summary of the review outcome is provided in Tables 1 to 3. Several project teams were asked to provide additional details on GPU performance and their impact on near future plans. However, the general consensus of the review committees was that ECP application projects have made excellent progress in FY20 and are well on track for a successful overall project completion.

Table 1: Review summary for KPP-1 applications. All listed FOM values are extrapolated to the full machine.

AD L4 project	Application area	Execution risk	Current FOM ratio
EXAALT	Atomistic materials	Low	350
ExaSky	Cosmology	Low	72.25
ExaSMR	Nuclear energy	Low	46.7
CANDLE	Precision medicine for oncology	Low	81.99
E3SM-MMF	Earth system science	Low	73
EQSIM	Earthquakes	Low	189
LatticeQCD	Nuclear physics	Medium	22.42
NWChemEx	Chemistry	Low	78.8
QMCPACK	Quantum materials	Low	37.1
WarpX	Accelerator physics	Low	114
WDMApp	Magnetic fusion energy	Medium	42.5

Table 2: Review summary for KPP-2 applications.

AD L4 project	Application area	Execution risk	Milepost schedule
Combustion-PELE	Combustion science	Medium	On track
ExaAM	Advanced manufacturing	High	On track
ExaBiome	Microbiome analysis	Medium	On track
ExaFEL	Laser analytics	Medium	On track
ExaSGD	Power grid	Medium	On track
ExaStar	Astrophysics	Medium	On track
ExaWind	Wind power	High	On track
GAMESS	Quantum chemistry	Medium	On track
MFIX-Exa	Chemical reactors	Medium	On track
Subsurface	Subsurface flow	High	3 months behind
Livermore	NNSA national security	Low	On track
Los Alamos	NNSA national security	Low	On track
Sandia	NNSA national security	Low	On track

Table 3: Review summary for KPP-3 co-design projects.

AD L4 project	Motifs	Current metric
AMReX	AMR, regular grids	9
CEED	FEM, unstructured grids	6
CODAR	Data I/O	1
CoPA	Particle methods, FFT	8
ExaGraph	Graph traversal	4
ExaLearn	ML, AI	1

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTINGS

ACRONYMS

AD Applications Development

AI artificial intelligence

ALCF Argonne Leadership Computing Facility

ALE arbitrary Lagrangian-Eulerian

AM additive manufacturing

AMR adaptive mesh refinement

ASC Advanced Simulation and Computing

ATDM Advanced Technology Development and Mitigation

BER Biological and Environmental Research

BES Basic Energy Sciences

CC coupled cluster

CD Co-Design

CFD computational fluid dynamics

CLR chemical looping reactor

CM Chemistry and Materials Applications

COE Center of Excellence

DAC data analytics computing

DAO data analytics and optimization

DEM discrete element method

DFET density functional embedding theory

DFT density functional theory

DFTB density functional tight binding

DNS direct numerical dimulation

DOD US Department of Defense

DOE US Department of Energy

DOF degrees of freedom

EA energy applications

EB embedded boundary

ECP Exascale Computing Project

EOS equation of state

ESS Earth and Space Science Applications

FEM finite element methods

FFT fast Fourier transform

FOM figure of merit

FRIB Facility for Rare Isotope Beams

HF Hartree-Fock

HI hardware and integration

HIP Heterogeneous-Compute Interface for Portability

ICF inertial confinement fusion

JFNK Jacobian-Free, Newton-Krylov

JLSE Joint Laboratory for System Evaluation

KPP key performance parameter

L3 Level 3

L4 Level 4

LCLS Linac Coherent Light Source

LES large eddy simulation

M&S modeling and simulation

MC Monte Carlo

MD molecular dynamics

MHD magnetohydrodynamics

ML machine learning

MSN mesoporous silica nanoparticles

NASA National Aeronautics and Space Administration

NCI National Cancer Institute

NDA nondisclosure agreement

NERSC National Energy Research Scientific Computing Center

NESAP NERSC Exascale Science Applications Program

NIF National Ignition Facility

NIH National Institutes of Health

NNSA National Nuclear Security Administration

NOAA National Oceanic and Atmospheric Administration

NSCI National Strategic Computing Initiative

NSF National Science Foundation

OLCF Oak Ridge Leadership Computing Facility

openPMD Open Standard for Particle-Mesh Data

OS Office of Science

PDF probability distribution function

PFC plasma-facing component

PIC particle-in-cell

QCD quantum chromodynamics

QMC quantum Monte Carlo

QMD quantum molecular dynamics

RANS Reynolds-averaged Navier-Stokes

SC Office of Science

SM Standard Model

SME subject matter expert

SMR small modular reactor

SSP Stockpile Stewardship Program

ST Software Technology

SYPD simulated-years-per-day

URANS Unsteady Reynolds-averaged Navier-Stokes

WBS Work Breakdown Structure

1. OVERVIEW OF APPLICATION DEVELOPMENT

Exascale systems enable game-changing advances in scientific, engineering, and national security applications critical to the US US Department of Energy (DOE)'s mission. Such progress requires close coordination among exascale application, algorithm, and software development to address key challenges, such as extreme parallelism, reliability and resiliency, complex memory hierarchies, scaling to large systems, and data-intensive science. For selected critical problems, the ECP is creating and enhancing the predictive capability of relevant applications through the targeted development of requirements-based models, algorithms, and methods along with the development and integration of required software technologies in support of application workflows.

Given the broad DOE and multi-agency demand for mission-critical modeling and simulation (M&S) and data analytics computing (DAC) applications, AD is contributing to the development of the ECP applications for DOE missions in science, energy, national security and the missions of other agencies, such as the National Institutes of Health (NIH), National Science Foundation (NSF), National Oceanic and Atmospheric Administration (NOAA), and National Aeronautics and Space Administration (NASA). For DOE alone, this scope encompasses full development support for application teams within the mission space of at least 10 DOE program offices. For other agency applications, the scope of AD is smaller and is one of partnership through the provision of selected staff to development teams in need of expertise in computer and computational science, applied mathematics, and high-performance computing (HPC).

To achieve these goals, AD includes six Level 3 (L3) Work Breakdown Structure (WBS) elements, each with multiple subprojects at Level 4 (L4). These are described in the following sections. Two key performance parameters (KPPs), KPP-1 and KPP-2, are used to measure the success of the AD application projects; KPP-3 is used to measure the success of the AD co-design projects. KPP-1 performance is measured through a uniquely defined FOM for each project. The meaning and requirements for each KPP are given in Table 4.

Table 4: KPP for ECP applications.

KPP ID	Description of scope	Threshold KPP	Objective KPP	Verification action/evidence
KPP-1	Performance improvement for mission-critical problems	50% of selected applications achieve FOM improvement ≥ 50	100% of selected applications achieve FOM stretch goal	Independent assessment of measured results that threshold goal is met
KPP-2	Broaden the reach of exascale science and mission capability	50% of selected applications can execute their challenge problem	100% of selected applications can execute their challenge problem stretch goal	Independent assessment of mission application readiness
KPP-3	Productive and sustainable software ecosystem	50% of the weighted impact goals are met	100% of the weighted impact stretch goals are met	Independent assessment verifying threshold goal is met

1.1 Early and Pre-Exascale Hardware

In FY20, the AD projects had access to two categories of hardware at the DOE Oak Ridge Leadership Computing Facility (OLCF) and Argonne Leadership Computing Facility (ALCF): (1) pre-exascale hardware that includes Summit at OLCF and (2) early exascale hardware. Summit contains NVIDIA GPUs supported by IBM multicore CPUs.[1] The early exascale hardware contains AMD and Intel hardware that are predelivery versions of the hardware that will be featured on Frontier and Aurora, respectively. The AMD early platforms Tulip, Poplar, and Lyra are supported by the Frontier Center of Excellence (COE) at Cray and the OLCF. The Intel early platforms Iris and Yarrow are supported by the Joint Laboratory for System Evaluation (JLSE) at ALCF.

[1] https://www.olcf.ornl.gov/olcf-resources/compute-systems/summit

1.2 2020 AD Review

In the FY19 AD review described in *Early Application Results on Pre-exascale Architecture with Analysis of Performance Challenges and Projections* [1], we evaluated the projects based on five charge questions that address:

1. defined "base" and "stretch" objectives;
2. progress on pre-exascale hardware;
3. defined FOMs and capability plans for KPP-1 and KPP-2, respectively;
4. defined 2020 and out year plans with defined risks and mitigation strategies based on early hardware progress; and
5. identified critical dependencies with Software Technology (ST) and Co-Design (CD) projects.

For the FY20 review, the projects are evaluated on three sets of charge questions for KPP-1, KPP-2, and CD projects. The KPP-1 charge questions are as follows.

1. Explain how your current KPP-1 value was calculated and how it relates to your exascale challenge problem, including any assumptions, caveats, simplifications, and so on.
2. Describe your performance on Summit (or equivalent) as completely as possible. Where possible, make meaningful performance comparisons, characterize GPU utilization, and identify bottleneck resources.
3. Describe your progress on Iris and/or Tulip to date. On a scale of 1–5, where 5 is the most confident, how confident are you that you will successfully meet the KPP-1 threshold on Aurora?
4. On a scale of 1–5, where 5 is the most confident, how confident are you that you will successfully meet the KPP-1 threshold on Frontier?
5. List the critical dependencies for your project. Which are the biggest risks to your project's successful completion?

The KPP-2 charge questions are as follows.

1. Summarize your progress toward executing your project's challenge problem, including a status of project mileposts.
2. Describe your performance on Summit (or equivalent) as completely as possible. Where possible, make meaningful performance comparisons, characterize GPU utilization, and identify bottleneck resources.
3. Describe your progress on Iris and/or Tulip to date.
4. On a scale of 1–5, where 5 is the most confident, how confident are you that you will successfully meet the KPP-2 threshold on Aurora?
5. On a scale of 1–5, where 5 is the most confident, how confident are you that you will successfully meet the KPP-2 threshold on Frontier?
6. List the critical dependencies for your project. Which are the biggest risks to your project's successful completion?

Finally, the CD charge questions are as follows.

1. Summarize your progress toward your KPP-3 objectives.
2. Describe your performance on Summit (or equivalent) as completely as possible. In particular, characterize GPU utilization and identify bottleneck resources.
3. Describe your progress on Iris and/or Tulip to date.

4. List the critical dependencies for your project. Which are the biggest risks to your project's successful completion?

As opposed to the FY18 and FY19 reviews, which both featured panels of external subject matter experts (SMEs), the FY20 review used the L3 leadership team as the subject matter reviewers. This choice was made due to the focus on application performance, particularly on early exascale hardware that has restrictive nondisclosure agreement (NDA) requirements, as opposed to science objectives. Also, due to restrictions imposed by COVID-19, the FY20 review was performed completely online. Reviews for each project were performed in 2 h blocks starting on October 27, 2020 and ending on November 12, 2020.

2. KEY PERFORMANCE PARAMETERS FOR AD

The AD focus area supports the development and evolved design of mission-critical science and engineering codes for efficient execution on exascale platforms. The ECP distinguishes between the code, which is typically a general capability, and an application, which uses the code to address a specific scientific or engineering question. A key concept is the definition of an exascale challenge problem. Each AD application code team must define an application challenge problem that is scientifically impactful and requires exascale-level resources to execute. Each exascale challenge problem targets a key DOE science or mission need and is the basis for quantitative measurements of success for each of the AD projects.

There are two measures of success used for AD application projects, referred to generically as the first and second of the ECP KPPs (KPP-1 and KPP-2). Projects are assigned exclusively to one of these two KPP groups, and each project is responsible for meeting the corresponding specific requirements. The KPP-1 applications (Table 5) and applications targeting KPP-2 (Table 6) are required to provide a detailed milestone plan that outlines all needed work to enable the successful execution of their exascale challenge problem. These milestone plans and the teams' progress in executing them are reviewed annually by AD leadership and external SMEs as part of the AD annual assessment. Progress toward KPP-2 is tracked between reviews with a dashboard to monitor timely milestone delivery. The KPP assignments were determined by the nature of the exascale challenge problem and the maturity of the individual code projects.

2.1 KPP-1

KPP-1 quantitatively measures the increased capability of applications on exascale platforms compared with their capability on the leadership-class machines available at the start of the project. Each application that targets KPP-1 is required to define a quantitative FOM that represents the rate of science work for their defined exascale challenge problem. FOM definitions are specific to an application area and are reviewed internally and externally to the ECP to confirm that they are appropriate representations of capability improvements for that domain. Because exascale challenge problems cannot be executed on petascale resources, the FOM will typically account for differences in problem size, numerical precision, algorithm complexity, and physical model enhancement to allow for an accurate measurement of the ultimate FOM improvement used to satisfy KPP-1.

For KPP-1, one key concept is the performance baseline, which is a quantitative measure of an application FOM that uses the fastest computers available at the inception of the ECP against which the final FOM improvement is measured. This includes systems at the ALCF, NERSC, and OLCF, such as Mira, Theta, Cori, and Titan—systems in the 10–20 PFlops range. The expectation is that applications will run at full scale on at least one of these systems to establish the performance baseline. In cases where this is not possible, the largest scale run is scaled to the full system, assuming perfect parallel efficiency. A challenging situation arises when the final exascale challenge problem requires capabilities that did not exist at the start of the project (e.g., new code coupling, new physics models, or algorithmic approaches). For these applications, approximate estimates are constructed from individual code components with the expectation that the baseline can be refined, if necessary, once the new capabilities are in place.

Applications that target KPP-1 are required to demonstrate improvements to their FOM throughout the project on pre-exascale platforms. The teams' progress in improving their FOMs and preparing their codes for exascale architectures is reviewed annually by AD leadership and external SMEs as part of the AD annual assessment. Progress toward KPP-1 is tracked between reviews with a dashboard to monitor each application's current FOM increase against their performance baseline.

Table 5: ECP applications targeting KPP-1.

Project name	Short description	Lead lab	Stakeholder program
LatticeQCD	Exascale lattice gauge theory opportunities and requirements for nuclear and high-energy physics	FNAL	DOE NP, HEP
NWChemEx	Stress-resistant crops and efficient biomass catalysts	PNNL	DOE BER, BES
EXAALT	Molecular dynamics at the exascale	LANL	DOE BES, FES, NE
QMCPACK	Find, predict, and control material properties	ORNL	DOE BES
ExaSMR	Coupled Monte Carlo neutronics and fluid flow simulation of small modular reactors	ORNL	DOE NE
WDMApp	High-fidelity whole device modeling of magnetically confined plasmas	PPPL	DOE FES
WarpX	Plasma wakefield accelerator design	LBNL	DOE HEP
ExaSky	Cosmological probe of the Standard Model	ANL	DOE HEP
EQSIM	Seismic hazard risk assessment	LBNL	DOE NNSA, NE, EERE
E3SM-MMF	Regional assessments in earth systems models	SNL	DOE BER
CANDLE	Accelerate and translate cancer research	ANL	NIH

Table 6: ECP applications targeting KPP-2.

Project name	Short description	Lead lab	Stakeholder program
GAMESS	Biofuel catalyst design	Ames	DOE BES
ExaAM	Additive manufacturing of qualifiable metal parts	ORNL	DOE NNSA, EERE
ExaWind	Predictive wind plant flow modeling	NREL	DOE EERE
Combustion-Pele	Combustion engine and gas turbine design	SNL	DOE BES, EERE
MFIX-Exa	Multiphase flow reactor design	NETL	DOE EERE
ExaStar	Demystify the origin of chemical elements	LBNL	DOE NP
Subsurface	Carbon capture, fossil fuel extraction, waste disposal	LBNL	DOE BES, EERE, NE, FE
ExaSGD	Reliable and efficient planning of the power grid	ORNL	DOE EDER, CESER, EERE
ExaBiome	Metagenomics	LBNL	DOE BER
ExaFEL	Light source-enabled analysis of molecular structure	SLAC	DOE BES
LANL ATDM	Ristra application	LANL	DOE NNSA
LLNL ATDM	MARBL multiphysics code	LLNL	DOE NNSA
SNL ATDM	SPARC for virtual flight testing and EMPIRE for electromagnetic plasma Physics	SNL	DOE NNSA

Because an exascale machine promises a rate approximately 50× the theoretical flop rate of the fastest currently deployed machine, the ECP sets the minimum FOM improvement aggressively at a factor of 50. The ECP supports complex multiphysics codes that put great demand on various aspects of the system: I/O capacity and bandwidth, memory bandwidth, and memory latency (i.e., not just floating-point instruction capacity and throughput). Many applications are not based on algorithms that can perfectly use specialized hardware features. Aside from improved use of hardware use, one key focus of the ECP is the development of innovative algorithms that can achieve the same accuracy more efficiently. In cases for which new methodologies are developed (e.g., using lower complexity algorithms or reduced iteration counts), AD projects must demonstrate equivalent or better accuracy relative to the baseline approach. Incentivizing algorithmic advances is critical to the long-term impact of the ECP in the computational science community. Although risky, any projects that are successful in this approach could have FOM ratios much greater than 50. However, the final KPP-1 calculation gives no additional credit for measurements beyond the target value of 50, so one extreme success will not skew the overall project metrics.

FOM formulations were initially developed by the subproject leads and were iterated upon and vetted over a 2 year period. This includes a panel of external SMEs; technical ECP leadership across the entire management team; and experts at ALCF, NERSC, and the OLCF, as well as in ECP-wide meetings and previous project reviews. Furthermore, the KPP-1 definition, including threshold and objective targets, was modified from the original plans based on extensive external feedback.

The challenge problems defined by all KPP-1 and KPP-2 applications represent ambitious but realizable goals that take into account all the risks and uncertainty of such a complex project. Given the presence of accelerated schedules, highly specialized hardware, evolving software and application-level libraries, and open questions about programming models and compiler technology, some of the applications are likely to fall behind their initial schedule. On the other hand, if many anticipated risks are never triggered or are readily mitigated, some or even all the applications might achieve their individual KPP goals earlier than expected. This best-case scenario is accommodated by defining objective KPP values for each application subproject, which are based on stretch goals. *Stretch goals* are extended challenge problem definitions (i.e., challenge problems that require additional physics capabilities, code coupling, more complex geometries, or, in some cases, even larger problem sizes). Stretch goals represent a best-case scenario that requires many key pieces to fall into place, but they stand as critical definitions of the most ambitious realizable goals each project can envision within the scope of the current project.

Given the specialized nature of the hardware and the breadth and complexity of the application projects, it is highly unlikely that all KPP-1 applications will meet or exceed the target FOM increase. Therefore, the ECP sets the threshold value for project success at 50% of KPP-1 applications, which is determined by the ECP to be an ambitious but attainable goal and concurred with by the ECP's DOE sponsors. The objective value for KPP-1 is that 100% of the application subprojects meet or exceed their target FOM increase and also demonstrate their stretch goal. This objective value is considered an extremely ambitious goal that will further drive the science and engineering goals of ECP applications.

2.2 KPP-2

KPP-2 is intended to assess the successful creation of new exascale science and engineering DOE mission application capabilities. Applications that target KPP-2 are required to define an exascale challenge problem that represents a significant capability advance in its area of interest to DOE. These challenge problem targets are reviewed internally and externally to confirm that they are impactful, challenging, tractable, and of interest to a key DOE stakeholder. The distinguishing feature of KPP-2 applications relative to those that target KPP-1 is the amount of new capability that must be developed to execute the exascale challenge problem. Many KPP-2 applications lack sufficient code infrastructure from which to calculate an FOM performance baseline (e.g., they started in the ECP as prototypes). Without a well-defined starting point at the 10–20 PFlops scale, it is unclear what FOM improvement would correspond to a successful outcome. A more appropriate measure of success for these applications is whether the necessary capability to execute their exascale challenge problems is in place at the end of the project, not the relative performance improvement throughout the project.

Applications that target KPP-2 (Table 6) are required to provide a detailed milestone plan that outlines all the work needed to execute their exascale challenge problem successfully. These milestone plans and the

teams' progress in executing them are reviewed annually by AD leadership and external SMEs as part of the AD annual assessment. Progress toward KPP-2 is tracked between reviews with a dashboard to monitor timely milestone delivery.

To quantitatively assess the successful completion of KPP-2, applications must demonstrate the capability to effectively use exascale hardware to execute their challenge problem. Because access to exascale resources might be limited and performance optimization might not yet be complete, KPP-2 applications can demonstrate exascale capability without running their challenge problem at full scale. This requires application teams to demonstrate: (1) parallel scalability sufficient to run at full exascale; (2) the ability to use specialized exascale hardware features, such as accelerators; and (3) the completion of all necessary physics and algorithmic capabilities to successfully perform the challenge problem. Internal and external review will confirm whether a team has satisfactorily met all three requirements. The metric for success is exascale capability; a code that runs on an exascale machine at the same rate or slower than on a pre-exascale machine will not be judged to be successful.

The ECP National Nuclear Security Administration (NNSA) applications are primarily focused on developing new and essential mission capabilities at exascale. All four NNSA ECP application projects (§§ 7.1, 7.2, 7.3, and 7.4) therefore target the ECP's KPP-2 metric. Because the national security nature of the NNSA challenge problems require a secure NNSA exascale computer (El Capitan), which will not be available before the ECP's current schedule to complete, progress and successful development of exascale capability by these applications cannot be assessed in the same way as the open DOE Office of Science (SC) applications. Instead, the ECP will leverage the NNSA Advanced Simulation and Computing (ASC) program milestone review and certification process by which these NNSA ECP applications will be assessed annually from FY19–23 for the necessary physics and algorithmic capabilities needed to execute their exascale challenge problems. The rigor of this process ensures that successfully completing these milestones through the end of the ECP does indeed verify that these applications can execute their exascale challenge problem once El Capitan enters its secure computing phase in FY24.

The applications that target KPP-2 are working toward a significant advance in simulation capability (i.e., physics and numerical fidelity) in a relatively short time. As such, it is unlikely that all applications will be able to fully complete these ambitious objectives. Thus, the ECP sets the threshold value for project success at 50% of KPP-2 applications, which is determined by the ECP to be an ambitious but attainable goal and is concurred with by the ECP's DOE sponsors. Because the review and assessment criteria are slightly different for the NNSA applications, two out of the three must demonstrate exascale capabilities to meet the KPP-2 threshold.

Like KPP-1 applications, each KPP-2 application defines a stretch challenge problem that is above and beyond the baseline exascale challenge problem. The objective value for KPP-2 is that 100% of the application subprojects demonstrate their stretch challenge problem.

2.3 KPP-3 for Co-Design

KPP-3 is used to measure the impact of co-design software products and the projects in the ECP's ST scope. ECP KPP-3 impact goals and metrics are the primary high-level means of connecting ECP co-design efforts to the ECP effort as a whole. Achieving these KPP-3 impact goals defines how the ECP's co-design centers are reviewed and how their success is determined.

A KPP-3 integration goal for co-design is defined to be its impact and use on its application customer codes, primarily the AD teams that are striving to meet KPP-1 and KPP-2 goals. The weights for the co-design center goals are determined by how many application teams rely on their software products. Of the co-design centers, AMReX and CEED are considered high-impact and will be assigned a weight of 2; CoPA is considered medium impact and will be assigned a weight of 1; and CODAR, ExaLearn, and ExaGraph are considered lower impact and will be assigned a weight of 1/2. This goal is explicitly tracked and reported for satisfying KPP-3 requirements.

Verifying the success of this goal will be documented as part of the capability and performance demonstrations on Aurora and Frontier needed to demonstrate completion of KPP-1 and KPP-2 objectives. In cases for which co-design capabilities are not explicitly required to meet KPP-1 and KPP-2 goals, separate integration runs will be performed for KPP-3 verification.

For all co-design centers, a passing score and a stretch goal on the number of applications that they will be

integrated into and demonstrated on an exascale platform (Aurora and/or Frontier) have been defined. The KPP-3 threshold is defined as 50% of the products meeting or exceeding their weighted impact goals. The weighted impact stretch goal is the maximum reasonably achievable integration score for a co-design center if capability integrations are successful with all potential ECP applications on both ECP exascale platforms. The KPP-3 objective is that 100 % of the products meet or exceed their weighted impact stretch goals.

3. CHEMISTRY AND MATERIALS APPLICATIONS

End State: Deliver a broad array of science-based chemistry and materials applications that can provide, through effective exploitation of exascale HPC technology, breakthrough modeling and simulation capabilities that precisely describe the underlying properties of matter needed to optimize and control the design of new materials and energy technologies.

The Chemistry and Materials Applications (CM) L3 area (Table 7) focuses on simulation capabilities that aim to precisely describe the underlying properties of matter needed to optimize and control the design of new materials and energy technologies. The underlying physics that governs these application areas is computationally challenging. Capturing quantum effects can introduce significant communication nonlocality and computational complexity, for example. Because efficiently scaling these methods to exascale is likely to be difficult, a key assumption of the CM WBS L3 is that the L4 subproject leads already have significant experience with their methods and algorithms on petascale HPC resources and thus have a good understanding of where the biggest challenges to scalability are mostly likely to lie. The ECP is providing an essential catalyst to help propel these efforts forward so that key DOE priorities can be achieved. Given that this is a broad and very fundamental area of research with applications to many technology areas, it is understood that the ECP cannot provide exhaustive coverage of this area.

Table 7: Summary of supported CM L4 projects.

WBS number	Short name	Project short description	KPP-X
2.2.1.01	LatticeQCD	Exascale lattice gauge theory opportunities and requirements for nuclear and high energy physics	KPP-1
2.2.1.02	NWChemEx	Stress-resistant crops and efficient biomass Catalysts	KPP-1
2.2.1.03	GAMESS	Biofuel catalyst design	KPP-2
2.2.1.04	EXAALT	Molecular dynamics at the exascale	KPP-1
2.2.1.05	ExaAM	Additive manufacturing of qualifiable metal parts	KPP-2
2.2.1.06	QMCPACK	Find, predict, and control material properties	KPP-1

3.1 LatticeQCD

Atomic nuclei and most particles produced by high-energy accelerators are tightly bound composites of quarks and gluons. The fundamental interaction of these quarks and gluons is known as the *nuclear* or *strong force*, which is one of the four fundamental forces of nature: strong, weak, electromagnetic, and gravity. These nuclear interactions are defined with mathematical precision by quantum chromodynamics (QCD), and HPC is required to predict the consequences of this underlying theory. The properties of the resulting bound states and the nature of their strong, highly nonlinear interactions are the central focus of nuclear physics and the context in which high-energy physics research must be conducted.

The strong interactions between quarks and gluons represent 99% of the mass in the visible universe. Understanding these interactions and the phenomena that result is the central goal of nuclear physics. The couplings between the quarks and the W, Z, and Higgs bosons lie at the heart of the Standard Model (SM) of

particle physics and can be studied, often with exquisite precision, by measuring the properties of the bound states formed from these quarks and gluons. QCD is the fundamental theory of the interactions between quarks and gluons and can be solved only through massive computation. Over the last three decades, QCD computations have driven and benefited from the spectacular advances in HPC. Computing at the exascale is essential for reaching two decadal challenges of central importance to nuclear and high-energy physics, which are the challenge problems of focus for the LatticeQCD project.

The advance to exascale capability over the coming decade offers exciting opportunities for groundbreaking discoveries in high-energy and nuclear physics. Exascale computing could realistically simulate the atomic nucleus and discover the first harbingers of new laws of nature, revealing a deeper theory that underlies the present "elementary" particles. These possibilities can be achieved only if new and impending advances in computer science can be harnessed to provide a software framework that allows lattice QCD code to efficiently exploit exascale architectures and application scientists to create and refine that code as new challenges and ideas emerge.

3.1.1 LatticeQCD: Science Challenge Problem Description

Six computations are representative of three of the common fermion actions that USQCD currently uses: (1) the highly improved staggered quark (HISQ) action, (2) the domain wall fermion (DWF) action, and (3) the Wilson-clover fermion action. The FOM for each action is determined from two benchmark components: the generation of a gauge configuration and a typical suite of measurements performed with that gauge configuration.

HISQ : This benchmark (Table 8) performs the calculations needed to measure meson masses and decay constants. The benchmark problem measures the rate of generating a new gauge configuration by using an molecular dynamics (MD) algorithm and the rate of making measurements on the gauge-field configuration.

DWF : As with the HISQ action, two FOMs for the DWF component were adopted. The first measures the rate at which a current state-of-the-art gauge field ensemble can be generated, and the second calculates a suite of observable by using this ensemble. Table 9 providea problem specifications.

Wilson-clover : The Clover benchmark has two components (Table 10). The first is the rate at which dynamical Clover fermion lattices can be generated by using an MD algorithm. Several solutions of the Dirac equation are computed and contracted to construct observables as part of the second component of the benchmark.

3.1.2 LatticeQCD: KPP Stretch Goal

The stretch goal is simply stated: instead of choosing the better system F for each fermion type, the team will compute the full FOM for both Aurora and Frontier on their own. The stretch goal is then that both the LatticeQCD base is optimized for both Aurora and Frontier to the point that FOM(Aurora) and FOM(Frontier) each exceed 50×.

3.1.3 LatticeQCD: Figure of Merit

The base FOM will be the arithmetic mean of the six-component FOMs. Those components comprises pairs of components corresponding to the three fermion types: HISQ, DWF, and Wilson-clover. For each fermion type, each pair comprises a gauge-configuration-generation component and a measurement component. Each of the component FOMs is defined to be

$$\mathrm{FOM}(B_a \to F_b) = \frac{t_a(B) f_a(B) n_b}{t_b(F) f_b(F) n_a}\,, \tag{1}$$

where B and F represent the baseline and final system; a and b represent baseline and target problems; and t, f, and n represent the wall time, fraction of the system used, assuming the benchmark run is part of a large ensemble, and complexity of the problem (Flops). After computing the FOM for each fermion type on

Table 8: LatticeQCD HISQ challenge problem details.

Functional requirement	Minimum criteria
Physical phenomena and associated models	Meson decay constants and masses from first principles quantum chromodynamics.
Numerical approach, algorithms	MD, sparse matrix solution, deflation.
Simulation details: problem size, complexity, geometry, and so on.	Generate part of a lattice ensemble with lattice spacing of 0.03 fm on a $240^3 \times 384$ grid with four flavors of highly improved staggered-fermion sea quarks at their physical masses but with degenerate up and down quarks. Specifically, run 4 molecular dynamics time units. Measure a standard set of meson decay constants and masses on those lattices.
Demonstration calculation requirements	An equilibrated ensemble is needed, which is estimated to require at least 1000 MD time units of evolution. Measurements could be done on a single lattice.
Resource requirements to run demonstration calculation	To equilibrate the lattice in preparation for the demonstration calculation, 150,000 socket-hours on the exascale machine are needed, where a performance of 100 GFlops per socket is assumed. To run the demonstration calculation of only two MD time units, approximately 10 socket hours for gauge-field generation are required. To demonstrate a single measurement, approximately 100 socket hours are required.

Table 9: LatticeQCD DWF challenge problem details.

Functional requirement	Minimum criteria
Physical phenomena and associated models	Study the decays of K, D, and B mesons. Examine simple single-hadron final states and more complex processes that involve multi-hadron final states, decay-induced mixing, long-distance effects, and E&M processes.
Numerical approach, algorithms	Use the methods of lattice QCD and a chiral fermion formulation. E&M effects are treated with infinite-volume methods. Linear and bilinear combinations of composite operators are renormalized non-perturbatively. Requires Lanzcos eigenvectors, deflation, all-to-all propagators, all-mode-averaging, open boundary conditions, and Fourier acceleration.
Simulation details: problem size, complexity, geometry, and so on.	The target lattice volume is $96^3 \times 384$ with a lattice spacing of $a = 0.055$ fm. The Wilson gauge action and Mobius DWF would be used.
Demonstration calculation requirements	• MC evolution for 5 time units of the physical mass, $96^3 \times 384$, $a = 0.55$ fm ensemble. Start with a replicated equilibrated configuration constructed from 162 periodic copies of a $32^3 \times 64$ configuration. • Standard suite of measurements on a single configuration.
Resource requirements to run demonstration calculation	25% of the full exascale machine for 6 h for evolution and for 10 h for measurements.

Table 10: LatticeQCD Wilson-clover challenge problem details.

Functional requirement	Minimum criteria
Physical phenomena and associated models	Hadronic correlation functions and energies from QCD. On an ensemble of gauge fields, construct Euclidean correlation functions within many-body systems.
Numerical approach, algorithms	Lattice QCD with a minimum grid size of $64^3 \times 128$ and lattice spacing of 0.091 fm. Use the hybrid MC MD algorithm and sparse matrix solutions. Analysis methods will use multiple right-hand side solvers.
Simulation details: problem size, complexity, geometry, and so on.	Generate part of an isotropic Clover ensemble with two light physical quark masses and one strange quark on a lattice size of $96^3 \times 256$ and a lattice spacing of 0.06 fm. Specifically, run 10 trajectories (MC time units) to observe the behavior and stability of the algorithm.
Demonstration calculation requirements	A fully equilibrated lattice is needed. This will require about 1000 MC time units. Measurement tests can be done on a single lattice.
Resource requirements to run demonstration calculation	Need to fully equilibrate an ensemble before measurements. Using Summit as a baseline, this equilibration is estimated to require about 10,000 Summit node hours. For the FOM, 10 MC time units are required, needing ∼3000 Summit node-hours. Will test on a range of machine sizes, but the minimum is 256 Summit nodes. For the measurement tests, a range of partitions will be tested on and can scale to 2500 Summit nodes. This test will require approximately 1000 Summit node hours.

Table 11: The six FOM-component baseline values from Titan and Mira. The machine fraction is based on 18,688 Titan nodes and 49,152 Mira nodes.

Component	$t(B)$(h)	Machine	$f(B)$	$t(B)f(B)$
DWF gen.	0.667	Mira	0.1667	3.97×10^{-2}
DWF meas.	5.26	Mira	0.6667	3.51×10^{-0}
HISQ gen.	1.58	Mira	0.2500	4.83×10^{-2}
HISQ meas.	1.73	Mira	0.2500	2.60×10^{-2}
Clover gen.	1.11	Titan	0.0548	7.57×10^{-4}
Clover meas.	0.0473	Titan	0.0068	5.73×10^{-5}

Aurora and Frontier, the team will choose the better result for each fermion type and take the arithmetic mean of those three numbers to obtain the *base* KPP goal.

FOM Update

The current LatticeQCD FOM calculation for each of our six components is summarized in Table 12. Values for the Mira/Titan baseline are shown in Table 11. The improvement ratios in the last column of Table 12 are calculated as in Eq. (1) for each component from values in the two tables. The current resulting average is 22.42. The figures in the two tables are based entirely on Mira/Titan-scale problems, so the difficulty factors in Eq. (1) were not needed. We are in the process of scaling up the problems to our challenge-problem size, in which case we will need them. For example, the HISQ measurement problem will grow from a $144^3 \times 288$ lattice on 144 Summit nodes to a $192^3 \times 384$ lattice on 432 Summit nodes and, in the process, decrease the lattice spacing from 0.042 to 0.03 fm. Similar increases are being applied to the other fermion discretizations.

Because target cases are too diverse to quantify, our significant reduction is not included in our FOM in the effort to calculate multiquark matrix elements. These matrix elements appear in calculations that

Table 12: Summary of the six current FY20 Q4 LatticeQCD component FOMs from Summit. The fraction of Summit is based on its full complement of 4608 nodes. The FOM in the last column is the ratio of the baseline value $t(B)f(B)$ in Table 11 to the $t(F)f(F)$ column in this table.

Component	$T(F)$(h)	$f(F)$	$t(F)f(F)$	FOM
DWF gen.	1.187	0.0278	3.30×10^{-2}	3.37
DWF meas.	40.05	0.0278	1.11×10^{-0}	3.15
HISQ gen.	1.51	0.0208	3.15×10^{-2}	12.56
HISQ meas.	0.47	0.0312	1.47×10^{-2}	29.45
Clover gen.	0.109	0.0069	7.57×10^{-4}	80.35
Clover meas.	0.0165	0.0035	5.73×10^{-5}	5.65
Mean				22.42

involve light nuclei, such as deuterium and lithium. The number of quark line routings in such cases grows factorially with the number of quarks, thus presenting an enormous combinatoric problem. During the ECP, the Jefferson Laboratory group used graph theory to organize the calculation in a way that largely reduced the computational requirement, making it possible to consider ab initio calculations for the first time at ECP challenge-problem sizes. The new method has been applied to the recent successful calculation of the decay rates of an exotic hybrid meson resonance before any experimental measurement [2].

3.1.4 LatticeQCD: Progress on Early and Pre-Exascale Hardware

Performance on Summit

The performance improvement on Summit comes in approximately equal parts from greater computing power and improved algorithms. Given the importance of algorithms, approximately half of our ECP effort is devoted to finding better algorithms. Here is a list of key algorithmic improvements:

- **Adaptive multigrid solvers:** Before the ECP, some of our team developed an adaptive multigrid solver for the clover fermion discretization, originally designed to optimize measurements for processes that involve light quarks. During the ECP, the Jefferson Laboratory group found a way to incorporate the multigrid solver in the generation of gauge-field configurations. This was the main reason for the improvement factor of 80.35 in the FOM table for this component. Applying multigrid methods to the other fermion discretizations is highly nontrivial. During this ECP, a combination of the Boston University group and NVIDIA discovered an implementation for the HISQ discretization. It was applied to the HISQ measurement component, contributing a good part of the 29.45 improvement factor. The same team recently discovered an implementation for the domain-wall discretization [3].
- **Communication-avoiding strategies for sparse matrix solves:** Limitations in the internode network present a serious bottleneck for our calculations, so it is important to find algorithms that limit or avoid communication.
  - Mixed-precision solvers: Most sparse-matrix solvers can be implemented with low-precision preconditioning with high-precision refinement. This strategy allows the bulk of the calculation to proceed with fewer bytes per communicated value, thereby reducing message sizes.
  - Split-grid methods for solvers and eigensolvers: In some cases, it is possible to reassign a full-machine-partition workload to multiple parallel subpartitions. Message traffic is reduced in the process. This method was developed by the Brookhaven-Columbia-Riken collaboration for eigensolvers and sparse-matrix solves, where it has proven to be cost effective.
- **Fourier-accelerated gluon-configuration generation:** With finer lattice spacings planned for exascale calculations, it takes much longer to generate statistically independent gauge configurations. This effect is manifested in growing autocorrelation times in the Markov chain that generates them.

This phenomenon is well-known, but there are no effective methods for mitigating it yet. Our ECP "critical-slowing-down" group has devoted some effort to finding such methods. Thus far, it has succeeded in obtaining a 13× decrease in autocorrelations with weak pure-gluon evolution.

- **Contractions:** The integration over fermion fields gives rise to a huge combinatoric challenge in the evaluation of the graphs via tensor contractions. With the finer lattice spacings planned for our project, the basis size that represents the particles grows, leading to increased complexity for many-body nuclear systems. Our project has devoted effort to finding optimal methods for evaluating these tensor contractions and has found a 3*times* improvement for meson systems and a 70× improvement for three-nucleon systems (tritium). These developments have made calculations for such systems viable on Exascale systems.
- **NVIDIA NVSHMEM and kernel fusion:** Some of our calculations enter the strong-scaling regime (e.g., the coarse-lattice phase of the multigrid algorithm). In this regime, message-passing and kernel-launching latencies limit performance. The NVIDIA NVSHMEM feature bypasses message passing interface (MPI), and kernel fusion reduced overhead. These features improve multinode performance by approximately 20 to 30 %.
- **NVIDIA tensor cores:** The NVIDIA tensor cores perform small-matrix operations with high efficiency. We are seeing some significant performance gains in some of our kernels when they are reorganized to take advantage of them.

Next Steps

In the coming year, we will continue the diversification of QUDA with particular focus on SyCL and HIP. We will continue to optimize Grid for Intel and AMD processors as they become available. In both cases, we will be connecting our high-level codes with QUDA and Grid and testing the full suite on available hardware. We will also continue implementing QDP-JIT and QDP++ on Intel and AMD architectures.

3.2 NWChemEx

The strategic goals of the NWChemEx project are as follows:

- To provide the molecular modeling capabilities needed to address two science challenges involved in the development of advanced biofuels: the design of feedstock for the efficient production of biomass and the design of new catalysts for the efficient conversion of biomass-derived intermediates into biofuels.
- To provide a framework for a community-wide effort to develop a next-generation molecular modeling package that supports a broad range of chemistry research on computing systems ranging from terascale workstations and petascale servers to exascale computers.

NWChemEx is based on NWChem, an application supported by the DOE SC Biological and Environmental Research (BER) program office, which is an open-source computational chemistry program that is being actively developed by an international consortium of scientists. NWChem is a high-performance parallel code that provides a broad range of capabilities for modeling molecular systems. The NWChemEx project is re-designing and re-implementing NWChem for pre-exascale and exascale computers. NWChemEx will develop high-performance, scalable implementations of two major physical models:

- Hartree-Fock and Density Functional Theory Methods. Hartree-Fock and density functional theory (DFT) methods are the foundations for the physical models to be incorporated in the NWChemEx framework. The implementation of these methods must be significantly revised to simulate the large molecular systems in the targeted science challenges on exascale computers.
- Coupled Cluster Methods. A robust suite of canonical, domain local, and explicitly correlated (F12/R12) coupled cluster (CC) methods will be implemented in NWChemEx. These methods are the "gold" standard in electronic structure theory and provide the level of fidelity required to address the above targeted science challenges. Although the canonical CC implementation is far more computationally intensive than domain local and explicitly correlated implementations, canonical CC methods are required to validate the approximate localized and reduced scaling CC methods.

In addition to the above, the NWChemEx project is developing density functional embedding theory (DFET) in order to describe an active site and its environment. Embedding techniques provide a natural and mathematically sound basis for seamlessly integrating subsystems with different electronic structure representations, enabling the active site of interest to be described with high accuracy CC methods, while using a lower fidelity method such as DFT to describe the impact of the environment on the molecular processes in the active site. Finally, a number of auxiliary computational methods are being implemented that will be needed to address the science challenges.

Although the NWChemEx project is driven by the two targeted science challenges, there are many other science challenges within the mission of the DOE that can be addressed using this and future versions of NWChemEx, including the development of new materials for solar energy conversion and next generation batteries, simulation of chemical processes in combustion, predicting the transport and sequestration of energy by-products in the environment, development of a science of synthesis, design of new functional materials, and design of separation ligands for critical materials needs.

3.2.1 NWChemEx: Science Challenge Problem Description

To guide the development of NWChemEx, the project is focusing on two interrelated target science problems (one base and one stretch) that are critical for the development of advanced biofuels: (1) the optimization of feedstocks for the efficient production of biomass for biofuels and other bioproducts on marginal lands and (2) the development of new catalysts for the efficient conversion of biomass-derived intermediates into biofuels and other bioproducts. The development of advanced biofuels is driven by both energy security and climate change considerations. A major goal of DOE's advanced biofuels program is to develop fuels that can use the existing infrastructure and replace existing fuels on a gallon-for-gallon basis. However, producing high-quality biofuels in a sustainable and economically competitive way is technically challenging, especially in a changing global climate. The NWChemEx project directly addresses one of the Priority Goals in DOE's *2014-2018 Strategic Plan*, namely, developing high-performance computational "models demonstrating that biomass can be a viable, sustainable feedstock" for the production of biofuels and other bioproducts.

Accurate quantum chemical simulation of the molecules and molecular processes that arise in the development of advanced biofuels is not feasible using current computational chemistry packages and existing computer systems. The molecular systems are complex and involve hundreds to tens of thousands of atoms in an active site that is embedded in an environment that may contain hundreds of thousands of atoms. In addition, the active sites themselves have a large and non trivial configuration space—variations in the spatial arrangement of the atoms that affect the reaction pathways, activation energies, and rates. The relationships between the structure and composition of the active sites on the one hand and reaction pathways and energetics on the other are poorly understood and lack predictive power. The two science challenges are briefly described below, and problem details are listed in Table 13.

Proton Controlled Membrane Transport in Biomass Cellular Materials (Base).

The first focal point for the NWChemEx project is transport across cellular membranes in response to biotic and abiotic stresses of importance to BER. Membrane transporters form gates between cells and the environment for the flow of metal ions as well as carbon, nitrogen, nutrients, and metabolic products and are key modulators of stress. An example is the Bax inhibitor that controls the transport of Ca^{2+} in transgenic sugarcanes. The process driving trans-membrane transport in the Bax inhibitor is poorly understood, although from experimental studies the mechanism appears to be proton controlled and involves two active sites, with one of those sites undergoing large conformational changes on protonation. It is critical to have a detailed molecular understanding of transport processes involved in stress responses to develop genetic modifications that lead to better stress-resistant crops.

Describing proton-controlled ion (Ca^{2+}) transfer in the Bax inhibitor in its local cellular environment requires modeling of hundreds of thousands of atoms to describe a suitable portion of the cellular membrane, the 3500 atom Bax inhibitor-1 protein, as well as a sufficient region of the immediate cytoplasmic environment. Currently proton-controlled transport simulations can only be performed using standard force fields that lack a description of the proton transfer process. Truly predictive modeling of this molecular system requires use of high-level quantum mechanical methods, describing $\sim 10^3$–10^4 atoms with CC methods embedded in an environment of $\sim 10^5$ atoms described by DFT to parameterize the proton hopping processes with chemical

Table 13: NWChemEx challenge problem details.

Functional requirement	Minimum criteria
Structure and energetics of molecules	Solution of the electronic Schrödinger equation to predict the structures and energetics of the reactants, products, and transition states involved in the conversion of propanol to propene.
Numerical approach, algorithms	Hartree-Fock, Density Functional Theory, and Coupled Cluster theory (both canonical and reduced scaling versions). Coupled Cluster theory is required to achieve an accuracy of 1 kcal/mol or better in the prediction of the energetics of molecular interactions, including barriers to chemical reactions.
Simulation details: problem size, complexity, geometry, etc.	There are two targeted science challenges: • Base: run calculations on fragments of the ubiquitin molecule, which is typical of proteins like the Bax inhibitor. The fragments begin with DGLRT, the 79-atom system used to benchmark NWChem, and end in ubiquitin (to be run on Frontier and Aurora). The calculation on Ubiquitin is representative of Science Challenge #1 and define the final KPP-1 and FOM. • Stretch: run calculations on the molecular system involved in the dehydration of propanol by H-ZSM-5 zeolite. The unit cell of the H-ZSM-5 zeolite is $Si_{96}O_{192}$, and the team will run calculations to predict the binding energy of water and propanol and their reactions in the zeolite cavity. These calculations are representative of Science Challenge #2 and will define the science stretch goals.
Demonstration calculation requirements	Iterative solution of the coupled cluster single and doubles (CCSD) equations followed by the calculation of the perturbative triples (T) correction. The latter is the most numerically intensive, time-consuming part of the CCSD(T) calculation and has a well-defined dependence on the computational details (number of electrons, number of occupied orbitals, number of virtual orbitals, etc.). Both the CCSD and (T) correction will be used to define the FOM.
Resource requirements to run demonstration calculation	The computing resources needed to run the demonstration base FOM calculation amount to the whole exascale machine for 2 hours. The calculation requires an aggregate memory amount of about 10 TB of data for the reduced scaling coupled cluster perturbative triples (T) calculation.

accuracy for subsequent use in, for example, simulation of proton dynamics using adaptive force fields and molecular dynamics and long-time conformational sampling at timescales of milliseconds of protonated and de-protonated states.

Catalytic Conversion of Biomass to Biofuels and Other Bioproducts (Stretch).

The second focal point for this project is the prediction of specific, selective, and low-temperature catalytic conversion of biomass to fuels and other products within complex interfaces of importance to DOE SC Basic Energy Sciences (BES) program office. Zeolites, such as H-ZSM-5, offer great promise for the catalytic conversion of renewable biomass-derived alcohols into fuels and chemicals. Compared to metal oxides with diverse surface and acid properties, zeolites have relatively well-defined and uniform Brønsted acid site structures, which makes them amenable to rigorous kinetic and theoretical investigations of the effect of acid strength and solvation environment and confinement on the reaction free energies. Although there have been a number of prior atomic-scale computational studies of these systems, unraveling the true complexity of the conversion process and identifying means of achieving conversions at lower temperatures and pressures is an unsolved problem. Nonetheless, there exists a body of experimental data for multiple chemical transformations of systems like propanol dehydration on which to benchmark new theoretical approaches and computational models.

The NWChemEx project will develop the capabilities needed to accurately model propanol dehydration. This requires (1) modeling the unit cell of the H-ZSM-5 zeolite along with propanol, water, and other species involved in the dehydration process, which will involve $\sim 10^2$–10^3 atoms, with CC theory, (2) embedding the unit cell in the larger zeolite environment using embedding techniques based on the DFT method, which will involve 10^4–10^5 atoms, (3) computing barrier heights and reaction energies to chemical accuracy (1 kcal/mol) with well-defined error bars for both the enthalpic and entropic terms, which can only be achieved with accurate CC methods and embedding methodologies, (4) inclusion of thermal effects on the reactants, products and transition state, and (5) predicting reliable chemical rate data for the reactions involved in the reaction network.

3.2.2 NWChemEx: KPP Stretch Goal

By the end of the Exascale Computing Project two exascale computing systems are expected to be available (Frontier at ORNL, Aurora at ANL) as well as a more complete set of computational chemistry capabilities. This will enable us to address far more complex molecular systems. Specifically, the team plans to model the conversion of 1-propanol to propene in the zeolite H-ZSM-5 as the exascale scientific stretch goal. Zeolites offer great promise for the catalytic conversion of renewable biomass-derived alcohols into fuels and other chemicals. Zeolites have relatively well-defined and uniform Brønsted acid site structures, the sites responsible for the conversion of alcohols to hydrocarbons, which makes them amenable to rigorous kinetic and theoretical investigations. In spite of a number of prior atomic-scale computational studies of these systems, unraveling the steps in the conversion process as well as the enthalpies, entropies and rates of each of these steps is still an unsolved problem.

The capabilities in NWChemEx will be used to rigorously characterize the steps proposed by Zhi, Shi, Mu, Liu, Mei, Camaioni, and Lercher [4] for the conversion of propanol to propene as the stretch exascale target science problem. One or more unit cells of the zeolite will be modeled along with propanol and other species involved in the dehydration process, e.g., water. This will involve calculations on an active site with $O(10^2$–$10^3)$ atoms using the high accuracy coupled cluster CCSD(T) method embedded in a DFT description of the larger zeolite environment of $O(10^4$–$10^5)$ atoms. In addition, this will involve the potential energy surface sampling methods to obtain enthalpies, entropies and rates. To be specific:

- Calculate the structures and energetics of the reactants, products, and intermediates involved in the conversion of 1-propanol to propene.
- Calculate the structures and barrier heights of the transition states involved in the dehydration of propanol.
- Calculate the rates of the reactions involved in the reaction network for the conversion of 1-propanol to propene.

3.2.3 NWChemEx: Figure of Merit

For the FOM of the NWChemEx project, a minimum size of the ubiquitin molecule using an aug-cc-pVTZ basis was selected for assessing the performance of NWChemEx with ORNL's Frontier computer system as the primary target. Ubiquitin is a protein molecule similar to molecules like the Bax inhibitor involved in the first science challenge, and there is an abundance of experimental data on ubiquitin and its fragments. Although it will be possible to run localized CC calculations on ubiquitin, a 1231-atom molecule, it will not be feasible to run canonical CC calculations on this molecule. Since one of the goals of the NWChemEx project is to validate the localized (DLPNO) implementations of the CC method to show accuracy within 1 kcal/mol, a sequence of ubiquitin fragments was generated starting with DGRTL, which has 79 atoms and is the largest molecule that can be modeled by NWChem on Titan, and ending with ubiquitin. This sequence of molecules is described in the report for Milestone 9.1 "Establishment of the Performance Baseline for NWChem" in Jira (ADSE11-166). This sequence will be used to assess the performance and scalability (with respect to molecular size) of the Hartree-Fock (HF), DFT, CCSD, and (T) methods being implemented in NWChemEx.

The protocol for calculating the FOM and KPP are as follows:

- To define a baseline, we performed an NWChem canonical coupled cluster calculations, CCSD(T), on as much of Titan as possible. These were completed right as Titan was being decommissioned and were run on as much of Titan as possible before the the machine was taken down. These simulations used DGRTL as the molecular system and, therefore, there are some challenges to make an estimate of the cost for a larger system like ubiquitin—especially for the CCSD part of the computation.

- Compute intermediate KPP values as we work on the applications utilizing Summit. Simulations of the canonical coupled cluster methods on DGRTL and on larger fragments will be run on Summit to obtain information about scaling for these systems. This scaling information is used to extrapolate the cost of running a canonical CCSD(T) calculation on the ubiquitin (or larger) system for the whole Summit computer. This provides an FOM for the NWChem code.

- Compute a Final KPP value using the Canonical CCSD(T) method. Perform an NWChemEx canonical coupled cluster calculation, CCSD(T), on the largest molecule that is feasible using 100 % (or as much of the system as is available) of the exascale computer in approximately two hours. Combined with scaling data for smaller molecular systems, an estimate will be made for running the NWChemEx CCSD(T) calculation on the ubiquitin (or larger) system to obtain a FOM for NWChemEx. The ratio of the FOM for NWChemEx divided by the FOM for NWChem will be used to define the Base KPP for NWChemEx. The Base KPP will largely represent the advances made in redesigning NWChemEx for exascale computers.

- Compute a Final "enhanced" KPP value using the DLPNO CCSD(T) method. A DLPNO-based NWChemEx CCSD(T) calculation will be run on the target molecule—currently, a minimum size of ubiquitin—on as much of the target machine as possible and will provide the FOM for NWChemEx. The ratio of the FOM for NWChemEx divided by the FOM for NWChem will define an enhancement factor that quantifies the impact of the algorithmic improvements made in NWChemEx—yielding an Enhanced KPP.

The ECP will be provided with both sets of numbers—the first as the Base KPP and the second as the Enhanced KPP, with the Enhanced KPP being the value that will be used for the NWChemEx ECP KPP.

As noted above, ubiquitin is far too large for canonical CCSD(T) calculations with NWChem on Titan (and even canonical NWChemEx on Summit), therefore the use of smaller fragments of this protein was investigated. The smallest of these fragments, DGRTL, is sufficiently small to run with current technology but large enough to provide timings that offer meaningful insights about the performance of NWChem. This protein has 79 atoms and is near the limit of what is computationally tractable at the present time. Further, this molecule has a H/(C+N+O) atom ratio, 1.05, which is close to that of ubiquitin, 1.03. The similarity of this ratio is critical for defining comparable calculations on the small (DGRTL) and large (ubiquitin) molecules. Assessments can be made about NWChem's behavior on a molecular system like ubiquitin using the data from calculations on DGRTL as shown below. As shown in Table 14, the DGRTL molecule is a

Table 14: Comparison of the computational parameters and estimates of the volume of computation for the molecular systems of interest for benchmarking NWChem (DGRTL) and NWChemEx (ubiquitin).

Molecular parameters	DGRTL	ubiquitin	Ratio ($= \frac{\text{ubiquitin}}{\text{DGRTL}}$)
N_{atom}	79	1231	15.58
H_{atom}/ (C_{atom}+N_{atom}+O_{atom})	1.05	1.03	0.98
$N_{electrom}$	292	4592	15.73
N_{mo}	424	6680	15.75
N_{occ}	146	2296	15.73
N_{virt}	278	4384	15.77
KPP $\times t \times \left(N_{occ}^3 N_{virt}^4\right)$	1.9×10^{16}	4.5×10^{24}	2.4×10^8

successful mimic of ubiquitin—the ratios for the various parameters involved in CCSD(T) calculations on the two molecules are confined to the range of 15.58–15.77—meaning that the ratios are consistent enough to provide predictive power.

FOM Update

The current KPP value of 78.8 is based on a simplified molecular system (DGRTL), a smaller basis set (cc-pVDZ as opposed to aug-cc-pVDZ), use of only the canonical CCSD(T), and use of Titan for NWChem and Summit for NWChemEx. As discussed previously, the extrapolations for the full usage of the machine is difficult. For NWChem, we have assumed linear scaling for the (T) portion (the best case scenario) and polynomial for NWChemEx (based on actual computation). The CCSD portion of the calculation uses the same polynomial scaling for both codes; however, this will be improved on using actually scaling information of NWChemEx on Summit.

3.2.4 NWChemEx: Progress on Early and Pre-Exascale Hardware

Since the NWChemEx software was built from the ground up, there is still an extensive amount of software that needs to be implemented—including the DLPNO CCSD(T) methods. However, significant parts of the software, such as the canonical CCSD and DFT methods have been implemented. The project team has made extensive in-roads to obtaining and improving performance across all platforms.

The general approach of the team for portability and scalability is to use separation of concerns and abstract programming interfaces to isolate performance critical portions of the code. Underneath these interfaces, the main programming model has been to use CUDA as the primary programming model and to convert these modules into HIP or SYCL using the appropriate tools. Of course, some hand tuning for each architecture will be required in this approach. However, our experience to date is that we can obtain very good performance in this manner.

A rough sketch of the architecture pieces in the following discussion is given in Fig. 1. TAMM, Tensor Algebra for Many-body Methods, is the tensor framework for the canonical and DLPNO CCSD(T) software. In addition to its own kernels, it uses TAL-SH for tensor operations on a single node which in turn uses a tensor transpose code, cuTT for CUDA and hipTT for HIP.

Performance on Summit

On Summit, significant improvements have been made to both the single node performance and scalability of the canonical CCSD(T) and on a memory reduced CCSD(T) using a Cholesky decomposition. Profiling on a single node (using both CPUs and GPUs) revealed data locality improvements and barrier-minimization opportunities that were implemented. In addition, improvements were made in multilevel parallelization process group issues. xGA was also improved for handling of irregular tiled arrays, performance of non-blocking operations, and to replace low performing MPI operations with higher performing ones. These improvements resulted in a significant improvement in single node performance. For example, comparing against the use of all of the CPUs on one node of Summit with NWChem to the use of all of the GPUs on a Summit node with NWChemEx, we see up to 2.2 times speedup on DGRTL using a small basis set. Last

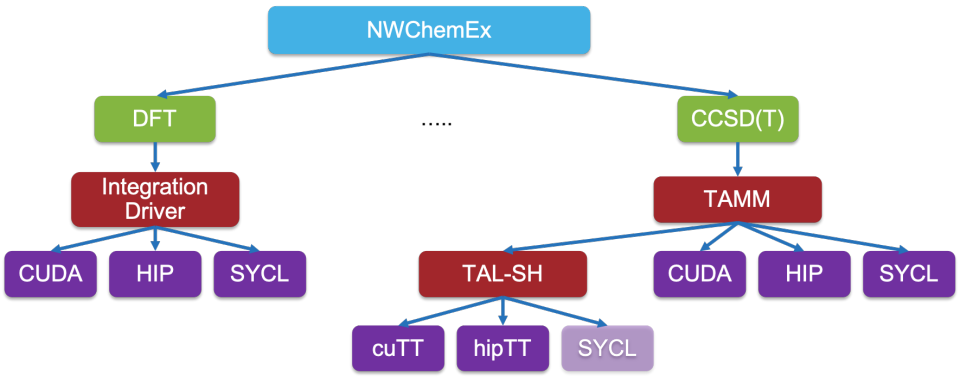


Figure 1: Rough sketch of the relevant NWChemEx architectural components.

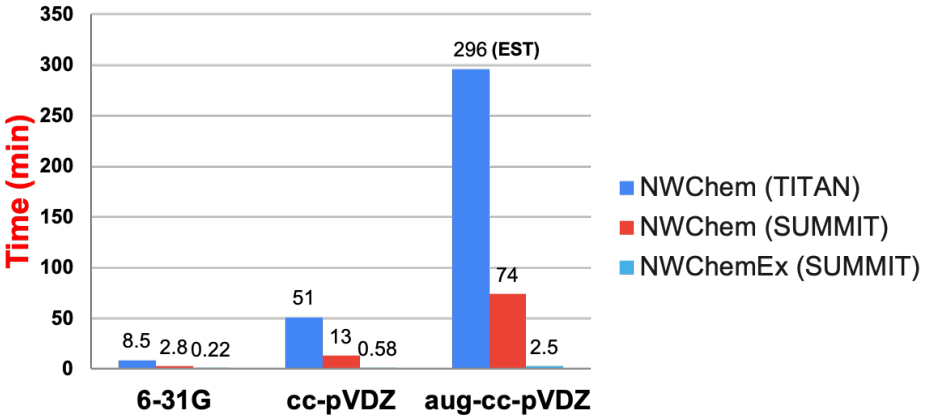


Figure 2: Time in minutes for DGRTL with various basis sets.

year, this same performance was a 50 % speedup.

Improvements in the CCSD algorithm are shown in Fig. 2. The figure shows the time in minutes for NWChem run on Titan in dark blue, for NWChem on Summit in red, and for NWChemEx on Summit in light blue. NWChem does not have a GPU implementation for CCSD and thus the improvements are significant. The simulations were run on 1/46 of Summit for the 6-31G basis, 1/21 of Summit for the cc-pVDZ basis, and 1/18 of Summit for the aug-cc-vPDZ basis. The NWChem Titan time for the latter basis set is an estimated number. These timings have been analyzed in more detail and a significant amount of the lost performance for NWChemEx is in the load imbalance which is currently being improved.

For the (T) portion of the code, comparing the NWChem performance on Summit (using GPUs) to the NWChemEx performance results in speedups ranging from 6 times for the DGRTL molecule in a 6-31G basis to 23 times with a cc-pVDZ basis. (Remember that the KPP is calculated based on the Titan numbers.) The (T) portion of the calculation benefits from the improvements discussed above, but also on specific optimization fusions across all contractions and balancing of operational intensity across the nodes. These improvements are discussed in more detail in an accepted SC20 paper. Additional improvements for the coming year will include using optimized block-level primitives, use of SUMMA-based methods to determine the best tile shape and processor grids on a particular architecture, and to further optimize the overlap of computation and communication.

The DLPNO CCSD code has been implemented and optimization is in progress—using code generation optimization where possible. Precision problems have been identified with the sparse index spaces and debugging is ongoing. Tensor intermediates are also being analyzed for reduced memory footprints or to use additional intermediate tensors to reduce operation cost. The DLPNO (T) code has been implemented and is

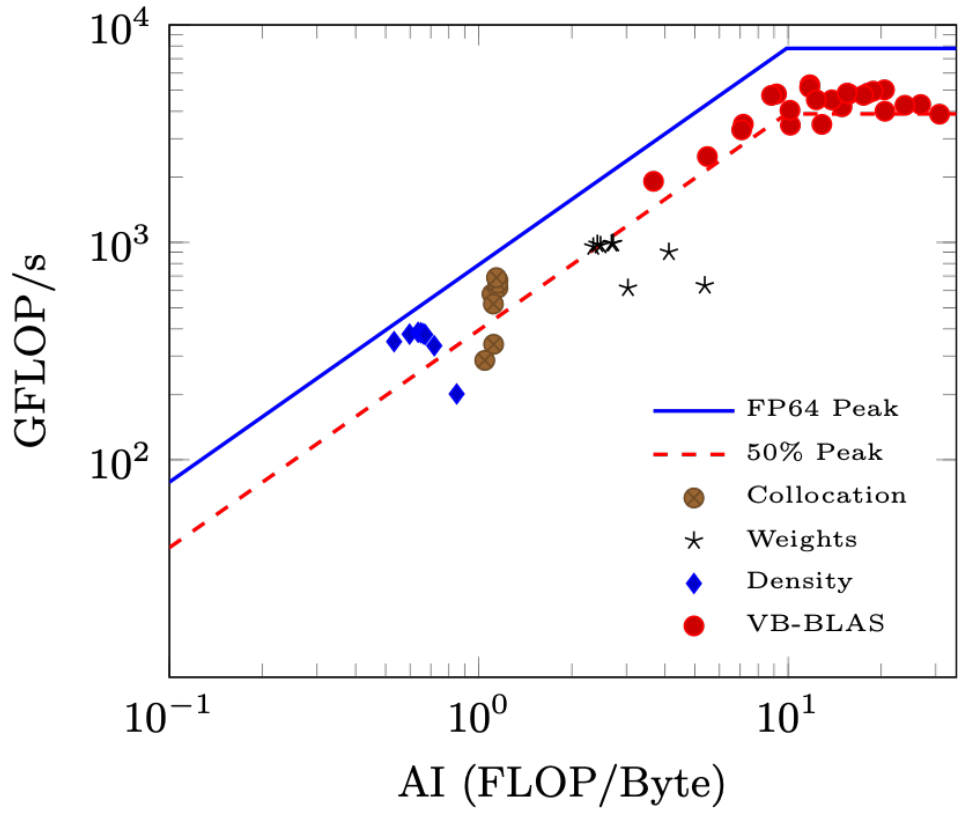


Figure 3: Summit roofline plot for key kernels on NVIDIA V100.

undergoing extensive verification and testing.

The Gaussian based DFT will be used as part of the stretch KPP and, thus, the project has worked to ensure that this code performs and scales well. One of the most expensive components of the DFT code is the numerical integration of the exchange-correlation (XC) potential. While embarrassingly parallel, the algorithms are prone to load imbalance. In keeping with our design and implementation philosophy of separation of concerns, the project has launched a general XC evaluation library, ExchCXX, for use on GPUs. A batched kernel design is used to mitigate kernel launch overhead due to the relatively small task sizes. In this approach logically identical tasks are merged into a single kernel with independent tasks executing on different thread blocks. Batched BLAS are accomplished using the MAGMA library. Figure 3 shows that most of the user defined kernels are bandwidth bound and that the variable sized batched (VB) BLAS are compute bound.

The XC evaluation of NWChemEx shows 5–10 times the speedup of the NWChem XC evaluation on Summit depending on the size of the system. Future work will look to improve the load balance and the performance of distributed reductions using tools from the Pagoda project.

Next Steps

A significant portion of the work in the coming year is to finish the DLPNO implementations and to improve their performance on Summit and early hardware. The framework of NWChemEx will go through a hardening phase and ensure that memoization, I/O, and restart work well. New functionality will include gradients for the Hartree-Fock and DFT portions of the code and the development of explicitly correlated F12 energies in the DLPNO formalism. Future activities will also include integration with the BES SPEC project and the ECP EXAALT project—specifically with LAMMPS.

3.3 GAMESS

Heterogeneous catalysis and the design of new catalysts is a grand challenge problem that will require the availability of exascale computers. To take full advantage of exascale architectures, it is critical that application software be developed that can exploit multiple layers of parallelism and take advantage of emerging low-power architectures that dramatically lower the energy/power cost without the significant deterioration of time to solution. This work will develop ab initio methods in the electronic structure program GAMESS, based on fragmentation methods that have been shown to scale beyond the petascale combined with high-quality quantum chemistry methods. To attain exascale performance, GAMESS is being refactored to take advantage of modern computer hardware and software, and the capabilities of the C++ libcchem code that is co-developed with GAMESS is being greatly expanded. Concurrently, performance analyses is being performed for the broad array of electronic structure methods in GAMESS on current and emerging architectures to assess their ability to decrease time to solution while decreasing energy demands. The new codes and algorithms that are being developed will be achieved on the heterogeneous catalysis problem, specifically by using mesoporous silica nanoparticles (MSN), requiring thousands of atoms as a template.

MSNs are highly effective and selective heterogeneous catalysts for a wide variety of important reactions, including carbinoalamine, which is a starter material for other structures. MSN selectivity is provided by "gatekeeper" groups that allow only desired reactants to enter the pore, keeping undesirable species from entering the pore. The presence of a solvent further complicates the problem. Accurate electronic structure calculations are needed to deduce the reaction mechanisms, including the effects of various solvents, and to subsequently design even more effective catalysts. The narrow pores (2–4 nm) can create a diffusion problem that can prevent product molecules from exiting the pore. Hence, in addition to elucidating the reaction mechanism, it is important to study the dynamics of the reaction process in which a sufficiently realistic cross section of the pore is included. It has been common to approximate a system such as this with a small model so that the small model might provide insight into the actual system. However, a recent computational study of MSN catalysis of carbinolamine formation demonstrated that small "toy" models are inadequate qualitatively and quantitatively.

3.3.1 GAMESS: Science Challenge Problem Description

The team will compute the chemical energetics on a model reaction with a representative MSN. An adequate representation of the MSN pore requires thousands of atoms, including solvent, with an appropriate basis set. For example, 5000 heavy atoms with the aug-cc-pVTZ basis set require more than 500,000 basis functions, not including the hydrogen atoms, the reacting molecules, and (especially) the solvent molecules. The challenge problem specifications are listed in Table 15.

The energy surface will be mapped via GAMESS calculations by using the EFMO + RI-MP2 methodology with refined calculation by using the EFMO+CR-CC(2,3) coupled cluster approach or GAMESS EFMO + QMCPACK QMC approach (as stretch goals) for more accurate reaction rates.

3.3.2 GAMESS: KPP Stretch Goal

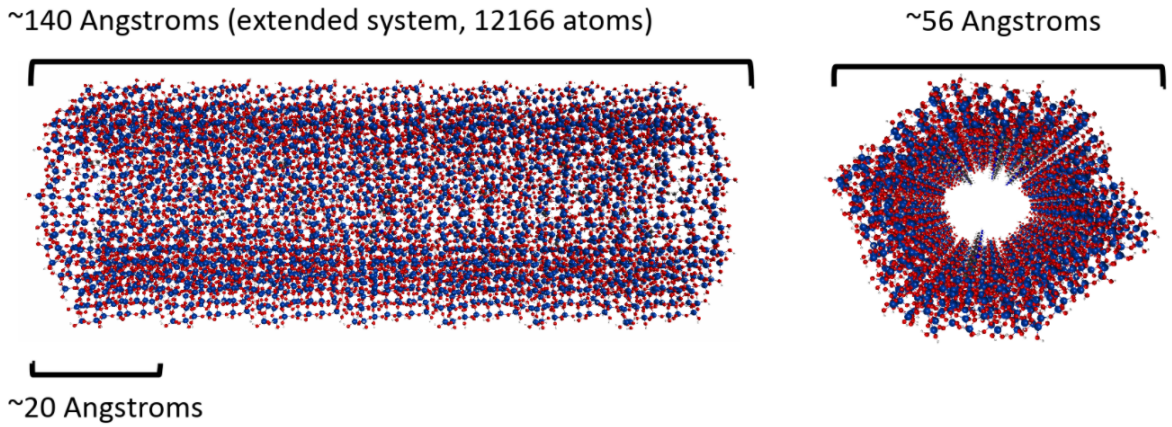
The base goal will be a system that comprises 1738 atoms plus solvent, giving an estimated total of 25,000 atoms. The stretch goal will be an expanded system with ∼76,000 atoms, including solvent. The stretch goal will allow a more realistic treatment of the diffusion problem. The baseline and stretch goals will include 20–40 points (energy + gradient) on the potential energy surface.

Additionally, the base goal will treat all chemistry in the system at the RI-MP2 level, whereas the stretch goal will apply CC and QMC to activation areas and nearby dimer fragments.

3.3.3 GAMESS: Capability Plan

FY19: Demonstrate the MSN problem with 1738 atoms and the following characteristics:

- (E)FMO + RI-MP2 Energies and
- tens of nodes on the CPU system.



(a) MSN model for the GAMESS base challenge problem (20 Å) and stretch challenge problem (140 Å)

Table 15: GAMESS challenge problem details.

Functional requirement	Minimum criteria
Physical phenomena and associated models	MSN Fragment energetics (reaction rates) and dynamics (diffusion rates) computations with at least 10,000 atoms for the pore + solvent. The go-to level of theory will be EFMO/RI-MP2 with an adequate basis set (e.g., 6-31G(d,p)) for the pore + catalyst + gatekeeper. The solvent will be treated with the same level of theory or with EFP. Final energies will be captured by using multilevel EFMO calculations with CC or QMC calculations for the reaction region and RI-MP2 elsewhere.
Numerical approach, algorithms	Configurations can be computed concurrently; each configuration will use the EFMO fragmentation approach to spatially parallelize the calculation of underlying quantum-chemistry methods, which are typically characterized by dense linear algebra-like operations: Hartree-Fock → RI-MP2 → CC/QMC.
Simulation details: problem size, complexity, geometry, and so on.	At least 10,000 atoms, comprising the MSN pore, reactants, and solvents. An estimated 1 million basis functions.
Demonstration calculation requirements	Demonstrate the ability to complete the science challenge problem by concurrently running a subset (1–10) of atomic configurations concurrently with EFMO-RI-MP2 on the full exascale system.
Resource requirements to run demonstration calculation	Full exascale machine for 2–4 h for each energy + gradient RI-MP2 calculation.

FY20: Demonstrate the MSN problem with 1738 atoms plus 6000 solvent atoms with the following characteristics:

- (E)FMO + RI-MP2 energies and gradients,
- RIMP2 accelerated with GPU,
- hundreds of GPU-accelerated Summit nodes,
- developed ability to use general Fock build from LibCChem, and
- RIMP2 mini-app compiled and working on Iris.

FY21: Demonstrate the MSN problem with 1738 atoms plus 12,000 solvent atoms with the following characteristics:

- EFMO I/O bottleneck reduced to less than 10–20 % of the runtime,
- developed general fragmentation method in FragLib (that subsumes current FMO and EFMO) codes,
- fully optimized general Fock build from LibCChem with open shell capability and analytic gradients,
- fully optimized one- and two- electron integral routines from newly developed LibAccInt in GAMESS and LibCChem, and
- fully analytic gradients with EFMO applicable to MSN.

FY22: Demonstrate the MSN problem with 1738 atoms plus 12,000 solvent atoms with the following characteristics:

- GPU-optimized RI CC code for the reaction center (∼20 atoms) with the remaining atoms treated with RI-MP2 and
- multilevel fragmentation for ∼30 points on the potential energy surface.

An initial RI-CCSD method was completed in FY20 Q4. RI-CCSD(T) is anticipated at the end of FY21 Q2, and RI-CR-CC(2,3) is anticipated at the end of FY21 Q4.

FY22 Complete challenge problem.

3.3.4 GAMESS: Progress on Early and Pre-Exascale Hardware

Performance on Summit

The GAMESS RI-MP2 code was off-loaded onto GPUs, and benchmarks were run on Summit. The results of these tests are shown in Table 16. The speedup improves as the size of the problem increases. The strong scaling of FMO/RI-MPI2 for the MSN system with 1738 atoms (32 fragments) on Summit is shown in Fig. 5a. With an added 2000 solvent water molecules, the EFMO/RI-MP2 strong scaling is shown in Fig. 5b. In both cases, FMO/EFMO is run on the CPU, and RI-MP2 is run on the GPU.

The GPU speedup of the GAMESS RI-MP2 code on Summit when using the GPUs vs. just CPUs is 17.2× for the largest basis set calculation on a representative molecule.

For the standalone HF code based on the general Fock build, the team has achieved nearly perfect parallelism for more than 20,000 atoms by using 95 % of Summit. A roofline performance analysis shows the code performing near the non-FMA floating-point operation peak on V100. The strong scaling for the HF code is illustrated in Fig. 6.

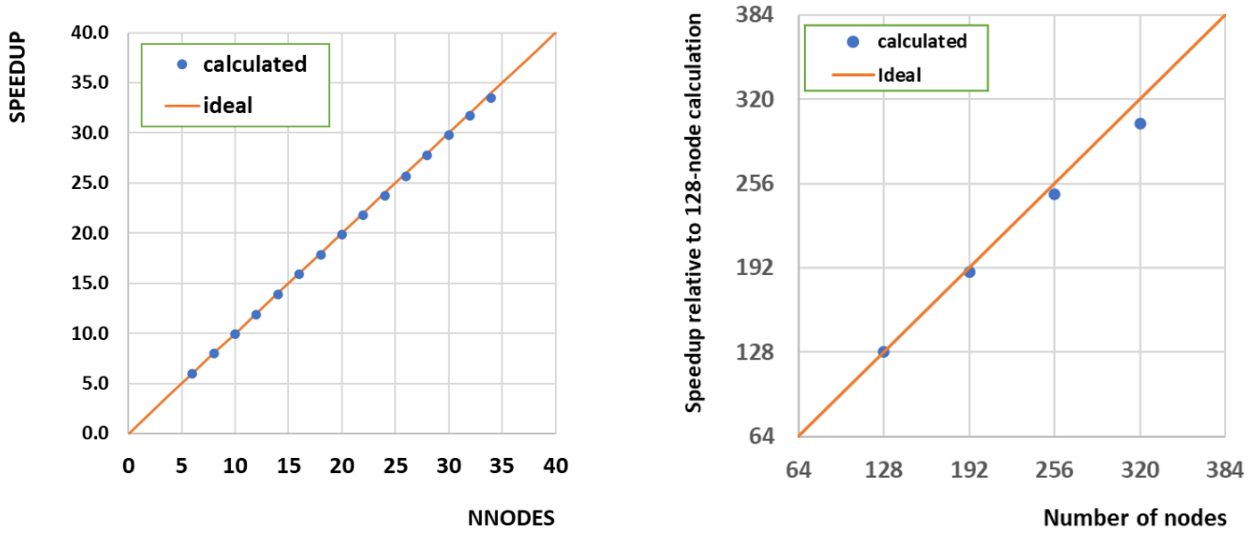
Next Steps

For Tulip, the team will investigate the use of existing rocblas routines to build an eigensolver. The team will modify the build system to accommodate building for NVIDIA and AMD. the team will also run multinode tests.

For Iris, the team will port the full HF stand-alone code to DPC++ and OpenMP off-load. The team will test the GAMESS branch with OpenMP off-load. The team will perform a roofline analysis on the RI-MP2 mini-app once the Intel Advisor works with MKL.

Table 16: Speedups of benchmark problems on Summit. CPU results are run on two Power9 cores, and GPU results are attained by using all six V100 GPUs. Wallclock times are in seconds.

		Wall clock time		
Basis set	N basis	CPU	GPU	Speedup
STO-3G	461	15.6	4.5	3.1
3-21G	779	74.9	9.1	8.2
3-21G(d)	887	106.9	11.1	9.6
6-31G	779	76.3	9.0	8.5
6-31G(d)	1205	232.6	17.2	13.5
6-311G	1169	248.2	19.4	12.8
6-311G(d)	1595	490.6	30.0	16.4
6-311G(d,p)	1697	574.6	33.4	17.2



(a) FMO/RI-MP2. **(b)** EFMO/RI-MP2.

Figure 5: Strong scaling on Summit for the MSN system.

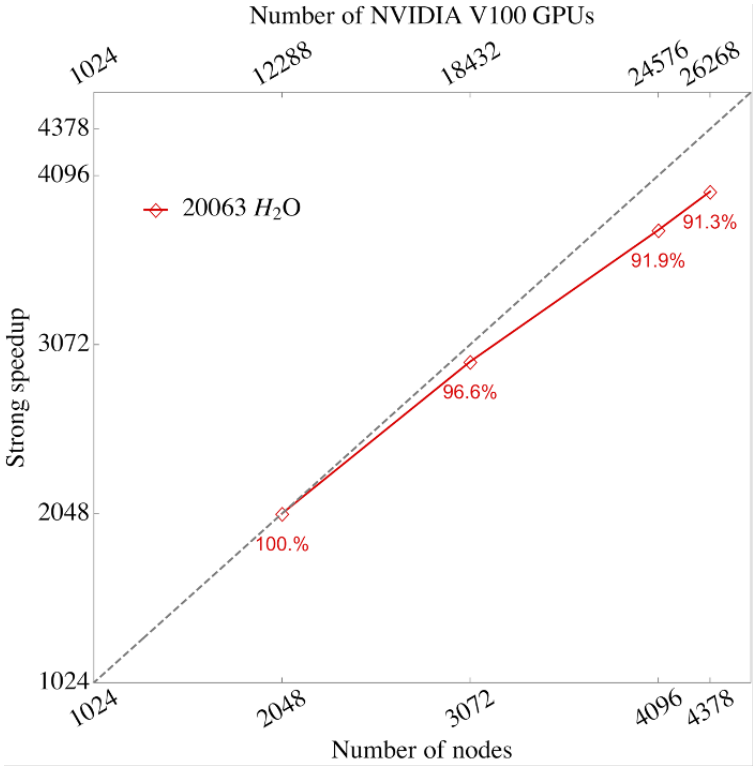


Figure 6: Strong scaling of the standalone HF code.

3.4 EXAALT

MD is a cornerstone of computational sciences. However, over and over, MD is prevented from achieving complete scientific success by the inability to simultaneously reach the necessary length and time scales while maintaining sufficient accuracy. Although the raw computing power available at the exascale should allow for a dramatic extension of the range of applicability of MD, conventional massively parallel codes suffer from poor strong scalability. This implies that a simple scale-up of current practices would only enable the simulation of much larger systems (billions or trillions of atoms) but would do little to improve current timescales (nanoseconds) and accuracy (empirical potentials). Because most challenging problems instead require accessing different regions in the accuracy (A), length (L), and time (T) simulation space (ALT), one of the team's key tools, MD, is in danger of missing out on the exascale revolution.

The EXAALT project combines three state-of-the-art codes—LAMMPS, LATTE, and ParSplice—into a unified tool that will leverage exascale platforms efficiently across all three dimensions of the ALT space. The new integrated capability will comprise three software layers. First, a task management layer will enable the creation of MD tasks, their management through task queues, and the storage of results in distributed databases. It will be used to implement various replica-based accelerated MD techniques, as well as to enable other complex MD workflows. The second layer is a powerful MD engine based on the LAMMPS code. It will offer a uniform interface through which the different physical models can be accessed. The third layer provides a wide range of physical models. In addition to the many empirical potentials implemented in LAMMPS, it will provide high-performance implementations of electronic structure-driven MD at the density functional tight binding (DFTB) level, as well as to SNAP, a high-accuracy machine-learned potential.

The project involves two science challenge problems. The first challenge problem is related to nuclear fission. Nuclear energy based on fission provides about 16% of the world's electricity. However, only 4–6 % of the uranium atoms in the primary fuel, UO_2, are burned, leaving behind a vast energy resource and creating a greater-than-necessary nuclear waste problem. One of the primary reasons is material integrity: as the fuel burns, radiation damage and fission gases accumulate, causing fuel swelling, pellet-clad interactions, and increased pressure on the clad. Because current burn-up levels are predicated on understanding how the fuel

evolves, improved models of fission gas evolution offer the potential for extracting more energy from the fuels. density functional theory—particularly DFT+U—has provided significant insight into the kinetics and thermodynamics of the defects that dictate fission gas evolution. These methods allow the electrons and holes that accompany defects to naturally distribute themselves. For instance, when an oxygen interstitial is inserted into UO_2, it creates two holes that reside on U ions, changing their oxidation state from U4+ to U5+. Critically, DFT+U calculations have identified larger defect clusters that contain up to four uranium vacancies as important for mass transport. However, DFT+U approaches are too computationally expensive to fully characterize these defects. In contrast, empirical potentials are affordable but cannot account for the charge redistribution, which is critical for correctly describing defect properties. For example, if holes are inserted by hand with explicit U5+ species whose pairwise interactions are different than for U4+, then the agreement with DFT + U calculations for static quantities is significantly improved. Unfortunately, this approach is incompatible with studying dynamics because holes must be able to redistribute as the geometry evolves. Furthermore, different defects dominate behavior, depending on experimental conditions (e.g., temperature and burn-up). Success involves knowing how these defects diffuse as a function of size, gas content, and temperature because this behavior forms the input to higher level fuel evolution models.

Solving this grand challenge will require a significant advance in the ability to perform high-accuracy, electronic structure-driven MD simulations on the timescales that are needed to observe the diffusion of defects while accounting for the changing polaron distribution. Given the size of these defects, relatively small systems (<1000 atoms) are sufficient. However, given the high barriers for uranium-defect evolution (∼2.5 eV), millisecond timescales will be required for the defects to move at the temperatures of interest (800–1300 K). This regime is inaccessible on present platforms. Simulation rates of only 1 µs/d for electronic-structure-based DFTB MD at the petascale are estimated. Therefore, the solution to this problem requires the development of a new simulation capability for the exascale that can increase the timescales by 1000×. Developing a computational capability to carry out this challenge problem at exascale is one of the stretch goals of EXAALT.

The second challenge problem relates to nuclear fusion. Realizing the promise of fusion as a commercially attractive twenty-first century energy source requires advanced structural materials capable of sustained operation in an extreme environment with high temperatures and high fluxes of helium, hydrogen isotopes, and neutrons. The performance demands on plasma-facing components (PFCs) of future fusion power plants are beyond the capability of current materials. Tungsten will be the divertor material in ITER and is the leading candidate material for DEMO and future fusion reactors. However, experiments indicate the possibility of substantial surface modification in tungsten exposed to low-energy plasma that contains helium. Experiments show that nanostructured fuzz, a nanoporous phase with tendrils on the order of tens of nanometers in diameter, forms on the surface when the surface temperature is between 1000 and 2000 K and the incident ion energies are between 20 and ∼100 eV. Such surface features will impact heat transfer and fuel retention, increase the rates of erosion through sputtering and dust formation, and embrittle the divertor. These modifications to the microstructure can lead to premature failure of the materials or quench the fusion reaction by cooling and destabilizing the plasma. Given the critical importance of understanding and controlling these microstructural changes, many possible formation mechanisms have been proposed. However, given the current lack of direct evidence on the nature of the microscopic mechanisms postulated to be responsible for fuzz growth, none of these models are widely accepted. However, some key facts are known. Transmission electron microscopy suggests that the nanometer-scale tendrils of fuzz and subsurface regions of tungsten contain gas bubbles and/or cavities, which suggests that bubble evolution is an important process in fuzz formation in tungsten. Fuzz growth is observed to proceed with no apparent saturation in thickness. This raises the question of how low-energy helium finds its way deep below tendril surfaces. Additional questions arise beyond the obvious question of the fuzz formation mechanism: (1) what factors control the temperature dependence of the helium accumulation and transition to fuzz formation, (2) how does low-energy helium penetrate through a thick tungsten fuzz layer to reach the bulk, and (3) what mechanisms control the continued growth of fuzz?

Although sophisticated efforts that leverage current leadership-class computing have enabled advances in understanding, a direct and unambiguous solution to this problem remains out of the reach of current capabilities. To identify the true origin of fuzz, simultaneous increases in time and length scales will be required, a feat only possible at the exascale. Answering these questions is anticipated to require accessing two regions of the ALT space: (1) 10^7 atoms over milliseconds with relatively inexpensive conventional

potentials to identify the nature of the mechanisms of roughening and early-stage fuzz growth and (2) 10^5 atoms over milliseconds at higher accuracy and ∼100–1000× the computational cost per atom to investigate the mechanisms that allow for helium transport along tendrils while accurately accounting for competing kinetics (e.g., trapping, desorption, agglomeration, bubble nucleation). Current simulation capabilities falling short of this target by a factor of 1000× largely explains why this crucial technological problem is still so poorly understood. This second challenge problem is used to define the threshold goal of EXAALT, and progress toward achieving the 50× threshold will be monitored through the KPP defined as follows.

3.4.1 EXAALT: Science Challenge Problem Description

Two exascale challenge problems are defined.

Fusion problem (base goal): The second target problem requires a dramatic extension of the reach of large-size long-time MD simulations. The team aims to simulate the evolution of a tungsten first-wall in conditions typical of fusion reactor operation. The primary target is to simulate a 10^5 atom system with a quantum-trained SNAP potential. This second challenge problem is used to define the threshold goal and hence a KPP that will quantify the team's progress.

Fission problem (stretch goal): The first target problem requires a dramatic extension of the reach of long-time, high-accuracy MD simulations to simulate the dynamics of defects in UO_2 on long timescales with quantum-accurate fidelity. The target is to simulate the evolution of fission gas clusters in 10^2 atom systems while accounting for the changing polaron distribution during the migration processes. Building a computational capability to carry out this problem at scale is a stretch goal of EXAALT. Therefore, no KPP is directly associated with this challenge problem.

Details are listed in Table 17.

3.4.2 EXAALT: KPP Stretch Goal

As discussed in the previous section, one key stretch goal of EXAALT is to develop a computational capability to carry out the fission science challenge problem efficiently at the exascale. The proposed stretch goal is to deploy a computational framework that can efficiently use the exascale machines on this problem. The team will strive to achieve a 50× speedup relative to its baseline, which is 5.7×10^{10} atom^3 time steps/s, but the measure of success will be the deployment of a high-quality computational framework. This will involve the following.

1. Ensuring that the task management infrastructure is scalable: This challenge is shared with the base KPP problem.

2. Efficiently using the hardware by having all of the critical kernels offloaded to accelerators and running efficiently: This is significantly harder for LATTE than for an empirical model because many functions contribute to the runtime in the former case. Furthermore, the relative cost of these different functions varies with the system size. This means that a large number of functions must potentially be ported to the accelerators to insure good performance over a range of system sizes.

3. Ensuring that the time-integration scheme is robust and stable enough to reach very long timescales: This is significantly harder for problems that require solving a self-consistent problem at each iteration than for conventional classical simulations. At each timestep, LATTE must equilibrate charges before computing the forces acting on each atom. This process is usually robust but can occasionally fail to converge, at which point it is not always clear how to proceed. The failure rate in serial is very low, but in a ParSplice setting, the number of replicas can be very large (e.g., 10,000), which dramatically amplifies the overall failure rate. Therefore, the team must develop extremely robust approaches that can keep the failure rates to extremely low levels.

The second stretch goal encompasses both challenge problems. It consists of developing an infrastructure to generate accurate physical models at scale. Indeed, if EXAALT achieves the aforementioned goals, simulations themselves will be able to efficiently and routinely leverage exascale resources. In this case,

Table 17: EXAALT challenge problem details.

Functional requirement	Minimum criteria
Physical phenomena and associated models	**Fusion problem:** The first problem consists of investigating the evolution of a tungsten surface under conditions relevant to exposure to a fusion plasma using atomistic simulations. The main physics of interest are the annealing mechanisms and characteristic timescales. This problem will be modeled by using a SNAP representation of tungsten. This will access the intermediate-accuracy/long-time/intermediate-size regime. **Fission problem:** The second problem pertains to the evolution of defects in nuclear fuels. The simulation will be carried out by using a tight-binding description of UO_2 that contains an individual oxygen/uranium vacancy complex. This will access the high-accuracy/long-time/small-size regime.
Numerical approach, algorithms	Parallel trajectory splicing—parallel MD
Simulation details: problem size, complexity, geometry, and so on.	Describe size/scale/properties of physical system **Fusion problem:** The reference simulation consists of a sublattice ParSplice simulation of a 10^5 atom system with a damaged tungsten surface typical of plasma-exposed conditions. The simulation will be carried out at $T = 800\,\mathrm{K}$, which is a fusion-relevant temperature, by using a SNAP representation of tungsten. The baseline FOM corresponds to a SNAP parameterization that uses 205 bispectrum components, or 205 descriptors of local atomic environments. A greater number of bispectrum components gives higher accuracy, and this number of components is consistent with the high accuracy desired for the final science challenge problem calculations. Improvements in the SNAP form developed under this ECP subproject, such as the new quadratic form or neural network extensions, might allow the team to ultimately achieve this same accuracy with fewer bispectrum components with reduced cost. Accuracy is quantified as the average error in predicted forces relative to a large database of DFT calculations. In the final benchmark calculation used for the Fusion FOM, either the baseline SNAP form or an improved SNAP form with accuracy equivalent to the one used for the baseline calculation will be used. **Fission problem:** The target simulation consists of a ParSplice simulation of a 96 atom UO_2 system that contains an individual oxygen/uranium vacancy complex at the DFTB + uranium level of theory by using the LATTE back end. The stretch goal is to develop the computational capability to carry out such a simulation at the exascale by using a combination of ParSplice, LAMMPS, and LATTE.
Demonstration calculation requirements	The team anticipates requiring only a few runs at scale to demonstrate this capability. To obtain an accurate performance benchmark on the order of 50,000 time steps on each replica ($\sim$50 ps of simulation time) are needed.
Resource requirements to run demonstration calculation	The team anticipates requiring on the order of $\sim$10 h on the full exascale machine to demonstrate both challenge problems.

parameterizing the physical model that will be used to carry out the simulations will become the main bottleneck. Obtaining these models is currently extremely time-consuming and labor-intensive because it requires generating and curating large sets of training and testing configurations, performing expensive DFT calculations on these configurations, performing the fitting procedure to obtain a model that achieves a sufficiently small discrepancy between predicted and measured values, and validating the model. Most often, this process must be repeated many times before a satisfactory result is obtained. This often involves human intervention and manual data assimilation. In light of this challenge, the second stretch goal is to develop a scalable infrastructure on which the whole aforementioned model parameterization workflow can be executed at scale, dramatically reducing the time required to obtain high-fidelity physical models that can be used in simulations. Finally, the team also aims to build an active learning framework in which the model will be improved on the fly as the simulation proceeds. Again, this would rely on the team's ability to execute a model parameterization workflow on the fly concurrently with a conventional ParSplice simulation workflow.

3.4.3 EXAALT: Figure of Merit

The FOM definition for EXAALT fusion problems is

$$\text{FOM} = \frac{N_t N_{\text{atoms}}}{t} \,, \tag{2}$$

where N_t is the number of time steps, and t is the wallclock time.

Two values of the FOM can be defined: the so-called "Raw MD" FOM and the "ParSplice" FOM. The first measures the throughput of the MD simulation that consumes most of the computing cycles in a ParSplice simulation, and the second measures the performance of an actual ParSplice simulation, which includes additional overhead not measured by the Raw MD value. This overhead is typically is on the order of 10–20 % of the raw FOM. The reference FOM value obtained on Mira was a raw MD FOM, but we will use the ParSplice FOM as the final measure of success because it is a better measure of actual application performance.

FOM Update

The latest FOM update was obtained on Summit. The benchmark was a standard ParSplice simulation of a tungsten system that contains 2000 atoms. This simulation did not use the sublattice variant of ParSplice in contrast to the final challenge problem, which will be run on a significantly larger system. However, the workload on each worker instance is actually in good correspondence with the final challenge problem, given that the domain decomposition used by sublattice ParSplice results in systems of ∼2000 atoms being allocated to the workers. Therefore, the workload is very similar to what is expected in the final challenge problem; the main difference is that there is a more complex task-management logic on the master process. Otherwise, the same kernels that act on data of the same size will consume essentially all the cycles.

The ParSplice simulations were carried out on 1000 nodes of Summit. Each worker process used one V100 GPU and one POWER9 core (6× per node). This experiment yielded an FOM of 288× over the Mira baseline when extrapolated to the whole of Summit. A corresponding "raw MD" FOM measure, this time using a heterogeneous set of worker processes (6× one V100 GPU and one POWER9 core plus 1×36 POWER 9 cores) per node, yielded a 350× speedup over the Mira baseline. The difference between the two FOM values is consistent with the expected overhead from ParSplice. This overhead does not scale with the number of worker instances but remains constant.

Current experience on pre-exascale systems is extremely encouraging. Pending major changes in hardware characteristics, these results indicate that our FOM target will likely be achieved on at least one exascale system.

3.4.4 EXAALT: Progress on Early and Pre-Exascale Hardware

Performance on Summit

Through a very valuable collaboration with NERSC/NESAP (R. Gayatri and N. Mehta), ECP/CoPA (S. Moore), NVIDIA (E. Weinberg), and HPE (S. Anderson), the SNAP kernels were highly optimized on Summit. As shown in Fig. 7, the performance of the SNAP kernels has increased 22 times from the pre-ECP

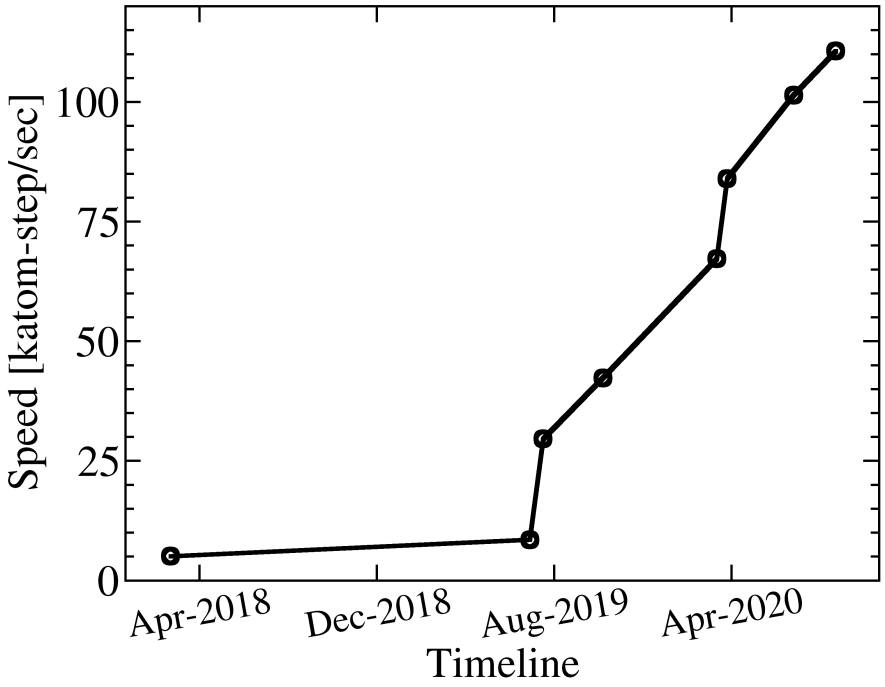


Figure 7: Evolution of the SNAP simulation rate over the last 2 years. Note the 22× increase in performance relative to the pre-ECP baseline.

baseline, of which a factor of 3.7× was obtained during FY20. The speedup was achieved through a range of optimizations, none of them dominating the overall performance increase. The most notable optimizations were:

- repeatedly changing the memory data layout of an array between kernels via transpose operations;
- refactoring loop indices and data structures to use complex numbers and multidimensional arrays instead of arrays of structs;
- refactoring some of the kernels to avoid thread atomics and use of global memory;
- judiciously using Kokkos hierarchical parallelism and GPU shared memory;
- fusing a few selected kernels, which helped eliminate intermediate data structures and reduced memory use;
- adding a new memory data layout, which enforced perfect coalescing and load balancing in one of the kernels;
- making data layouts of certain matrices symmetrical, which reduced memory overhead and the use of thread atomics on GPUs; and
- precomputing certain parameters.

The overall throughput increase also contains contributions from a refactoring of deeply nested loops, which resulted in a net decrease in the number of flops that are executed. A detailed summary of the optimization process is available in a recently submitted article [5].

The SNAP kernels are now highly optimized. Although some fine-tuning could improve the performance, the kernels are approaching the compute-bound limit, which suggests good performance on even more powerful GPUs.

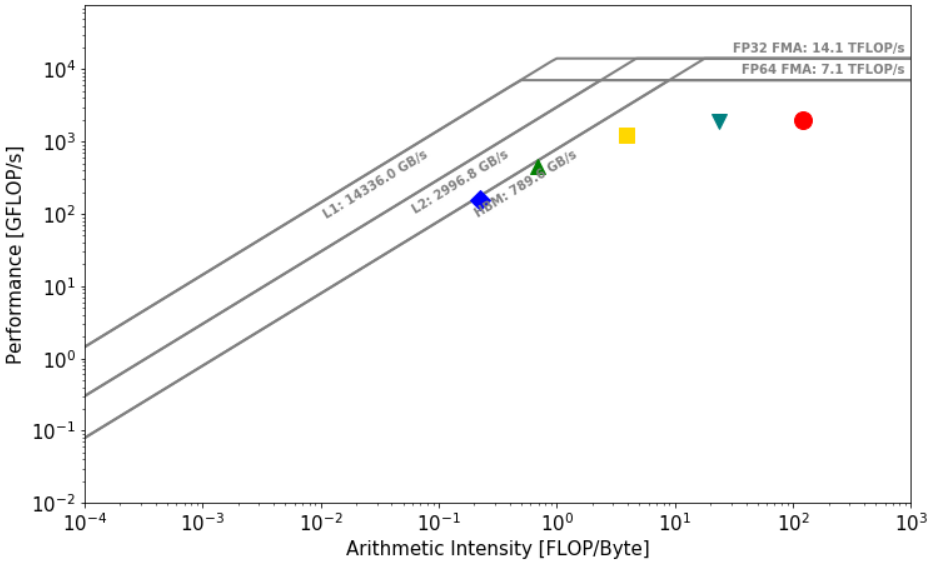


Figure 8: Roofline analysis of the production SNAP version on V100 at the DRAM level.

We also made significant progress toward our stretch goal on Summit. LATTE was refactored, leading to a 7.8× increase in performance on POWER9. The refactored version was also ported to the BML/PROGRESS libraries developed by the CoPA project. The combined stack compiles and runs correctly on Summit. However, GPU performance is low, which is likely due to the small sizes of the systems that were considered (only 96 atoms). This workload is insufficient to saturate the GPUs, leading to no performance improvement versus running on the POWER9. On larger systems, the CoPA team has observed up to 15× performance increase on the V100 for a molecular system, which is suggestive that a similar speedup could be achieved if our system was expanded. A careful scaling study of the performance on V100 will be carried out shortly.

Next Steps

We will begin optimizing the production version on MI60 and MI100 on Tulip. We will also update TestSNAP with the production version to have a good measure of the performance of the fallback OpenMP off-load solution that would be used if the Kokkos implementation on the final exascale machines is delayed. The performance of TestSNAP will also be assessed on ATS.

Regarding our stretch goal, we will characterize the performance of LATTE on the test beds and address the bottlenecks that will be identified. We will also characterize the performance of LATTE on systems of various sizes to identify the optimal conditions in which to carry out the simulations as a compromise between simulation rate and efficiency on GPUs.

Finally, the machine learning (ML) activities will continue to be a focus. The implementation of novel techniques to generate diverse training data to train the ML-based models is ongoing. We are also optimizing the architecture of the neural networks used as part of the extension of SNAP. Finally, in collaboration with the NWChem team, we will integrate EXAALT and NWChem so that the generation of training data can be executed at scale, which will considerably decrease the time needed to parameterize physical models for new materials. This capability will enable us to take the first steps toward a powerful integrated simulation/ML capability.

3.5 ExaAM

ExaAM, the exascale additive manufacturing (AM) project, is developing the Integrated Platform for Additive Manufacturing Simulation (IPAMS), which is a collection of capabilities to simulate metal AM processes at the fidelity of the microstructure. By directly incorporating microstructure evolution and the effects

of microstructure within AM process simulation, ExaAM is enabling the design of AM components with location-specific properties and the acceleration of performance certification.

AM is revolutionizing manufacturing, allowing for the construction of complex parts not readily fabricated by traditional techniques. Additionally, AM offers the possibility of constructing "designer materials" by adjusting process control variables to achieve spatially varying physical properties. additive manufacturing is a unique application area due to its strategic importance to both US industry and federal agencies, such as DOE, including NNSA, DOD, NASA. Although there has been significant interest and investment in AM, the fraction of this investment devoted to modeling and simulation is relatively small and is focused on developing reduced-order models for industry use rather than high-fidelity predictive models. ExaAM represents a unique opportunity to leverage DOE investments to address challenges that require exascale resources and, to the team's knowledge, that are not being addressed by other AM modeling and simulation efforts.

In AM, a geometric description of the part is processed into 2D slices. A feedstock material is melted, and the part is built layer by layer. In metal AM, the feedstock is often in wire or powder form, and the energy source is a laser or electron beam. ExaAM is focusing on powder bed processes in which each layer is approximately 50 µm. Hence, a part that is 1 cm tall would require 200 layers, each consisting of spreading new feedstock powder and one or more passes of the laser or electron beam to sinter and/or melt the powder in appropriate locations.

A complex interplay between multiple physical phenomena at spatial and temporal scales that spans orders of magnitude determines the performance of the final manufactured part. These include heat transfer (e.g., conductive, convective, radiative, and evaporative), fluid flow, melting and solidification, and solid-solid phase transformation. These phenomena are directly influenced by controllable process parameters, such as the pattern by which each layer is melted, diameter, magnitude, speed of the energy source, and so on. In turn, these influence the microstructure throughout the part, which determines local properties, residual stress, and—ultimately—performance (e.g., strength, modal properties, service life). This sequence is often referred to as the *process-structure-property-performance (PSPP)* relationship. There are significant gaps in understanding the fully integrated sequence for AM processes; filling these gaps where possible and quantifying uncertainties where it is not is crucial to unlocking the potential of AM.

The physical processes involved in AM are similar to those of welding—a field with a wealth of experimental, modeling, simulation, and characterization research over the last decades. Unfortunately, although calibrated and approaching predictive capability, the simulation tools developed for welding and other similar processes are inadequate for AM processes as demonstrated by the inability to predict the failure rate for new AM parts, which can be as high as 80%. The team believes that this is largely because the PSPP relationship is traditionally analyzed in an uncoupled manner, relying on tabular databases unable to adequately capture the implicit, dynamic, nonequilibrium nature of AM processes.

One ExaAM goal is to remove those limitations by coupling high-fidelity mesoscale simulations within continuum process simulations to determine microstructure and properties by using local conditions. Typically, thermomechanical finite element models are employed at the macroscopic part scale; finite volume or finite element models for fluid dynamics and heat transfer to capture the melt pool dynamics and solidification at millimeter scales; mesoscale approaches (e.g., discrete elements, cellular automata, kinetic MC, phase field models) to simulate melting, solidification, and microstructure formation at the micron scale; and polycrystal plasticity models to develop the microscale mechanical property relationships.

Although no single code can capture all the relevant physics required, several codes have been developed to simulate phenomena similar to those required for AM within the DOE complex and the broader computational science community. ExaAM is leveraging several of those existing capabilities, enhancing and extending them as needed and developing new capabilities when necessary.

A full ExaAM simulation comprises five stages, as shown in Table 18.

For the purposes of estimating computational resources, the focus is on stages 1–3 since the cost of stages 0 and 4 is relatively small and can be performed on capacity HPC computational platforms. Additional capabilities that are required as risk mitigation or to provide parameters for models above (e.g., subgrain scale solidification microstructure evolution via phase field and OpenFOAM for melt pool simulation) are also omitted from the remaining discussion.

Table 18: Computational simulation stages in an ExaAM simulation.

Stage	Exascale simulation	Required computational capability	ExaAM components
0	Approximate full-part build simulation	Macroscale thermomechanics	Diablo
1	Prediction of "as-built" microstructure	Coupled thermomechanics, fluid flow, and microstructure evolution	Diablo + TruchasPBF + ExaCA
2	Prediction of "late-time" microstructure	Solid-solid phase transformations and other mesoscale phenomena during cooling	MEUMAPPS-SS
3	Prediction of micromechanical properties	Response of the predicted microstructure to representative forces at a sufficient number of locations and synthesis of macroscale constitutive models from microscale properties	ExaConstit
4	Full-part build simulation	Macroscale thermomechanics using locally accurate constitutive properties	Diablo

3.5.1 ExaAM: Science Challenge Problem Description

ExaAM is developing a collection of simulation capabilities for performing process-aware performance modeling of additively manufactured parts by using locally accurate properties predicted from microstructures that develop based on local processing conditions. This capability will be demonstrated by simulating the Inconel 625 (IN625) build of the complex bridge structure developed for the 2018 National Institute of Standards and Technology (NIST) AM-Bench Conference known as AMB2018-01 (Fig. 9). A full description can be found on the NIST website.[2]

The threshold (i.e., base) simulation will be performed at the location at which measurements were performed, 2.5 mm above the base plate, for one of the thick legs. Since the alloy selected is IN625, which does not exhibit significant microstructure changes due to solid-solid phase transformation and precipitate formation during the build process, Stage 2 can be neglected from the threshold problem. Stage 2 would be required for other materials, such as IN718 or Haynes282, so stage 2 appears as a stretch science goal.

Tables 19 and 20 describe Stages 1 and 3 in detail for the threshold challenge problem. The following caveats apply to the challenge problem.

- This estimate is for the threshold challenge problem only. The actual goal would be to predict microstructure and properties throughout AMB2018-01, which would require significantly more computational resources.
- This estimate includes a significant amount of flexibility (e.g., number of layers in Stage 1, number of representative volume elements (RVEs) in Stage 3), allowing adjustment based on accuracy needs, better-than-expected performance, or lower-than- expected performance.
- Memory is not anticipated to be a limiting factor.

3.5.2 ExaAM: KPP Stretch Goal

The planned stretch science goals for ExaAM are defined in Table 21.

[2] https://www.nist.gov/ambench/amb2018-01-description

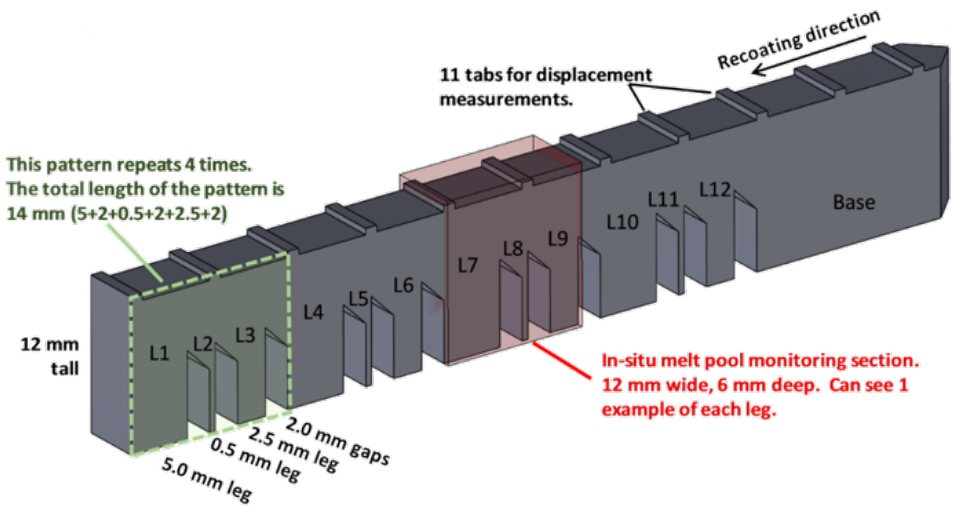


Figure 9: NIST AM-Bench AMB2018-01 bridge structure used for the ExaAM challenge problem. L1, L2, and L3 are thick, thin, and medium legs, respectively. A section is a set of three legs along with the upper bridge structure.

Table 19: ExaAM challenge problem details for Stage 1, "As-Built Microstructure Prediction."

Functional requirement	**Minimum criteria**
Physical phenomena and associated models	Thermomechanics, fluid flow, heat transfer with phase change (melting and solidification), microstructure evolution.
Numerical approach, algorithms	Time-dependent Lagrangian FEM with nonlinear material models, time-dependent Eulerian FVM, cellular automata.
Simulation details: problem size, complexity, geometry, and so on	For the purposes of this estimate, neglect the computational cost of the far-field thermomechanical component To obtain thermal history profile, each layer requires 250 domains of $1.0 \times 0.3 \times 0.2$ mm with 5 µm zones for 12.5 million time steps (0.5 µs time step size). Microstructure evolution occurs on the same set of domains but at a 1 µm resolution and a similar time step requirement.
Threshold simulation requirements	A representative volume of AM microstructure requires coupled thermomechanical/fluid flow and microstructure development simulation of five layers.
Resource requirements to run threshold calculation	Based on scoping simulations on Summit, the team estimates requiring approximately 50 Summit nodes for 4 h. Execution on Frontier should require $<$ 1 h for a similar number of nodes.

Table 20: ExaAM challenge problem details for Stage 3, "Micromechanical Property Prediction."

Functional requirement	Minimum criteria
Physical phenomena and associated models	Elastic-plastic response using polycrystal plasticity.
Numerical approach, algorithms	Time-dependent Lagrangian FEM with nonlinear crystal plasticity material models using grain conforming mesh.
Simulation details: problem size, complexity, geometry, and so on	Each RVE is a $100 \times 100 \times 100\,\mu m$ domain that contains approximately 1000 grains at $1\,\mu m$ resolution (1 million zones).
Threshold simulation requirements	For each RVE, up to 1% strain along 20 loading conditions at 10 temperatures, or 200 simulations for each RVE.
Resource requirements to run demonstration calculation	Approximately 10 locations (i.e., RVEs) will be required, resulting in 2000 independent ExaConstit simulations. Based on scoping simulations on Summit, each simulation will require approximately 1 h on two Summit nodes (4000 nodes for 1 h for all 2000 ExaConstit simulations). Execution on Frontier should require 10–15 min on a similar number of nodes.

Table 21: ExaAM science stretch goals.

Description	Motivation	Components
Predict microstructure and local properties throughout AMB2018-01	Process optimization requires knowledge of local microstructure and properties throughout.	TruchasPBF, ExaCA, ExaConstit, Diablo
Inform development of reduced-order models for AMprocess simulation	Topology and shape optimization require faster-running models for process simulation.	TBD
Detailed solidification simulation (IN625, IN718, and/or Haynes282)	Subgrain microstructure and nucleation model to inform grain-scale microstructure (ExaCA).	AMPE and/or Tusas
In situ annealing (IN718 and/or Haynes282)	Capture microstructure changes in materials the exhibit significant solid-solid phase transformations and precipitation during a build (Stage 2 in the workflow).	MEUMAPPS-SS
Heat treatment (IN625)	Capture microstructure changes during post-build heat treatment (hours) of AMB2018-01	MEUMAPPS-SS.
Determine local crystal model from annealed state (IN625, IN718)	Capture changes in the local crystal model from solid-sold phase transformations and precipitation during cooldown and annealing.	ExaConstit
Powder-resolved process simulation	Capture details of melt pool behavior to optimize process parameters to minimize porosity due to keyholing and so on.	ExaMP

Table 22: ExaAM mileposts.

FY19:	Demonstrate the ability to perform thermomechanics + melt pool and melt pool + microstructure simulations of overlapping melt pools for an arbitrary scan strategy.
FY20:	Demonstrate the ability to predict microstructure and properties for at least one location of AMB2018-01.
FY21:	Demonstrate the ability to predict microstructure and properties at one location of thick leg of AMB2018-01 and perform an initial part-scale build by using self-consistent properties.
FY22:	Demonstrate the ability to predict microstructure and properties at multiple locations of AMB2018-01 and perform a part-scale build by using self-consistent properties. Determine the number of locations required for challenge problem.
FY23:	Perform challenge problem simulations.

3.5.3 ExaAM: Capability Plan

A full ExaAM simulation comprises the five stages listed in Table 18. Completing the challenge problem simulation requires developing the capabilities represented by these five stages and linking them in a sequential workflow. The mileposts below represent a sequence of steps that demonstrate progress toward that ultimate goal.

3.5.4 ExaAM: Progress on Early and Pre-Exascale Hardware

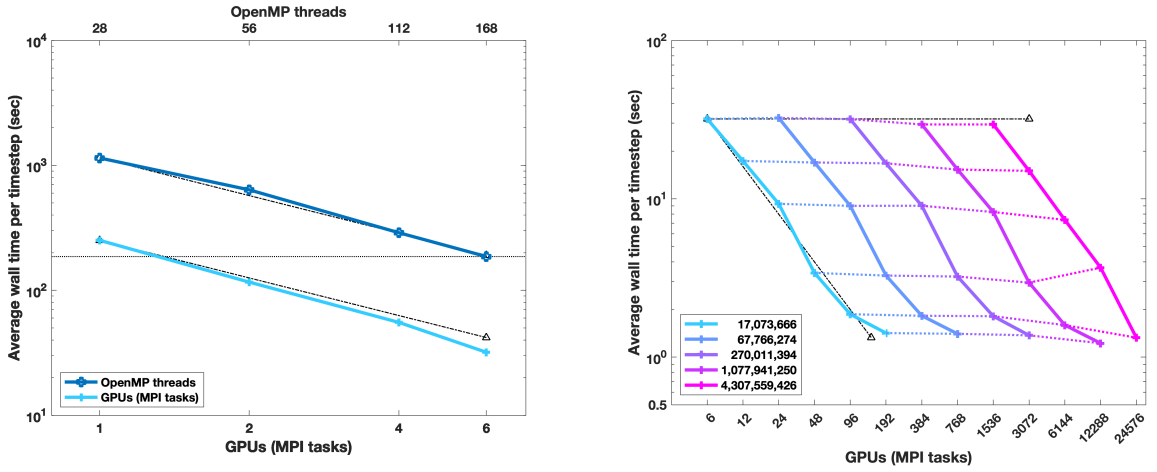
Performance on Summit

Although we anticipate the need to implement new physics as we continue to analyze validation experiments, the basic physics required to simulate metal AM was implemented in each component code, and our FY20 capability milestone focused on the coupling or links between the components in our computational workflow. Specifically, we determined the required information to be transferred and enabled each component to send and receive this information on the ORNL Summit computer system.

The ExaAM FY20 demonstration milestone is intimately tied to our FY20 milepost, "Predict microstructure and properties for at least one location of AMB2018-01." As described, the ExaAM computational workflow and exascale utilization model are driven by the coupling between AM processing, material microstructure, material properties, and part performance (PSPP) with the goal of creating AM process-aware material models. An initial AM process-aware material model and a built simulation that uses it is our FY21 milepost. Each milepost demonstrates progression toward completing this workflow. In FY20, we demonstrated "Process to Property" for at least one location in the AMB2018-01 build. Specifically, we:

- enabled the coupling of key elements of the ExaAM workflow from melt-pool scale process simulation through the simulation of microstructure development to the calculation of location specific material properties;
- iIdentified location (midway in thin leg) and scale (20 layers) for statistically significant simulations;
- simulated the melt and resolidification of 20 successive layers at one location of AMB2018-01 using the thermal code TruchasPBF;
- transferred thermal histories to the microstructure development code ExaCA to compute the grain-scale microstructure everywhere within the 20 layers; and
- carved out an RVE containing hundreds of crystalline grains and computed the location-specific material properties using the code ExaConstit.

All these simulation were performed on Summit, and we found the detailed microstructure and property components to be the most computationally intensive. A detailed analysis of the performance of these components on Summit is presented as follows.



(a) Strong and weak single-node, OpenMP and GPU scaling. **(b)** Strong and weak multinode, multi-GPU scaling.

Figure 10: Tusas performance on Summit.

Tusas (Stage 1)

Tusas is a general, flexible code for solving coupled systems of nonlinear partial differential equations. Tusas was originally developed for phasefield simulation of solidification. The Tusas approach comprises an unstructured Lagrange finite element spatial discretization of the fully coupled nonlinear system, which is treated explicitly or implicitly (Euler, Trapezoid, BDF2, or IMR) in time with a preconditioned Jacobian-Free, Newton-Krylov (JFNK) method. The preconditioning strategy in Tusas is based on block factorization and an algebraic multigrid that allows an efficient, implicit time integration. The JFNK method only requires a residual from an implementation standpoint, so Tusas allows a flexible framework because it only requires users to implement code for a residual equation with the configuration of the nonlinear system and preconditioner performed at runtime.

In addition to performing well in parallel across multiple CPUs via MPI and OpenMP, one of Tusas' strengths [6, 7] is the ability to run efficiently across multiple GPUs via MPI and CUDA. Because the residual fill is composed entirely of local operations, it is performed entirely on the GPU by using a distance-1 element graph coloring within each MPI subdomain. Element graph coloring avoids shared memory race conditions and atomic operations on CPU threads and GPUs. Additionally, inner products within the GMRES solver are implemented on threads and each GPU via OpenMP and Cuda within the `Belos` and `Kokkos` packages in `Trilinos` [8].

To demonstrate strong and weak parallel multinode and multi-GPU scaling, we considered simulations for a 3D dilute binary alloy solidification with discretizations consisting of 17,073,666, 67,766,274, 270,011,394, 1,077,941,250, and 4,307,559,426 unknowns. The discretizations consist of meshes with $512 \times 128 \times 128$, $512 \times 256 \times 256$, $512 \times 512 \times 512$, $512 \times 1024 \times 1024$, and $512 \times 2048 \times 2048$ bilinear hexahedral elements.

Figure 10a shows the CPU time required for time integration to a fixed time on a single node of Summit as a function of the number of GPUs and number of OpenMP threads with 17,073,666 unknowns. Ideal strong scaling is demonstrated for OpenMP threads with up to 168 threads, and for up to 6 GPUs. A factor of approximately 6× is achieved by using 6 GPUs over 168 threads. Figure 10b shows the CPU time required for time integration to a fixed time as a function of the number of GPUs across multiple nodes, using six GPUs per node. Specifically, Fig. 10b shows strong and weak scaling on `Summit` on up to 24,576 GPUs (4096 nodes) with up to 4,307,559,426 unknowns. Strong scaling is depicted by solid lines on Fig. 10b where each color depicts a fixed problem size. Weak scaling is depicted by horizontal markers with dashed lines. We acknowledge that further optimizations can be performed on our GPU implementation. These preliminary scaling studies demonstrate that Tusas effectively scales strongly and weakly on thousands of GPUs with billions of unknowns and is particularly suited for emerging heterogeneous architectures.

MEUMAPPS-SS (Stage 2)

The original version of MEUMAPPS-SS is written in Fortran 90 and uses a parallel fast Fourier transform (FFT) package, P3DFFT [9], which allows MPI decomposition of the computational domain in two directions (a "pencil decomposition"), which then enables the use of scalable 3D FFTs in which the 1D FFTs are performed with the FFTW library [10]. The scaling of the Fortran 90 version of MEUMAPPS-SS on Summit with and without GPU off-load using OpenACC is shown in Fig. 11. The replacement of OpenACC with OpenMP off-loading and of P3DFFT with heFFTe, a GPU-enabled scalable 3D FFT library [11], is under active investigation.

Development of a more modern and modular version of MEUMAPPS-SS written in C++ is also underway. This version uses Kokkos [12] for performance-portable node-level parallelization. Current development is focused on using the CUDA back end for NVIDIA GPUs, but the code is ready to leverage the HIP back end for AMD GPUs and back ends for other architectures as they become available. Calculations with the C++ version of MEUMAPPS-SS can perform GPU-enabled 3D FFTs by using the AccFFT [13] or heFFTe libraries. Both libraries use pencil MPI domain decompositions and use the FFTW library for 1D FFTs on CPUs and the cuFFT library [14] for 1D FFTs on GPUs.

In typical MEUMAPPS-SS simulations, ∼80 % of the runtime is spent solving linear elasticity equations for mechanical equilibrium at each time step. Therefore, the test problem for GPU speedups on Summit was chosen to be a mechanical equilibrium calculation. For calculations with a 400^3 grid on one Summit node (42 IBM Power9 CPU cores and six NVIDIA Volta V100 GPUs), the Fortran/OpenACC version of MEUMAPPS-SS exhibited a 7× nodal GPU speedup (for which the CPU-only baseline used all 42 CPU cores) and a 10× per-MPI-task GPU speedup (for which the CPU-only baseline used six CPU cores, one for each GPU). The C++/Kokkos version exhibits a 3× nodal GPU speedup and a 16× per-MPI-task GPU speedup. Despite the lower nodal GPU speedup, the runtime for the fastest single-node C++/Kokkos calculation is nearly half that of the fastest single-node Fortran/OpenACC calculation, likely due to a combination of improved memory management and improved FFT library performance. For both codes, most of the runtime was spent in FFT library calls, which highlights the importance of high-performance FFT libraries to this portion of the project.

ExaConstit (Stage 3)

ExaAM uses three open-source code bases for these finite element methods (FEM) simulations, two of which (ExaCMech and ExaConstit) are supported by ExaAM. The crystal plasticity constitutive response is handled by the library ExaCMech [15]. It is coupled with ExaConstit [16], a general quasistatic nonlinear solid mechanics velocity-based finite element application built on the MFEM framework [17]. ExaCMech uses the RAJA library [18] to abstract away the CUDA and HIP-type kernel launches. RAJA is used to wrap the entire constitutive model into one large compute kernel that is launched on the device. Within ExaConstit, different GPU strategies are being examined. The dominant computational cost within ExaConstit is the series of linearized system solves within a Newton-Raphson scheme to obtain the velocity field. In the currently employed approach, the linearized system matrix is never fully assembled on the GPU. Instead, matrix-free methods are used that perform the action of the stiffness matrix [19–21]. One disadvantage of using matrix-free methods is the loss of traditional preconditioners that are used within most FEM implementations. Therefore, ExaConstit resorts to a matrix-free Jacobi preconditioner. Depending on the assembly method—partial assembly (PA) on the GPU [19, 21], element assembly (EA) on the GPU [20], or full assembly (FA) on the CPU—ExaConstit exhibits different strong scalability on Summit. This is shown in Fig. 12 for meshes with 1 and 8 million linear hexahedron elements in simulations in which the polycrystal undergoes monotonic loading out to 1 % strain. The PA method exhibits perfect scaling, FA exhibits acceptable scaling, and the EA exhibits poor scaling. Any GPU-only simulation used six GPUs and six CPUs per node, and the CPU-only simulations used all CPUs on a given node. Although the EA method exhibits poor scaling, it is the most performant and allows us to run the hundreds of simulations necessary for the challenge problem concurrently. Further speedups and reduced start-up costs could be obtained by using more efficient linear and nonlinear solvers.

Next Steps

As discussed, the basic physics and inter-component coupling in our computational workflow was established. For FY21, we will turn our capability milestone to optimizing GPU utilization for exascale readiness. Although most of our components are written with GPU utilization in mind using the Kokkos and RAJA potability

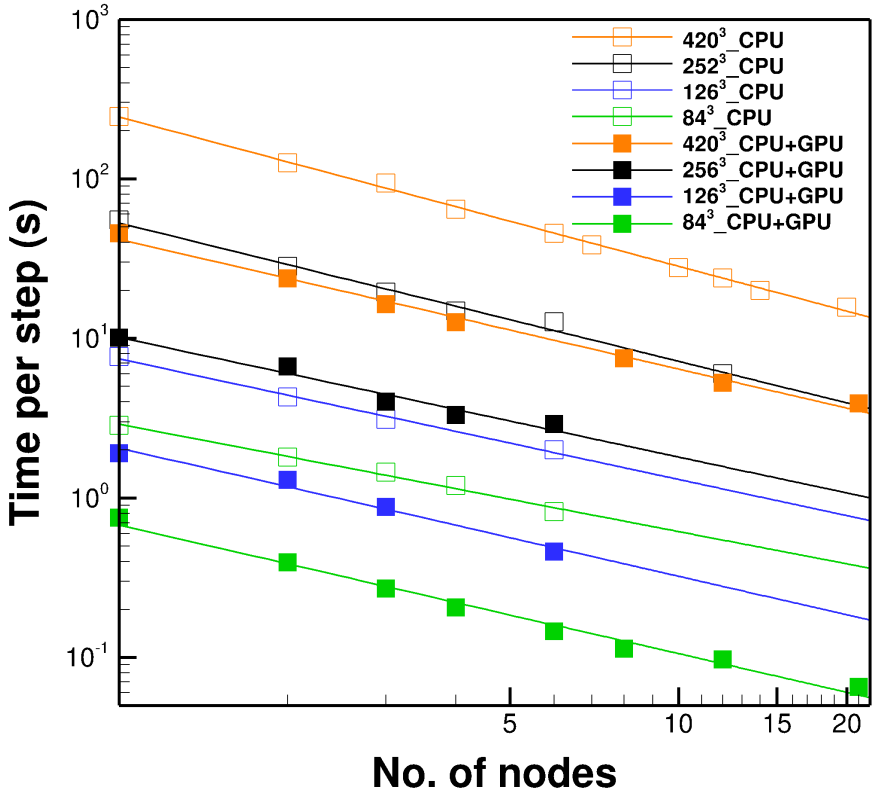


Figure 11: MEUMAPPS-SS scaling on Summit.

layers, we anticipate significant optimization from analyzing data layout and data movement.

Our FY21 demonstration milestone is intimately tied to our FY21 milepost, "Perform initial part-scale build using self-consistent properties (process-aware material model)." Specifically, we will:

- identify multiple thermomechanically distinct locations in the AMB2018-01 build,
- execute our process-to-property workflow for these locations;
- develop and execute workflows for the ensemble of orientations for each microstructure.
- develop an AM-specific flow stress model for use in build simulation, and
- perform full part build simulation (Diablo) by using an AM-specific material model.

We initiated a collaboration with the ECP Workflow Project to use this use-case to build an automated workflow manager for the ExaAM project.

3.6 QMCPACK

The ability to computationally design, optimize, or understand the properties of energy-relevant materials is fundamentally contingent on the existence of methods to accurately, efficiently, and reliably simulate them. Quantum-mechanics-based approaches must necessarily serve as a foundational role since only these approaches can describe matter in a truly first-principles (i.e., parameter-free) and thus robust manner. Materials design has progressed from the study of simple bulk properties to targeting collective effects in strongly correlated materials, such as magnetic ordering, phase transitions, and quantum coherence. This requires a fundamentally different set of computational tools than have been used in the past. quantum Monte Carlo methods are ideal candidates for this because they robustly deliver highly accurate calculations of complex materials that do not artificially bias solutions of a given character. Significantly, with increased

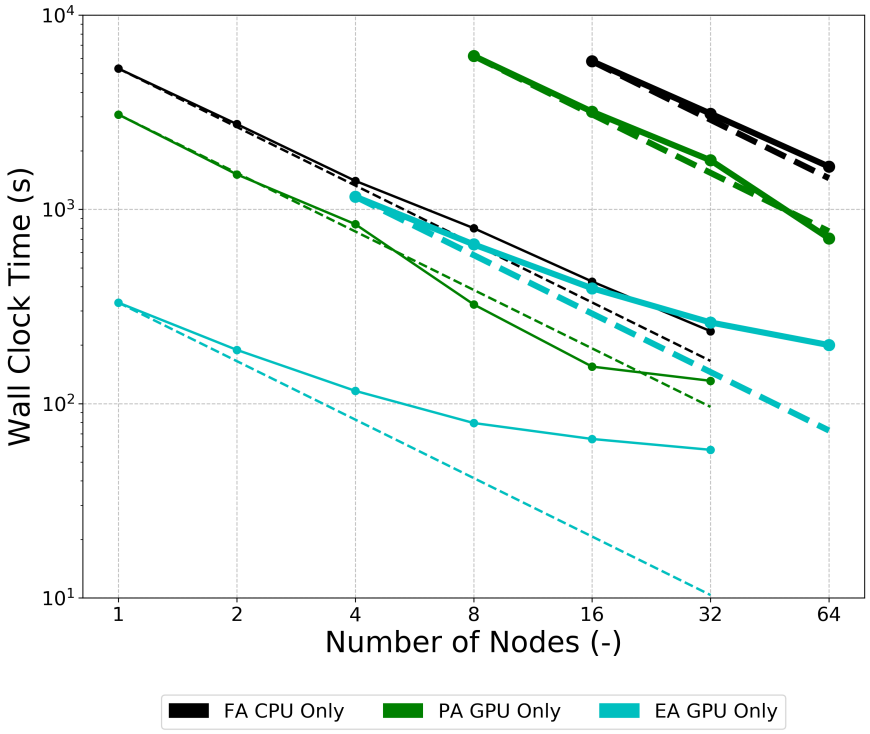


Figure 12: Strong multinode, multi-GPU scaling on Summit for various assembly methods. Thinner lines represent the 1 million element mesh, and the thicker lines represent the 8 million element mesh. Dashed lines represent perfect scaling for each assembly method and mesh size.

computer power, the few approximations in these methods can be tested and systematically reduced, which is not possible with other first-principles methods. Saritas, Ming, Du, and Reboredo [22] provides a recent study of defects in phosphors.

The trade-off is that the computational demands of the QMC method are large. For example, petascale computers have allowed for calculations of the magnetic exchange in a copper oxide important for understanding the mechanism of high-temperature superconductivity to be performed. However, these calculations involved a highly symmetric supercell containing only 56 atoms when a realistic model that considered the defects and dopants of actual superconductors would require at least several hundreds of atoms. The 10 year challenge problem is to simulate transition metal oxide systems of approximately 1000 atoms to 10 meV statistical accuracy, such as complex oxide heterostructures that host novel quantum phases, by using the full concurrency of exascale systems. The additional power and parallelism of exascale QMC will provide the essential predictive and quantitative capability for these and related materials that lie well beyond the capabilities of existing methods. Exascale provides the opportunity for highly impactful and enabling benchmark accuracy calculations on these materials, providing the reference calibration data that is missing from essentially all current quantum mechanics-based materials calculations. This capability will be highly useful across materials science, nanoscience, and physics communities, particularly where experimental data are costly or difficult to obtain.

3.6.1 QMCPACK: Science Challenge Problem Description

The challenge problem is to calculate the cohesive energy of a large supercell of nickel oxide (NiO) by using QMCPACK and diffusion QMC to an accuracy of 0.010 eV per NiO formula unit at capability scale in a reasonable and scientifically productive amount of wallclock time (e.g., <1 d). The team anticipates a minimum 256 atom supercell up to a 1024 atom supercell, as specified in the team's original proposal, but the current FOM is more flexibly defined and includes the formal power law scaling of the method with system size. Details on the challenge problem are provided in Table 23.

NiO was selected as emblematic of the science challenges involving the complex physics of transition metal oxides. This classic Mott insulator (more accurately a charge transfer insulator) defies nonempirical predictions by other methods. NiO is also part of the class of materials that is being studied by a DOE BES-funded Computational Materials Sciences Center. Success for the NiO problem will indicate that a high and productive rate of computational work could be achieved for other challenging materials, including those with strong electronic correlations, novel magnetic states, and a host of novel quantum phases.

Although reaching the FOM indicates an ability to measure the total energy to a good accuracy with reasonable time to solution, a highly productive science tool requires significantly more functionality, including a wider range of wave functions, a large range of observables (e.g., electron density, forces, density matrices), a sophisticated trial wave function optimization scheme, support for multiple QMC methods, and viable sources of input trial wave functions. The design templates established by reaching the FOM should be transferable since these additional capabilities add computational cost and are thus thought to increase the ease of mapping to different architectures. One key stretch goal is to ensure that low-symmetry materials can be studied. Although these do not change the electron count and thus formal computational cost, memory requirements are greatly increased. Therefore, this requires support for localized orbitals to reduce the memory usage and/or successful development of latency-hiding techniques to enable the use of slower or remote memory.

3.6.2 QMCPACK: KPP Stretch Goal

One stretch goal is to complete the QMC workflow for a QMC calculation of a general low-symmetry non-bulk system (e.g., defect or interface) by using localized orbitals.

Exact achievable size/complexity will depend on the memory of A21 and Frontier, and memory reductions are achievable by using the localized RMG orbitals compared with delocalized schemes (7× for 512 atom NiO, 6.5 Bohr localization).

RMG localized orbital implementation and support in QMCPACK must be sufficiently capable and portable.

Table 23: QMCPACK challenge problem details.

Functional requirement	Minimum criteria
Physical phenomena and associated models	Predict the cohesive energy of a large supercell of NiO by using QMCPACK and diffusion QMC to an accuracy of 0.010 eV per NiO formula unit. The FOM formula allows for the calculation of an arbitrary supercell size, but a 1024 atom supercell calculation on exascale systems is anticipated. To ensure that the chosen target problems are realistic and representative, trial wave functions and pseudopotentials are specified that are the same as in the team's recent publications [23] (i.e., they passed scientific peer review and obtained sufficiently accurate results).
Numerical approach, algorithms	QMC, basis-set, many-body WF approach.
Simulation details: problem size, complexity, geometry, and so on	The problem is defined by a NiO primitive bulk cell multiplied to the chosen supercell size. This sets the number of valence electrons in the computational problem, which scales in cost with the cube of the number of electrons.
Demonstration calculation requirements	Calculations should execute to completion, and the resultant statistical analysis of the equilibrated QMC data should yield of 0.01 eV/formula unit error bar.
Resource requirements to run demonstration calculation	Depending on the simulated problem size, a complete FOM run is expected to use the full exascale machine for 1–4 h. The FOM can also be accurately estimated by measuring throughput at the full scale of the machine.

3.6.3 QMCPACK: Figure of Merit

The FOM for QMCPACK is defined as:

$$\text{FOM} = \frac{N_{\text{samples}} N_{\text{elec}}}{\text{wall time}} \,. \tag{3}$$

The baseline FOM is 1.004×10^{14} on 18,000 Titan nodes. QMCPACK v3.5.0 obtains a FOM of 1.91×10^{14} on 18,000 Titan nodes through the use of the "Fahy" determinant update variant.

FOM Update

The end of FY20 FOM for QMCPACK based on a 512 atom NiO supercell is now 37.1, based on 95 % scaling from a single node run on Summit. This FOM is obtained with the open-source LLVM/clang development compiler, which can now be used with QMCPACK as a result of improvements during the year. Performance essentially matches that of the IBM XL compiler (37.4).

3.6.4 QMCPACK: Progress on Early and Pre-Exascale Hardware

During FY20, the project emphasis has been on improving the initial proof of concept performance portable implementation developed in FY19, as well as on trialing and optimizing this new version on Summit and early access hardware and software. This new implementation involves a significant update to the architecture of the application to increase the available numerical work for accelerator execution by enabling batching of operations over groups of MC walkers. The "legacy" CUDA implementation employed this concept for efficiency and is used as a reference. Therefore, the measured performance tests the performance of any new hardware/software and the updated application architecture.

The initial performance-portable implementation of the real space QMC algorithms uses OpenMP target off-load. Although a few OpenMP 5.x features are desired, the implementation primarily depends on an efficient and performance implementation of standard OpenMP 4.5 off-load capabilities. The primary difficulty faced by the project has been the immature nature of OpenMP implementations, whether in open-source or vendor proprietary compilers and runtimes. The QMCPACK project worked with the ECP SOLLVE project to improve open-source LLVM/Clang toward production quality (19 closed and six open bug reports) and contributed two patches directly to LLVM to increase performance. Over 20 bug reports were each submitted to AMD and Intel for significant problems, and additional feedback was provided to Cray for their compiler. All compilers have significantly improved, but important issues remain. Many of these issues directly impact achievable performance.

Performance on Summit

Performance on Summit is measured against the legacy CUDA implementation for a range of problem sizes. Figure 13 shows the current performance. The performance is twice that of the legacy CUDA implementation for very large problems; however, for even a 3072 electron problem (256 atom NiO), performance is significantly before the legacy CUDA application. Correctness is also checked via the application test suite, which was expanded throughout the year, particularly in the areas of unit tests and in deterministic integration tests. When the latest LLVM development compiler (approximately the LLVM 11) is used, the full test suite now passes on Summit. As a result, QMCPACK on Summit no longer depends on the IBM XL compiler, and the application can adopt C++17, which is well supported by LLVM.

Performance characteristics were studied and profiled. For the largest problem sizes, the performance-portable implementation is faster than legacy CUDA implementation primarily due to more efficient matrix updates and linear algebra. These are the cubic scaling parts of QMC and dominate for large problems sizes. For smaller problems, performance is poor primarily due to the lack of full OpenMP "target nowait" support for true asynchronous execution with the current LLVM compiler and OpenMP runtime. As a result, the CPU host thread can not progress and either queues additional GPU work or runs GPU-inefficient tasks. A partial implementation is currently underway via the ECP SOLLVE project. The legacy CUDA implementation launches these kernels asynchronously and thus is more efficient. Additionally, many particle-oriented operations—primarily distance table computation—are still run on the CPUs. These operations will be moved to the GPUs, and some data transfers will consequently be removed. These two changes are

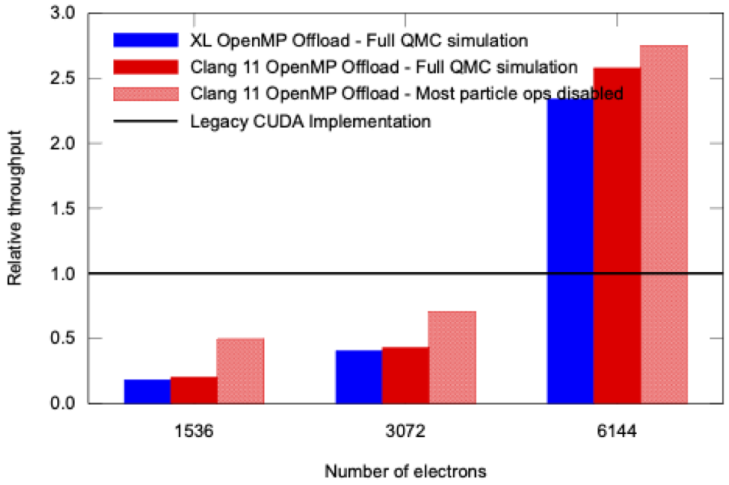


Figure 13: Throughput of real-space diffusion QMC performance-portable implementation relative to legacy CUDA implementation for various problem sizes. Overall, 6144 electrons correspond to 512 NiO atoms. For this large problem size, the new implementation has more than twice the throughput compared with the legacy CUDA. However, for all smaller problems, the performance is significantly slower.

expected to greatly improve performance for smaller problem sizes. Large problem performance will also increase, although to a lesser extent, yielding an improved FOM.

Besides progress within the ECP, achieving these performance increases is critically important for science production in 2021 on Summit by Innovative and Novel Computational Impact on Theory and Experiment (INCITE)-supported users of the QMCPACK code.

Next Steps

Based on the results obtained thus far, the new architecture for performance portability appears capable of delivering a highly successful FOM on Frontier or Aurora. The changes to the code base have been extensive due to the updates in memory handling and execution needed to obtain high and portable performance. Expanded testing is aiding the refactoring and redesign and allowing the legacy parts of the application to be rewritten, where necessary. As planned in the FY21 milestones, the project will continue to assess and optimize performance on pre-exascale hardware and move more functionality to the new performance-portable implementation in a phased manner. These developments will also enable production science on Summit by using exactly the same code base that will be used on the exascale machines. Progress depends particularly on advances in the maturity of the OpenMP target off-load implementations in the open-source and vendor compilers. The project will continue to work with the ECP SOLLVE project and vendors to prioritize critical feature implementations and bug fixes.

4. ENERGY APPLICATIONS

> **End State:** Deliver a broad array of science-based computational applications able to provide—through the effective exploitation of exascale HPC technologies—breakthrough modeling and simulation solutions that yield high-confidence insights into a set of critical problems and challenges that positively impact the nation's energy security.

The energy applications (EA) L3 area (Table 24) focuses on modeling and simulating existing and future technologies for the efficient and responsible production of energy to meet the growing needs of the United States. The applications in this WBS L3 generally require the detailed modeling of complex facilities and multiple coupled physical processes. Their goal is to help overcome obstacles to efficiently and safely deliver

Table 24: Summary of supported EA L4 projects.

WBS number	Short name	Project short description	KPP-X
2.2.2.01	ExaWind	Predictive wind plant flow modeling	KPP-2
2.2.2.02	Combustion-Pele	Combustion engine and gas turbine design	KPP-2
2.2.2.03	ExaSMR	Coupled MC neutronics and fluid flow simulation of small modular reactors	KPP-1
2.2.2.04	MFIX-Exa	Multiphase flow reactor design	KPP-2
2.2.2.05	WDMApp	High-fidelity whole device modeling of magnetically confined plasmas	KPP-1
2.2.2.06	WarpX	Plasma wakefield accelerator design	KPP-1

energy.

These applications are highly complex and involve modeling intricate geometric details and including a broad range of physical phenomena. One key additional requirement for EA is the broader community adoption of the computational models and methods developed in the project or, in some cases, the virtual datasets or physical insights that result from simulations carried out in the ECP. Additionally, applications are expected to influence—directly or indirectly—design choices for exascale hardware and software.

4.1 ExaWind

The scientific goal of the ExaWind project is to advance fundamental understanding of the flow physics governing whole wind plant performance, including wake formation, complex-terrain impacts, and turbine-turbine interaction effects. Greater use of the nation's abundant wind resources for electric power generation, reaching 30 % of US electrical supply, will have profound societal and economic impacts, strengthening US energy security through greater diversity in its energy supply, providing cost-competitive electricity to key regions across the country, reducing greenhouse-gas emissions, and reducing water used in thermoelectric power generation.

One key challenge of the wide-scale deployment of wind energy in the utility grid without subsidies is predicting and minimizing plant-level energy losses, which are currently estimated to be 20 % in relatively flat areas and much higher in regions of complex terrain. Current methods for modeling wind plant performance fall far short due to insufficient model fidelity and inadequate treatment of key phenomena combined with the lack of the computational power needed to address the wide range of relevant length scales associated with wind plants. Thus, the exascale challenge is the predictive simulation of a wind plant that comprises $O(100)$ multimegawatt wind turbines sited within a 10 km $\times$ 10 km area with complex terrain, involving simulations with $O(100)$ billion grid points. These predictive, physics-based, high-fidelity computational models validated with targeted experiments will drive innovation in the blade, turbine, and wind plant design processes by providing a validated "ground truth" foundation for new turbine design models, wind plant siting, operational controls, and the reliable integration of wind energy into the grid.

This multidisciplinary project embodies a systematic development of the modeling capability and computational performance and scalability required for effective exascale simulations. The project plan builds progressively from predictive petascale simulations of a single turbine—in which the detailed blade geometry is resolved, meshes rotate and deform with blade motions, and atmospheric turbulence is realistically modeled—to a multi-turbine array in complex terrain.

This new M&S capability will establish a virtual wind plant test bed that will revolutionize the design and control of wind farms and result in a significant advance in the ability to predict the response of wind farms to a wide range of atmospheric conditions.

The primary application codes in the ExaWind environment are Nalu-Wind, AMR-Wind, and OpenFAST. Nalu-Wind is an unstructured-grid, acoustically incompressible computational fluid dynamics (CFD) code written in C++, and it is a wind-specific version of the Nalu code, an large eddy simulation (LES) research code developed at Sandia National Laboratories. OpenFAST is a whole-turbine simulation code written in Fortran 2003 that grew out of FAST version 8. Nalu-Wind contains the infrastructure for unstructured-grid discretization of the underlying models, and it heavily uses the Trilinos Sierra Toolkit, which provides an unstructured mesh, in-memory, parallel-distributed database. The linear systems can be solved with *hypre*, Trilinos, or some combination of the two. In a new hybrid modeling approach adopted in 2020, the team created AMR-Wind, which is a structured-grid incompressible flow CFD solver with an adaptive mesh refinement capability, and is built on the AMReX libraries. The modeling approach will have a Nalu-Wind model surrounding the turbine blades, where it is important to resolve the blade boundary layers with body-conforming meshes. The blade meshes reside within a larger structured-grid AMR-Wind mesh. Coupling between meshes is handled with the with the overset approach for which mesh connectivity and constraints are created with the Topology Independent Overset Grid Assembler (TIOGA). In collaboration with the DOE Wind Energy Technologies Office High-Fidelity-Modeling project, Nalu-Wind and AMR-Wind are being continually appended and improved with wind-specific capabilities, including a new time step algorithm, fluid-structure-interaction capabilities, overset-mesh capabilities, and hybrid Unsteady Reynolds-averaged Navier-Stokes (URANS)/LES models.

4.1.1 ExaWind: Science Challenge Problem Description

The ExaWind challenge problem is a predictive simulation of a wind farm with tens of megawatt-scale wind turbines dispersed over an area of $50\,\mathrm{km}^2$. The goal is to capture crucial phenomena that are under-resolved in today's models, including wake formation, complex-terrain impacts, wake-atmosphere interaction, turbine-turbine interaction, and blade boundary-layer dynamics. This target requires an M&S capability that resolves turbine geometry and uses adequate grid resolution down to micron scales within the blade boundary layers. The resolution must capture the upstream chord-scale atmospheric turbulent eddies, the generation of near-blade vorticity, and the propagation and breakdown of this vorticity within the turbine wake to a distance of many rotor-diameters downstream. This application uses the Nalu-Wind CFD code and the OpenFAST turbine-simulation code that were specifically designed for wind turbine and wind farm simulations. The simulation will require a hybrid Reynolds-averaged Navier-Stokes (RANS)/LES turbulence model, fluid-structure interaction, and atmospheric turbulent flow.

The simulation will contain at least four megawatt-scale turbines (e.g., NREL 5 MW reference turbines) organized in a 2×2 array and residing in a $3\,\mathrm{km} \times 3\,\mathrm{km}$ domain with a height of at least 1 km. A hybrid-RANS/LES model will be employed for which a URANS model will be used near turbine surfaces, and an LES model will be used in the wake region. The simulation will have a mean wind speed at the turbines' rated speed (e.g., 11.4 m/s for the NREL 5 MW reference turbine). The model will require at least 20 billion grid points and 100 billion degrees of freedom (DOF) to resolve the system, and near-blade grid spacing will be such that the viscous sublayer within the RANS region is resolved. A successful simulation will require an optimized solver stack that minimizes the time per time step. A scientifically meaningful simulation duration will be for at least one domain transit time (about 370 s for the $3\,\mathrm{km} \times 3\,\mathrm{km}$ domain at 11.4 m/s). The team will demonstrate that such a simulation is feasible within 4 weeks of system time. Details on the challenge problem are given in Table 25.

4.1.2 ExaWind: KPP Stretch Goal

The stretch-goal simulation will contain at least nine megawatt-scale turbines (e.g., NREL 5 MW reference turbines) organized in a 3×3 array, and residing in a $4\,\mathrm{km} \times 4\,\mathrm{km}$ fluid domain with complex terrain and a height of at least 1 km. A hybrid RANS/LES model will be employed for which an unsteady RANS model will be used near turbine surfaces, and an LES model will be used in the wake region. The simulation will have a mean wind speed at the turbines' rated speed (e.g., 11.4 m/s for the NREL 5 MW reference turbine). The model will require at least 30 billion grid points and 150 billion DOF to resolve the system, and near-blade grid spacing will be such that the viscous sublayer within the RANS region is resolved. A successful simulation will require an optimized solver stack that minimizes the time per time step. A scientifically meaningful

Table 25: ExaWind challenge problem details.

Functional requirement	Minimum criteria
Physical phenomena and associated models	Acoustically incompressible fluid flow with fluid-structure interaction. A hybrid RANS/LES model in which the unsteady-RANS model will be used near turbine surfaces and an LES model will be used in the wake region.
Numerical approach, algorithms	Hybrid unstructured-grid and structured-grid finite volume solvers with overset meshes and implicit pressure projection. Linear and nonlinear structural finite element models.
Simulation details: problem size, complexity, geometry, and so on	Four megawatt-scale turbines in a 2×2 array residing in a volume of fluid with dimensions $3\,\text{km} \times 3\,\text{km} \times 1\,\text{km}$. Minimum grid size of 20 billion points with 100 billion DOF; the mesh will be refined to resolve the viscous sublayer on blade surfaces.
Demonstration calculation requirements	Simulation will be run with a time step size that represents statistically steady flow and corresponds to $C = O(1)$ in most of the LES regions and $C \gg 1$ in the RANS regions, where C is the Courant number.
Resource requirements to run demonstration calculation	1 hour at full system utilization.

simulation duration will be for at least one domain transit time of about 500 s for the $4\,\text{km} \times 4\,\text{km}$ domain at 11.4 m/s. The team will demonstrate that such a simulation is feasible within 4 weeks of system time.

4.1.3 ExaWind: Capability Plan

Successfully simulating the ExaWind challenge problem will require a complete set of physics models for hybrid RANS/LES simulations, meshes capturing complex moving geometry and an extreme range of scales, and solver algorithms and infrastructure that are optimized for next-generation platforms (i.e., GPUs). The ExaWind software stack has three interconnected solvers:

- Nalu-Wind, an unstructured-grid finite-volume solver built on Trilinos STK that can use linear solvers and preconditioners from Trilinos or *hypre*;
- AMR-Wind, a structured-grid background solver built on AMReX that is coupled to Nalu-Wind through the TIOGA overset-mesh interface; and
- OpenFAST, a wind turbine simulation code that includes models for blades, tower, control system, and other features.

An inherent challenge for incompressible-flow CFD is the need to solve global Poisson-type (i.e., pressure) and Helmholtz-type (i.e., momentum and scalar) linear systems at every time step at least once. In the original ExaWind solver pathway, a global Nalu-Wind model was envisioned, and there was the need to rebuild those linear systems and preconditioners at every time step due to mesh motion. These every-time-step requirements make a CPU plus accelerator (i.e., host plus device) solver pathway for which linear systems and preconditioners are created on the CPU and passed to the accelerator for solution impractical because the data-movement costs are expected to be too high. The new hybrid-solver approach alleviates some of these every-step costs in that linear-system graphs will remain unchanged, although linear-system coefficients and preconditioners must be reconfigured at every step. The ExaWind capability plan is thus designed to maximize the portion of the CFD solver stack that can run on the GPUs. However, the OpenFAST turbine models are planned to remain on the CPUs.

In ExaWind, the Nalu-Wind and Trilinos components are preparing for next-generation architectures with Kokkos, a parallel-performance abstraction layer. The ExaWind team has been converting the Nalu-Wind kernels to use Kokkos for threadable operations. The Trilinos software stack is being converted to Kokkos

parallelization under ECP/Advanced Technology Development and Mitigation (ATDM) funding. This is being done in close collaboration with ExaWind, and members of the ExaWind team include core members of the Kokkos, Kokkos-Kernels, STK, and MueLu teams. The ExaWind team is also working closely with the hypre team in preparing *hypre* for the effective use of GPUs. The hypre strategy for performance portability is to use accelerator-specific APIs (e.g., NVIDIA, CUDA, AMD HIP). AMR-Wind relies on the AMReX abstraction layer in which the architecture-specific APIs are hidden from AMR-Wind.

The potential for the ExaWind codes to predict the complex flow dynamics of wind farms depends on the validity of the underlying models. For practical grid resolutions, hybrid RANS/LES models have well-known deficiencies, and they need additional models to predict the transition between separated and attached flows, which are important in wind turbine dynamic stall events. Hence, improving the turbulence models in the ExaWind software stack has been and will continue to be an integral part of the project.

The following is a series of yearly mileposts that must be met for ExaWind to successfully complete its challenge problem.

FY20: Demonstrate host + device simulations on wind-relevant static-mesh simulations (e.g., highly resolved parked-turbine simulation) for which linear-system creation is accomplished on the host and solves are performed on the device. These simulations will establish the performance of linear-system solves on the device from which performance-improvement plans will be devised. Understanding the weak and strong scaling performance and identification of hot spots that could benefit from off-loading to the device are important.

FY21: Demonstrate a full-turbine simulation with moving meshes and the full solver stack (AMR-Wind + Nalu-Wind + OpenFAST) and where linear solves and overset search are performed on the device. These simulations will establish the performance of linear system and preconditioner creation on the device and will further inform solve-phase improvements. Weak and strong scaling performance and progress toward sufficiently small time per time step will be documented. Simulations will be demonstrated up to at least 10 billion grid points.

FY22: Demonstrate multi-turbine simulation with full hybrid structured/unstructured moving grid with the maximized amount of the software stack running on the device. Having a structured-grid background solver is crucial for achieving the ExaWind time per time step requirements. Such simulations will require a robust and accurate decoupling of the elliptic linear systems. Performance results will indicate the likelihood of successfully simulating the ExaWind challenge problem. Simulations will be demonstrated up to at least 20 billion grid points.

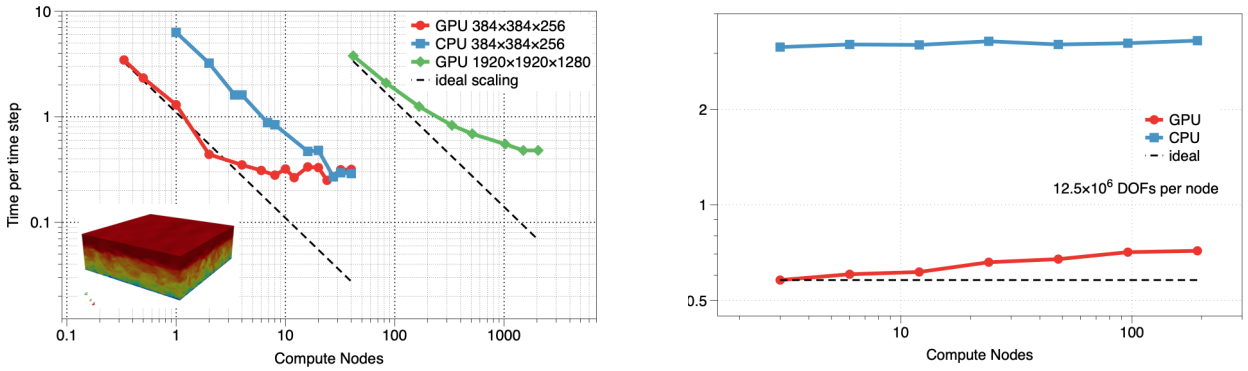
FY23: Demonstrate using 1 h of the full exascale system that the ExaWind challenge problem can be executed with sufficiently small time per time step, then pursue additional system time to complete the full challenge problem.

4.1.4 ExaWind: Progress on Early and Pre-Exascale Hardware

Performance on Summit

The ExaWind team used Summit extensively in FY20 for performance testing with a focus on strong and weak scaling and on CPUs and GPUs. This section describes the representative results. As described previously, the two primary CFD solvers are Nalu-Wind and AMR-Wind. Under the new hybrid solver strategy, AMR-Wind is the principal solver for the background turbulent atmospheric flow, and Nalu-Wind is the principal solver for the near-turbine flow, including blade boundary layers. As such, we examined independent solver performance in the target applications (i.e., AMR-Wind for atmospheric boundary layer [ABL] simulations and Nalu-Wind for blade-resolved turbine simulations). Performance testing of the coupled hybrid-solver configuration is the focus of the FY21 Q2 milestone.

Figure 14 shows the time per time step strong- and weak-scaling performance for ABL simulations. Strong scaling is shown for 38 million and 4.7 billion grid points. The strong-scaling limit is about 1.2 million grid points per GPU. For the 38 million grid point simulations, GPUs can offer a significant speedup over CPUs. The time step size for the 38 million grid point simulations is 0.5 s, and these simulations demonstrate that these ABL simulations can be run at faster than real time. Weak scaling (up to 2.5 billion grid points) shows



(a) Strong scaling.

(b) Weak scaling.

Figure 14: Strong-scaling (a) and weak-scaling (b) studies performed with AMR-Wind on Summit for atmospheric boundary layer simulations. The strong-scaling limit was observed around 1.2 million grid points per GPU. The inset in (a) shows a representative simulation result.

that CPU-based simulations show near-ideal scaling. GPU-based simulations show deviation from perfect weak scaling but still maintain excellent efficiency.

As described previously, Nalu-Wind can use various combinations of Trilinos and *hypre* to solve the underlying linear systems. Figure 15 shows the time per time step strong scaling performance on Summit for 23 million grid point blade-resolved simulations of the NREL 5-MW reference turbine. The left-hand side of Figure 15 shows results for which either *hypre* or Trilinos is used for the mass-continuity equations (i.e., the pressure Poisson system), and Trilinos is used for all other equations (i.e., momentum and scalars). The best performance is clearly seen with *hypre* on the CPUs. The right-hand side of Figure 15 shows strong scaling results for the best solver configuration of those examined, which uses *hypre* for all equations. With enough work for the GPUs, the *hypre* GPUs simulations can outperform those on the CPUs on a per-node basis. The time step size for these simulations is only 4 ms, so additional strong-scaling improvement is crucial for ExaWind's success for practical time to solution. Because of the hybrid-solver strategy, the individual Nalu-Wind models will be of modest size (e.g., less than 15 million grid points), so the team is less concerned with the weak scaling of Nalu-Wind.

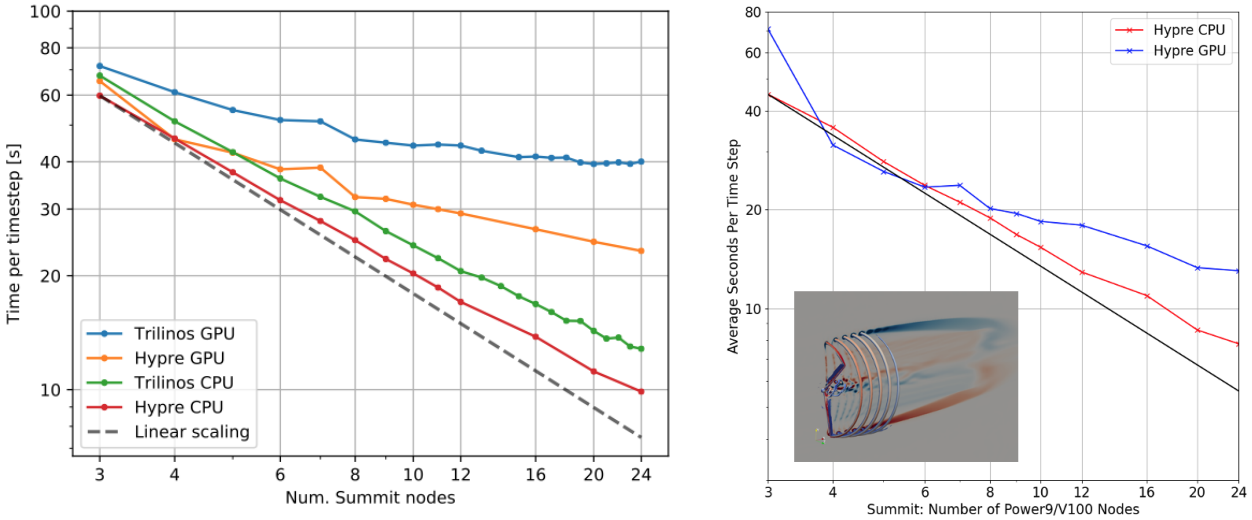
Next Steps

The ExaWind team will continue to push strong and weak scaling of AMR-Wind and Nalu-Wind on Summit. Although Nalu-Wind development on the pre-exascale systems is on hold as the team waits for Trilinos support, the team will continue to advance AMR-Wind. FY21 activities will focus on evaluating and improving GPU performance of the hybrid-solver strategy, establishing a next-generation hybrid RANS/LES turbulence modeling approach, and performing some of the highest fidelity wind energy simulations to date.

4.2 Combustion-Pele

Aggressive national goals to significantly reduce petroleum use and greenhouse gas emissions require significant improvements in all aspects of the nation's energy use. Combustion processes have historically dominated electrical power production and transportation systems. Despite significant advances in improving the efficiency and reducing the costs of alternative energy sources, combustion-based systems are projected to dominate the marketplace for decades. Consequently, these systems must be optimized for energy efficiency and reduced emissions.

This project is structured around providing a combination of first-principles direct numerical dimulation (DNS) and near first-principles (DNS/LES hybrids) simulations to advance understanding of fundamental turbulence-chemistry interactions in device-relevant conditions. The exascale motivating problem is to perform high-fidelity simulations of the relevant processes in a low-temperature reactivity-controlled compression ignition (RCCI) internal combustion engine. The relevant processes include turbulence, mixing, spray



(a) **(b)**

Figure 15: Strong-scaling studies performed with Nalu-Wind on Summit for blade-resolved simulations of the NREL 5-MW reference turbine. (a) Results in which the continuity equation was solved with *hypre* or Trilinos, and other equations were solved with Trilinos. (b) The best-configuration results for which all equations are solved in *hypre*. The inset in (b) shows representative simulation results.

vaporization, low-temperature ignition, flame propagation, and soot/radiation. RCCI is thermodynamically favorable relative to existing engines and thus holds the promise of groundbreaking efficiencies while operating in a regime that limits pollutant formation. The road map toward this exascale-era motivating problem includes simulating a multi-injection low-temperature diesel jet into an open domain with a large alkane fuel undergoing two-stage ignition processes, simulating dilute spray evaporation and mixing, and simulating multi-injection with fuels of varying reactivity in a geometry that influences the mixing field. The latter of these forms the challenge problem to demonstrate a new exascale capability.

The motivating problem that anchors the team's proposed development is a sufficiently realistic simulation of the in-cylinder processes in an internal combustion engine by using low-temperature combustion, for which RCCI is the exemplar. The enabled exascale-era simulations will address key scientific questions regarding the mixture formation effects, multistage ignition of a diesel surrogate fuel, lifted flame stabilization, jet re-entrainment affected by cylinder-wall geometry, and emissions. The simulation will account for the isentropic compression, subsequent injection of the high-reactivity fuel, and combustion processes in a compression ignition engine. Necessary physics include gas compression and models of fuel injection process, spray vaporization (i.e., injection of liquid fuel sprays into high-pressure conditions), mixing, and four combustion processes—autoignition, flame propagation, soot, and thermal radiation—all in a non trivial engine geometry. The scenario involves kinetically controlled processes in turbulent combustion, including ignition, extinction, and emissions. The application for this project, Pele, implements a hybrid LES/DNS approach in the compressible and low-Mach limits in which the team will refine to the DNS limit by using the machinery of adaptive mesh refinement (AMR), as necessary, to capture turbulence/chemistry interactions while restricting resolution to that required for a high-fidelity LES model far from the flame. The physical problem characteristics and the computational approaches required to be performant on the exascale architecture used to address them are summarized in Table 26.

Progress is needed on several challenging but tractable fronts. Existing simulation tools must evolve to perform well on exascale architectures, maintaining existing physics capability along with performance portability. Algorithmic and implementation issues involve new memory management and data layouts that respect memory systems of emerging architectures, new load balancing and communication strategies, communication-avoiding linear solvers that trade communication for computation, strategies for asynchronous

Table 26: Physical characteristics and computational approaches.

Characteristic/need	Approach
Impulsively started jets with disparate scales between fronts and turbulence. (Outer scales: 10 cm, milliseconds; inner timescales: micrometers, nanoseconds)	Dynamic adaptive mesh refinement
High-speed injection followed by subsonic conditions downstream	Compressible and low-Mach capabilities
Long-time horizons to set up turbulence for studying fundamental TCI	Hybrid DNS/LES (non-reacting LES, DNS for flame)
Lean, rich, and low-, intermediate-, and high-temperature chemistry critical in multistage ignition and formation of soot precursors.	Accurate and detailed thermochemistry of hydrocarbon fuels derived from theory-driven automatic mechanism development and reduction
Liquid fuel injection	Lagrangian polydisperse spray model
Coupling between mixture preparation and emissions	Detailed kinetics including emissions, soot model with radiation
Mixture preparation dependent on re-entrainment of combustion products	Realistic piston dish and cylinder wall geometry

task execution, and in situ analytics approaches. Improved modeling of physical processes is also needed. The fidelity of physics models that are tractable at the exascale greatly exceeds that of current petascale codes. High-fidelity physical models for nonideal fluid behavior, sprays, soot, and radiation, as well as nontrivial geometry, will enable a significant improvement in the realism of combustion simulations as typified by the target problem. Thirdly, advances in numerical algorithms will be needed to treat new and improved physical process models, optimize linear solvers to obtain good convergence rates with realistic geometry, improve coupling of the various physical processes to expose parallelism while maintaining a high order of accuracy, and optimize solution algorithms to handle the effects of nonideal chemistry.

4.2.1 Combustion-Pele: Science Challenge Problem Description

The specific science-based challenge problem is derived from the road map toward the motivating exascale era problem. Specifically, the challenge problem demonstrates the ability to simulate the interaction of two fuels with varying reactivity under a multipulse injection strategy into engine-relevant geometry. It is a baseline for a series of simulations that will enable the impacts of effects to be isolated, such as spray evaporation on mixture fraction and temperature, alternative fuels, and the design of strategies to control combustion phasing and subsequent combustion rates. The problem will be tractable under a realistic allocation by using the full capabilities of an exascale machine; within the projected 50–100× increase in productive capability of *Frontier* beyond the capabilities of *Titan*, the simulation will execute in approximately 2–4 weeks of wall time. Challenge problem details are given in Table 27.

4.2.2 Combustion-Pele: KPP Stretch Goal

As a stretch goal, multiphase fuel injection and soot emissions will be simulated in Pele to provide more realistic mixture formation (i.e., inhomogeneities of fuel and enthalpy distributions) conditions for downstream combustion processes and mechanistic understanding of particulate generation from fuel wall films in gasoline direct injection engines. Downstream combustion and emissions processes in engines are exquisitely sensitive to upstream mixture formation. Spray droplets will be treated with Lagrangian parcels, which will be injected at the inlet adopting the multipulse injection strategy of our KPP baseline problem. The droplet size distribution resulting from the upstream spray atomization processes will be obtained offline from existing volume-of-fluid DNS of atomizations, LES of engine spray combustions, and/or engine experiments. The

Table 27: Combustion-Pele challenge problem details.

Functional requirement	Minimum criteria
Physical phenomena and associated models	Multiscale CFD with reacting flows in low-Mach and compressible formulations with DNS+LES turbulence resolution with multispecies chemistry.
Numerical approach, algorithms	Time-explicit and deferred correction strategies in a compressible and projection-based low-Mach formulation, respectively. Finite volume spatial discretizations on block-structured AMR grids with embedded boundaries. Hybrid DNS/LES to enable fully resolved (i.e., DNS) treatment in which turbulence chemistry interaction occurs and modeled (i.e., LES) treatment to reduce resolution requirements in low-heat releases portions of the flow.
Simulation details: problem size, complexity, geometry, and so on	Gas-phase simulation of four multiple jets interacting in a $\frac{1}{4}$ scale geometry (2.5 cm diameter) derived from a production engine piston bowl with a flat head and centered fuel injector. Multiple pulses include a low-reactivity and high-reactivity fuel that capture cross-mixing and reactions between fuels by using ∼30–35 species; 1.5 ms physical simulation time; four levels of hierarchical mesh refinement; finest grid 1.25 µm; and a realistic environment of >50 bar.
Demonstration calculation requirements	Restart from the checkpoint that gives us a realistic development of plume into geometry obtained by running the case with restricted resolution (two levels of mesh refinement) and additional refinement added at restart for a total of four levels, then run 10–20 time steps to compute a realistic grind time that can be used to estimate the cost of the full-time horizon.
Resource requirements to run demonstration calculation	Estimate to run 20 time steps is 1.5 h by using the anticipated full system.

Table 28: Combustion-PELE mileposts.

Year	Target
FY19	• Complete EB functionality in PeleLM. • Create baseline GPU port of PeleC achieving $S/T > 3.5$ for non-EB-reacting calculation.
FY20	• Compare CUDA vs. OpenACC for PeleC-GPU and select strategy for PeleLM. • Prototype GPU chemistry strategy in PeleC and PeleLM. • Optimize PeleC-GPU to achieve $S/T > 10$ for EB-reacting calculation.
FY21	• Achieve $S/T > 15$ for PeleC for gas-phase combustion in piston bowl geometry (EB) with ndodecane chemistry (35 species). • Achieve $S/T > 10$ for PeleLM for gas-phase combustion in piston bowl geometry (EB) with ndodecane chemistry (35 species).
FY22	• Achieve $S/T > 16$ for PeleLM or $S/T > 30$ for PeleC. • Generate restart file for challenge problem measurement through low-resolution equivalent to challenge problem.

DNS of turbulent flame-wall interactions in the presence of fuel films will also be performed in Pele. Data from the DNS will augment optical engine experiments to understand and predict the effects of temperature and fuel/air equivalence ratio stratification on thermal pyrolysis and soot formation/transport from the fuel wall films.

Recent advances in ML/AI software and hardware technologies—such as the deployment of Summit with TPUs, GPUs, and CPUs-provide the potential to reframe the role of high-fidelity simulations in enabling progress in combustion research and engineering design. Traditionally, the scientific process involves a "human in the loop" to analyze a device-level experiment or challenge and abstract out a set of high-fidelity simulations to be performed and used to develop reduced order models that in turn enable parameter exploration, optimization, digital twins, and control strategies. With our stretch goal, we will attempt to demonstrate two aspects of this workflow for which ML and AI can be transformative. First, AI techniques currently show great promise for developing reduced order models (e.g., surrogate DNS, LES models) and control strategies. Second, the judgement of the scientist based on experience and heuristics to determine the necessary fidelity of the simulations for model development is critically important for effectively addressing the motivating question of interest. As part of our stretch goal, we will demonstrate the use of ML and AI techniques to take a description of a question of interest along with a device-level simulation to help formulate the problem to be solved in terms of characteristics, such as the accuracy of the chemical mechanism, turbulence treatment, multiphysics closure models, and the number of realizations. We will also explore the ability of surrogate models via ML and AI that replicate the quantity of interest with similar accuracy as the original high-fidelity simulation but at considerably lower computational cost.

4.2.3 Combustion-Pele: Capability Plan

The remaining direct barriers to executing the challenge problem require the completion of (1) adding embedded boundary (EB) geometry to PeleLM, (2) achieving necessary performance of PeleLM and PeleC on anticipated architectures, and (3) conducting sufficient demonstration calculations to exercise the requisite capabilities. For performance metrics, the measurements that can be readily made and are relevant are performance comparisons on a Summit node vs. a Theta node. These figures work backward from assuming a linear extrapolation with the same percentage of peak floating point operations per second between Summit and Frontier; as details of Frontier and development hardware become available, it will be more logical to convert the metrics into a Frontier node. Mileposts to measure necessary progress toward achieving the necessary capabilities are listed in Table 28.

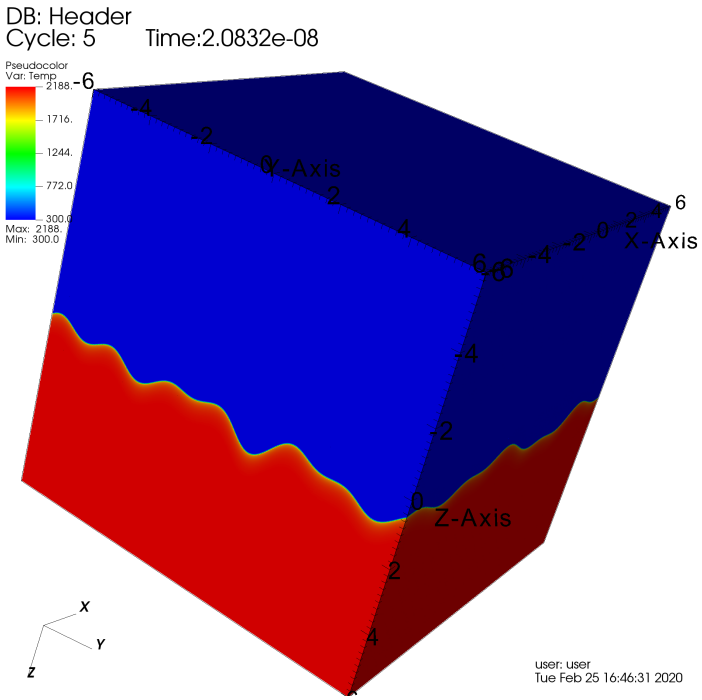


Figure 16: Plot of PMF problem run in PeleC.

4.2.4 Combustion-Pele: Progress on Early and Pre-Exascale Hardware

Performance on Summit

PeleC

The following plots summarize the performance of PeleC on Summit. Figure 16 shows a plot of the premixed flame (PMF) case used for benchmarking PeleC on Summit. In all benchmark cases, the DRM19 chemistry is used, which has 21 species. Figure 17 shows strong-scaling results after the initial port of the code to AMReX's GPU framework. This work involved porting the original F90 kernels to C++ lambdas. A speedup of approximately 154× is observed from the original F90 kernels to PeleC running on Summit's GPUs. Moving from the F90 kernels to an entirely C++ application resulted in a 2× speedup on the CPU. However, the most performant CPU configuration is on the Eagle machine at NREL, which is an Intel Skylake-based machine. Comparing this configuration with Summit's GPUs results in an 18× speedup. Strong scaling results are close to ideal on the CPU but are less desirable on the GPU due to the overhead of host to device and vice versa data transfers currently required when running PeleC on the GPU.

Next, the SUNDIALS ODE integrator library was integrated into PeleC to try to gain more performance on the GPU. Figure 18 demonstrates these results by comparing different ODE integrators available to PeleC. On the CPU, DVODE is the original ODE integrator chosen because it is the best performer on the CPU. DVODE is only available in the original F90 code. However, using the ARKStep integrator in SUNDIALS on the GPU improves PeleC performance 6× over DVODE.

Figure 19 shows the strong-scaling performance of PeleC with an EB problem running on Summit GPUs. EB routines are observed to be mostly insignificant in contribution to runtime. PeleC running on Summit's GPUs also allows users to obtain an equivalent time per time step by using 16× fewer nodes than if they were running on the CPU nodes on Eagle. Figure 22 shows a plot of the piston bowl problem by using the EB capability.

Figure 20 shows the weak scaling performance of PeleC on Summit. Using 4 million cells per node with two levels of AMR, weak scaling survives up to 2048 nodes, which is a 9 billion cell problem. At 4096 nodes and 18 billion cells, weak scaling efficiency drops off from 65 to 20 %. Therefore, weak scaling issues must be

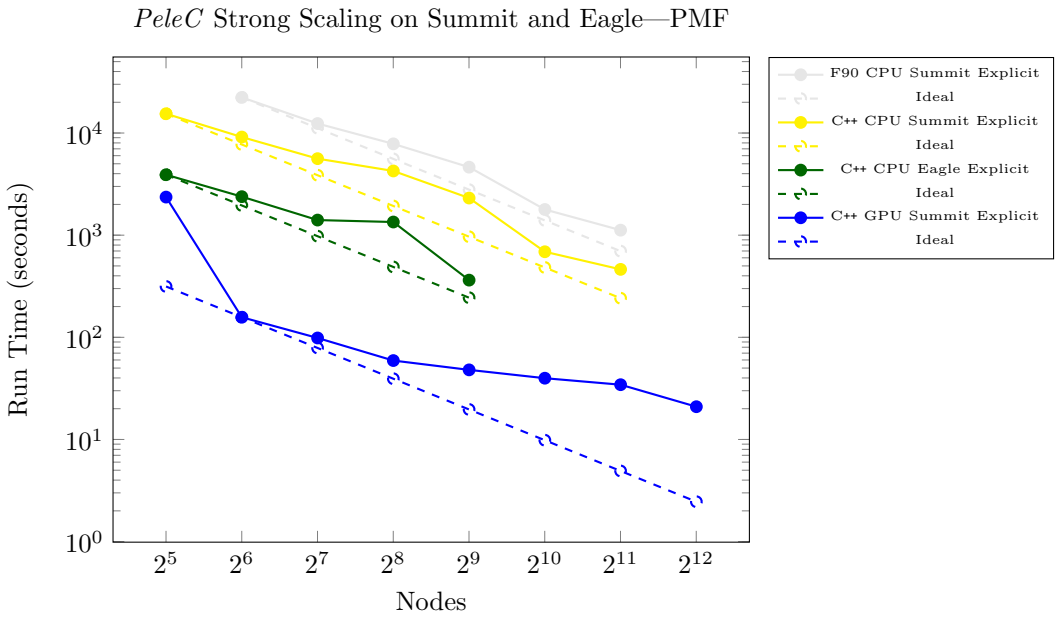


Figure 17: Strong scaling of PMF case with DRM19 chemistry on the Summit and Eagle machines. There were 360 million cells with two levels of AMR. The Intel 2018.4 compiler was used on Eagle, and built-in RK64 ODE integrator was also used.

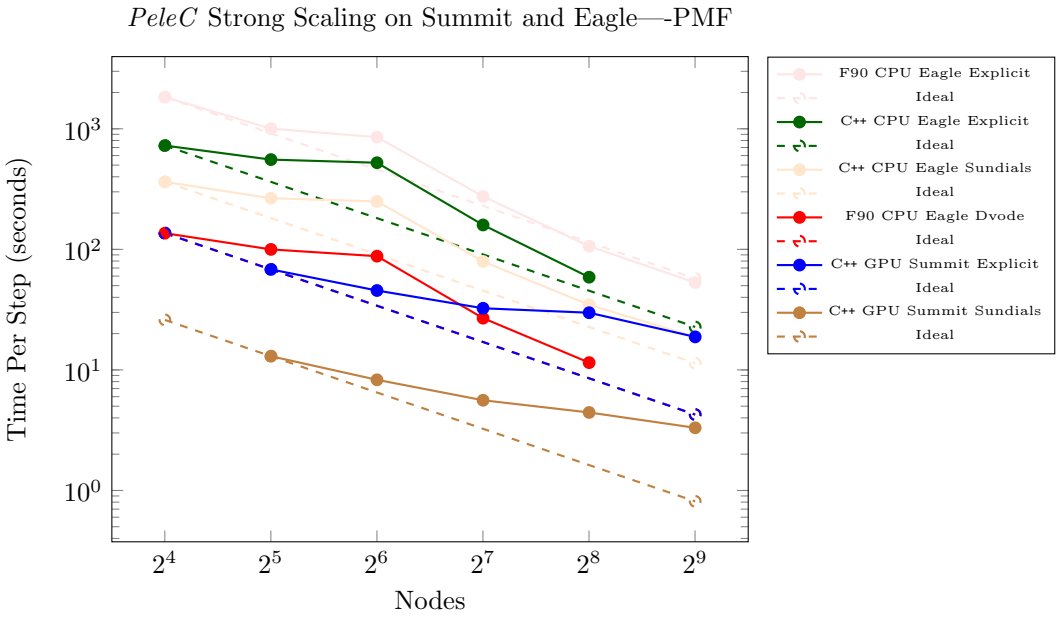


Figure 18: Strong scaling of PMF case with DRM19 chemistry on the Summit and Eagle machines after implementation of SUNDIALS ODE integrator. There were 164 million cells with two levels of AMR. The Intel 2018.4 compiler was used on Eagle.

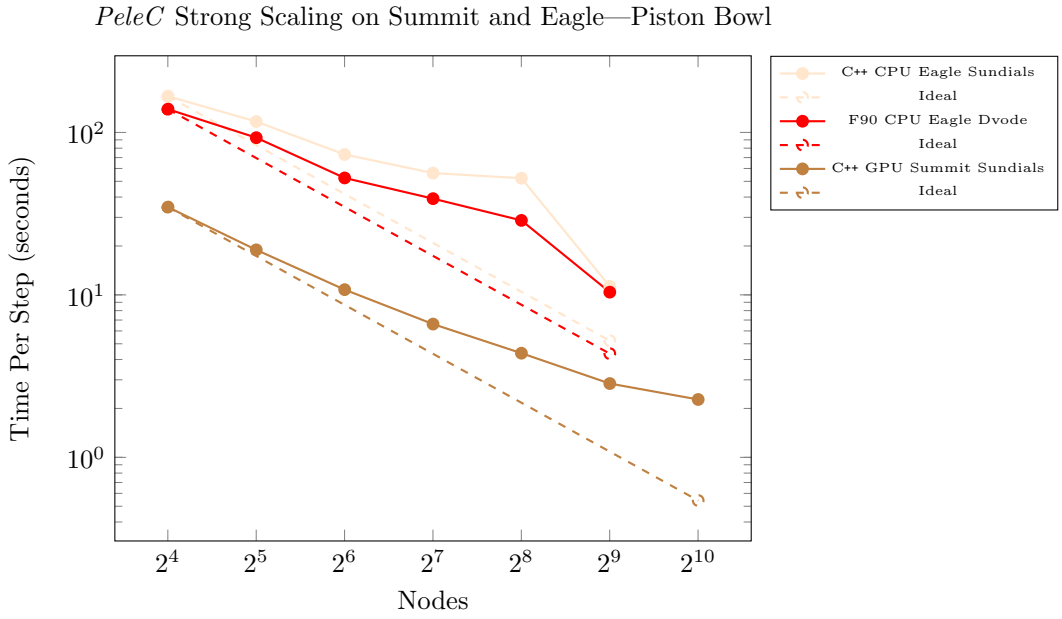


Figure 19: Strong scaling of the Piston Bowl case with DRM19 chemistry on the Summit and Eagle machines after embedded boundaries ported to GPU. There were 246 million cells with with levels of AMR. The Intel 2018.4 compiler was used on Eagle.

addressed to handle the 45 billion cell challenge problem. Table 29 shows the routines that contribute most to PeleC's runtime in the weak-scaling study.

PeleLM

The following section discusses the performance of the PeleLM port the Summit machine at ORNL. The first port, labeled *PMF*, sets up the idealized premixed flame configuration, as discussed previously for the PeleC tests. AMR is used in the following cases noted to generate multiple levels of grid refinement centered on the flame surface by tagging cells for refinement based on the presence of flame intermediate species. A series of these PMF configurations is generated to enable a weak-scaling study by doubling the domain size in each direction normal to the mean flow direction. In this way, most of the computational aspects of the setup remain constant as the problem is scaled to larger domains. An initial case is sized to fill the available GPU memory on a single Summit node. The larger cases are run on an increasing number of nodes proportional to the domain size so that the loading of each node remains constant.

Two other test configurations analyzed here include flow past cylinder (FlowPC) and flame past cylinder (FlamePC). In both cases, a cylindrical solid body is placed inside a domain with inflow-outflow boundaries in the coordinate-aligned mean flow direction. Periodic conditions are enforced normal to the mean flow. In the nonreacting FlowPC case, the conditions are such that the flow separates as it is diverted around the cylinder, and time-dependent fluctuations form as the flow reattaches on the downstream side of the solid body. The FlamePC case is similar except that a pair of flames is initialized in the fluctuating wake of the cylinder. The initial flame is created by using a 1D auxiliary laminar flame solution computed with CANTERA. The domain size in the direction of the cylinder axis is doubled for each simulation used to evaluate the weak scaling capabilities of PeleLM, including EB.

Table 30 shows the wallclock time spent in the various algorithm components of PeleLM on a single node of Summit obtained by inserting timers in the code around all aspects of the algorithm associated with each operation, including component-specific allocations and setup work. The single-level PMF case contains ∼9.4 million cells, and the two-level case contains ∼5 million cells. Table 30 shows that the acceleration due to the use of the GPU resources is a strong function of the numerical characteristics of each process. Integrating the chemical reaction ODEs is nearly 60× faster when using the GPU hardware compared with a

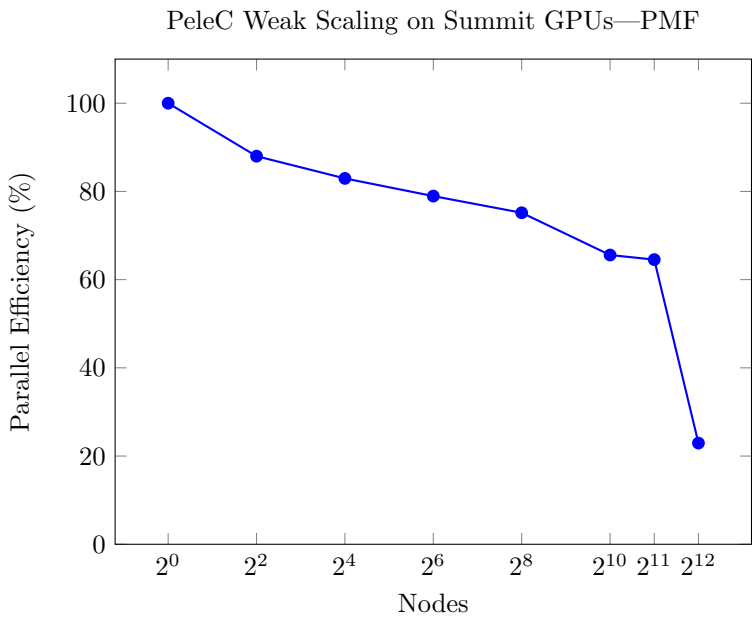


Figure 20: Weak scaling of the PMF case with DRM19 chemistry. There were 4,325,376 cells per node with two levels of AMR. The plot is based on the average time per time step for the single node case.

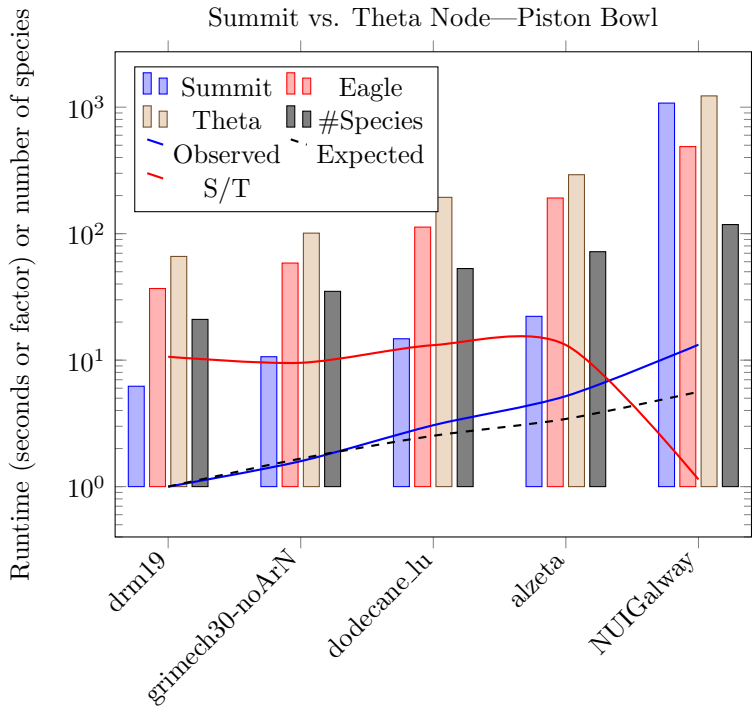


Figure 21: Speedup of Summit vs. Theta (S/T) results for chemistry models with an increasing number of species. There were 600,000 million cells, one level of AMR, and 10 time steps.

PMF - 1 node function	ncalls	excmin	excavg	excmax	max
fKernelSpec()	2834	8.29900	10.2300	11.6500	38.51%
FabArray::ParallelCopy()	539	2.65700	**5.7030**	9.9190	32.79%
PeleC::react_state()	59	4.65200	5.6770	6.4570	21.34%
PeleC::advance_hydro_pc_umdrv()	56	1.96600	2.4220	2.6680	8.82%
AsyncCpy	245	1.74300	2.1840	2.5640	8.48%
PeleC::getMOLSrcTerm()	84	1.40100	1.7110	1.8970	6.27%
AllocsInARKODE	245	0.53150	0.7002	0.8072	2.67%
FillBoundary_finish()	56	0.07919	0.1886	0.3886	1.28%

PMF - 2048 nodes function	ncalls	excmin	excavg	excmax	max
FabArray::ParallelCopy()	539	13.410000	**19.1100**	25.9800	52.26%
fKernelSpec()	2849	7.458000	10.5000	13.1700	26.50%
PeleC::react_state()	59	4.031000	5.9670	8.0800	16.26%
PeleC::advance_hydro_pc_umdrv()	56	1.759000	2.4390	3.0040	6.04%
AsyncCpy	246	1.479000	2.2000	2.7270	5.49%
PeleC::getMOLSrcTerm()	84	1.263000	1.7160	2.0490	4.12%
DistributionMapping::LeastUsedCPUs()	10	0.273800	1.0300	1.7530	3.53%
AmrMesh::MakeNewGrids()	7	0.339700	1.4100	1.4700	2.96%

PMF - 4096 nodes function	ncalls	excmin	excavg	excmax	max
FabArray::ParallelCopy()	539	6.964000	**96.3800**	103.300	75.65%
fKernelSpec()	2834	7.527000	10.6500	74.680	54.72%
PeleC::react_state()	59	4.133000	6.0710	20.570	15.07%
AsyncCpy	245	1.536000	2.1940	12.590	9.23%
PeleC::reflux()	12	0.001965	1.3720	10.290	7.54%
AllocsInARKODE	245	0.343800	0.9188	8.668	6.35%
PeleC::advance_hydro_pc_umdrv()	56	1.802000	2.5460	8.336	6.11%
DistributionMapping::LeastUsedCPUs()	10	1.178000	3.7620	5.436	3.98%

Table 29: Routines contributing most to PeleC's runtime for the PMF weak-scaling study.

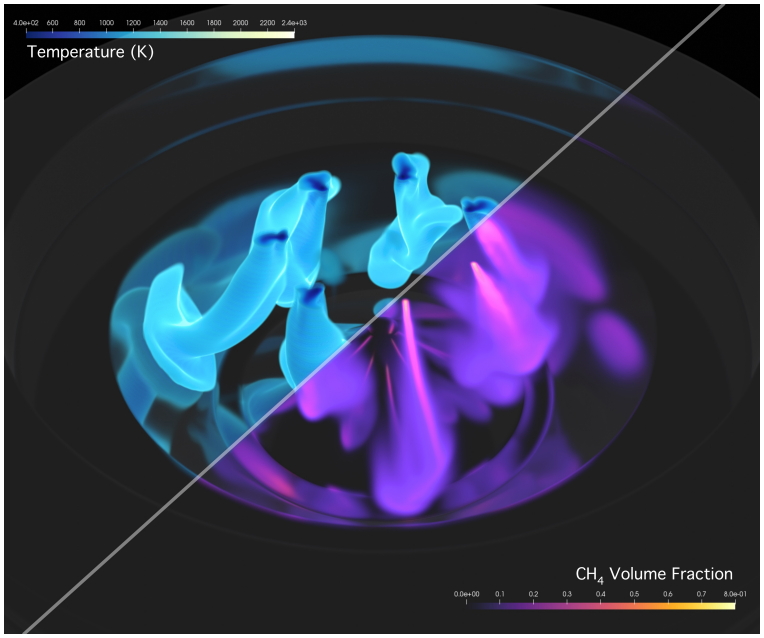


Figure 22: Plot of the piston bowl problem run in PeleC.

distributed CPU-only implementation, whereas the other operations experience a more modest improvement. The chemistry acceleration in the two-level case is considerably less dramatic, achieving 30× improvements, mainly because the finer level comprises smaller boxes for which the speedup is less important. This illustrates the conflicting performance aspects of AMR on GPU in which where the AMR algorithm objective is to add as little additional work as possible where it is needed (small boxes in region of interests), whereas GPU launches are efficient on large boxes. The stencil-based operations and those with particularly low-arithmetic intensity, such as linear solvers, experience particularly low levels of acceleration. This is likely due to their dependence on communication operations, which must move data from device to host, then across the network, then from host to device. In some sense, the data shown here represent a best-case scenario in the current implementation—since there is no EB geometry, the multigrid linear solvers are constructed to use a relatively large number of coarsened levels for V-cycle relaxations. When EB is included, the number of coarsened levels available for geometric multigrid becomes a strong function of the geometric complexity of the domain. The iterative solvers become increasingly inefficient, and the arithmetic intensity of these parts of the code becomes unacceptably low.

Region	Time (s)			Percent	
	MPI+OMP	MPI+Cuda	speedup	MPI+OMP	MPI+Cuda
Single-level PMF					
PeleLM::advance::reactions	288.8	4.968	58.1	81.99	31.83
PeleLM::advance::diffusion	28.05	4.242	6.61	7.97	27.18
PeleLM::advance::project	14.86	1.228	12.1	4.23	7.90
PeleLM::advance::mac	1.167	0.571	2.0	0.33	3.66
PeleLM::advance::velocity_adv	2.769	0.4656	5.9	0.79	3.17
PeleLM::advance::scalar_adv	3.199	0.3756	8.5	0.91	2.41
Two-level PMF					
PeleLM::advance::reactions	164.4	5.348	30.7	47.12	21.83
PeleLM::advance::diffusion	74.32	9.516	7.8	21.43	33.70
PeleLM::advance::project	40.44	2.323	17.4	12.08	8.33
PeleLM::advance::mac	5.81	1.591	3.66	1.67	5.63
PeleLM::advance::velocity_adv	11.25	1.444	7.79	4.51	5.59
PeleLM::advance::scalar_adv	3.8	0.404	9.4	1.09	1.43
PeleLM::mac_sync	15.38	3.946	3.90	4.39	13.94

Table 30: Contributions of the different components of PeleLM algorithm to the wallclock time of a single time step. Speedup is computed as the ratio between the MPI+OMP and the MPI+Cuda times.

Figures 23–25 show the results of a weak scaling study with each data point corresponding to the wallclock time of a single coarse time step. On both CPU and GPU, the plots show a linear loss of parallel efficiency as we move from a single node to multiple nodes. Figure 23 shows that the GPU performances of the **PMF** configuration described in Table 30 on a single node are maintained all the way to 2048 Summit nodes.

Figures 24 and 25 show the results of a three-level FlowPC case and the single-level FlamePC case, respectively. Although the speedup obtained on GPU compared with CPU is less important than that of the PMF case, the weak-scaling data indicate that the added constraints on the linear solver induced by using EB do not prevent us from achieving relatively good scaling properties.

All the graphs indicate a loss of parallel efficiency as the problem is spread over more compute nodes. To provide more detailed indications as to what is causing this loss, Figure 26 shows the weak scaling data of PeleLM on GPU for the FlamePC test case and includes the timing of two main components of the algorithm: chemistry and diffusion. In this case, the diffusion (i.e., linear solver) operators clearly account for a significant amount of the loss of parallel efficiency.

Figure 27 shows the results of a strong scaling GPU study based on the PMF configuration. We adjusted the problem definition to fully load a single node of Summit and compute the wallclock time to complete a

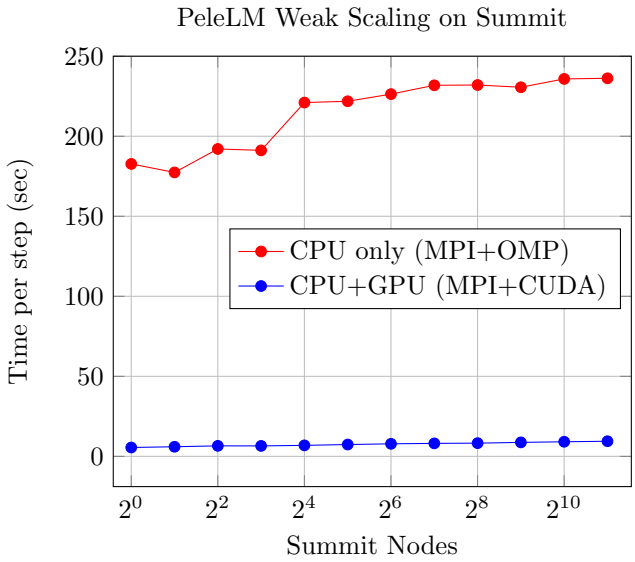


Figure 23: Weak scaling of PeleLM for the single-level PMF test case.

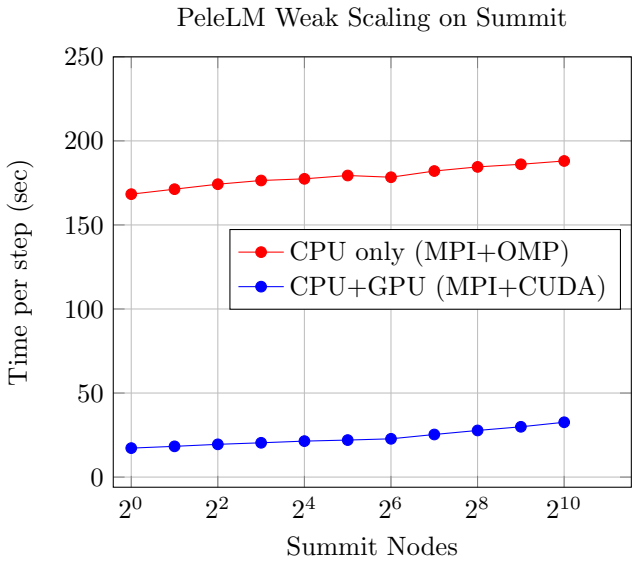


Figure 24: Weak scaling of PeleLM for the three-level **FlowPC** test case.

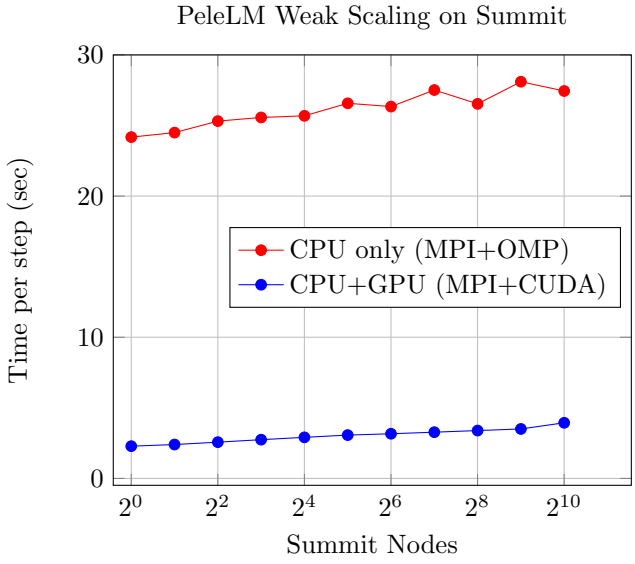


Figure 25: Weak scaling of PeleLM for the single-level **FlamePC** test case.

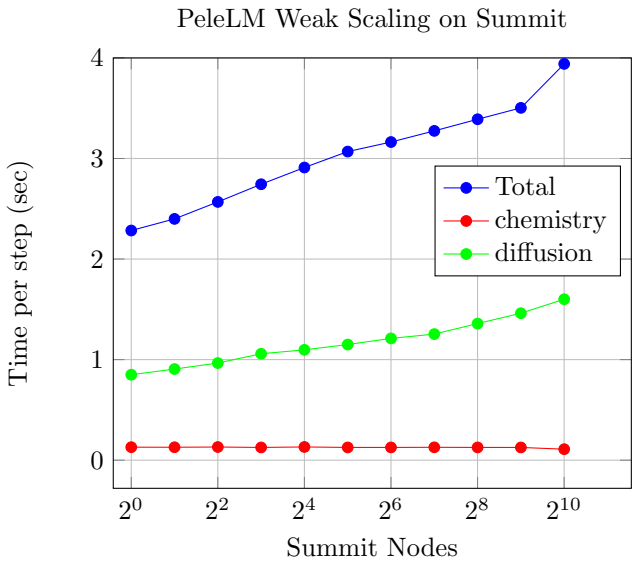


Figure 26: Weak scaling of PeleLM on GPU for the single-level **FlamePC** test case.

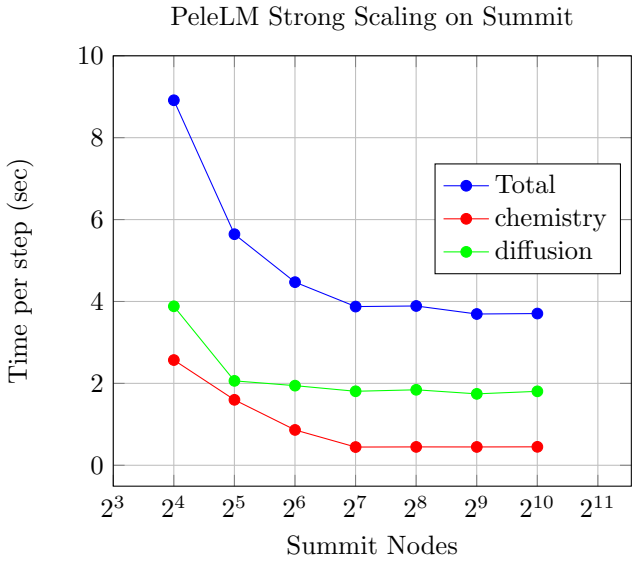


Figure 27: Strong scaling of PeleLM for the single-level **PMF** test case.

full time step of the algorithm. We then distributed the memory and storage for the same problem over twice the number of nodes and repeated the run, continuing until there was no improvement in runtimes.

The results of Fig. 27 clearly indicate the limited ability of PeleLM to achieve good strong-scaling performances on GPU. In particular, the performances of the diffusion rapidly deteriorates. To provide a more detailed analysis of the bottlenecks that limit the strong scaling, the NVIDIA Nsight-System profiling tool was used. An example of the results is presented in Fig. 28; the top image was obtained by using 16 Summit nodes, and the bottom image was obtained by using 64 Summit nodes. These images show the entire call stack of PeleLM with the various pieces of the algorithm clearly visible and the diffusion and chemistry highlighted. In accordance with the results of Fig. 27, the diffusion time is almost unchanged as the number of nodes is multiplied by four and its relative contribution is thus increased.

This tool also provides quantitative measurements regarding the time spent in the different routines of the linear solvers mostly comprising the diffusion time. A fairly constant amount of time is dedicated to the *FillBoundary* operation, which is responsible for the communications of the ghost cells data between adjacent boxes. This operation, which is key during the diffusion solves, is dominated by MPI communications between ranks followed by *HostToDevice* memory copies. Because the overhead of memory copy operations on heterogeneous CPU-GPU systems is higher than that of homogeneous CPU systems, this represents a significant performance bottleneck for the linear solver as the granularity of the data is increased.

PeleMP

PeleMP is the multiphysics Pele code that couples with PeleC and PeleLM to accomplish the KPP stretch goal. Currently, PeleMP contains a spray modeling functionality that is coupled with both Pele codes on CPUs and with PeleC on GPUs.

To evaluate the performance of PeleC coupled with the PeleMP spray module, a weak scaling test was performed on Summit. The test was a quiescent flow with uniformly distributed spray droplets that convect. There were 8 million particles per node. Two nested additional levels of refinement were fixed in the center of the domain.

Figure 29 shows the time per time step from the weak-scaling test performed on Summit. Additionally, the figure lists the parallel efficiency; at 216 nodes, the parallel efficiency is 69 %. These values are consistent with the those seen by AMReX particle tests. Obtaining runtimes at higher node counts was prevented due to a bug when particle counts exceeded the size of an integer; this is currently being investigated.

Figure 30 shows the speedup incurred by using GPU hardware. A particle-laden Taylor-Green vortex problem was used, and the number of particles increased. The tests were run on eight Summit nodes and eight Cori Haswell nodes. The figure shows that switching from CPU to GPU architecture resulted in an

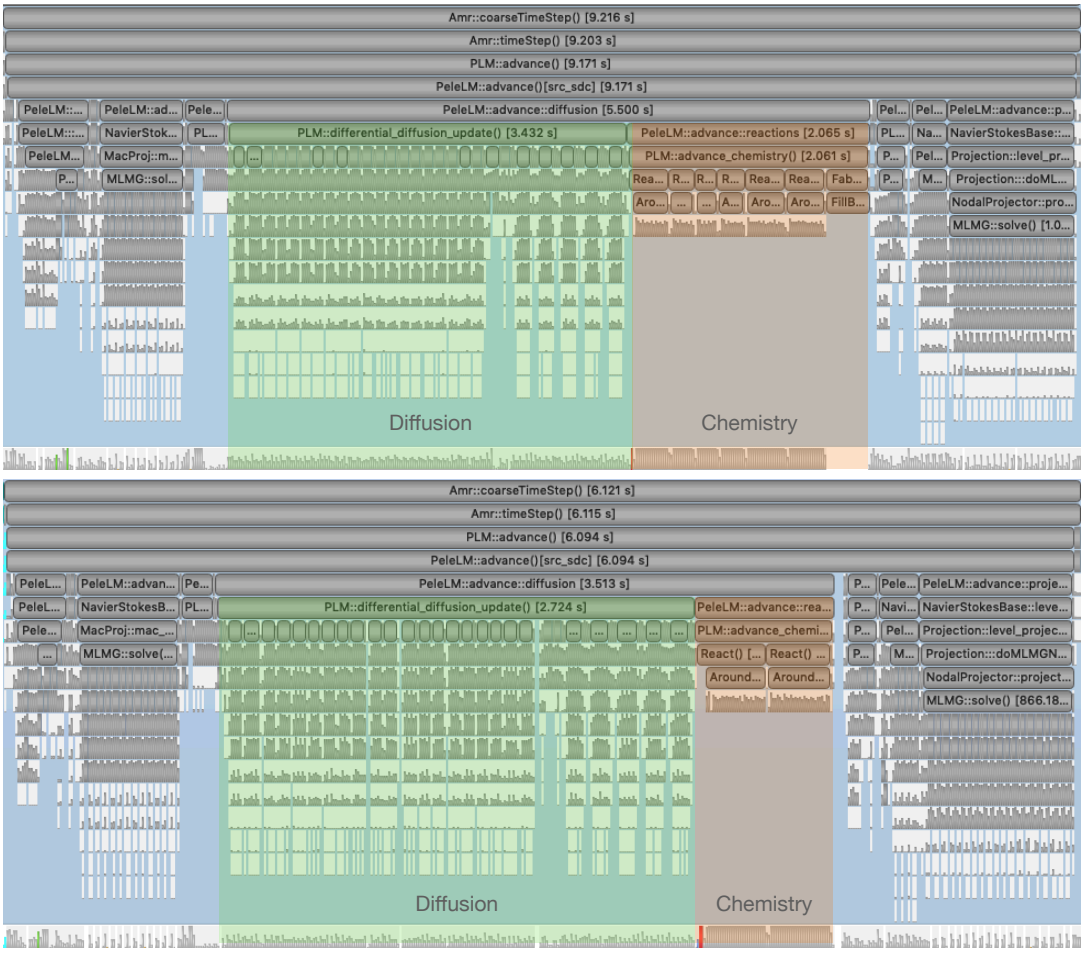


Figure 28: Example of NVIDIA NSight-System output for PeleLM obtained with 16 Summit nodes (top) and 64 Summit nodes (bottom) in the PMF strong-scaling case.

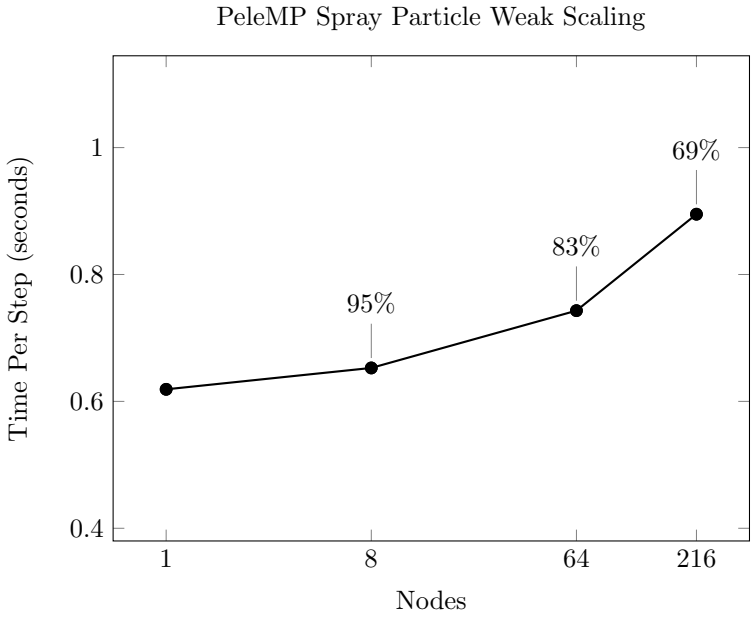


Figure 29: PeleMP coupled with PeleC weak-scaling performance.

almost a 100× speedup for flows without particles. The performance improvements diminished but remained significant as particles were added to the domain. Since higher particle loads can be offset by increasing the number of particles per parcel, significant performance issues are not expected for practical applications.

Particle creation and distribution are the known performance bottlenecks. Currently, particles are created—either at initialization, during injection, or for ghost and virtual particles—on the CPU and are copied to the device. This operation is expensive and can negatively impact weak-scaling performance, particularly for problems that involve spray injection. Future work will prioritize particle creation directly on the device to avoid this issue. Additionally, future effort will overlap particle communication with the reaction solver to effectively hide the cost associated with the particle distribution.

The Grit particle library is another module used for the spray modeling portion of the KPP stretch goal. Grit is a Kokkos-based performance-portable software library that serves as an alternative to the native AMReX particle library, which is implemented to couple with a finite-difference flow solver, S3D, and with PeleC for coupled Eulerian-Lagrangian spray simulations. We successfully used Grit to verify the governing equations for sprays. We were particularly looking at the mass and energy conservations in the presence of evaporation and the transport properties used in the models. We also took advantage of the simpler data structures of Grit to expand modeling capabilities. Grit can model the impingement of sprays on surfaces, which employs a distance-function-based algorithm to advance the sprays near the boundaries. The developed impingement model is ready to be implemented in Pele with its embedded boundaries.

Next Steps

The next steps involve working with AMReX developers to increase the weak-scaling efficiency needed to run the challenge problem. Other dominant time-consuming routines in Pele involve integrating ODEs for chemistry and reactions. The Pele team will work with the SUNDIALS team to increase the performance of reaction calculations on the GPU. A second option for integrating ODEs involves a tool called SINGE, which can compile reaction routines into a GPU device assembly code. SINGE was originally created for CUDA, but converting generated code to other languages will also be pursued. A profile of PeleC on the V100 for the piston bowl problem shows that the `react_state()` routine that calls SUNDIALS accounts for 68 % of the runtime. This leaves `getMOLSrcTerm()` at 22 % left to be gained in PeleC routines directly, which the team will pursue. For PeleLM, the linear solvers required for the diffusion step will be the focus of the next round of optimizations. We will incorporate several new features that are currently in development to improve performance, including working with multiple components simultaneously, and other communication-avoiding

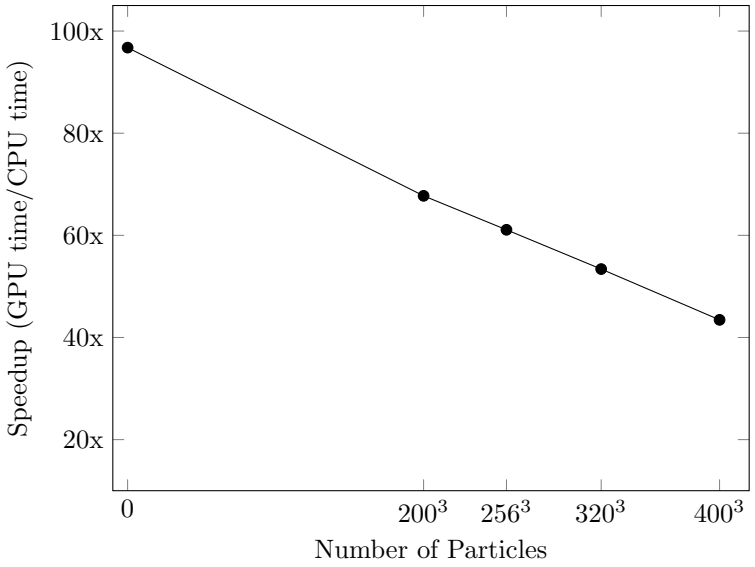


Figure 30: GPU/CPU performance comparison for PeleMP coupled with PeleC.

strategies. AMReX has many tunable parameters that affect the efficiency of the communication, which will be necessary for exploring and understanding. Lastly, there are also more opportunities for asynchronous behavior throughout the codes likely able to be exploited. For PeleMP, we will apply the aforementioned performance optimizations and couple the spray module with PeleLM for use on GPUs. Additionally, the spray functionality will be integrated with embedded boundary, and models for soot and radiative heat transfer will be added to PeleMP.

4.3 ExaSMR

In the nuclear industry, small modular reactors (SMRs) offer affordable nuclear energy-based electricity production while avoiding some of the traditional limitations that encumber large nuclear reactor designs, such as high capital costs and long construction timelines. The current US nuclear fleet was built on a multidecade history of experimental and operational data; for licensing, SMRs leverage that experience but depend on M&S for design optimization. However, industry-class computing is based on heavily parameterized, coarse models of reactor phenomena. This provides a compelling opportunity for high-resolution calculations that can benchmark and influence these engineering-class simulations.

The Consortium for the Advanced Simulation of Light Water Reactors DOE energy innovation hub has demonstrated the value of HPC to the nuclear industry by deploying highly accurate MC neutronics on DOE leadership-class platforms. The datasets generated by these first-of-a-kind simulations were used to validate the start-up calculations of the new Westinghouse Electric Company AP1000 reactor. Although this represented a significant advance, petascale computing and application limitations restrict the calculations to reactor start-up conditions. In contrast, the objective in the exascale SMR project ExaSMR is to provide benchmarks for multicycle operational design parameters for SMRs by 2025.

The ExaSMR approach is to integrate MC neutronics and CFD—the most accurate numerical methods available for operational reactor modeling—for efficient execution on exascale systems. ExaSMR builds on a base of applications that have demonstrated high efficiency and excellent scaling on current petascale leadership-computing levels. The ExaSMR effort will provide value to nuclear fuel vendors and the broader nuclear community through the generation of highly detailed benchmark datasets of operational nuclear reactors. The MC neutronics method presents significant challenges related to the random access of unordered data on hierarchical memory architectures; CFD presents the challenge of achieving high floating-point efficiency on sparse linear algebra problems. Furthermore, the MC particle transport and CFD implementations developed will complement other projects in the DOE that are based on similar methods. Requirements for these numerical motifs will also help to inform choices in hardware, runtime systems, and programming models for

exascale systems.

4.3.1 ExaSMR: Science Challenge Problem Description

The ExaSMR challenge problem is the simulation of a representative NuScale SMR core by coupling continuous-energy MC neutronics with CFD. Features of the problem include the following:

- representative model of the complete in-vessel coolant loop,
- hybrid LES/RANS turbulence model or RANS plus an LES-informed momentum source for treatment of mixing vanes, and
- pin-resolved spatial fission power and reaction rate tallies.

Details on the challenge problem specification are given in Table 31.

The simulation objective is to calculate reactor start-up conditions that demonstrate the initiation of natural coolant flow circulation through the reactor core and primary heat exchanger. The driver application, ENRICO, performs inline coupling of the Nek5000 CFD module with MC through a common API that supports two MC modules: Shift, which targets the Frontier architecture at ORNL, and OpenMC, which targets the Aurora system at ANL.

Minimum neutronics requirements for the coupled simulation are as follows:

- full-core representative SMR model containing 37 assemblies with 17×17 pins per assembly and 264 fuel pins per assembly item,
- depleted fuel compositions containing $O(150)$ nuclides per material,
- 10^{10} neutrons per eigenvalue iteration,
- pin-resolved reaction rate with a single radial tally region and 20 axial levels, and
- six macroscopic nuclide-independent reaction rate tallies.

Minimum CFD requirements for the coupled simulation are as follows:

- assembly bundle mesh models with momentum sources from a resolved CFD calculation on a representative spacer grid and
- full-core mesh 200×10^6 elements and 70×10^9 DOF.

4.3.2 ExaSMR: KPP Stretch Goal

The ExaSMR stretch goal is to improve the fidelity of the MC and CFD models. For the MC problem, this will mean increasing the number of radial rings per fuel pin tally region from one to three and tallying nuclide-specific microscopic reaction rates instead of nuclide-independent reaction rates. With over 150 nuclides per fuel material, this will result in a substantially increased memory footprint and add significant computational cost. Tallying microscopic reaction rates will demonstrate the ability to perform isotopic depletion calculations. For the CFD problem, the stretch goal is to replace the modeling of mixing vanes by using an LES-informed momentum source with an explicit representation of mixing vanes with a hybrid LES/RANS turbulence model. This addition dramatically increases the number of spatial DOF needed to resolve the problem and places more stringent requirements on the time steps needed to maintain numerical stability.

Table 31: ExaSMR challenge problem details.

Functional requirement	Minimum criteria
Physical phenomena and associated models	Eigenvalue form of the linear Boltzmann transport equation with quasistatic nuclide depletion neutronics coupled to hybrid RANS/LES (or equivalent accuracy model), incompressible CFD with Boussinesq approximation, or low-Mach incompressible CFD with nonzero thermal divergence.
Numerical approach, algorithms	The neutronics solver is an MC particle transport; the CFD solver is a spectral finite element on unstructured grids. Physics are coupled by using a quasistatic approximation.
Simulation details: problem size, complexity, geometry, and so on	The neutronics model has a minimum of 200,000 tally bins and 10 billion particle histories per eigenvalue iteration. The CFD model has a minimum of 200 million elements and 70 billion DOF.
Demonstration calculation requirements	Run 30 eigenvalue cycles (10 inactive and 20 active) to estimate the MC particle tracking rate and 1000 time steps toward steady-state convergence in the CFD solve. Only one nonlinear (Picard) iteration is required. To facilitate comparison with the baseline measurement, the neutronics portion of this calculation requires six macroscopic reaction rates and one radial region per pin.
Resource requirements to run demonstration calculation	2 h at full system.

4.3.3 ExaSMR: Figure of Merit

The coupled simulation FOM for ExaSMR is a harmonic weighted average of the neutronics and CFD performance,

$$\mathrm{FOM} = \frac{2}{\mathrm{FOM}_{\mathrm{MC}}^{-1} + \mathrm{FOM}_{\mathrm{CFD}}^{-1}} , \tag{4}$$

where each physics FOM is a ratio of a measured work rate to the baseline work rate, that is,

$$\mathrm{FOM}_{\mathrm{MC}} = \frac{W_{\mathrm{MC}}^{\mathrm{measured}}}{W_{\mathrm{MC}}^{\mathrm{baseline}}} , \tag{5}$$

$$\mathrm{FOM}_{\mathrm{CFD}} = \frac{W_{\mathrm{CFD}}^{\mathrm{measured}}}{W_{\mathrm{CFD}}^{\mathrm{baseline}}} . \tag{6}$$

The individual physics work rates are given by

$$\mathrm{W}_{\mathrm{MC}} = \frac{\text{neutrons}}{\text{wallclock time}} , \tag{7}$$

$$\mathrm{W}_{\mathrm{CFD}} = \frac{\text{degrees-of-freedom}}{\text{wallclock time}} . \tag{8}$$

The MC work rate is computed using only the active cycle particle tracking rate, which is more computationally intensive and typically occupies most of MC runtime. If ensemble averaging is used to perform the CFD calculation, the number of DOF used in the work rate measurement includes the sum over all ensembles. The demonstration problem to evaluate the final FOM value will be a coupled MC-CFD calculation with the relevant work rates for the physics components extracted from this calculation. This is done to facilitate comparison with the baseline calculations, which were performed by running each code independently due to the lack of a coupled physics capability during the baseline measurements.

The current baseline measurements are:

- $\mathrm{W}_{\mathrm{MC}} = 10.4$ million neutrons per second (Mn/s) on Titan and

- $W_{CFD} = 3.0$ billion DOF per second on Titan.

FOM Update

Updated FOM measurements were performed on Summit. Because the GPU-enabled versions of the physics codes are not yet supported in coupled physics simulations within ExaSMR, the work rates were computed by executing the MC and CFD codes independently. An MC simulation of a full SMR core—the same one used to compute the baseline work rate on Titan—was performed on 4096 Summit nodes with 24.6 billion neutrons per eigenvalue cycles (1 million neutrons per GPU). This simulation achieved a particle tracking rate of $W_{MC} = 242$ Mn/s, corresponding to an FOM of 23.3. Extrapolating to the entire 4608 Summit nodes results in a projected FOM of 26.2.

The NekRS code was used to perform a simulation of a full SMR core. The computational model included 174 million spatial elements of degree 7, corresponding to 59.7B total DOF. Executing on 3620 Summit nodes, NekRS achieved a work rate of $W_{CFD} = 506$ billion DOF per second and an FOM of 168.7. Extrapolating this value to the full Summit machine corresponds to a projected FOM of 214.7. A calculation for the same computational model on the entire machine resulted in a work rate of $W_{CFD} = 494$ billion DOF per second and an FOM value of 164.7. This lower value is because the computational model did not have enough DOF to fully saturate the machine; executing a larger problem on the full machine is expected to yield a work rate closer to the extrapolated work rate for the 3620 node case.

For these Summit simulations, the current single-physics achieved FOM values are:

- $\text{FOM}_{\text{MC}}^{\text{Summit}} = 23.3$ and
- $\text{FOM}_{\text{CFD}}^{\text{Summit}} = 168.7$.

This equates to a combined ExaSMR FOM of:

$$\text{FOM}_{\text{achieved}}^{\text{Summit}} = 40.9\,. \tag{9}$$

Using the work rates extrapolated to the full Summit machines gives:

$$\text{FOM}_{\text{extrapolated}}^{\text{Summit}} = 46.7\,. \tag{10}$$

4.3.4 ExaSMR: Progress on Early and Pre-Exascale Hardware

Performance on Summit

The Shift and NekRS codes have been heavily used on NVIDIA GPUs, particularly on Summit. As illustrated in Fig. 31, MC transport exhibits very favorable weak scaling behavior when the number of particles per GPU remains constant. This trend holds equally well for CPU and GPU implementations. Thus, if a single GPU can be used effectively, it is very likely that an entire supercomputer can be used. Because of the significant amount of branching and indirection, MC transport is inevitably bound by memory latency rather than bandwidth or floating point operations. Furthermore, because the parallel regions in transport codes must implement complex physics models, many kernels require a significant number of registers (thread-local storage). This register pressure results in reduced occupancy on the device, limiting the ability of the GPU to hide the latency of memory accesses via context switching. Several algorithmic modifications were investigated in a CUDA solver within the Shift code to reduce thread divergence and the number of registers used per kernel, thus increasing occupancy. The impact of these algorithmic modifications in Shift is demonstrated in Table 32.

Development activities within the NekRS code were conducted in close collaboration with the CEED project. NekRS supports execution on a variety of computing platforms by using the OCCA performance portability model. OCCA supports a variety of back ends, including CUDA, OpenMP, and OpenCL. Algorithmic optimizations for fluid flow problems in NekRS have included the introduction of new preconditioners, including reduced-precision approaches, as well as runtime-optimized GPU-aware gather/scatter and projection operations. Figures 32 and 33 demonstrate the weak- and strong-scaling behavior of NekRS, respectively, on problems related to the ExaSMR challenge problem. The weak-scaling efficiency from 87 nodes to the full

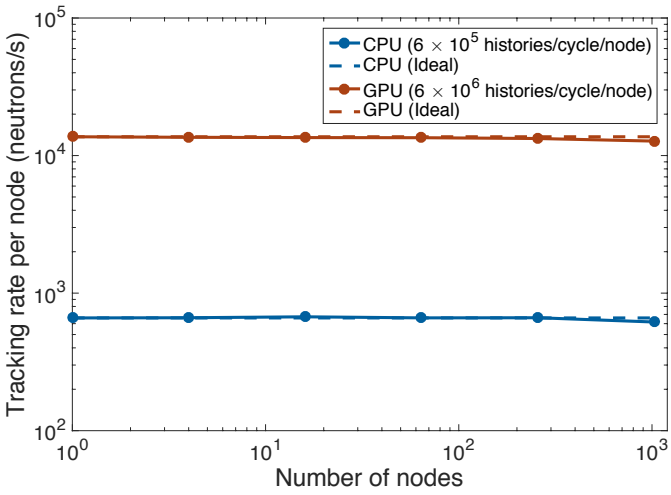


Figure 31: Shift weak scaling on Summit. Parallel efficiency at 1024 nodes is approximately 93%.

Table 32: Active cycle particle tracking rate (10^3 n/s) on K40 GPU at different occupancies for fresh fuel SMR core during active cycles.

	Algorithm		
Occupancy (%)	History-based	Event-based	Flattened event-based
12.5	3.7	3.4	8.2
25.0	–	5.8	13.3
37.5	–	–	14.5
50.0	–	–	16.9
62.5	–	–	18.0

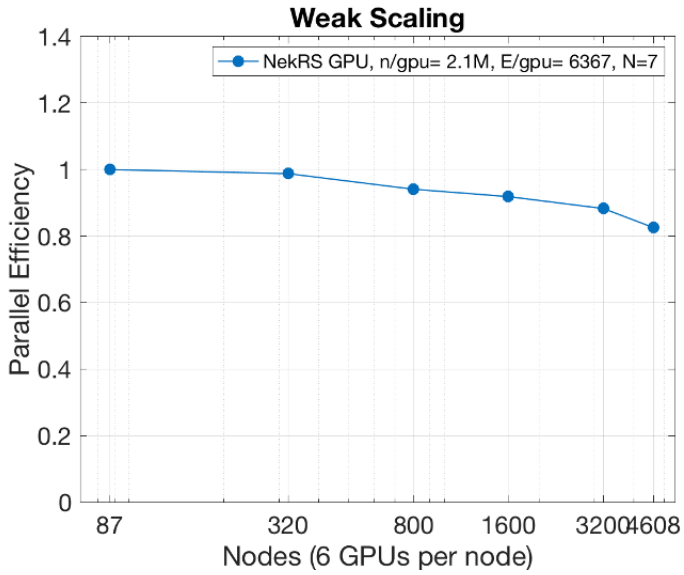


Figure 32: NekRS weak scaling on Summit. Parallel efficiency at 4608 nodes is slightly above 80 %.

machine remains above 80 %, and the strong scaling from 1810 to 4608 nodes is approximately 70 %. The degradation in strong scaling behavior beyond around 3620 nodes is due to an insufficient amount of parallel work to fully saturate all the GPUs on Summit. Much higher efficiency is expected to be achieved on a problem with more DOF.

Next Steps

The primary near-term effort for the ExaSMR project is to complete the initial ports of each physics code to its respective pre-exascale hardware. This process will enable an initial performance assessment on problems closely related to the ExaSMR challenge problem on the most representative available hardware and allow any components or kernels that are at an elevated risk to be identified with respect to KPP-1 FOM targets. Development and optimization efforts can then focus on the most critical areas. The completion of the code porting is expected to take significant time because it will likely require close interaction with hardware vendors and the potential resolution of bugs or issues in the vendor software components.

Additional effort will also target the continued development of the ENRICO multiphysics driver that will be used to performed the coupled-physics simulations necessary for ExaSMR's challenge problem. In particular, emphasis will be placed on enabling the NekRS and Shift GPU capabilities, allowing the first coupled-physics calculations with MC and CFD using GPU hardware. This work will result in the first FOM measurement from a multiphysics simulation.

4.4 MFIX-Exa

Carbon capture and storage (CCS) technologies (e.g., oxy-fuel combustion, chemical looping combustion, post-combustion capture systems) offer the most promising approaches for reducing CO_2 emissions from fossil fuel power plants. The large-scale commercial deployment of CO_2 capture technologies requires understanding of how to scale laboratory designs of multiphase flow reactors to industrial sizes. However, the direct scale-up of such reactors is unreliable, and the current approach requires building and testing physical systems at increasingly larger intermediate scales. The cost in dollars and development time of having to build and extensively test systems at multiple intermediate scales is prohibitive. Developing high-fidelity computational tools capable of simulating such systems to enable the design and optimization of emerging CCS technologies is critically important for controlling costs and reducing the risk of designs that do not meet performance standards. The development of such tools and the simulations required are impossible without exascale computations. This work specifically targets the scale-up of chemical looping reactors (CLRs) by using NETL's

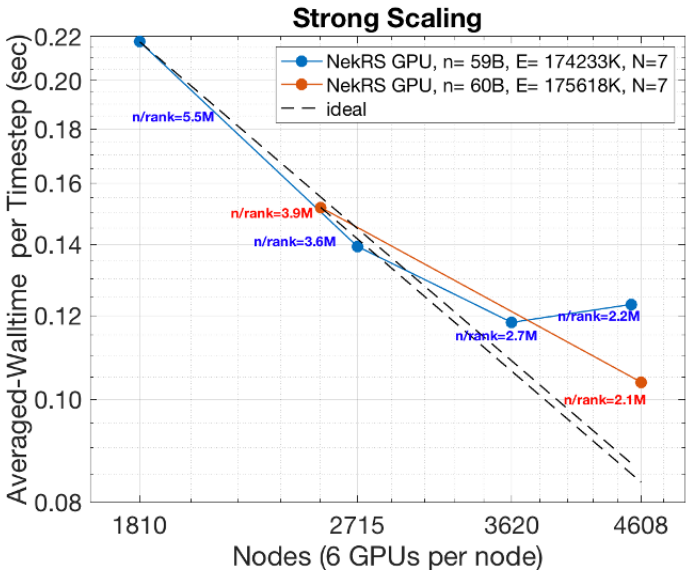


Figure 33: NekRS strong scaling on Summit.

50 kW CLR as a basis through the creation of MFIX-Exa, a scalable computational fluid dynamics-discrete element method model (CFD-DEM) code, which is the next generation of the highly successful NETL-based MFIX code.

CFD-DEM is an approach that allows individual particles (discrete element method (DEM) portion) to be tracked within a continuum fluid phase (CFD portion). To date, the focus of existing MFIX CFD-DEM efforts has been on validating and developing physical models in the context of a relatively basic computational framework. MFIX-Exa will integrate expertise in HPC directly with expertise in multiphase modeling and will outperform the existing MFIX by orders of magnitude.

4.4.1 MFIX-Exa: Science Challenge Problem Description

The challenge problem is a CFD-DEM simulation of NETL's 50 kW CLR, which contains 5 billion particles for a sufficiently long physical time so that exit gas compositions reach a pseudo-stationary state, enabling the evaluation of reactor performance. In chemical looping combustion, multiple reactors are used to separate combustion into two distinct processes: reduction and oxidation. In the fuel reactor, a solid oxygen carrier supplies oxygen for combustion and is reduced by the fuel. The reduced oxygen carrier is sent to the air reactor where it is regenerated to its oxidized state. The air reactor produces a hot, spent air stream that is used to create steam to drive a turbine for power generation. Then, the oxygen carrier is returned to the fuel reactor, restarting the reduction-oxidation cycle. The challenge problem requires the full-loop CLR geometry to be represented, covering all five flow regimes, including interphase momentum, mass, and energy transfer. The MFIX-Exa code is an exascale-enhanced version of the classic MFIX application that improves fidelity by employing DEM instead of the traditional low-order models. Challenge problem details are given in Table 33.

4.4.2 MFIX-Exa: KPP Stretch Goal

The stretch goal is to run the challenge problem simulation until the gas concentrations at the fuel-reactor exit reach a stationary state, a simulation-time that cannot be determined a priori. The initial conditions for the simulation will be determined from lower order simulations ("bootstrapping approach") to better approximate the solids inventories in the three main components of the CLR: the air reactor, fuel reactor, and loop seal. The cross section-averaged CO_2 mass fraction at the exit of the fuel reactor will be plotted as a function of time. The plot is expected to show a rapid change in the CO_2 mass fraction during an initial period. Subsequently, the CO_2 mass fraction is expected to fluctuate around a steady mean value. After

Table 33: MFIX-Exa challenge problem details.

Functional requirement	Minimum criteria
Physical phenomena and associated models	Low-Mach number fluid flow with multi-species, reactive transport coupled to discrete solid particle transport.
Numerical approach, algorithms	The CFD-DEM numerical method is based on MFIX-DEM. The continuity and momentum balance equations are solved for the gas phase, and the equations account for the volume occupied by particles and the momentum transfer with the particles. All the particles are tracked by solving the kinematic equation and the linear and angular momentum equations. The particle-particle collisions are modeled by using a soft-sphere approach, and the contact forces are modeled by using a spring-dashpot-slider model. The numerical accuracy of the code will be periodically established by conducting verification tests, such as those described in the milestone report ECP-MFIX-Exa-2017-12, which demonstrate a high degree of numerical accuracy. The numerical parameters and convergence criterion used in the verification tests will also be used for the challenge problem. Model validation is outside the scope of the current project and will not be conducted.
Simulation details: problem size, complexity, geometry, and so on	Geometry: The system geometry contains a fuel reactor, air reactor, cyclone, and loop seal. Components are connected via piping with conditions ranging from a dense granular flow (fuel reactor to air-reactor) to dilute transport (air-reactor to cyclone). The approximate dimensions of the fully assembled reactor are 4.2 m high, 1.1 m wide, and 0.22 m deep. Grid: A uniform grid size of 1 mm will be used for the entire domain. Because of the CLR's torus-like shape, it is anticipated that three-fourths of the domain will not contain active computational cells and thus will be removed to reduce memory usage. The domain can be fully covered by roughly 40,000 individual grids of 32^3 cells each; if only one-fourth of the domain is relevant, one should need closer to 10,000 grids or a total of roughly 330 million cells. Fluid: The fluid will be modeled as a mixture of methane (CH_4), hydrogen (H_2), steam (H_2O), carbon monoxide (CO), carbon dioxide (CO_2), oxygen (O_2), and nitrogen (N_2). There will be 12 unknowns per grid cell: velocity (3), pressure (1), temperature (1), and composition (7). Fluid density will be evaluated as function of temperature, pressure, and composition. Fluid volume fraction (void fraction) is computed from particle location and volume data by using a compact support kernel. Particles: The solids oxygen carrier will be modeled with approximately 5 billion monodisperse spherical particles with a 200 µm diameter. Particles will be treated as a mixture of hematite (Fe_2O_3) and Wüstite (FeO). There will be 12 unknowns per particle: position (3), translational and rotational velocities (6), temperature, and composition (3). The main chemical reactions: • air reactor: $2FeO + \frac{1}{2}O_2 \rightarrow Fe_2O_3$ and • fuel Reactor: $4Fe_2O_3 + CH_4 \rightarrow 8FeO + CO_2 + 2H_2O$.
Demonstration calculation requirements	Initial conditions: Minimizing the initial transient time from the guessed initial condition to the fully developed condition is a considerable challenge in reacting CFD-DEM. Lower fidelity and/or coarser simulations will be used to establish reasonable initial conditions for the challenge problem, including fluid and particle flow patterns, temperatures, and chemical compositions. The full science simulation will be run for a sufficiently long physical time so that exit gas compositions reach a stationary state, enabling the evaluation of reactor performance. Boundary conditions: A mixture of CH_4 and N_2 will be used to fluidize the fuel reactor, and a mixture of O_2 and N_2 will be used to fluidize the air reactor. N_2 will be used to fluidize the loop-seal and move gas in the standpipe and L-valve to facilitate solids movement. An operating temperature of 1100 K is targeted for the system, and outlets in the cyclone, loop seal, and fuel reactor will vent to atmospheric pressure. No heat loss through the reactor walls will be considered.
Resource requirements to run demonstration calculation	Using the approach developed in FY21 to bootstrap physically relevant initial conditions, the team anticipates only needing to take 10 time steps to approximate the time necessary for a full science simulation at realistic operating conditions in which the particle distribution throughout the CLR is relatively constant in time. The time required for such simulations, estimated by extrapolating the time to solution for existing simulations, will be 2 h at full system.

disregarding the initial period, the CO_2 mass fraction will be time-averaged over successive intervals equal to the gas residence time in the fuel reactor. If a few successive time-averaged values have less than a 5 % difference between them, then the simulation will be considered to have reached a stationary state, and the stretch goal will be considered to have been achieved.

4.4.3 MFIX-Exa: Capability Plan

The primary technical capabilities needed to execute the challenge problem include the ability to:

1. solve the coupled mass, momentum, and energy equations for the fluid and particles in the full-loop reactor geometry;
2. bootstrap the initial conditions from a lower resolution run to the final high-resolution geometry; and
3. run the compute-intensive kernels in the MFIX-Exa code on the GPU accelerators.

To date, the team has the ability to do the following.

1. Solve the coupled momentum equations for the fluid and particles in a simplified approximation to the full-loop reactor geometry: The team replaced the original SIMPLE algorithm with an approximate projection method and replaced the original BiCGStab linear solver with geometric multigrid supplied by AMReX. A novel level set method for efficiently calculating particle-wall collisions was developed and implemented.
2. Run the MFIX-Exa code at various scales and resolutions: The initial 1:1 mapping of grids to MPI ranks was replaced by a more general AMReX-based gridding strategy that allows much greater flexibility in grid creation and removal.
3. Off-load several key particle-intensive kernels from CPUs to GPUs: The original MFIX code could not have been extended to GPUs.

The development schedule through FY23 is as follows.

- FY20:
  - Implement an MFIX-PIC solids model to be used as the lower resolution solids model to be deployed in the initial condition bootstrapping approach.
  - Incorporate heat and mass transfer between particles and fluid due to chemical reactions for both DEM and particle-in-cell (PIC) solids models.
- FY21:
  - Incorporate an initial condition bootstrapping approach by using the PIC solids model and develop a methodology to switch between the two solids models.
  - Incorporate a generalized geometry generation approach to allow greater flexibility in modifying the reactor geometry to account for changes in the physical system.
- FY22:
  - Assess a mixed solids model for a dense granular region as a risk mitigation strategy. If the mixed solids model is successful, then integrate it into the MFIX-Exa algorithm.
  - Optimize on the prototype for target architectures (e.g., replacing CUDA regions if CUDA is unavailable on the target machine).
- FY23:
  - Optimize for target architecture, including fine-tuning the load balancing strategy to reflect the realities of the new architecture.

Table 34: Inclusive time and percent of the total runtime for particle kernels obtained from an MFIX-Exa single GPU analysis on Summit.

Kernel name	Time	Percent
`calc_particle_collisions()`	49.12	29.84
`update_neighbors()` *(AMReX routines)*	23.34	14.18
`update_particle_velocity_and_position()`	13.43	8.16
`update_particle_species()`	5.52	3.35
`update_particle_enthalpy()`	3.01	1.83
`substep_init()`	1.09	0.66
`calc_wall_collisions()`	0.91	0.55

4.4.4 MFIX-Exa: Progress on Early and Pre-Exascale Hardware

Performance on Summit

A single GPU analysis and weak-scaling study were executed to assess the performance of MFIX-Exa on Summit. For both tests, the fluid was treated as a multicomponent ideal gas that comprises seven species: CH_4, H_2, H_2O, CO, CO_2, O_2, and N_2. Particles were treated as a mixture of Fe_2O_3 and FeO. Density, enthalpy, and species mass were advected for both the fluid and particles, and an open-system constraint was used so that the evolution of species densities and enthalpy is consistent with the equation of state (EOS). Particles are assumed to be incompressible with fixed volumes. The fluid and particles were fully coupled though interphase momentum, energy, and mass transfer. Three chemical reactions with constant reaction rates were used to represent the reduction of Fe_2O_3 to FeO:

- $Fe_2O_3 + H_2 \rightarrow 2FeO + 2H_2O$,
- $Fe_2O_3 + CO \rightarrow 2FeO + CO_2$, and
- $4Fe_2O_3 + CH_4 \rightarrow 8FeO + CO_2 + 2H_2O$.

For the single GPU test, a cylindrical geometry was used that is similar in aspect ratio to the CLR fuel reactor geometry. The domain was decomposed with a $256 \times 64 \times 64$ fluid mesh with two-thirds of the computational cells being covered (i.e., excluded from the solution because they are "outside" the embedded boundary). The fluid was advanced for 50 fluid time steps with 50 sub-cycles per fluid step (2500 total steps) used to evolve the particles. The fluid advance accounted for 37 % of the total solve time with most of the time spent in the linear solves for the nodal and MAC projections. The time spent coupling the fluid and particle solvers (e.g., deposition of particle volume, calculation and deposition of interphase transfer terms) accounted for only 3 % of the total runtime. Lastly, the particle advance consumed the remaining 60 % of total runtime. The inclusive time and percent of total runtime for particle kernels is provided in Table 34 where the particle-particle collision detection/force calculation and an update of particle velocity and position are shown as the most expensive MFIX-Exa kernels. A roofline analysis, shown in Fig. 34, indicates that the particle kernels are memory-bound, which explains why the update for position and velocity consumes a significant portion of the total runtime. Presently, we are investigating changes to the particle data structure (e.g., converting from an array-of-structures to a structure-of-arrays) to improve memory access.

A weak-scaling study was conducted by using a simple box geometry to ensure an even load distribution. Specifically, a doubly periodic box with a mass inflow and pressure outflow was constructed. The fluid mesh size was set to 2.5× the particle diameter, and eight particles were placed in each cell. The domain was decomposed into uniform grids of 64^3 fluid cells and approximately 2.1 million particles. There was a 1:1 grid-to-MPI mapping, and the problem is weak-scaled in size by successively doubling the number of grids in each direction. The fluid and particle compositions were as previously described. The fluid was advanced for 10 fluid time steps with 50 particle sub-cycles per fluid step (500 steps total). The total time for the fluid advance, particle advance, and coupling was averaged overall all 10 steps.

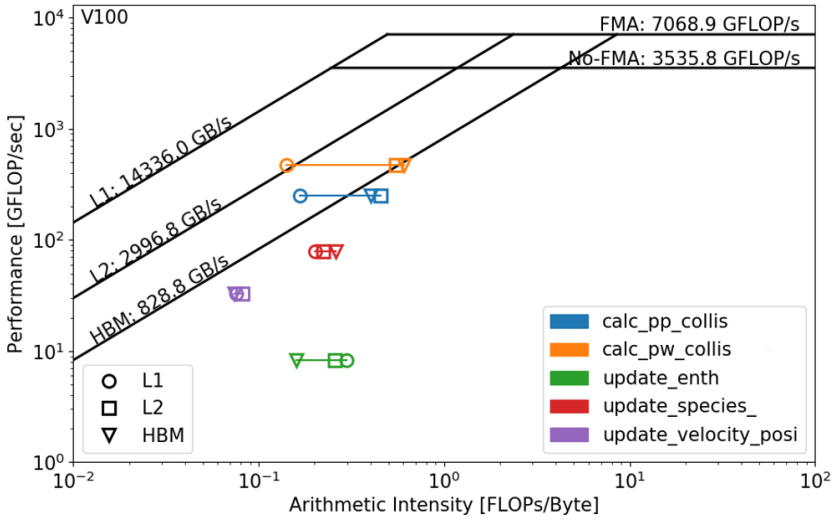


Figure 34: Roofline plot of key MFIX-Exa particle kernels.

The scaling efficiency and average time per step are shown in Fig. 35. As with the single GPU study, most of the fluid solve is spent in linear solves for the nodal and MAC projections. The study was repeated with HYPRE as the multigrid bottom solve; however, no improvement in performance was observed. The particle solve showed little reduction in efficiency after 32 GPU cases, at which point the problem is sufficiently large so that no grid has the same MPI rank for any of the six faces. Unlike the single GPU study, the particle solve is dominated (∼90 %) by updating neighbor particles. Specifically, the particle-particle collision algorithm requires that the position and velocity of "ghosted" particles from adjacent grids be updated after each particle sub-cycle, which is significantly more costly than calculating collisions. Lastly, the efficiency of the fluid-particle coupling follows the efficiency of the fluid solve. However, the computing accounts for less than 4 % of the total time step.

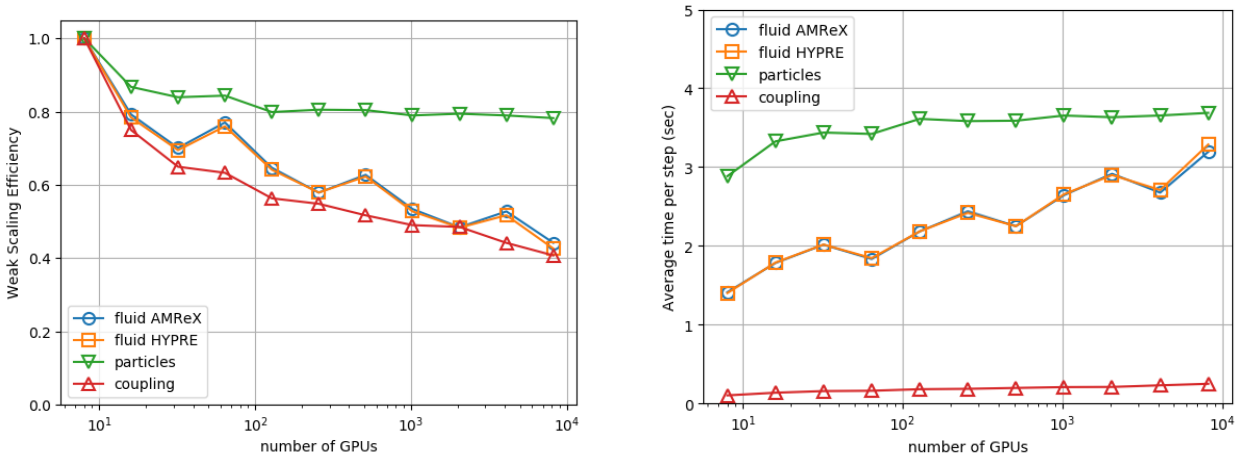
Finally, a variation on the weak scaling was conducted to compare CPU and GPU performance. Shown in Fig. 36, the GPU version is approximately 1.8× faster than the CPU version at the largest Summit node count.

Next Steps

We are currently investigating changes to the particle data structure to obtain better performance on GPU. Specifically, we currently use an array of structures to store particle data, whereas a structure of arrays might allow for better cache reuse. Along these lines, we are looking at sorting particle neighbor lists and different loop constructs (e.g., loop over collisions as opposed to looping over particles) to reduce data movement.

4.5 WDMApp

The Whole Device Model Application project (WDMApp) strives to develop a high-fidelity model of magnetically confined fusion plasmas, which is urgently needed to plan experiments on ITER [24] and optimize the design of future next-step fusion facilities. These devices will operate in high-fusion-gain physics regimes not achieved by any of the current or past experiments, making advanced and predictive numerical simulation the best tool for the task. WDMApp is focused on building the main driver and coupling framework for the more complete Whole Device Model (WDM), the ultimate goal being to complete a comprehensive computational suite that will include all the important physics components required to simulate a magnetically confined fusion reactor. The main driver for the WDM will be coupling two advanced and highly scalable gyrokinetic codes, XGC and GENE. XGC is a PIC code optimized for the treating the edge plasma, and GENE is a continuum code optimized for the core plasma. WDMApp aims to take advantage of the complementary nature of these two applications to build the most advanced and efficient



(a) **(b)**

Figure 35: The scaling efficiency and average time per step for the MFIX-Exa weak scaling study on Summit GPUs.

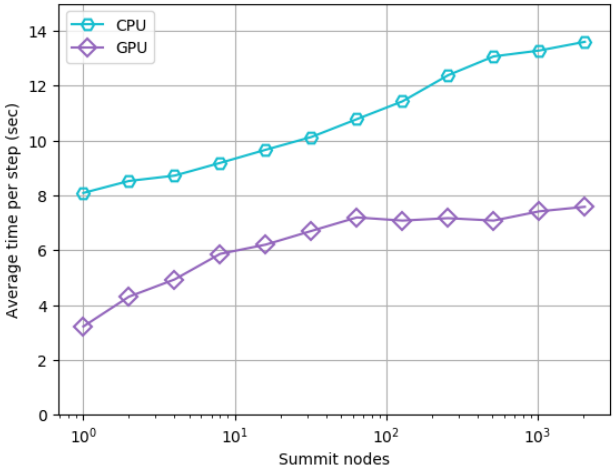


Figure 36: MFIX-Exa comparison of CPU and GPU weak scalings.

Table 35: WDMApp challenge problem details.

Functional requirement	Minimum criteria
Physical phenomena and associated models	5D gyrokinetic plasma model with Coulomb collisional kernel.
Numerical approach, algorithms	Structured grid, continuum solver with spectral and finite-difference spatial schemes in the plasma core coupled to unstructured grid, particle-in-cell solver with finite element and finite-difference schemes in the edge region.
Simulation details: problem size, complexity, geometry	Minimum problem size of 100 million solver vertices covering the whole core and edge region. Minimum time step of one-tenth the ion sound wave period. Minimum number of particles is 1 trillion in XGC.
Demonstration calculation requirements	2500 time steps.
Resource requirements to run demonstration calculation	3 h to run 2500 time steps on 100% of exascale computer.

whole device kinetic transport kernel for the WDM. One main thrust of the project is the coupling framework End-to-End Framework for Fusion Integrated Simulation (EFFIS) 2.0, which will be further developed for the exascale and optimized for coupling most of the physics modules that will be incorporated in the WDM. The current MPI+X implemented in the main GENE and XGC applications will be enhanced with communication-avoiding methods, task-based parallelism, in situ analysis with resources for load optimization workflows, and deep memory hierarchy-aware algorithms.Alternate code-couplings will also be considered for risk mitigation (i.e., the PIC code GEM in the core region).

The resulting exascale application will be unique in its computational capabilities and could have a transformational impact on fusion science, such as studying a much larger and more realistic range of dimensionless plasma parameters than ever before and the rich spectrum of kinetic micro-instabilities that control the quality of energy confinement in a toroidal plasma—including tokamaks and stellarators—with the core and the edge plasma strongly coupled at a fundamental kinetic level based on the gyrokinetic equations.

4.5.1 WDMApp: Science Challenge Problem Description

The exascale science challenge problem is the high-fidelity simulation of whole-device burning plasmas applicable to a high-confinement ("H-mode") advanced tokamak regime, specifically an ITER steady-state plasma that aims to attain a tenfold energy gain. The physics objective is to predict two of the most important indicators for energy confinement in the H-mode: the plasma pressure "pedestal" height and shape. Realizing the H-mode with high edge plasma pressure and mild pedestal gradient is critical to the performance and success of ITER. Fusion burn efficiency is virtually determined by the height of the pressure pedestal at the edge. The team's strategy will be to use WDMApp, which is focused on coupling the continuum code GENE and/or GEM in the core region and the PIC code XGC at the edge.

The targeted minimum requirement for accuracy is 5% relative error. Because of the intrinsically turbulent nature of some important fusion plasma observables, higher accuracy is neither cost-effective nor realistic in most experiments or simulations. The computational hardware size target will be 100% of the exascale platforms on which the FOM will be compared against the baseline FOM established on 50% of Titan. The minimum physics problem size will be 100 million solver vertices, and the minimal time step will be one-tenth of the ion sound wave period. More detailed information is provided in Table 35 on the minimum criteria for the FOM runs.

4.5.2 WDMApp: KPP Stretch Goal

The stretch goal for the project is to carry out core-edge coupled electrostatic turbulence simulations to model gyro-kinetic ions and drift-kinetic electrons in ITER geometry on an exascale facility. In addition to the

FOM enhancement produced by the hardware upgrade to the exascale facility, the KPP stretch goal would realize substantial FOM achievement through the algorithmic improvement of coupling two gyro-kinetic codes mostly by exchanging fluid moment data augmented by the infrequent exchange of probability distribution function (PDF) data. Further improvements are expected from an asynchronous electron push in GPU and improvements in code-coupling efficiency and load balancing.

4.5.3 WDMApp: Figure of Merit

The science challenge problem for WDMApp pertains to a realistic ITER plasma. The FOM is defined as:

$$\text{FOM} = \frac{N_m N_t N_p}{\text{wallclock time}} \times 10^{-10}\,, \tag{11}$$

where N_m is the number of solver vertices, N_t is the number of time steps in specified wallclock time (60 s), and N_p is the number of particles per mesh vertex in the edge region. The baseline FOM is 1.76 on Titan based on a PDF data exchange.

FOM Update

Current FOM measurements on GENE-XGC were calculated on Summit with kinetic electrons under the following conditions.

- The circular model-plasma run was performed on 518 nodes, 512 of which were used for XGC and six of which were used for GENE with XGC's GPU capabilities enabled.
- XGC was run with six MPI processes per node, one GPU per MPI process, and 28 OpenMP threads per MPI process. GENE was run with 42 MPI processes per node.
- The number of whole domain solver vertices used was $N_m = 9{,}640{,}480$, the number of particles in XGC per vertex was $N_p = 8922$, and the wall-time per time step was 61.4 s.

These numbers give a value of FOM = 8.4. Extrapolating to full Summit (4608 nodes) gives FOM = 74.8. Using the baseline value of 1.76 from full Titan, the updated GENE-XGC FOM extrapolated to full Summit is 42.5.

We have also performed FOM updates for GEM-XGC with kinetic electrons and the following conditions.

- A realistic DIII-D plasma run was performed on 1056 nodes, 960 of which were used for XGC and 96 of which were used for GEM.
- XGC and GEM were both run with six MPI processes per node, one GPU per MPI process, and 28 OpenMP threads per MPI process.
- The number of whole domain vertices used was $N_m = 12{,}072{,}384$, the number of particles per vertex was $N_p = 10{,}020$ for XGC ($N_p = 512$ for GEM), and the wall time per time step was 51.1 s.

Using these numbers, the FOM for GEM-XGC extrapolated to full Summit is 35.1.

4.5.4 WDMApp: Progress on Early and Pre-Exascale Hardware

Performance on Summit

The XGC code base was converted to C++ and was fully integrated with Cabana/Kokkos, which allows more efficient GPU porting. The following kernels were converted to C++ and ported to GPUs:

- electron push, converted to C++, used Cabana/Kokkos;
- ion and electron scatters, partially converted to C++, used Cabana/Kokkos; and
- collisions, converted to C++, used Kokkos.

The C++ and Cabana/Kokkos conversion enabled more pre-exascale performance gains on Summit.

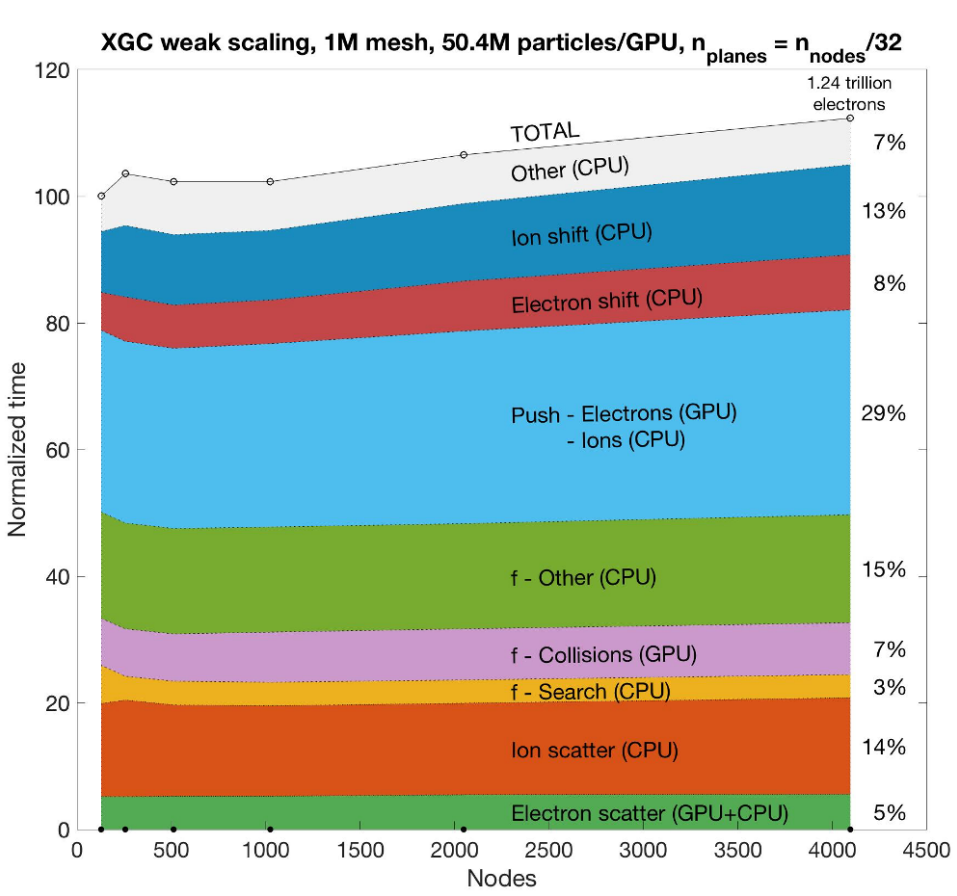


Figure 37: Cabana/Kokkos electron push on Summit.

- Whole code (CPU + GPU) is 12 % faster than 1 year ago for the entire production code.
- The C++ Cabana/Kokkos electron push is 28 % faster than the Fortran Cabana/Kokkos electron push in V100.

The performance of the Cabana/Kokkos electron push is shown in Fig. 37.

The GENE and GEM plasma core codes were also ported and tested on Summit. Figure 38 shows a weak scaling on Summit and compares performance on CPUs and GPUs. We observed a performance degradation as the number of nodes increases so that the speedup goes from 15 to 6.2× as the number of nodes varies from 1 to 512 (6–3072 GPUs). However, because GENE will be using <40 nodes but XGC uses >4000 Summit nodes, the speedup degradation at high node count should not be an issue.

GEM performance was also measured on Summit. As shown in Fig. 39, GEM weak-scales well to 1024 Summit nodes. This performance is sufficient for a coupled GEM-XGC simulation in which GEM occupies ∼10 % of the total nodes.

4.6 WarpX

Particle accelerators are a vital part of the DOE-supported infrastructure of discovery science and have a broad and critical range of applications in industry, security, energy, the environment, and medicine. Particle accelerators are used in many areas of fundamental research—such as elementary physics, nuclear physics, material science, chemistry, and biology—and even play a role in astrophysics and cosmology. Thirty percent of all Nobel prizes in physics since 1939—and four of the last 14 Nobel prizes in chemistry—have been enabled by particle accelerators. Beyond fundamental research, the investments in accelerator technologies for discovery science have led to the development of tens of thousands of particle accelerators for various

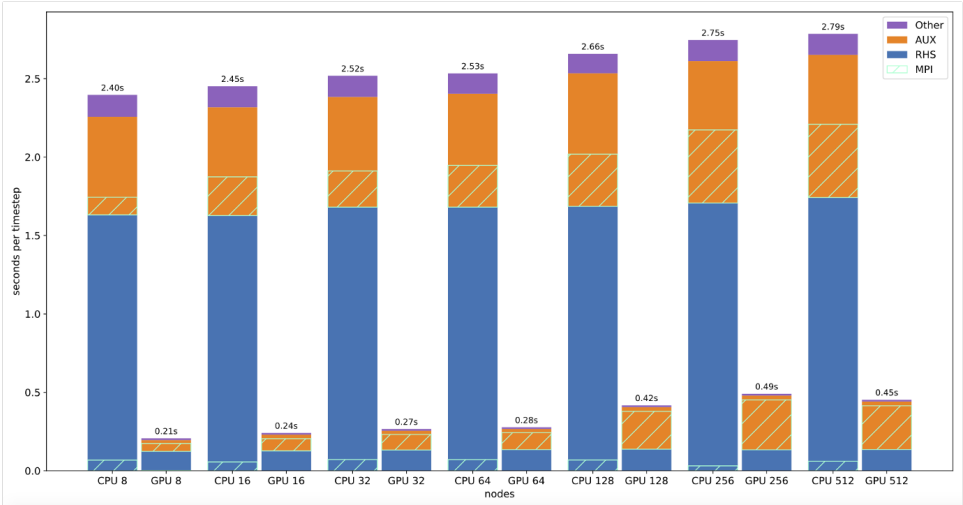


Figure 38: Weak scaling comparison of GENE CPU/GPU on Summit.

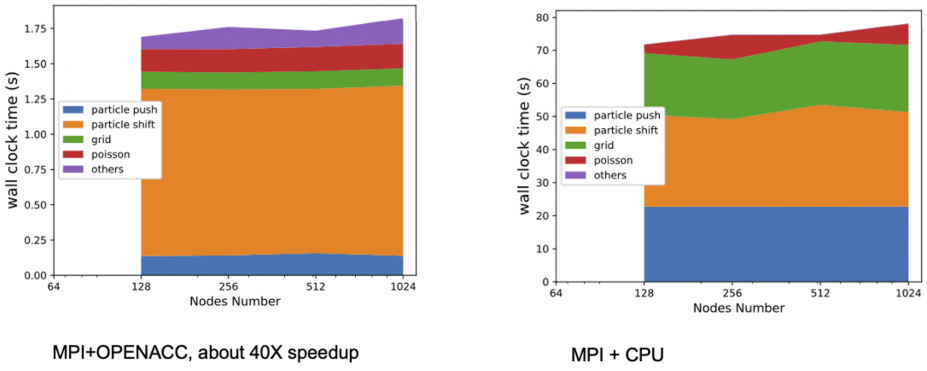


Figure 39: Weak-scaling comparison of GEM on Summit.

applications that impact lives, such as sterilizing food and toxic waste, implanting ions in semiconductors, treating cancer, and developing new drugs.

For most applications, the size and cost of the accelerators are limiting factors that can significantly impact the funding of projects or the adoption of solutions. Among the candidate new technologies for compact accelerators, the advent of plasma-based particle accelerators stands apart as a prime game-changing technology. The development of plasma-based accelerators depends critically on high-performance, high-fidelity modeling to capture the full complexity of the acceleration processes that develop over a large range of space and timescales. However, these simulations are extremely computationally intensive due to the need to resolve the evolution of a driver (laser or particle beam) and an accelerated beam into a structure that is orders of magnitude longer and wider than the accelerated beam. Studies of various effects—including the injection, emittance transport, beam loading, tailoring of the plasma channel, and tolerance to nonideal effects (e.g., jitter, asymmetries) that are needed for the design of high-energy colliders—will necessitate a series of tens or hundreds of runs. This will require speedups that are orders of magnitude over the present state of the art, which will be obtained by combining the power of exascale computing with the most advanced computational techniques. This project will combine the PIC code Warp technology and the AMR framework AMReX (§ 8.4) into a new code (WarpX) and port the software to exascale platforms. WarpX will incorporate the most advanced algorithms that have been developed and validated by the lead teams, including the optimal Lorentz boosted frame approach, scalable spectral electromagnetic solvers, and mitigation methods for the numerical Cherenkov instability. To ensure speed and scalability, WarpX will take advantage of the latest features that the team has developed in portable vectorization algorithms, hierarchical parallelism, and GPU programming, as well as AMReX's dynamic gridding capabilities, to load balance the combined computational work associated with the particles and mesh.

On exascale supercomputers, the new software will enable the exploration of outstanding questions in the physics of the transport and acceleration of particle beams in long chains of plasma channels, such as beam quality preservation, hosing, and beam-breakup instabilities. These new breeds of virtual experiments, which are impossible with present technologies, will bring huge savings in research and lead to the design of a plasma-based collider and even bigger savings by enabling the characterization of the accelerator before it is built.

4.6.1 WarpX: Science Challenge Problem Description

The exascale challenge problem is the modeling of a chain of tens of plasma acceleration stages. Realizing such an ambitious target is essential for the longer range goal of designing a single- or multi-teraelectron volt electron-positron high-energy collider based on plasma acceleration technology. The WarpX application uses AMReX for AMR and employs PIC methodology to solve the dynamic Maxwell equations to model the accelerator system.

The minimum completion criteria are designed to demonstrate that the project is on track toward modeling multi-teraelectron volt high-energy physics colliders based on tens to hundreds of plasma-based accelerator stages. The main goal is to enable the modeling of an increasing number of consecutive stages to reach higher final energy and to increase the precision of the simulations by performing simulations at higher resolutions in a reasonable clock time. The goals are as follows:

- achieve a total plasma accelerator energy gain of at least 100 GeV by using five accelerator stages or more and
- achieve a maximum wallclock time that is the same as the baseline.

Details on the challenge problem are given in Table 36.

4.6.2 WarpX: KPP Stretch Goal

The project goal is to demonstrate the capability to perform the modeling of tens of consecutive multi-gigaelectron-volt stages. For a multi-teraelectron volt collider design, hundreds to thousands of stages will be necessary. The KPP stretch goal is thus naturally to reach an FOM that is as high as possible without the constraint limit on boost from algorithms ($B_A < 5$). Therefore, the stretch goal is to demonstrate the capability to model as many consecutive multi-gigaelectron-volt stages as possible.

Table 36: WarpX challenge problem details.

Functional requirement	Minimum criteria
Physical phenomena and associated models	Dynamic Maxwell equations in the presence of a time-varying electromagnetic field with laser-driven and charged-particle source terms.
Numerical approach, algorithms	PIC using finite-difference (i.e., FDTD) and/or pseudo-spectral (i.e., FFT-based) analytical time domain temporal discretization with FFT-based field solve on AMR grids with solution in Lorentz-boosted frame.
Simulation details: problem size, complexity, geometry, and so on	Minimum of 10^{11} grid cells and 2×10^{11} macroparticles. Minimum number of five chained accelerator stages. Laser driver on the order of 1 PW peak.
Demonstration calculation requirements	Run a minimum of 100 time steps so that performance is accurately measured by using a preloaded plasma column. Measure FOM for the FDTD and FFT-based solvers.
Resource requirements to run demonstration calculation	2 h on 100% of the machine.

4.6.3 WarpX: Figure of Merit

The FOM for WarpX is defined as:

$$\text{FOM} = \frac{(\alpha N_c + \beta N_p) N_t B_A}{\text{wallclock time}} \,, \tag{12}$$

where N_c is the number of grid cells; N_p is the number of particles; α, β are the relative cost of grid push and particle push; and N_t is the number of time steps. The term *Boost from Algorithms* (B_A) includes additional boost coming from algorithm developments and is constrained by $B_A \leq 5$.

The FOM without the B_A provides the "raw" speed from running the same problem faster or at higher resolution (more grid cells, particles, and time step) on a newer and bigger machine in the same time (based on strong/weak scaling). Beyond this, speedup comes from (1) a lower number of grid points and particles using AMR and (2) a lower number of time steps using a novel large time step (LTS) algorithm. Hence, B_A is the product of two possible algorithmic boosts:

$$B_A = B_{\text{AMR}} \times B_{\text{LTS}} \,, \tag{13}$$

where B_{AMR} is the ratio of the number of cells in the simulation at highest resolution without AMR and the number of cells with AMR, and B_{LTS} is the Courant condition of the novel LTS solver divided by the Courant condition of the standard solver.

The current baseline measurement is $\sim 2.2 \times 10^{10}$ on 6625 compute nodes of Cori (FOM scaled to the full machine).

FOM Update

The latest FOM measurement, which was performed on July 2, 2020, is $\sim 2.5 \times 10^{12}$ on 4263 compute nodes of Summit (FOM scaled to the full machine), giving an improvement of ~114 over the baseline. Table 37 summarizes the FOM progress to date.

4.6.4 WarpX: Progress on Early and Pre-Exascale Hardware

Performance on Summit

The following section summarizes some key lessons learned from our experience with porting WarpX to Summit. These fall under three main areas: issues relating to memory management and overall footprint, issues relating to parallel communication, and the importance of cache utilization on GPU platforms.

Table 37: Progress in the FOM measurement over time. The code is either the original Warp code (baseline) or WarpX. The date is the date when the measurement was taken (month/year). The machine is which computer was used to make the measurement. The nodes are how many nodes on which the measurement was performed; there are 9,668 KNL nodes on Cori and 4,608 nodes on Summit. The FOMs were extrapolated from the number of nodes that the measurement was taken on to the full machine.

Code	Date	Machine	Nodes	FOM
Warp	3/2019	Cori (KNL)	6625	2.2e10
WarpX	3/2019	Cori (KNL)	6625	1.0e11
WarpX	6/2019	Summit	32	8.6e11
WarpX	6/2019	Summit	1000	7.8e11
WarpX	9/2019	Summit	2560	6.8e11
WarpX	1/2020	Summit	2560	1.0e12
WarpX	2/2020	Summit	4263	1.2e12
WarpX	6/2020	Summit	4263	1.4e12
WarpX	7/2020	Summit	4263	2.5e12

Memory Optimization

With the trend toward GPU computing, the importance of optimizing codes for memory consumption has increased. If we consider only device memory, then each Summit node has six NVIDIA V100 GPUs with 16 GB of memory each for a total of only 440 TB, which is substantially less than Cori phase II. This means that if users want to run in a mode in which the problem fits entirely on the GPUs—which is desirable considering the performance penalties associated with frequent host/device data transfers—then users cannot actually run as big of a problem on Summit as they could on Cori. This makes reducing the memory footprint of a simulation code very impactful in terms of enabling production-level science calculations.

Reducing the memory footprint can also have performance implications. Originally in WarpX, every particle stored persistent values for the electric and magnetic fields interpolated to the particle's position. In addition to the storage overhead, these values must be communicated every time particles change MPI domains and shuffled around in memory every time particles are sorted (Section 4.6.4). Additionally, if the performance of a GPU kernel is memory-bound, meaning its performance is limited by the rate at which data can be transferred from main memory to the streaming multiprocessors on the GPUs, then increasing the arithmetic intensity of those kernels by streaming less data and recomputing values on the fly can improve their overall performance.

Recently, WarpX removed the persistent electric and magnetic fields at the particle positions in favor of regathering these values inside GPU kernels as they are needed. For this, the field gathering and particle pushing kernels were fused together in the PIC loop, resulting in fewer data that must be streamed to the processors in a given time step. In addition to reducing the memory footprint by a factor of ∼1.6, this also led to a ∼25 % speedup in the overall runtime on several key benchmarks. When the field values at the particle positions are needed more than once in a step—for example, when modeling additional effects, such as ionization, or using certain diagnostics—the gather operation is performed multiple times.

Finally, we are currently exploring other means of reducing the overall memory footprint of WarpX, including exploiting single/mixed precision and employing compression.

Memory Arenas

Dynamic memory allocation is many times more expensive on GPU than CPU architectures. This fact combined with common programming patterns involving temporary variables can lead to drastic performance penalties on GPU systems. For example, consider the code in Listing 1. This snippet demonstrates how to loop over mesh data via the AMReX data structures. The `MFIter` object instructs each MPI rank to loop over the grids it owns. For each grid, we resize a temporary scratch space called `tmp`, then launch a `ParallelFor` kernel to do some calculations with it. The `Elixir` is not essential to the point, but it keeps the scratch space alive in memory until the kernel is finished working with it; this is needed due to the asynchronous nature of

GPU kernel launches. If every call to `resize` the buffer ended up triggering `cudaMalloc` and `cudaFree` calls, this could easily end up becoming the dominant cost of this routine. Another are in which this occurs is in out-of-place sorting and partitioning operations, which require a temporary buffer in which to store the result.

One way to mitigate this is to refactor application codes to keep temporary buffers alive in memory instead of letting them go out of scope. However, this is error-prone and labor-intensive. Instead, AMReX provides several memory arena classes, which allocate memory in large chunks and distribute pieces of it as the application runs. Thus, although WarpX frequently uses temporary variables, there are no calls to `cudaMalloc` or `cudaFree` during most time steps.

These arenas have several different options for managing memory fragmentation; currently, the default in AMReX is to use a "first fit" strategy. AMReX provides memory arenas that use host, device, pinned, and managed memory. WarpX uses these arenas for all of its mesh and particle data structures. By default, when running on NVIDIA GPUs, WarpX places all of its core data in managed memory.

Listing 1: Example of `ParallelFor`. This code can be compiled to run on CPU with OpenMP or GPU with CUDA, HIP, or DPC++.

```
FArrayBox tmp;
for (MFIter mfi(mf); mfi.isValid(); ++mfi)
{
  const Box& bx = mf.tilebox();
  tmp.resize(bx);
  Elixir eli = tmp.elixir();
  auto const& tmp_arr = tmp.array();
  amrex::ParallelFor(bx,
  [=] AMREX_GPU_DEVICE (int i, int j, int k) noexcept
  {
      compute_tmp(i, j, k, tmp_arr);
  })
}
```

Communication Optimization

Once the initial port of WarpX to NVIDIA GPUs was complete, the initial experience was that compute kernels, such as current deposition and field gathering, were much faster on V100 hardware than on KNL. However, this was not true for the AMReX parallel communication routines. The primary reason for this was that the parallel communication routines involved many small "copy on intersection" routines between neighboring boxes, especially when packing and unpacking MPI send and receive buffers. These operations involved little to no computation but launched many small kernels that packed and unpacked data buffers. Thus, the dominant cost in these routines was the latency associated with the kernel launches, which could be fused into a fewer number. After optimization, each MPI rank makes only one kernel launch to pack and unpack its MPI buffers, which led to greatly improved performance on Summit.

Cache utilization

As with CPU-based many-core architectures, rearranging computations so that they properly exploit the memory hierarchy can result in significant performance increases on V100 GPUs. This section discusses a case study on how periodic sorting of particle data so that it is processed in a cache-friendly way can greatly improve the performance of PIC operations, such as field gathering and current deposition on V100. First, we will briefly describe the current deposition algorithm we use and how it differs between CPU and GPU runs.

Current Deposition In PIC codes, most operations are straightforward to parallelize since particles can be threaded over and processed independently without needing to worry about potential race conditions.However, charge and current deposition operations require special consideration since there is the potential for collisions when threading over particles because multiple threads might attempt to update the same cell simultaneously.

In WarpX, our approach to concurrent scatter operations in particle deposition kernels varies, depending

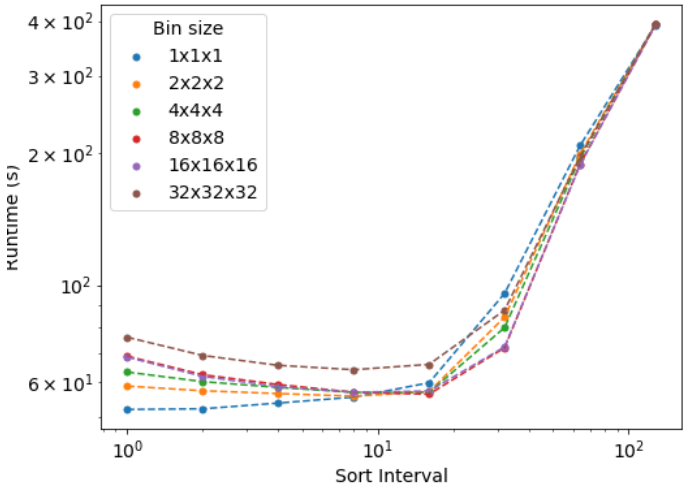


Figure 40: The effect of sorting the interval (i.e., sorting every N time steps) and sorting the bin size on the overall performance on a uniform plasma benchmark. The x-axis shows the sorting interval, and the y-axis shows the overall time to take 100 steps, including the cost of the sorting. A sorting interval >100 means that the particles are never resorted during the run.

on whether we are running with OpenMP or CUDA/HIP/DPC++ as the parallel back end. With OpenMP, the particles on a grid are sorted onto smaller subregions called *tiles*. OpenMP threads map to tiles, which begin processing them simultaneously. Each OpenMP thread deposits particles onto its own private deposition buffer with enough cells to capture the support of all the particles on the tile. There is no need for atomics at this stage since each thread has its own buffer. After deposition onto the buffer is complete, the buffer values are atomically added to the values for the full grid by using atomic writes. Thus, atomics are only needed on a per-cell basis, not a per-particle basis. However, when running on GPUs, we perform atomics write directly to global memory for each particle. Along with periodic sorting, this is sufficient to achieve good performance on NVIDIA V100 GPUs.

Particle Sorting Sorting the particles by their spatial locations periodically on each grid so that particles that are near each other in memory also interact with cells that are near each other in memory exploits the memory hierarchy on the GPUs more effectively than processing them in an unordered fashion. This is particularly true when particles are moving with high velocities and frequently change cells. In that case, even if particles are sorted at a particular time, they will rapidly become disordered, leading to significant performance degradation in the particle-mesh operations.

We differentiate between binning, which computes a permutation array that assigns particle indices to cells with user-defined bin size, and sorting, which uses this permutation array to actually reorder the particle data in memory. Cache utilization requires full sorting, but for many operations, knowing the cell-sorted indices is sufficient. AMReX provides a GPU-capable implementation of the counting sort algorithm that can be used to perform both of these operations. Internally, it is built by using a GPU implementation of parallel prefix sum, which is based on Merrill and Garland [26] and works on NVIDIA AMD, and Intel GPUs.

In addition to the presented cache-utilization optimization, sorting and/or binning particles is needed to model particle-particle interactions. By default, the PIC method only models particle-mesh interaction and mesh updates. WarpX implements binary collisions, which depend on a prior binning of neighboring particles, to address various applications in accelerator and beam physics.

Figure 40 shows the results of a parameter study in which the bin size and sorting interval were varied. For example, a bin size of $2 \times 2 \times 2$ and a sorting interval of 4 mean that particles were sorted into $2 \times 2 \times 2$ supercells every four timesteps. On this problem, the optimal sorting is to sort by cell (i.e., a bin size of

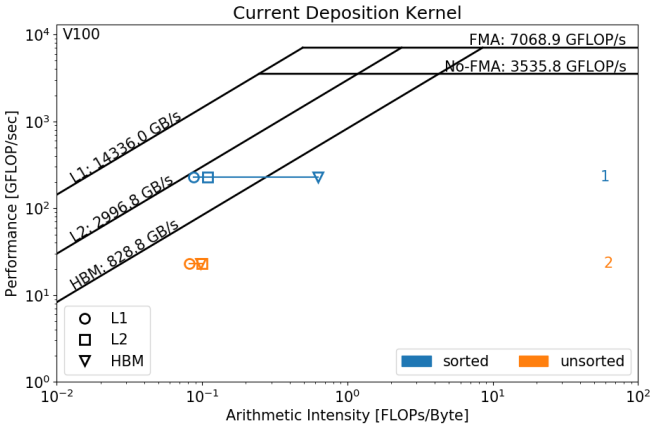


Figure 41: Roofline analysis of the third-order Esirkepov current deposition [25] kernel in WarpX on a single V100 GPU with and without particle sorting. In the memory streaming limit, three different lines are shown, corresponding to the bandwidths of the L1 and L2 caches and the bandwidths for the main high-bandwidth memory (HBM) on the GPU. Likewise, in the compute-bound regime, two different values are used for the peak floating point performance with and without taking advantage of fused multiply-add instructions. The arithmetic intensity is measured three times for each kernel via the memory traffic for each level of the memory hierarchy. For the sorted version, the arithmetic intensity is significantly lower for the L1 and L2 data points, which shows that we are achieving substantial reuse in both levels of cache. Conversely, because the data points are all on top of each for the unsorted run indicates that without sorting, the degree of reuse is poor.

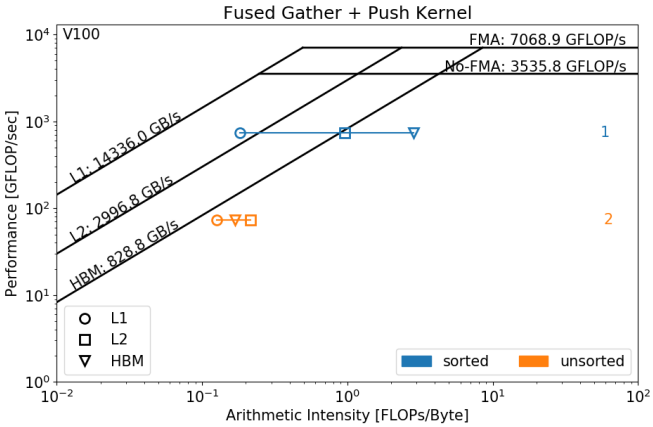


Figure 42: This figure is the same as Fig. 41 but is for the fused gather and push kernel in WarpX. Again, there is substantial cache reuse when sorting is employed, although performance still appears to be limited by HBM bandwidth for this kernel, even with sorting.

$1 \times 1 \times 1$ every time step, and the difference between sorting optimally and not sorting at all is a factor of ∼7.5 with most of the improvement coming from the current deposition and fused gather and push kernels. However, this very frequent sorting interval is specific to this problem in which the particles change cells more often than in most WarpX applications. Currently, the default in WarpX used throughout § 4.6.4 is to sort the particles by their PIC cell every four time steps.

Although the `Redistribute()` function in AMReX does not maintain this cell-sorted order for particles that left one grid and were migrated to another, this only applies to particles that have changed grids—typically only a small subset of the total that are near the "surface." Most particles on a grid will maintain their sorted order in between `Redistribute()` calls.

Figures 41 and 42 show the results of a roofline analysis on the current deposition and fused gather and push kernels in WarpX, which are the two most computationally expensive operations. Our analysis followed the methodology of Yang, Kurth, and Williams [27]. For this test, we used a uniform plasma setup with eight particles per cell and gave the particles a large thermal velocity so that they frequently change cells. To eliminate the effects associated with unified memory paging, we ran the problem for 100 steps and only profiled the last one.

The roofline analysis reveals three things. First, as already demonstrated, sorting the particles gives significantly better performance on V100 GPUs than not sorting them. Second, the arithmetic intensity measured via the memory bandwidth for the L1 and L2 caches is significantly lower than for HBM, which indicates that we are getting significant reuse in both cache levels in the sorted run. Third, the arithmetic intensity for the current deposition for the sorted run is directly against the streaming limit for the L2 cache. This indicates that the performance of this kernel is now limited by the L2 cache bandwidth. On the other hand, gather and push is likely still limited by HBM bandwidth. Taken together, these results suggest that these kernels should achieve significantly better performance on the A100, which has a larger L2 cache and higher HBM bandwidth than the V100.

Performance Results

This section provides current performance results on Summit on a uniform plasma test case.

Weak scaling study To test the scaling of WarpX in an idealized setting and gauge the speedup associated with using accelerated nodes, we performed a weak-scaling study by using a uniform plasma setup on OLCF's Summit supercomputer. The base case for this scaling study used a $256 \times 256 \times 384$ domain with a box size of 128^3 and ran on one Summit node; thus, on the GPU-accelerated runs, each GPU was responsible for processing two 128^3 sized boxes. Particles were initially distributed uniformly with eight particles per cell. We used the standard "Yee" finite-difference Maxell solver for these runs with Esirkepov current deposition and third-order shape functions. The number of Summit nodes was doubled with the number of cells and particles therein in the x-, y-, or z- directions while holding everything else constant, maintaining a constant workload per node. We continued this process up to 2048 nodes—about half of the Summit machine.

The results are shown in Fig. 43. We performed this scaling study twice: once using all six GPUs per Summit node and again using only the POWER9 CPUs. For both runs, we used six MPI tasks per node. So that all 42 cores on the node were active, we used one GPU per MPI task for the GPU-accelerated runs and seven OpenMP threads per task for the CPU-only case. Using these results, we can characterize the weak-scaling behavior of the CPU and GPU versions of the WarpX, as well as see the overall speedup obtained on Summit by using the accelerators. In both cases, the code scales well up to 2048 nodes. The weak-scaling efficiency, defined as the total time taken for 100 time steps on one node divided by the total taken on 2048 nodes, is 81 % for the GPU case and 90 % for the CPU case. The difference in scaling efficiency between the CPU and GPU can be attributed to the fact that because the local work is significantly faster when using the V100s, communication operations such as `FillBoundary`, which are inherently harder to scale, become relatively more expensive. Additionally, the speedup from the accelerators at all scales tested was a factor of ∼30. This speedup refers to the total runtime, including the time associated with host/device memory traffic and communication, not to isolated compute kernels.

Strong Scaling Study We also conducted a series of strong scaling tests by using uniform plasma problem setup very similar to the one used before. The only difference is that the box size was set to 64^3 to allow for more GPUs/MPI tasks to be used as the problem is strong-scaled. There is some overhead associated with

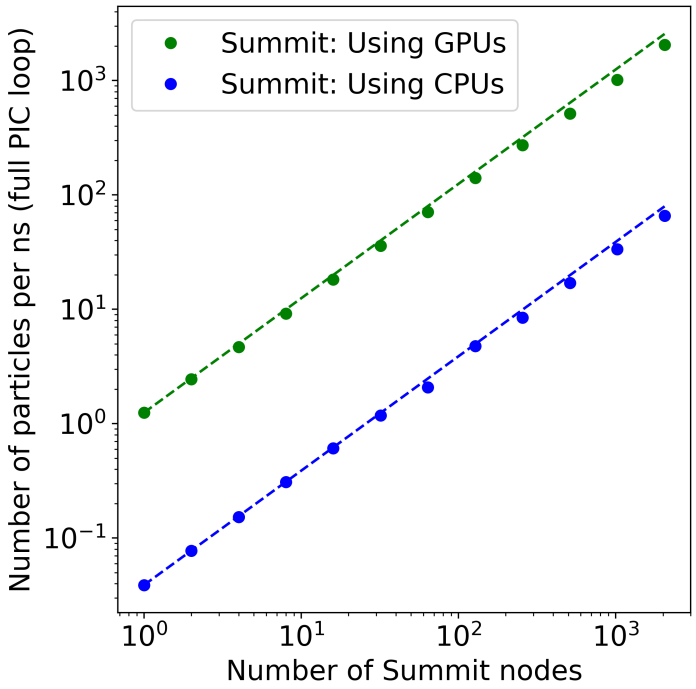


Figure 43: Results of a weak-scaling study on a uniform plasma setup on Summit. The x-axis shows the number of Summit nodes, and the y-axis is the number of particle advances per nanosecond. The CPU and GPU versions of the code both scale well, and the overall speedup associated with using the accelerators is $\sim$30.

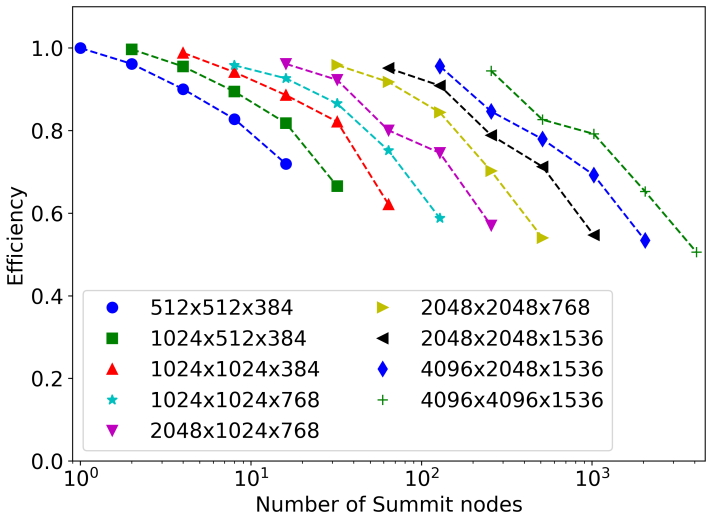


Figure 44: Strong-scaling studies for a variety of problem sizes. Each tick type refers to a different problem size. The x-axis shows the number of Summit nodes, and the y-axis shows scaling efficiency, which is defined as the time that a run should take, assuming perfect strong scaling within a problem size and perfect weak scaling from the base problem size, divided by the actual runtime.

doing this because the surface to volume ratio of ghost cells is higher with smaller boxes. Other than the box size, the parameters are all the same as before.

We used a series of problem sizes, each scaled up by a factor of 2 in terms of the number of cells and particles in the domain. For each, we conducted a series of five runs, increasing the number of MPI tasks by a factor of 2 each time. Thus, assuming perfect strong scaling, the runtime should have decreased by a factor of 16 in the fifth run. By the time we multiplied the number of MPI ranks by 16, this problem had reached the point at which the compute work and the communication work take approximately the same amount of time, so we would not expect the problem to scale further than that.

The smallest scaling study in this series goes from 1 to 16 nodes, and the largest goes from 256 to 4096 nodes, which is nearly the entire machine. The scaling efficiency, defined as the time a run should take, assuming perfect strong scaling within a problem size and perfect weak scaling from the base problem size divided by the actual runtime, is plotted in Fig. 44. The efficiencies after strong scaling by a factor of 16 for each problem size vary from approximately 70 % for the smallest case to approximately 50 % for the largest.

Next Steps

WarpX now runs at scale on Summit, and a model of a chain of up to 10 consecutive plasma accelerator stages was demonstrated on Summit. Further optimizations include reducing the size and communication costs of guard regions when using the pseudo-spectral Maxwell solver and using single precision for part or whole of the particle-in-cell loop. A new algorithm that allows time steps to be used above the limit for standard particle-in-cell methods was developed and will be validated on plasma accelerator simulations on Summit. The development of mesh refinement will also be pursued with the integration of the latest algorithmic developments.

For AMD GPUs, our focus will be on further investigating and optimizing key kernels. For example, we will explore ways to reduce the number of registers used in the current deposition kernel whose occupancy is currently limited by the number of registers. For Intel GPUs, we will start building and running WarpX on ATS nodes with the latest Intel HP GPUs. Because the latest Intel GPUs have native double precision support and high-bandwith device memory, a more detailed performance analysis will be carried out. For both AMD and Intel GPUs, we will also start building and running WarpX on multiple GPUs.

New and Ongoing Integrations with ECP S&T

Table 38: Summary of supported ESS L4 projects.

WBS number	Short name	Project short description	KPP-X
2.2.3.01	ExaStar	Demystify the origin of chemical elements	KPP-2
2.2.3.02	ExaSky	Cosmological probe of the Standard Model	KPP-1
2.2.3.03	EQSIM	Seismic hazard risk assessment	KPP-1
2.2.3.04	Subsurface	Carbon capture, fossil fuel extraction, waste disposal	KPP-2
2.2.3.05	E3SM-MMF	Regional assessments in earth systems	KPP-1

WarpX added a first integration with Ascent for in situ visualization, which produced first figures and videos for the aforementioned multistage milestone on Summit, including GPU-accelerated compute and in-memory visualization with VTK-m. Because first functional correctness was achieved, the ongoing collaboration with the Alpine (Ascent) team will focus on providing more functionality for domain-specific visualization needs, providing advanced rendering options for highlights, relaxing initial limitations in the composition of rendered scenes, deploying software, and performing performance optimizations.

We successfully created a first functional I/O implementation compatible with the Open Standard for Particle-Mesh Data (openPMD) [28]. openPMD is a generalized metadata format that we use to unify the input and output of data in a suite of particle accelerator codes. The first bindings to ADIOS2 and HDF5, abstracted via our own `openPMD-api` middleware layer, were implemented. The first developer tests that compressed ADIOS2 data with ZFP were also successful. All our current I/O methods suffer from varying performance regressions at scale and in read performance. Due to high flexibility and good tunability, the upcoming year will focus on performance tuning of ADIOS2 for I/O workflows.

We continue to integrate our software stack with ECP Spack to ensure that we can build the full functionality of WarpX in a consistent deployment. We are already using Spack environments for development and continue to contribute further recipes and updates, including binary deployment via E4S.

5. EARTH AND SPACE SCIENCE APPLICATIONS

End State: Deliver a broad array of comprehensive science-based computational applications able to provide, through effective exploitation of exascale HPC technologies, breakthrough modeling and simulation solutions to fundamental issues and scientific questions centered on key planetary processes and the origin of the universe.

The Earth and Space Science Applications (ESS) application L3 area (Table 38) spans fundamental scientific questions from the origin of the universe and chemical elements to planetary processes and interactions affecting life and longevity. These application areas treat phenomena where controlled and fine resolution data collection is extremely difficult or infeasible, and in many cases fundamental simulations are the best source of data to confirm scientific theories and predict critical phenomena.

The key objective in the area of ESS is to utilize exascale resources to carry out simulations of phenomena with massive ranges of space and temporal scales, and where controlled data collection is extremely limited or impossible. These applications are critical to mankind's well-being and understanding of fundamental questions of the universe, and in many cases advanced simulation is the most effective vehicle for gaining insight into these processes. As their computing requirements are massive, it is critical to develop these codes to make efficient use of exascale computing resources.

5.1 ExaStar

This project is developing a new code suite, Clash, which will be a component-based multiphysics AMR-based tool kit that accurately simulates coupled hydrodynamics, radiation transport, thermonuclear kinetics, and

nuclear microphysics for stellar explosion simulations. Clash comprises the FLASH and Castro multiphysics codes and associated modules, which will reach exascale efficiency by building on current multicore and manycore efficient local physics packages integrated into a task-based asynchronous execution framework based on current AMR technology. The fundamental goal in Clash development is to understand the production of the chemical elements in these explosions, particularly those heavier than iron. Although astronomical observations reveal that the production of the heaviest nuclei began early in galactic history, it is unknown how and where these elements were formed. To address this topic via laboratory measurements, a series of Nuclear Science Advisory Committee Long Range Plans have supported the construction of radioactive ion beam facilities, culminating in the Facility for Rare Isotope Beams (FRIB). Although FRIB is designed to acquire extensive data on the nuclei relevant for astrophysical nucleosynthesis, its science end goal cannot be met unless those experimental data are integrated into high-fidelity simulations of stellar explosions—supernovae and neutron star mergers—that define the conditions under which such heavy element production most likely occurs. Through a better understanding of the sites at which the heaviest elements are made, Clash can help focus experimental efforts at FRIB on those reactions of greatest influence.

5.1.1 ExaStar: Science Challenge Problem Description

The ExaStar challenge problem is a 3D simulation of the first 2 s of evolution after the iron core bounce of a core-collapse supernova. The progenitor star model will be chosen at runtime from the best available models. The most likely progenitor models are: (1) the solar metallicity 12 $\text{solar}_{\text{mass}}$ progenitor of Sukhbold, Woosley, and Heger [29], which was chosen because it represents the "center" of the distribution of massive stars that produce core collapse supernovae (CCSNe), and (2) the binary merger model of Menon and Heger [30], which was chosen because it is believed to closely mimic the progenitor system of SN 1987a, the only CCSNe from which there are multi-messenger signals to date.

The physical domain will extend from the center of the star outward to fully enclose the helium shell of the evolved star. The precise location of this radius is progenitor-dependent, but it is always more than 10,000 km. The maximum spatial resolution enabled with AMR will be at least 1 km at the surface of the proto-neutron star (i.e., in approximately the inner 100 km of the event). At least 20 energy groups will be used to resolve the spectra of neutrinos of all types (i.e., electron, mu, tau, and their anti-particles) from 0 to 300 MeV. An approximation to general relativistic gravity that uses at least 12 moments in a multipole approach will be used, with the option to have a more realistic treatment (e.g., conformally flat approximation), if possible. A set of tabulated neutrino-matter interaction rates—which include emission, absorption, scattering, and pair production from various nuclear and nucleonic processes—will be used. This table will be coupled to a set of tabulated quantities derived from a high-density EOS that will provide pressures, entropies, and all other required thermodynamic values as required by, for example, the hydrodynamics. The available set of coupled rates and EOS tables will include, the SHF0 EOS of Steiner, Hempel, and Fischer [31] at a minimum. Details on the challenge problem are listed in Table 39.

5.1.2 ExaStar: KPP Stretch Goal

The ExaStar stretch goal is to perform a 3D simulation of the initial phases of a neutron star merger. Unlike the core collapse supernova challenge problem, this stretch goal requires a general relativistic (GR) dynamical space-time solver and a GR magneto-hydrodynamics solver, which represent a significant advance in code capability. The initial conditions of the problem will comprise two neutron stars in a quasi-circular orbit with a separation sufficient to complete two to three orbits before mergers. The simulation domain will cover 3000 km (roughly 300 neutron star radii) with a resolution that uses AMR of order 100 m. Instead of Newtonian gravity, a dynamical space-time solver that uses a high-order finite differencing scheme will be employed with corresponding Kreiss-Oliger dissipation terms to discretize space-time variables in the BSSN formalism. Additionally, a general relativistic ideal magneto-hydrodynamics solver will use the metric from the space-time solver to advance the fluid dynamics while ensuring a divergence-free magnetic field. They will use a generalized version of the two-moment neutrino transport module that includes the relevant general relativistic terms with an order of 20 energy groups used to resolve the spectra of neutrinos. The transport will use a tabulated neutrino-matter interaction rate coupled to a set of tabulated quantities derived from a high-density EOS. Because the physical conditions in the dynamical phase of mergers do not involve complex nuclear reactions, only a small reaction network, including nucleons and alpha particles, will be required.

Table 39: ExaStar challenge problem details.

Functional requirement	Minimum criteria
Physical phenomena and associated models	Core-collapse supernova, compressible hydrodynamics, self-gravity, nuclear burning, specialized equation of state, neutrino transport.
Numerical approach, algorithms	Finite volume shock-resolving Eulerian solvers, Poisson solver, ODE, 0D calculations from tables for EOS, discontinuous Galerkin finite elements for neutrino transport.
Simulation details: problem size, complexity, geometry, and so on	Maximum spatial resolution enabled with AMR will be at least 1 km at the surface of the proto-neutron star (i.e., in the inner 100 km); 20 energy groups to resolve the spectra of neutrinos of all flavors (i.e., electron, mu, tau, and their anti-particles) from 0 to 300 meV; an approximation to general relativistic gravity using at least 12 moments in a multipole approach will be used with the option to have a more realistic treatment (e.g., conformally flat approximation), if possible.
Demonstration calculation requirements	Partial evolution from a post-bounce configuration, which can be evolved to this point in spherical symmetry. The evolution must be long enough to provoke several epochs of AMR mesh generation and restriction.
Resource requirements to run demonstration calculation	Full exascale machine for 2 h.

The simulation will cover roughly the first 20 ms of the merger or if a black hole forms for a few milliseconds after formation, which is sufficient to determine the amount of dynamical mass ejected and the composition and asymptotic kinetic energy of the outflows.

5.1.3 ExaStar: Capability Plan

The capability matrix for ExaStar is shown in Table 40.

5.1.4 ExaStar: Progress on Early and Pre-Exascale Hardware

Performance on Summit

All physics modules required for the ExaStar base challenge problem were implemented, verified with test problems, and ported to GPUs. These modules were integrated into the FLASH code and exercised on supernova simulations of reduced dimensionality, spatial resolution, and evolution in time.

The calculation of neutrino radiation transport using the two-moment radiation transport module *Thornado* represents possibly the most computationally expensive component of the astrophysical simulation. ExaStar developed, implemented, and tested various nonlinear solvers for implicit neutrino-matter coupling in an IMEX scheme for two-moment transport. Profiling results on Summit, shown in Fig. 45 (left panel), indicate that a nested Anderson acceleration (AA) scheme has the best performance, being a factor of two faster than a coupled Newton or coupled AA method. The nested solvers require fewer outer iterations, reducing the number of calls to expensive neutrino opacity kernels. The nested AA solver is moderately ($\sim 10\times$) faster than a nested Newton's method due to the latter's additional dense linear algebra cost. The transport modules, along with other physics modules in the ExaStar ecosystem, will rely on good dense linear algebra libraries for good performance.

The fidelity of the neutrino transport was enhanced by completing the tabulation of the base neutrino opacity set with improved microphysics, including emission/absorption, elastic scattering, neutrino-electron scattering, and pair processes. Profiling on Summit (Fig. 45, right panel) indicates that the bottleneck for *Thornado* transport calculations is table interpolations for the opacity. The cost is mitigated when using GPUs, and a $10\times$ speedup was achieved relative to CPUs. Slight differences in performance are seen between openACC and OpenMP with off-load, which could be compiler-related.

Table 40: Capability matrix for ExaStar.

	FY20	FY21	FY22	FY23
Neutrino Physics	Fully couple electron-flavor neutrinos and anti-neutrinos. Exercised on radiation test problem.			nu-nubar + muons
Radiation Transport	Two-moment discontinuous Galerkin method, no-frame effects.	Transport informed by multi-angle MC methods in marginally optically thin regions. Beta version of frame-effects in two-moment transport.	Fully implemented frame-effects in two-moment transport.	Fully implemented frame effects in two-moment transport; integration of MC transport support.
Gravity	Asynchronous multipole solver (Newtonian). Performance compared with previous solver.	GR hydrodynamics with fixed metric.		Full space-time solver and GR hydrodynamics.
Hydro/MHD		Implementation and testing of GR MHD solver.	GR MHD tested and exercised on science problem.	Optimized performance of MHD solver.
Network	Large (160 species), profiled on GPU.	Large.	Large.	Larger.
EOS	Tabulated with GPU-enabled interpolation.	Gaussian process enabled. EOS	Hybrid EOS with smooth transition.	Best available combination.
Integration	Run and profile integrated core collapse simulation run through bounce with reduced dimensionality/resolution.	Run and optimize integrated core collapse simulation through bounce with higher dimensionality/resolution.	Run and profile integrated neutron star merger disk simulation with best available physics.	Run integrated core collapse simulation through bounce and shock propagation with full dimensionality, target resolution and highest fidelity physics.
GPU readiness		Hydro solver in Castro working with HIP.		

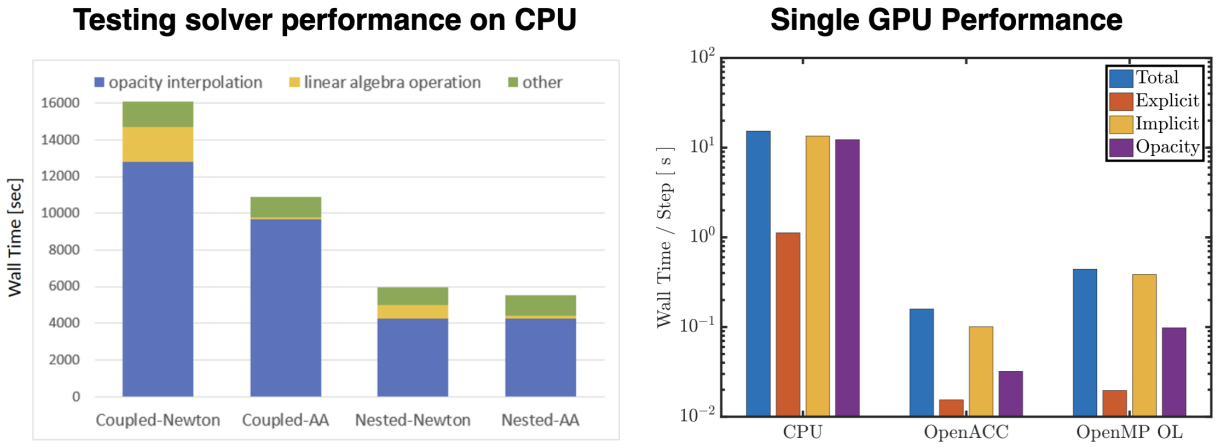


Figure 45: Performance of the *Thornado* radiative transport module on Summit. (Left) Code performance using different nonlinear solvers; a nested Anderson acceleration scheme provides the fastest results. (Right) Profiling the components of the transport calculations. The implicit part of the calculation, which treats neturino-matter coupling, dominates over the explicit part, which treats radiation advection. Of the implicit part, the calculation of opacities through table interpolation is the most costly component. A 10× speedup is achieved on GPUs by using openACC or OpenMP with off-load.

ExaStar has developed new physics modules needed for its stretch goal. Initial testing was performed, although these modules have not yet been integrated into the full FLASH framework. A general relativistic space-time solver was implemented by using an explicit time integration. This solver leverages AMReX abstractions for massively parallelizable data structures and iterators for mesh data. It includes AMR with sub-cycling in time and a higher order time integrator. Code generators are used to allow for precise unpacking of the complex set of Einstein equations, translating from Python to AMReX usable C. The compilable expression of the right-hand side of the evolution equations, which use higher order spatial discretizations, convert PDEs to ODEs via a method of lines. The space-time solver was ported to GPUs, and the results were confirmed to be identical to CPU up to numerical roundoff. Figure 46 shows the strong scaling performance for a test calculation, run in this case on NERSC's Cori machine with NVIDIA Tesla V100. The run used one MPI rank per GPU and the equivalent of 15 Summit nodes. High occupancy is confirmed to provide the most efficient results.

Other physics modules used in ExaStar—such as nuclear reaction networks, hydrodynamics, and EOS—were ported to GPUs in previous years and demonstrated to have similar speedups on Summit. Integrated simulations run on Summit using the FLASH code with improved physics modules have followed a core collapse supernova simulation that captures the key features of stellar collapse, core bounce, and shock formation.

Next Steps

Continuing work will improve the physical fidelity of ExaStar simulations. The Thornado transport module will be generalized to include relativistic effects and enhanced microphysics. As the two-moment formulation of the transport problem approximates the angular distribution of the radiation field, an MC transport method was implemented within the AMReX framework that solves the Boltzmann equation in full generality. Although MC might be too computationally expensive, ExaStar will explore a hybrid approach in which the neutrino emissivity is split in a weighted fashion with the two-moment method preferentially applied in the interior regions of the system where the radiation is more nearly isotropic and with the MC method applied in the exterior, optically thin regions where MC is more efficient. Further work will implement and test such a hybrid algorithm; if the approach proves unfeasible, the MC could still be used to derive

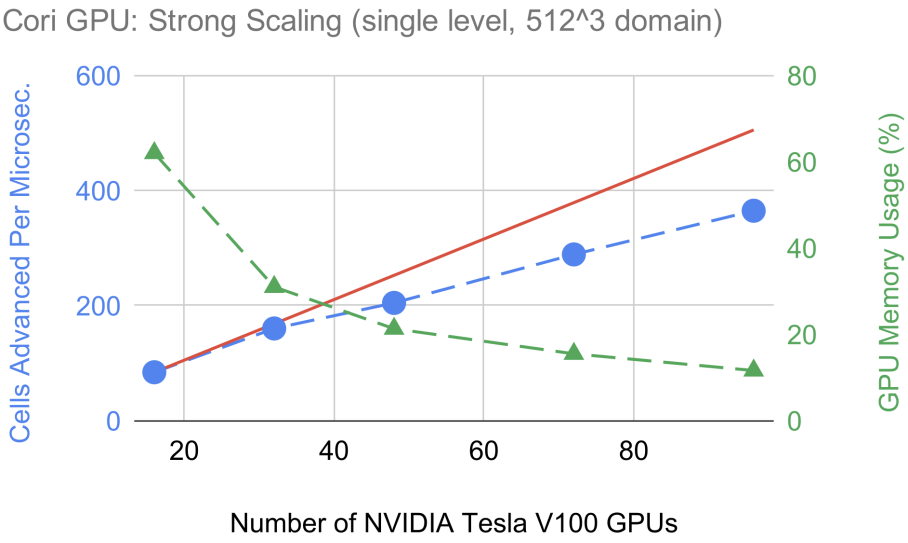


Figure 46: Strong scaling of the space-time solver module run on NERSC's Cori with NVIDIA Tesla V100 using the equivalent of 15 Summit nodes. One MPI rank per GPU was used with all integrator/RHS functions on the GPU. Blue points are a comparison of the scaling results relative to the ideal (red line). The green line shows the percent memory usage with high occupancy being more efficient.

improved analytic closures required by the two-moment module.

The new space-time solver will be implemented and tested on AMReX adaptive meshes. Additionally, new development will replace the Newtonian hydrodynamics in FLASH with a module for general relativistic magneto-hydrodynamics. A prototype module called *Spark* was already put into FLASH and run on initial benchmark calculations for relativistic shocks with a fixed space-time metric. Future work will port Spark to GPUs and integrate it with the space-time solver, enabling the full general relativity required for the ExaStar stretch goal of a neutron-star merger simulation.

Design specifications of an orchestration system will be completed, and the base functionality of the system will be implemented. The orchestration system has two primary components. The first is a runtime system responsible for data movement between devices and launching kernels. The second is an offline toolchain that can parse meta-information encoded by the component writers, indicating various options for configuring the kernels within the component, matching with the specifications of the target platform, and generating the code and information needed by the runtime to operate correctly. Work will complete the design specification for the toolchain through the process of prototyping within the design space.

Because significant changes are being made to the code infrastructure and data movement algorithms in FLASH, performance bugs must be caught. A performance monitoring system will be incorporated into the daily test suite that will allow us to closely observe code behavior. This will be in addition to any targeted benchmarking that is done with every main section of the orchestration system.

ExaStar is modernizing the Paramesh AMR package as a risk mitigation plan in the event that the performance of AMReX in FLASH proves suboptimal. Initial studies will address asynchronizing primary communication, steps such as ghost-cell fill, regridding, and flux-correction. The process will start by converting communications in ghost-cell fill to one side and specifying meta-information that must accompany the data for the target to be able to process it asynchronously.

With an eye toward ExaStar's base challenge goal, we will run and evaluate a science problem to test the hydrodynamics and two-moment neutrino transport coupling for a dynamic core-collapse matter background, including the existing Newtonian gravity solver. This will use a collapsing massive star as the initial condition to test the hydrodynamics and transport coupling with best available neutrino physics in a dynamic

environment. Profiling will provide further information on any bottlenecks and inefficiencies in the integrated simulation that covers the physical conditions anticipated in the challenge problem.

5.2 ExaSky

Modern cosmological observations carried out with large-scale sky surveys are unique probes of fundamental physics. They have led to a remarkably successful model for the dynamics of the universe and several breakthrough discoveries. Three key components—dark energy, dark matter, and inflation—are signposts to further breakthroughs because they all reach beyond the known boundaries of the Standard Model of particle physics. A new generation of sky surveys will provide key insights into questions raised by the current paradigm and provide new classes of measurements, such as neutrino masses. They might lead to exciting new discoveries, including those of primordial gravitational waves and modifications of general relativity. Sophisticated, large-scale simulations of cosmic structure formation are essential to this scientific enterprise. These simulations not only shed light on some of the biggest challenges in physical science but also rank among the largest and most scientifically rich simulations run on supercomputers today. Existing machines do not have the performance or memory needed to run the next-generation simulations that are required to meet the challenge posed by future surveys whose timelines are parallel to that of the ECP. The ExaSky project extends the HACC and Nyx cosmological simulation codes to efficiently use exascale resources as they become available. The Eulerian AMR code Nyx complements the Lagrangian nature of HACC; the two codes are being used to develop a joint program to verify of gravitational evolution, gas dynamics, and subgrid models in cosmological simulations at a very high dynamic range.

To establish accuracy baselines, there are statistical and systematic error requirements on many cosmological summary statistics. These statistics include the density fluctuation power spectrum, the halo mass function, the halo bias as a function of mass, the weak gravitational lensing shear power spectrum, and kinematic and thermal Sunyaev-Zeldovich effects for galaxy clusters. There are also several cross-correlations, such as the density-halo cross power and cosmic microwave background cross-correlation with large-scale structure. The accuracy requirements are typically scale-dependent, large spatial scales being subject to finite-size effects and small scales being subject to several more significant problems, such as particle shot noise and code evolution errors, including subgrid modeling biases. Strict accuracy requirements were already set by the observational requirements for DOE-supported surveys ,such as the CMB-Stage 4 (CMB-S4), Dark Energy Spectroscopic Instrument (DESI), and Rubin Observatory's Legacy Survey of Space and Time (LSST), which typically are sub-percent (statistical) over the range of well-observed scales. Systematic errors must be characterized and controlled, where possible, to the percent level or better. All of these error controls must be satisfied when running the challenge problem simulations at the >50× FOM requirement. For a recent exploration of cosmological simulation errors in hydrodynamic simulations by ExaSky, see Emberson, Frontiere, Habib, Heitmann, Larsen, Finkel, and Pope [32]; this paper uses the ExaSky-developed CRK-HACC code that implements the recently developed CRK-SPH method [33] in a cosmological setting. The final challenge problem runs will be carried out with a new set of subgrid models for gas cooling, UV heating, star formation, and supernova and AGN feedback, which are now under active development.

The simulation sizes are set by the scales of the cosmological surveys. The challenge problem simulations must cover boxes of linear size up to the few gigaparsec scale with galaxy formation-related physics modeled down to roughly 0.1 kpc, a dynamic range of one part in 10 million, improving the current state of the art by an order of magnitude. Multiple box sizes will be run to cover the range of scales that must be robustly predicted. In the smaller boxes, the mass resolution of the simulations will go down to roughly 1 million solar masses for the baryon tracer particles and about five times this value for the dark matter particles; the largest boxes will involve larger particle masses by roughly two orders of magnitude. The final dynamic range achieved depends on the total memory available on the first-generation exascale systems.

5.2.1 ExaSky: Science Challenge Problem Description

The ExaSky science challenge problem will eventually comprise two very large cosmological simulations run with HACC that simultaneously address many science problems of interest. Setting up the science challenge problem requires multiple simulations, which will be completed before the arrival of the exascale systems. These involve building subgrid models by matching against results from very high-resolution galaxy formation

Table 41: ExaSky gravity-only challenge problem details.

Simulation target	Specifications
Initial conditions	Multiparticle Gaussian random field initial conditions using a specified linear power spectrum at the initial redshift based on ExaSky's 3D SWFFT with up to 30,000 cube FFTs.
Boundary conditions	Periodic (box size of 3 Gpc/h).
Resolution	Force resolution, $\sim$1 kpc, mass resolution $\sim 2 \times 10^8$ solar masses (gravity-only run), and $\sim 10^9$ solar masses (dark matter), $\sim 2 \times 10^8$ solar masses (baryons) for the hydrodynamic simulation. This corresponds to $23{,}040^3$ particles for the first simulation and $2 \times 12{,}288^3$ for the hydro simulation.
Physics	N-body gravity via spectral particle-mesh (long-range) and direct particle-particle or tree/FMM (short-range); Lagrangian hydrodynamics with CRK-SPH. Subgrid models for UV cooling and heating; star, black hole, and galaxy formation and associated effects; supernova; and AGN feedback.
Science outputs	Summary statistics for matter and velocity fields, Lyman-alpha forest, weak lensing shear, halo properties and halo spatial statistics, halo merger trees, and tSZ and kSZ statistics. Light-cone outputs, galaxy summary statistics, strong/weak lensing for galaxy clusters, and sky maps for optical and CMB observables. Multiprobe cross-correlations.

astrophysics codes via a nested-box simulation approach, a medium-scale set for parameter exploration. The final two large-scale challenge problem runs on the exascale platforms will be based on these results.

The challenge problem runs are of two different types. The first is a large-volume, high-mass, and force resolution gravity-only simulation (Table 41), and the second is a corresponding hydrodynamic simulation that includes detailed subgrid modeling (Table 42). The second simulation will include hydrodynamics and detailed subgrid modeling. The main probes targeted with these simulations are strong and weak lensing shear measurements, galaxy clustering, clusters of galaxies, and cross-correlations that are internal to this set and with CMB probes, such as CMB lensing and thermal and kinematic Sunyaev-Zeldovich effect observations. The challenge problem runs will have the same cosmology and simulation volume; they will also share the same random phases in the initial conditions. This will allow us to investigate the effects of baryonic physics on cosmological probes via a direct comparison across the two simulations.

There are two different optimization strategies for the ExaSky challenge problem. The first is to maximize the performance of the gravity and hydro solvers, and the second is to develop a new generation of subgrid models in which known empirical results are emergent rather than enforced, as is mostly the case currently. The aim with this more physics-based approach is to achieve a more consistent approach in which the final results become independent of code parameters and are more easily interpreted. The new subgrid models not only must be optimized for the exascale platforms but they also affect the temporal resolution of the simulations. Once implemented, more time steps are needed as more fine-grained physics is being included and resolved. More than the raw performance of the main solvers, which is very good and will remain so, this feature will strongly affect the computational requirements for running the challenge problem.

5.2.2 ExaSky: KPP Stretch Goal

The stretch goal will be to increase the number of particles in the simulations, depending on the memory available on the systems. The improved mass resolution will increase the computational intensity and will increase the wallclock requirement for the demonstration calculation. The current estimate for the stretch goal is based on a $30{,}720^3$ particle problem (compared with the $23{,}040^3$ particles for the base gravity-only problem) and $2 \times 15{,}360^3$ for the hydrodynamics problem. This will increase the demonstration runtime by a

Table 42: ExaSky subgrid challenge problem details.

Simulation target	Specifications
Physical phenomena and associated models	Multiscale cosmic structure formation—gravitational evolution, gas dynamics, and subgrid models for astrophysical processes, including several feedback mechanisms.
Numerical approach, algorithms	Lagrangian (n-body) with CRK-SPH hydrodynamics for the HACC code; multiple data-intensive algorithms, including AI/ML, in HACC's CosmoTools analysis library.
Simulation details: problem size, complexity, geometry, and so on	Multitrillion multispecies particle simulation with fully representative subgrid modeling for the challenge problem; results must be available at low redshift (near the current epoch).
Demonstration calculation requirements	(1) Limited number of time steps with a fully representative simulation for the solvers but not for the subgrid models; ability to run to low redshift. (2) Smaller scale simulation with subgrid models fully implemented, run with a larger number of time steps, again to low redshift.
Resource requirements to run demonstration calculation	The demonstration case will require a large fraction of an exascale system for about 48 h of runtime.

factor of 2, at a minimum.

In terms of additional capabilities, it is important that several data-intensive and AI/ML-oriented analysis capabilities be available on the exascale systems. As Table 41 makes clear, the science outputs from the simulations are complex and require multiple analyses carried out with HACC's CosmoTools framework in all three analysis modes: in situ, co-scheduled, and off-line. Consequently, the ExaSky project will stress test several capabilities of the exascale systems (e.g., file I/O, AI/ML performance) aside from the purely computation-oriented requirements.

5.2.3 ExaSky: Figure of Merit

In principle, the ExaSky FOM requires a discussion of several factors to arrive at a well-defined single number that can simultaneously capture a multidimensional set of requirements. In terms of overall throughput or delivered performance, the basic issues can be divided into three components: (1) increase in problem size (weak scaling), (2) node-level computational performance (strong scaling), and (3) the addition of new science capabilities, such as physics complexity, which in the case of ExaSky is a statement about subgrid models. Provided that weak-scaling targets are met, increases in problem size should provide a linear increase in the FOM; likewise, provided that the strong-scaling targets are met, problems at a fixed size should run proportionally faster. Although ExaSky comprises two simulation codes, HACC and Nyx, because the final large-scale simulations for the exascale challenge problem will be run with HACC, the FOM will be based on results with HACC alone. To summarize, a scaled-down version of the challenge problem run at low redshift will be used to determine code throughput in three modes: gravity-only, gravity + hydro, and gravity + hydro + subgrid. The results from these runs will be combined to yield a single FOM with weight factors chosen to represent the importance of each particular test with respect to the challenge problem runs.

The problem with factoring the complexity of the subgrid models into the FOM is that because these models are continuously evolving and were not run—and will not be run—on systems such as Mira and Titan, it is very difficult to estimate what the performance ratios would be across platforms, especially if the baseline platforms are unavailable or if it is not helpful to port new applications to obsolete systems. Fortunately, the subgrid model performance from the FOM can essentially be eliminated for two reasons: (1) the subgrid models are entirely local and as such do not affect the weak scaling performance of the code and (2) the primary effect of having subgrid models is twofold: an increase in the time for an individual short-range computational map that combines gravity and hydrodynamic forces and a reduction in the actual

value of the short-range time step (i.e., higher time resolution) once the subgrid models are incorporated. The latter effect can be very significant, leading to increases in the overall number of time steps by one or two orders of magnitude, whereas the actual increase in time due to the additional work per time step is only on the order of 10–20%. For this reason, the FOM remains controlled by the time spent in the main solvers (i.e., gravity + hydro); therefore, the performance of these can be used to determine this value without needing to consider how the new subgrid models would perform on the baseline platforms.

The current baselines for the FOM were established by gravity-only runs on Intel KNL platforms (Cori II, Theta) and on the IBM BG/Q systems Mira and Sequoia. CPU/GPU runs include simulations on Titan, SummitDev, and Summit. Initial (nonradiative) CRK-SPH hydro runs were performed on Cori II, Theta, SummitDev, and Summit. Details of the baseline information are presented in the ExaSky milestone report ADSE01-29, *MS3/Y2: Summit Performance Metrics for HACC*. Because of HACC's demonstrated excellent weak- and strong-scaling performance, it is easy to convert wallclock numbers from one set of simulation runs to another if the physical parameters of the runs are kept unchanged (i.e., cosmological parameters and force and mass resolution are held invariant, whereas the number of compute nodes can be varied).

Therefore, the HACC FOM calculation is based on the following steps (ADSE01-29).

1. Establish code scaling on the reference and evaluation systems. HACC must: (1) strong-scale at settings typical of the challenge problem requirements and (2) weak-scale to the full size of the evaluation system. Both of these requirements must also be satisfied for the reference system.

2. Run representative problems on both systems. A smaller problem run on the reference system can be appropriately scaled to the one on the evaluation system by using the known weak-scaling performance. Also, compute the per-time-step throughput for the inverse of the time taken to run one time step in a code configuration typical of the late universe where the time-stepping is the most compute-intensive.

3. Compute the throughput ratios of the evaluation system to the reference, making sure to scale the reference system performance to 20 PFlops (peak).

4. If desired, apply this methodology separately to the gravity-only part of the code, to the gravity + hydro, and to the final gravity + hydro + subgrid model code versions. This is useful because intermediate FOMs for the solvers can be generated even if all the subgrid models are not implemented. The final set of FOMs will be applied by using Summit as the base system with appropriate scaling based on the known FOMs for Summit referred to the previous generation of machines.

FOM Update

The ExaSky FOM is is designed to represent the most important components and goals of the challenge problem, and it is straightforward to implement. The ingredients are the number of simulation particles $N = n_p^3$, where n_p is the number of particles per dimension; the time to solution, t, measured per time step; and separate runs performed for the gravity and hydro versions of HACC. With this information, the ExaSky FOM is defined as:

$$\mathrm{FOM}_{\mathrm{ExaSky}} = \sqrt{\left(\frac{n_p^3}{t}\right)_{\mathrm{grav}} \left(\frac{n_p^3}{t}\right)_{\mathrm{hydro}}}\,. \tag{14}$$

As discussed, the FOM does not include subgrid models because these are continuously evolving and take up only 10–20 % of compute time, as well as because the time step size depends on the included subgrid model physics. The current FOM setup and measurements are the same as for the final challenge problem; the test size and attained performance will increase on the final exascale systems.

The FOM baseline was established on Theta, a KNL system at ALCF, by running HACC and CRK-HACC on 3072 nodes. The FOM was then measured by running the two codes on Summit at the OLCF on 4096 nodes. In 2019, the FOM ratio was 23.72. In 2020, because of improvements on the hydro solver implementation for GPUs by a factor of about 9×, the FOM ratio increased to 72.25, meaning that our effort has already achieved the target ratio of 50. We expect further improvements on the FOM in the coming year, but our focus will be mostly on adding more physics in the subgrid models.

5.2.4 ExaSky: Progress on Early and Pre-Exascale Hardware

Performance on Summit

The GPU implementation within HACC was ported from OpenCL to CUDA to better optimize performance. We restructured the code memory to use fundamental vector data types, which is important for achieving high memory bandwidth utilization on GPUs. HACC uses all six GPUs per node with each allocated to one MPI rank.

A new extreme-scale simulation called Farpoint was performed on Summit in 2020. This simulation was supported by an ALCC award and was run with ∼2 trillion particles. The box size was 1 Gpc/h with a particle mass of $\sim 5 \times 10^7$ solar masses. This is a very high mass resolution for such a large simulation box and is challenging for the code and analysis tools. This simulation provided an excellent opportunity to analyze the performance and load-balancing of the time-stepper; analyze the performance of and finalize the implementation of CosmoTools, the HACC analysis framework; and implement the final improvements to our I/O strategy. To run Farpoint effectively on Summit, we implemented a significant in situ analysis suite within CosmoTools. Improvements included a speedup of the halo finder by using GPUs and a new threaded implementation (10× speedup) and added additional tools (e.g., lightcone construction) and properties (e.g., each halo in the simulation now has 78 separate properties). In many ways, the Farpoint run functioned as an early test bed for running the challenge problem.

In conclusion, we demonstrated excellent performance and scaling for the gravity-only version of HACC on Summit, resulting in four complete science runs on 4096 nodes each. The Farpoint simulation has a higher mass resolution than the challenge problem and provided the opportunity to stress test HACC and CosmoTools, which allowed us to identify and fix all relevant bottlenecks. This particular line of development for HACC on Summit is now complete.

Regarding CRK-HACC, we have a new implementation of the CRK-SPH solver in CUDA by using a hybrid algorithm based on our fast P^3M gravity implementation. The simulation domain is decomposed into independent subvolumes that reside and evolve on the GPU, avoiding transfer latencies to the host. These subvolumes can be distributed across the entire machine to significantly improve load-balancing. The interaction trees are flattened to only interact with base leaves, simplifying the data structures and associated indexing; the high memory bandwidth of the GPU for contiguous data overcomes the efficiency loss from discarding tree levels. As a result of these improvements, we attained a 9× performance enhancement. Subgrid model implementation is proceeding quickly; all subgrid models were refined and implemented in the new GPU hybrid solver. These include radiative cooling, star formation and supernova feedback, and AGN feedback.

Next Steps

The next steps involve focused work on porting the full set of performance-sensitive kernels to the pre-exascale test beds. We will also be performing more at-scale runs on systems at ALCF (Polaris) and NERSC (Perlmutter, where both HACC and Nyx are NERSC Exascale Science Applications Program (NESAP) codes). The main priorities for HACC are: (1) efficiently implementing the hydro kernels and (2) implementing the full suite of subgrid models. There are several other important activities for running the challenge problems that relate to I/O, in situ analysis, data reduction, resilience, and comparative visualization, but progress on these is unlikely to be bottlenecked by performance issues on GPUs. Nyx development to higher AMR levels and verification tests against HACC are two important overall goals for ExaSky. The first stage of verification tests across the two codes has gone well, as described in milestone reports, and we aim to focus on achieving sub-percent agreement on certain metrics.

5.3 EQSIM

Large earthquakes present a significant risk around the world and are a large issue across the DOE mission space ranging from the safety of DOE's own inventory of one-of-a-kind mission-critical facilities to all major US energy systems (e.g., electric/gas distribution systems, renewable energy production facilities, nuclear power plants). Beyond the DOE enterprise, addressing earthquake risk, both from the standpoint of life safety and damage/economic impact, is a significant societal challenge for virtually every element of the built environment, including transportation, health, data/commerce, and all urban infrastructure. The

tremendous developments that occur in HPC, data collection, and data exploitation can help advance earthquake hazard and risk assessments. As computational power increases, the reliance on simplifying idealizations, approximations, and sparse empirical data can diminish, and attention can be focused on dealing with the fundamental physics uncertainties in earthquake processes. Regional-scale ground motion simulations are becoming computationally feasible, and simulation models that connect the domains of seismology and geotechnical and structural engineering are becoming feasible.

The EQSIM application development project focuses on creating an unprecedented computational tool set and workflow for earthquake hazard and risk assessment. Starting with a set of the existing codes—SW4, a fourth-order, 3D seismic wave propagation model; NEVADA, a nonlinear, finite displacement program for building earthquake response; and ESSI, a nonlinear finite-element program for coupled soil-structure interaction—EQSIM is building an end-to-end capability to simulate from the fault rupture to surface ground motions (earthquake hazard) and—ultimately—infrastructure response (earthquake risk). The ultimate goal of EQSIM development is to remove computational limitations as a barrier to scientific exploration and understanding of earthquake phenomenology, as wells as to practical earthquake hazard and risk assessments.

5.3.1 EQSIM: Science Challenge Problem Description

Traditional earthquake hazard and risk assessments for critical facilities have relied on empirically based approaches that use historical earthquake ground motions from many different locations to estimate future earthquake ground motions at a specific site of interest. Because ground motions for a particular site are strongly influenced by the physics of the specific earthquake processes, including the fault rupture mechanics seismic wave propagation through a heterogeneous medium and site response at the location of a particular facility, earthquake ground motions are very complex with significant spatial variation in frequency content and amplitude. The homogenization of many disparate records in traditional empirically based ground motion estimates cannot fully capture the complex site-specificity of ground motion. Over the last decade, interest in using advanced simulations to characterize earthquake ground motions (earthquake hazard) and infrastructure response (earthquake risk) has accelerated significantly. However, the extreme computational demands required to execute hazard and risk simulations at regional scale have been prohibitive. One fundamental objective of the EQSIM AD project is to advance regional-scale ground motion simulation capabilities from the historical computationally limited frequency range of 0–2 Hz to the frequency range of interest for a breadth of engineered infrastructure of 0–10 Hz. Another fundamental objective of this project is to implement an HPC framework and workflow that directly couples earthquake hazard and risk assessments through an end-to-end simulation framework that extends from earthquake rupture to structural response, thereby capturing the complexities of interaction between incident seismic waves and infrastructure systems.

To achieve the overall goals, two fundamental challenges must be addressed. First is the ability to effectively execute regional-scale forward ground motion simulations at unprecedented frequency resolution with much larger, much faster models. Achieving fast earthquake simulation times is essential for allowing the necessary parametric variations needed to span critical problem parameters (e.g., multiple fault rupture scenarios). Second, as the ability to compute at higher frequencies progresses, there will be a need for better characterization of subsurface geologic structures at increasingly fine scales; thus, a companion schema for representing fine-scale geologic heterogeneities in massive computational models must be developed. To evaluate regional-scale simulations and assess progress on the application developments of this project, a representative large regional-scale model of the San Francisco Bay Area (SFBA) was created. This model includes all the necessary geophysics modeling features, such as 3D geology, earth surface topography, material attenuation, nonreflecting boundaries, and fault rupture models. For a 10 Hz simulation, the computational domain includes approximately 203 billion grid points in the finite difference domain. The SFBA model provides the basis for testing and evaluating advanced physics algorithms and computational implementations. Challenge problem details are given in Table 43.

5.3.2 EQSIM: KPP Stretch Goal

Ultimately, soft near-surface sediments can exhibit nonlinear softening behavior under strong earthquake ground motions. The representation of soil nonlinearity can be approximated in several ways (e.g., a series of equivalent linear simulations that progressively modify soil properties); however, it would be desirable to directly model the localized nonlinearity that can occur in near-surface sediments. This is a very

Table 43: EQSIM challenge problem details.

Functional requirement	Minimum criteria
Physical phenomena and associated models	Earthquake simulations, including representative fault rupture mechanics, wave propagation through heterogenous 3D geologic structure, and appropriate coupling between regional geophysics and local soil/structure models.
Numerical approach, algorithms	Geophysics simulations will be executed with a fourth-order, summation-by-parts finite difference program (SW4) that will require extensive advancement to achieve ground motion simulation goals. Infrastructure simulations will be based on appropriate coupling of regional-scale geophysics simulations with local soil-structure models.
Simulation details: problem size, complexity, geometry, and so on	Regional-scale simulations will typically encompass a large urban region surrounding the urban environments of interest and the regional earthquake faults (sources) of interest. A representative model for the SFBA was developed with a finite difference domain, including on the order of 200 billion grid points for high-frequency resolution simulations.
Demonstration calculation requirements	The EQSIM science demonstration runs, which are performed annually in the project milestone plan to establish current application FOM, will revolve around the SFBA model with a simulation of a representative $M = 7$ earthquake on the Hayward fault and a simulation of a corresponding 90–120 s of earthquake motions. These runs measure the project's annual progress toward the exascale challenge problem.
Resource requirements to run demonstration calculation	Based on the results to date and the outlined objectives, EQSIM is estimated to require ∼80–90 % with total integrated wallclock machine time usage on the order of 90–150 h for one earthquake scenario simulation. One earthquake realization will need 3–5 h.

computationally challenging problem and would be a stretch goal for EQSIM, depending on overall progress and performance on Exascale platforms.

5.3.3 EQSIM: Figure of Merit

The EQSIM project has established an application FOM that simply and clearly expresses the computational and science goals of the overall effort. The FOM reflects the fact that the objective is to achieve high frequency simulations in the shortest possible wallclock time. The FOM is executed in reference to the frequency resolution and execution time of the regional scale SFBA model and reflects the fact that doubling the frequency resolved in the ground motion simulations requires essentially 16 times the computational effort,[3] and thus computational effort varies as frequency resolved to the fourth power. The initial application FOM was originally computed as:

$$\text{FOM} = \frac{f_{\max}^4}{\text{wallclock time} \times 7.6}\,, \tag{15}$$

where $f_{\max}$ is the highest frequency (Hertz) resolved in the regional ground motion simulation, the wallclock time (hours) is for one full rupture scenario simulation for a large earthquake (typically simulating on the order of 90 s of physical earthquake rupture and subsequent wave propagation), and 7.6 is a normalization factor so that the application is baselined to a FOM of 1.0 in the first regional scale simulation performed with the SW4 application at the start of the project with a $V_{s\,\min} = 500$ m/s in the regional model.

FOM Update

As work progressed with the performance evaluations on the regional scale model, it became apparent that it would also be desirable to explicitly reflect the dependency on the minimum geologic shear wave velocity included in the regional model because the model grid discretization is dependent on the minimum shear wave velocity in the model. Additionally, regional simulations illustrate that to achieve realistic simulations of risk, it would be necessary to reduce $V_{s\,\min}$ below 500 m/s to reflect the soft sediments on the bay margins of the SFBA model. Thus, the final application FOM is defined as:

$$\text{FOM} = \frac{f_{\max}^4}{\text{wallclock time} \times 7.6}\left(\frac{500}{V_{s\,\min}}\right)^4, \tag{16}$$

where $V_{s\,\min}$ is the smallest geologic shear wave speed included in the computational model (m/s), typically associated with near-surface soft sediments. The current FOM measurement from September 2020 on Summit is 189.

5.3.4 EQSIM: Progress on Early and Pre-Exascale Hardware

Performance on Summit

EQSIM successfully transitioned to Summit in FY19 and continued to make improvements that pushed performance boundaries in FY20 with the completion and implementation of several advanced capabilities, including:

- the development of mesh refinement for the near-surface, curvilinear portion of the SW4 geophysics code computational grid, including extensive testing to ensure the underlying fourth-order accuracy of SW4 was maintained and that the earthquake fault-rupture source crossing the depth of the curvilinear mesh was accurately represented;
- enhancements to the EQSIM workflow, including automating the coupling of regional geophysics wave propagation simulations to local engineering models of soil/building systems through the domain reduction method and the creation of a workflow to allow for the automated simulation of multiple earthquake fault rupture scenarios in a single parallel computation;

[3] Doubling the frequency resolution of the model requires reducing the mode grid size 2× in each of the three directions and halving the integration time step size, resulting in a 16× computational increase.

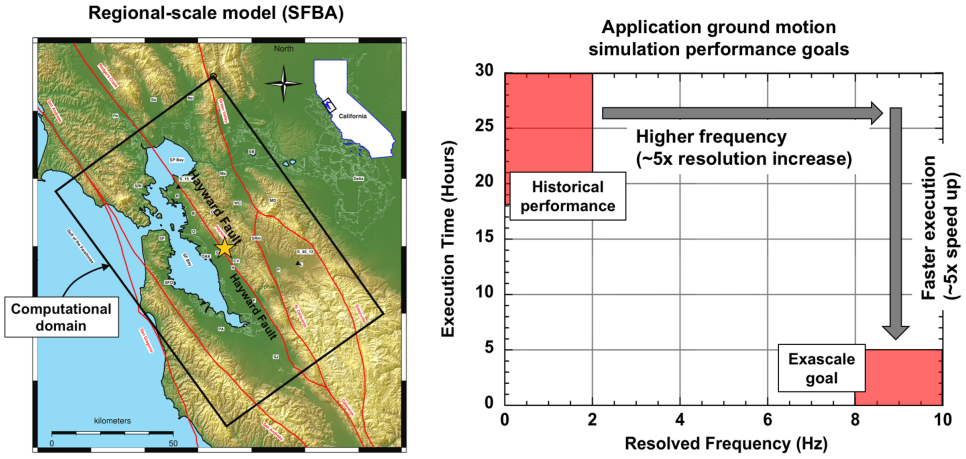


Figure 47: Extent of the SFBA regional model (left). Historical regional simulation performance before the EQSIM ECP project (upper left red box) and the EQSIM exascale goal of 10 Hz simulations in 3–5 h wallclock time (lower red box).

- extensive accuracy and performance testing of geophysics/engineering code coupling to link the fault-to-structure workflow, which included the first incorporation of nonlinear models of soft near-surface sedimentary soils in support of the EQSIM stretch goal of representation of nonlinear soil response;
- improvements to parallel I/O that provide more efficient data storage and processing based on the use of HDF5 data containers, which was particularly important to the cross-domain coupling of geophysics and engineering codes;
- the incorporation of all the aforementioned advancements in the RAJA version of SW4 for execution readiness on GPU platforms.

EQSIM continues to highly leverage ECP S&T project developments, including RAJA, to expedite the transition of all new capabilities to GPU platforms, ExaIO to improve I/O and data management capabilities, and—for the first time—Alpine/ZFP to develop data compression capabilities that will be required to fully realize the potential of fault-to-structure simulation capabilities.

The EQSIM framework performance is measured relative to a regional-scale simulation of a magnitude 7 earthquake on the Hayward fault in the EQSIM SFBA regional model, as shown in Fig 47. The ultimate EQSIM exascale goal is the ability to execute regional end-to-end simulations at a frequency resolution relevant to engineering structures, resulting in a target frequency of an unprecedented 10 Hz. Figure 48 and Table 44 illustrate the progress to date, starting with the first regional simulation of an M7 event at the start of the EQSIM project in 2017 to the best performance achieved on Summit in FY19 (point E) and most recently on Summit in September 2020 (point F). The performance advancements were achieved first through a combination of algorithm enhancements and code optimization for specific compute platforms (progress from point A to point D) and then through the transition to the Summit GPU-based system that provided a substantial performance boost (progress from point D to Point E) and finally due to the breakthrough completion of mesh refinement in the curvilinear grid of SW4 (progress from point E to point F). The corresponding increase in the EQSIM FOM is similarly illustrated in Fig. 48 in which the FOM has progressed from an FOM of 1 for the first regional scale simulations to an FOM of 189 in the most recent September 2020 performance assessment. The performance increases achieved from successfully moving to the

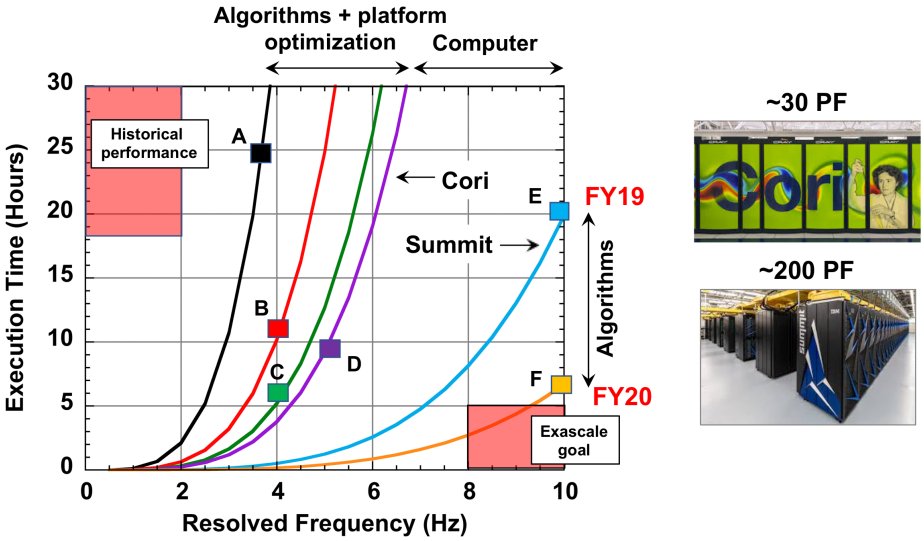


Figure 48: Performance increases realized since the start of the EQSIM project, including best FY19 performance (point E) and best FY20 performance (point F) on Summit for a minimum model shear wave velocity ($V_{s\,\mathrm{min}}$) of 500 m/s.

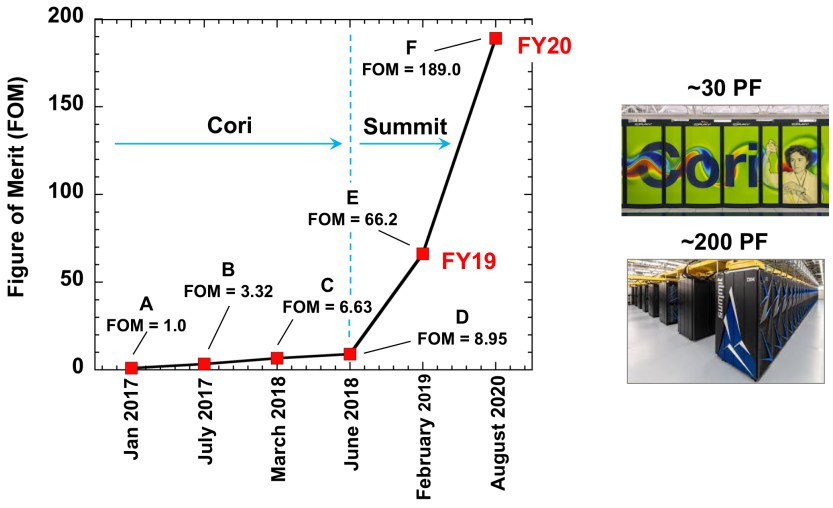


Figure 49: FOM increases realized since the inception of the EQSIM project, including the best FOM achieved in FY19 (point E) and FY20 (point F) on Summit. FOM values are the actual achieved run performance, not the performance extrapolated to the use of the full computer.

Table 44: Progression of EQSIM ground motion simulations with SW4.

Benchmark simulation (platform)	Code attributes	Frequency resolution (Hz)	Number of compute nodes	wallclock time (h)	FOM
A (Cori)	Initial run of SW4 ported to Cori	3.67	2048	23.9	1.0
B (Cori)	SW4 with optimized hybrid MPI/OpenMP loops	4.17	6528	12.0	3.32
C (Cori)	SW4 with Cartesian mesh refinement	4.17	4000	6.0	6.63
D (Cori)	SW4 using all of the Cori computer	5.0	8192 (all of Cori)	9.2	8.95
E (Summit)	Initial run of SW4 ported to the Summit computer	10.0	1200 (1/4 of Summit)	19.9	66.2
F (Summit)	Fall 2020 run of SW4 including enhanced I/O, curvilinear, and Cartesian mesh refinement	10.0	1024 (<1/4 of Summit)	6.9	189

Summit platform have been substantial, as illustrated in Fig. 48 and 49. If similar increases can be realized from the transition to exascale systems, then an unprecedented and transformational level of earthquake simulation performance will be achievable.

The performance increases illustrated in Fig. 48 are relative to a minimum model shear wave velocity of 500 m/s, which is a typical cutoff frequency used in historical simulations. Also, based on SFBA model parametric studies that have been performed, a minimum shear wave velocity of at least 250 m/s clearly must be achieved to fully represent the ground motions for soft sediment regions that surround the San Francisco Bay and thus the associated risk to infrastructure. This will require significant additional computational performance increases on exascale platforms, as discussed in the next section

Next Steps

Now that the significant multiyear task of developing and implementing mesh refinement in the curvilinear grid of SW4 is complete, the last main performance-enhancing algorithm for forward earthquake simulations has been accomplished. In 2021, several workflow improvements remain, including the full implementation of data compression to allow for the practical storage of ground motions from an entire near-surface volume of a regional domain. Compression testing performed in 2020 indicated that a very large reduction in data size can be achieved with the ZFP libraries (i.e. a reduction in stored data from 73 to 0.151 TB). This will be essential for fully realizing the ability to execute strong-coupling between regional geophysics and local engineering models throughout the regional domain on demand and fully parallelizing the local soil island/building simulations. Using the substantial performance increases achieved in 2020, the first end-to-end simulations for the SFBA at 10 Hz resolution will be performed for the annual performance assessment in 2021. Additional work on the advancement of full waveform inversions to allow for the use of historical measured small earthquake data to improve regional geophysics models will continue in 2021.

In terms of continued progress toward exascale goals, it will be necessary to achieve regional simulations with a lower $V_{s\,\mathrm{min}}$ in the regional model. Preliminary tests with a $V_{s\,\mathrm{min}}$ of 250 m/s indicated a substantial increase in computational effort with the reduction of $V_{s\,\mathrm{min}}$ from 500 to 250 m/s, as shown in Fig. 50. The achievement of fast regional simulations at 250 m/s will require exascale platforms.

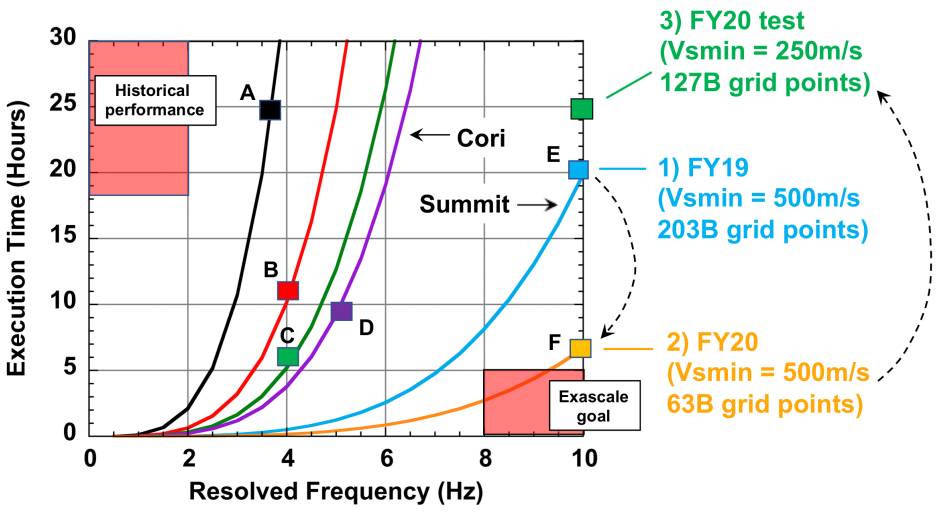


Figure 50: Preliminary exploration of SFBA regional simulations with $V_{s\,\mathrm{min}}$ lowered to 250 m/s; point 2 indicates a 10 Hz SFBA simulation with $V_{s\,\mathrm{min}} = 500$ m/s, and point 3 indicates a 10 Hz SFBA simulation with $V_{s\,\mathrm{min}} = 250$ m/s.

5.4 Subsurface

Understanding and predicting reservoir-scale behavior affected by the long-term integrity of the hundreds of thousands of deep wells that penetrate the subsurface for resource use are urgent challenges. The performance of a wellbore hinges on the behavior of very thin interface features that control the leakage of fluids along the boundary between the well casing and cement. Similarly, the leakage of buoyant fluids (e.g., CO_2) through caprocks might be controlled by micron-scale asperities in fracture networks that are themselves subject to geomechanical and geochemical modification. At the reservoir or field scale (∼1–10 km domain size), multiphase flow and reactions in fractured porous media are typically modeled by using continuum models that use averaged quantities and bulk parameters that do not fully take into account thermal-hydraulic-chemical-mechanical (THCM)-related heterogeneity at different spatial and temporal scales. A more rigorous treatment is to resolve the pore-scale (0.1–10 µm) physical and geochemical heterogeneities in wellbores and fractures to improve the ability to predict the evolution of these features when subjected to geomechanical and geochemical stressors. The ultimate challenge is to integrate the complex multiphysics processes that occur at multiple scales, from the micro to the kilometer scale in a high-resolution reservoir simulator. Meeting this challenge requires the use of innovative multiscale coupling approaches and exascale computing.

The Subsurface project addresses this exascale computing challenge by coupling two mature code bases: (1) Chombo-Crunch, developed at LBNL, which currently handles Navier-Stokes and Darcy flow coupled to multicomponent geochemical reaction networks, and (2) the GEOSX code, developed at LLNL, which handles geomechanical deformation and fracture+Darcy flow at a variety of scales.

A science challenge problem was developed that focuses on the evolution of a single fracture in wellbore cement, beginning at Stage 1 with diffusion-controlled reaction and weakening of the cement that leads to fracturing. The propagation of the fracture as a result of further chemical reaction and fluid pressure-driven deformation is simulated with 1 µm resolution within the fracture and is coupled to a coarser resolution (10 µm) representation of the porous cement adjacent to the evolving fracture. The resulting challenge problem is estimated to require 1 trillion grid cells with 16 trillion DOF once the hydraulic, mechanical, and chemical variables are included. Based on prior experiments and modeling, the challenge problem is estimated to extend for 10 d of simulation to capture the evolving fracture and associated reaction fronts.

5.4.1 Subsurface: Science Challenge Problem Description

There are a wide range of processes that occur in the subsurface that involve the evolution of fractures, including opening and closing due to some combination of mechanical and chemical stresses. In this project, the team focuses on the failure of a wellbore for CO_2 sequestration in saline reservoirs as the single science challenge problem with the consideration of a wellbore segment of up to 100 m and times up to 1 year. Wells are considered to be high-risk pathways for fluid leakage from geologic CO_2 storage reservoirs because breaches in this engineered system could connect the reservoir to groundwater resources and the atmosphere. The geologic carbon storage community has raised further concerns about wellbore stability because acidic fluids in the CO_2 storage reservoir, alkaline cement meant to isolate the reservoir fluids from the overlying strata, and steel casings in wells are inherently reactive systems. This is of particular concern for storing CO_2 in depleted oil and gas reservoirs with numerous legacy wells engineered to variable standards.

The wellbore stability problem involves four physical processes that must be considered to model the challenge problem.

1. Geochemically driven fracture initiation: Initial crack growth near the wellbore occurs as a result of chemical corrosion when acidic CO_2 contacts alkaline cement. In these zones, enhanced transport rates contribute to chemical dissolution and weakening of the cement. Transport is expected to be dominated by diffusion due to the initially low permeability of the cement, leading to a compact reaction front in the porous cement.

2. Mechanically driven fracture propagation: This process involves the growth of a fracture based on the stress/deformation field near the fracture tip. A fracture criterion is given for the rock; fracturing will occur when this criterion has been exceeded. In the context of the challenge problem, this process is driven by the fluid pressure within the fracture, which concentrates stress at the fracture tip.

3. Fracture sealing: This process is driven by reactive flow in the open fracture and can include: (1) the closure of flow pathways by the mechanical compression of the asperities (i.e., collapse of fracture pillars) or (2) the deposition of minerals in the fractures due to their supersaturation.

4. Chemically induced fracture growth: This process is also driven by reactive flow in the open fracture and can include: (1) sustained fracture growth by dissolution of the cement, potentially leading to "wormholing," and (2) increased stress due to the deposition of minerals in the fracture (i.e., mineral precipitation-induced fracturing). The second phenomenon can occur where the Gibbs free energy for mineral precipitation exceeds the strength of the rock.

Subsurface energy applications, including the science challenge problem, are modeled at a large scale known as the *field* or *reservoir scale*, $O(0.1\text{–}1\,\text{km})$ and $O(10\,\text{y})$, with spatial resolution as fine as 1 mm locally (e.g., near the wellbore) and time steps of minutes to hours for computationally tractable simulations. Problems are typically analyzed with limited or no coupling between mechanical, hydraulic, and chemical processes that control fracture initiation and growth. Furthermore, the equations of motion are based on an effective medium, parameterizing subgrid flow, and transport processes as bulk properties (e.g., permeability and reaction rate) that do not represent the true tortuosity of flow paths or the reactive surface area of the material.

In contrast to the conventional treatment of wellbore failure, the accurate prediction of fracture evolution depends on the microscale resolution of fracture asperities (i.e., pillars) controlling permeability and chemical reactivity. In particular, high resolution is needed in the vicinity of the fracture tip where chemical corrosion combines with the focused stress field to propagate the fracture. As in the classical subcritical fracture growth literature [34], the overall rate of fracture growth in these zones is controlled by coupled processes that occurs at the fracture tip.

Microscale resolution is also needed to accurately predict fracture permeability since real rough fractures are typically held open by asperities (i.e., pillars) of this scale. Chemical corrosion (i.e., dissolution) or mechanical corrosion (i.e., pressure solution) of these asperities occurs at the same micron scale. Chemical dissolution might actually have two opposing effects: (1) the formation of channels or wormholes in the fracture plan that focus flow, thus accelerating the opening of the fracture, or (2) the dissolution of fracture asperities, thus allowing the fracture to close under the ambient confining stress. The overall domain size of

Table 45: Subsurface challenge problem details.

Functional requirement	Minimum criteria
Modeled physics	Coupled flow, multicomponent reactive transport, and fracture mechanics and deformation
Numerical approach, algorithms	Chombo-Crunch: AMR, finite volume embedded boundary method for flow and transport, level set GEOSX: finite element solid mechanics
Domain size	$1\,cm^3$ (e.g., $10\,cm \times 1\,cm \times 0.1\,cm$) near fracture tip
Grid resolution midrule Number of grid cells	1 µm 1 trillion
Degrees of freedom	16 trillion (six flow/mechanical variables, 10 solute transport variables)
Domain decomposition and load balancing	32,768 grid cells per box per core
Resource requirement to run exascale challenge problem demonstration (based on current architecture)	475,000 Cori KNL nodes (50× more than available); 80,000 Summit nodes
Simulation time for exascale challenge computation	10 days of simulated diffusion-reaction in Portland cement, reaction-induced fracture evolution ⇒ 8640 global time steps ⇒ 4 weeks of machine time

the fracture tip, O(cm), is important because the resulting fractures can become the conduits for reactive flow (CO_2 saturated brine) in contact with highly reactive alkaline cement, potentially leading to wormholing that causes more rapid wellbore cement failure and "runaway" borehole failure.

For the exascale challenge problem, one fracture tip with pore-scale resolution is tracked. The localized subdomain needed to resolve reactive transport processes at microscale resolution during fracture propagation is a domain size up to 10 cm (in the length of the wellbore)×1 cm (along an azimuth in the cement annulus)×1 mm (in the radial direction) with 1 µm resolution. This domain size is assumed to be the minimum domain needed to capture coupled reactive transport and mechanics effects in a fracture (e.g., pillar collapse). Up to 10 cm is an adequate length with respect to the aspect ratio of the fracture. In the cross section of the fracture, 1 cm in the azimuthal direction accounts for the length scale for an REV. A domain on the order of 1 mm in the radial direction is required to capture diffusive transport over long timescales.

The resolution of 1 µm is required to explicitly resolve the reactive surface area of reacting materials in the cement fill of the wellbore. This domain-to-grid resolution ratio at the pore scale conservatively requires the equivalent of 30,000,000 KNL cores (475,000 KNL nodes) based on petascale Chombo-Crunch domain decomposition and load-balancing sweet spot of one 32^3 box of cells per core. With a current benchmark for petascale capability of 600,000 KNL cores (9000 nodes) for 24 nm resolution of a 100 µm block of shale, the challenge problem is well into the regime of exascale resources because it is 50× the current capability. Resources based on Summit Volta GPUs can also be estimated. Because Summit has the same HBM as Cori KNL (16 GB HBM) and six cards per GPU node, the requirement would be 80,000 Summit nodes. The problem specifications are summarized in Table 45.

5.4.2 Subsurface: KPP Stretch Goal

The stretch problem focuses on simulating 100 m of a single wellbore with proposed cell resolutions down to 2 mm. The specifications for this problem are listed in Table 46. The wellbore scale component of this problem is a GEOSX simulation that uses continuum/Darcy scale assumptions with coupled physics that comprise solid mechanics, multicomponent multiphase flow, and fracture mechanics. Although the wellbore scale (GEOSX) component of the stretch challenge problem will be developed independently of the base

Table 46: Subsurface stretch problem details.

Functional requirement	Specification
Modeled physics	Fracture mechanics, multiphase multicomponent flow, reactive transport/geochemistry
Numerical approach, algorithms	GEOSX: finite element solid mechanics with REV-based upscaling from Chombo-Crunch base challenge problem simulation data
Domain size	100 m × 0.8 m × 0.1 m high-resolution zone
Grid resolution	2 mm resolution
Number of grid cells	∼800 million elements
DOF	∼10 billion DOF
Domain decomposition and load balancing	Two ranks per node
Resource requirement to run stretch problem (based on current architecture)	1600 Summit nodes
Simulation time for stretch goal	6 months of simulated time; execution time depends on the performance of the team's linear solver strategy

challenge problem, it is intended that the capabilities from the base challenge problem will be coupled with the wellbore scale through methods developed within the project (ADSE05-21).

5.4.3 Subsurface: Capability Plan

To achieve the target simulation of the exascale challenge problem on Frontier by the end of the project, development is divided into two tracks: multiphysics capability demonstration and performance portability software for accelerator-based machines. The first peg point between the two tracks was September 2020 when a multiphysics capability demonstration was targeted for the end of FY20 as a milestone but is essentially a milepost, and underlying software portability to GPUs for Chombo-Crunch and GEOSX was also pegged at the end of FY20 as a milestone. Performance portability for each new architecture thereafter precedes multiphysics capability demonstration on the architecture. Multiphysics capability on a given architecture depends on the performance portability of Chombo-Crunch and GEOSX individually on the architecture before a full multiphysics coupled code demonstration.

1. September 2020: Multiphysics capability demonstration. Subsurface is developing a multiphysics capability that couples Chombo-Crunch flow and reactive transport with GEOSX mechanics to model the fracture evolution of CO_2 invasion in wellbore cement. In this milepost, experimental data were used to validate the coupling approach that we propose to use to solve the challenge problem. We modeled the open cement fracture fabricated for experiments in Walsh, Mason, Frane, and Carroll [35]. The domain is approximately 3.5 cm long and 1.5 cm × 1.5 cm in the cross section. We simulated the problem at the pore scale by using Chombo-Crunch with approximately 5 µm resolution. GEOSX simulates the mechanical deformation as a whole on a similar size domain with 100 µm resolution. The domain-to-resolution ratio for this problem is a necessary and intermediate step toward the exascale challenge problem of 10 cm × 1 cm × 0.1 cm with 1 µm resolution. The algorithmic and modeling components of this milepost are represented in the stories of Jira Epic 98 (milestone ADSE05-20).
   (a) Chombo-Crunch continuum Darcy-scale diffusion-reaction capability with variable coefficient porosity.
   (b) Pore-continuum coupling with fracture evolution.
   (c) Representation of reaction-induced solid mass modifications in GEOSX.

(d) Experimental validation of wellbore cement geochemical reactions [**Li:2017**].

(e) Experimental validation of wellbore cement alteration by CO_2-rich brine [35].

2. September 2021: GPU-based multiphysics capability demonstration on Summit. This milepost extends to a pre-exascale GPU-based machine (i.e., Summit). We will use the fully coupled code, Chombo-Crunch, implemented in Chombo4, which is based on the performance portability back end EBProto.

3. June 2022: Capabilities for the simulation of a wellbore problem in GEOSX.

4. June 2023: Performant multiphysics simulation of challenge problem on Frontier. Subsurface is a simulation of single-fracture propagating in wellbore cement due to an attack of CO_2-saturated fluid. We will track a high-resolution zone $10\,\text{cm} \times 1\,\text{cm} \times 0.1\,\text{cm}$ ($= 1\,\text{cm}^3$) with 1 µm resolution in the fracture (Chombo-Crunch) and 100 µm resolution of the entire cubic centimeter volume (GEOSX). This domain size is the minimum domain needed to capture the effects of coupled flow, reactive transport, and mechanics in a fracture.

5.4.4 Subsurface: Progress on Early and Pre-Exascale Hardware

The performance portability of Chombo-Crunch on Summit GPUs is being accomplished by implementing the C++ library called Proto. Proto is an abstraction layer that provides a high-level representation for discretization operators on data defined on a single rectangle, replacing the low-level Fortran/C-subset approach currently used. In Proto, the principal data type (i.e., class) is a collection of values defined at each point in a rectangular patch (`BoxData`). `BoxData` corresponds to a multidimensional rectangular array but with the rectangular patch corresponding to the indices over which the array is defined as a separate data type, similar to the `FArrayBox` class in Chombo and BoxLib. The main change from the previous approach in Chombo is that all the performance-critical discretization methods that were represented as low-level loops are now represented in applications code as compositions of two high-level operators applied to `BoxData`s.

- Application of stencils (apply). In Proto, stencils are first-class objects that are separately defined and archived.

- Pointwise applications of user-defined functions applied to the values of the `BoxData` at each point in the rectangular patch (`forall`).

Furthermore, Proto is not limited to Chombo-based applications.

However, Chombo-Crunch is an application code that relies heavily on embedded boundary, finite volume discretizations of operations and data. EB calculations are primarily used to compute solutions on complex geometries, but they also provide a general approach to tracking sharp interfaces, such as those by multiple media (e.g., multiphase, multimodel). Because the nature of these algorithms is semistructured, EB calculations require more complex data structures. The EBProto performance portability layer extension to non-EB Proto is designed to meet this complexity.

- The stencils, which vary from point to point on cut cells, are computed on the host and evaluated on the device.

- Data live on a graph structure that can be more complex than a simple array. These data live on the device and require complex indexing.

- Structures are built for fast stencil evaluation and pointwise function evaluation.

- We provide a dictionary of stencil implementations that users can easily accessed. Most users will never have to see the graph or moment information.

Our performance engineering strategy is to extend the non-EB Proto performance engineering optimizations to EBProto. Because Proto has no EB capability, EBProto is an extension of Proto that requires new data holders and geometry generation in Chombo4. We extend `Proto::forall` and `Proto::stencil` to EB. Furthermore, for EBProto to support Chombo-based application codes, such as Chombo-Crunch, a

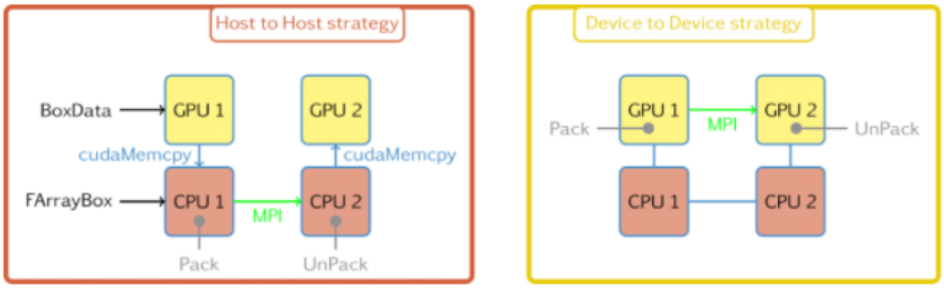


Figure 51: Comparison of host-to-host strategy vs. device-to-device strategy.

Table 47: Weak-scaling performance optimizations on Summit. All times are measured in seconds.

	Six GPUs, one node	3072 GPUs, 512 nodes
	Total time (exchange)	Total time (exchange)
Baseline	1208 (957)	1304 (947)
Device-to-device	304 (43)	398 (91)
Other optimizations	115 (30)	180 (70)

Proto/EBProto-enabled version of Chombo, called Chombo4, was developed. The next section reports the performance results for benchmarks on GPU architectures.

Performance on Summit

We identified several performance bottlenecks on Summit along with our solution outcome.

- We replaced the exchange of ghost data mediated by the CPU with functionality that is completely resident on the GPU, gaining more than a 10× improvement in performance of exchange. A schematic is shown in Fig. 51.
- Integer stencil offset calculations on GPU are slow. We precomputed offsets on CPU and bundled them in kernel launch payload.
- `nvcc` does not inline pointwise functions on the device. We wrapped them in a class that enables optimization analogous to inlining.
- We replaced Thrust library calls with native implementations for arithmetic operations on arrays and use special syntax for accumulating norms to minimize barrier synchronizations on the device.
- We minimized kernel launch costs by fusion, which are not visible to the application.

The weak-scaling performance from these optimizations are shown in Table 47.

The following rooflines were generated with the Chombo4::Euler example and nsight-compute. We observe the following in Table 48 and Figs. 52 to 55:

- the whole application is memory-bound (Fig. 52);
- three main kernels account for ∼67 % of the total time;
- kernels are memory-bound (Figs. 53 to 55); and
- the GFlops are two times higher for this simulation than in Table 48 because the patch size influences the workload.

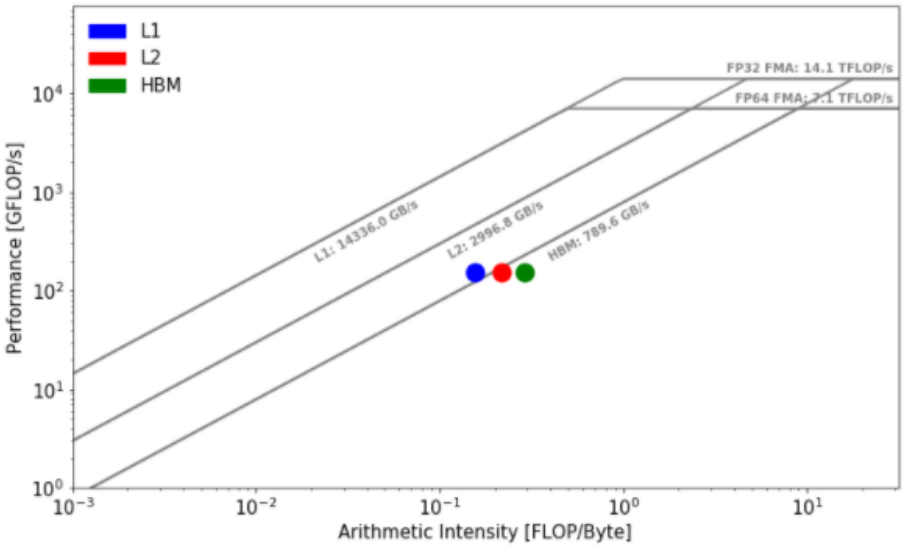


Figure 52: Roofline of all kernels.

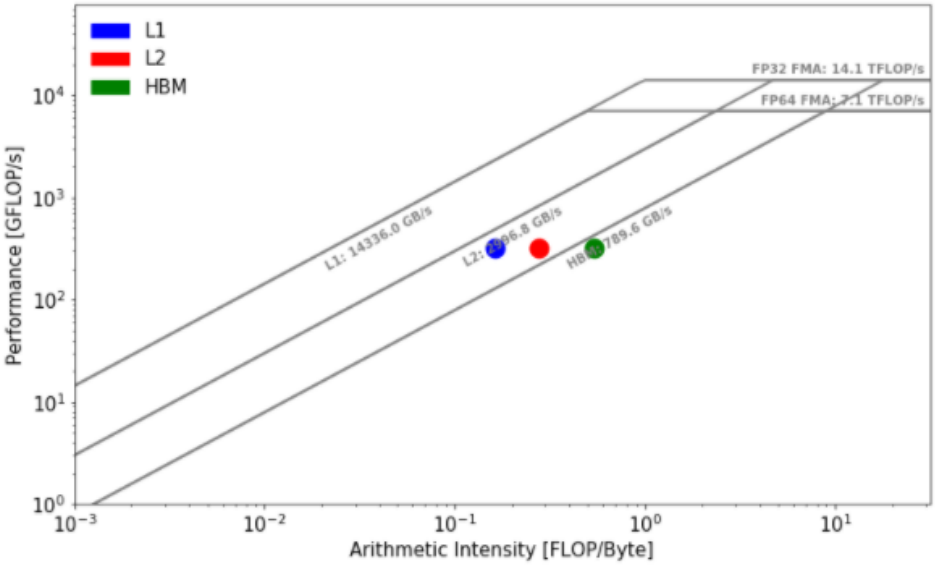


Figure 53: Hierarchical roofline of stencilIndexer kernel (all stencils together).

Table 48: Kernel performance information used to build rooflines. Data are obtained from NSight Compute with the Chombo4/Euler example with eight patches of 64^3 elements and one step. Here, AI is the arithmetic intensity.

	Ratio (%)	GFlops	L1 AI	L2 AI	DRAM AI
Whole simulation (kernel only)	100	156.1513	0.1561	0.2183	0.2918
`StencilIndexer` (all stencils)	27.8	320.3305	0.1640	0.2765	0.5437
`Indexer::getFlux`	21.8	75.5640	0.0931	0.1231	0.1373
`Indexer::upwindState`	18.0	145.4033	0.1932	0.2302	0.2517

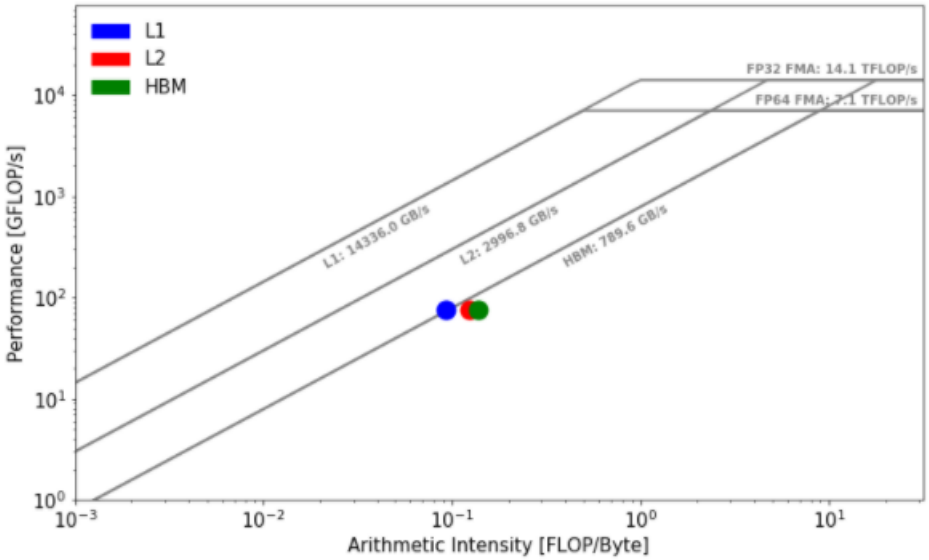


Figure 54: Hierarchical roofline of indexer:getFlux.

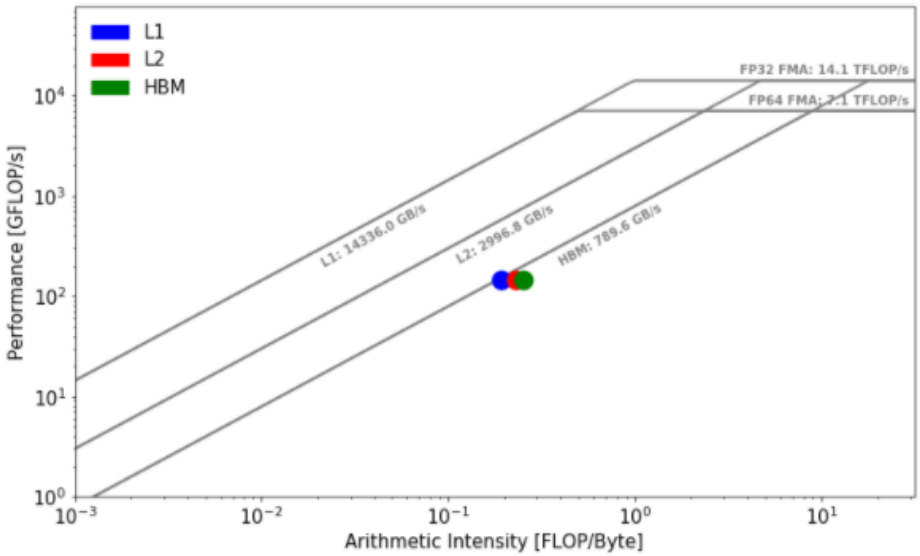


Figure 55: Hierarchical roofline of indexer:unpwindState.

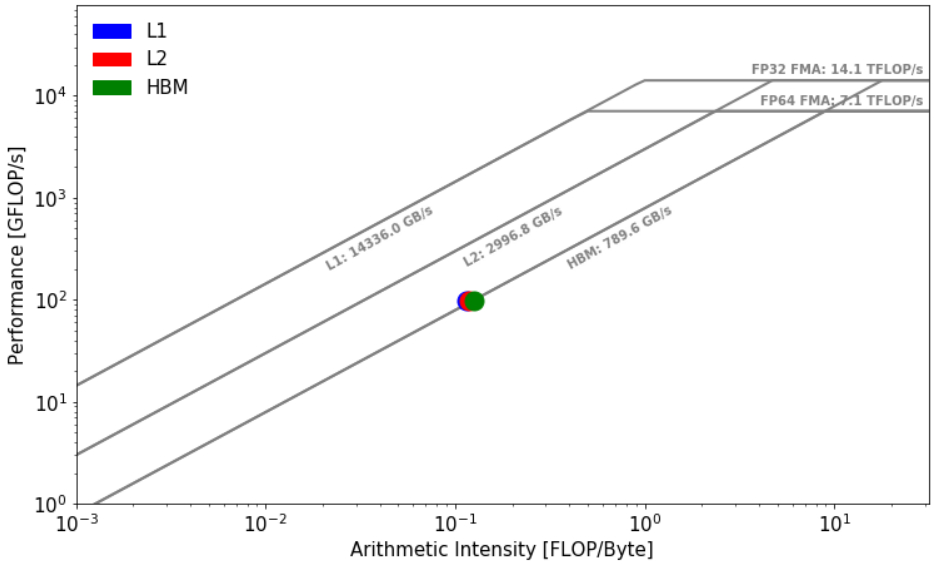


Figure 56: `Indexer` (`Forall`): 98.2 GFLOPS `ebkernel1`.

EBHelmholtz benchmark The following rooflines in Figs. 56 to 59 were done with the applyHelmholtz test and nsight-compute for our EBHelmholtz benchmark. The kernels—`Indexer`, `indexer_i`, and `StencilIndexer`—are memory-bounded. Regarding EBStencil, there are two issues for the poor performance: (1) no coalesced memory access from noncontiguous memory (2) and the workload is too low to achieve decent performance. EBProto will benefit greatly from the device-to-device strategy for the exchange used in Proto depicted in Figs. 51 (work in progress).

5.5 E3SM-MMF

The goal of the Energy Exascale Earth System Model (E3SM)-MMF project is to develop a cloud-resolving earth system model with the throughput necessary for multidecade, coupled high-resolution climate simulations. This next-generation model could substantially reduce significant systematic errors in precipitation found in current models due to its more realistic and explicit treatment of convective storms. Consequently, it will improve the ability to assess regional impacts of climate change on the water cycle that directly affect multiple sectors of US and global economies, especially agriculture and energy production. Current earth system models possess a limited ability to model the complex interactions between the large-scale, mostly 2D baroclinic atmospheric motions and the smaller scale 3D convective motions found in clouds and individual storms. These motions and their interactions, to the first order, determine the spatial distributions and characteristics of regional precipitation. Complexities include the microscale chemistry and physics of cloud formation and the impacts of anthropogenic climate change on cloud formation. Properly resolving the key processes involved in cloud formation requires resolution (grid spacing) on the order of 1 km in the atmosphere. It is possible to run such resolution on today's $O(10)$ petascale computing systems but only at great expense and for very short times (i.e., several simulated days). Running conventional climate models at this resolution, for 100 y simulations, requires a 5000× increase in computing resources.

For exascale, the team thus considers a multiscale modeling framework (MMF) approach to cloud-resolving modeling, often referred to as *superparameterization*, which offers significant opportunities for unprecedented model skill improvement that has not yet been fully explored due to limited computing resources. This project will integrate a cloud-resolving convective parameterization (i.e., superparameterization) into the DOE E3SM by using the MMF and will explore its full potential to scientifically and computationally advance climate simulation and prediction. The superparameterization will be designed to fully use GPU-accelerated systems and will also involve refactoring and porting other key components of the E3SM model for GPU systems. The acronym *E3SM-MMF* refers to the superparameterized version of the E3SM being developed

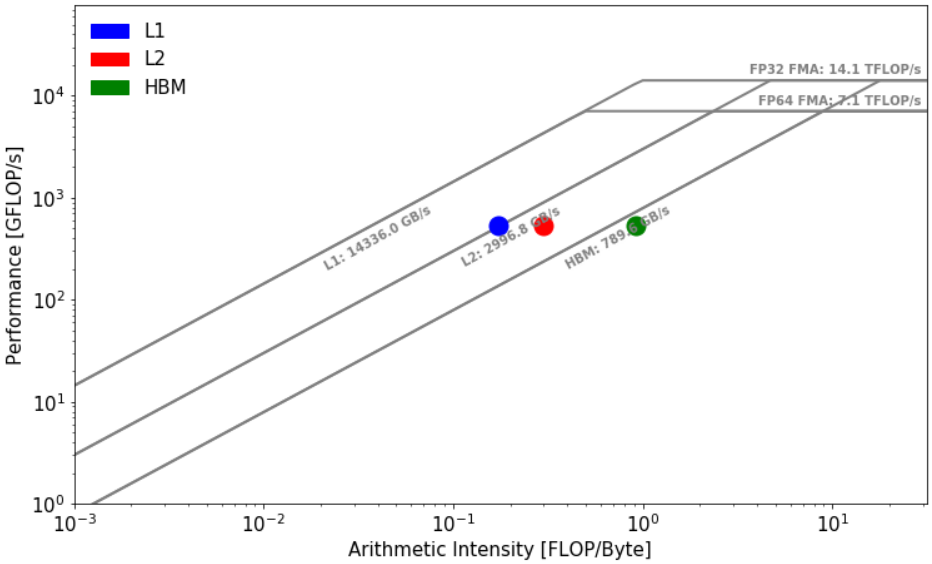


Figure 57: `StencilIndexer`: 531 GFLOPS `ebkernel2`.

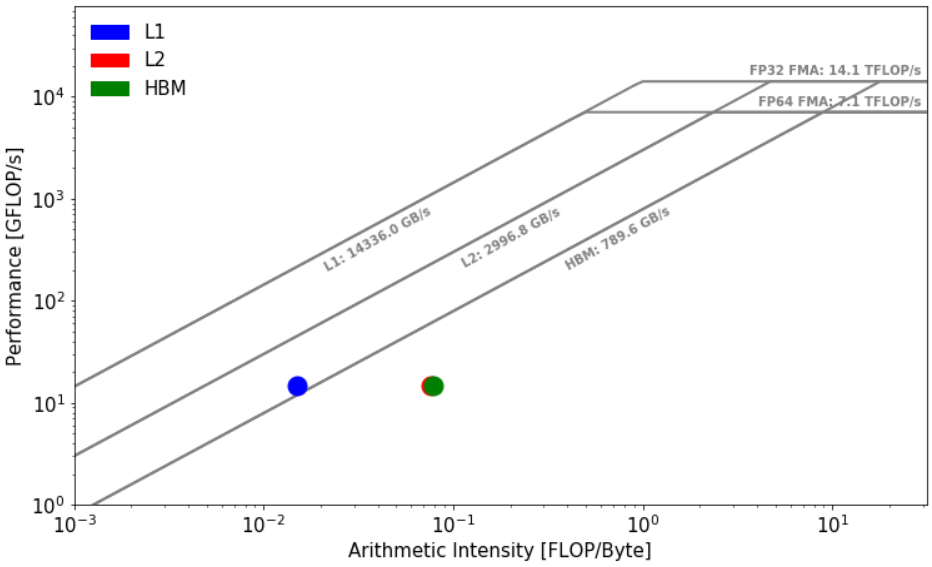


Figure 58: `AggStencil` (`EBSTencil`): 14.77 GFLOPS `ebkernel3`.

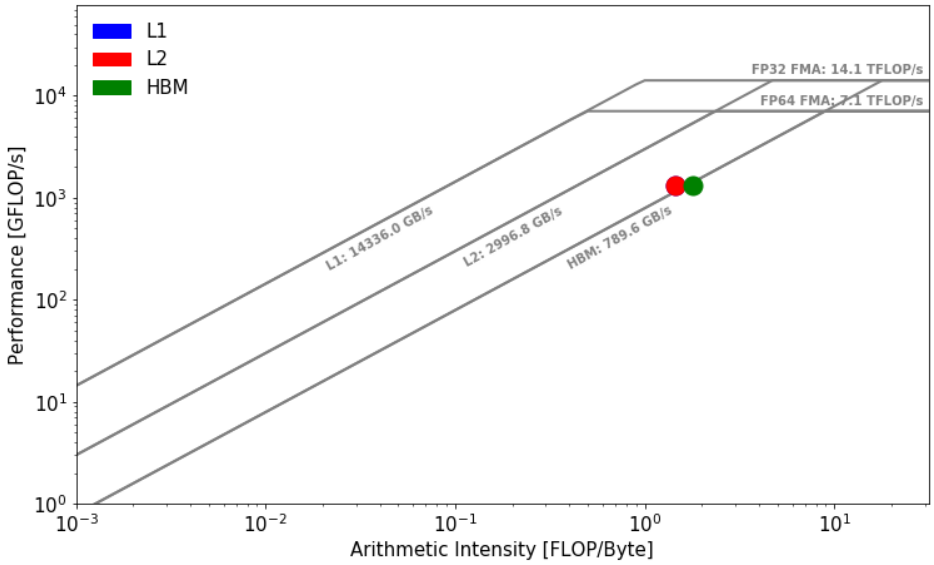


Figure 59: `Indexer_i` (`ebForall`): 1308 GFLOPS `ebkernel4`.

under this ECP project.

5.5.1 E3SM-MMF: Science Challenge Problem Description

The overarching challenge problem is to develop an Earth system model with a fully weather-resolving atmosphere and cloud-resolving superparameterization, an eddy-resolving ocean, and ice components, all while obtaining the necessary throughput to run 10–100 member ensembles of 100 year simulations in less than 1 calendar year.

The challenge problem size has several aspects. The first is to achieve cloud-resolving resolution in the atmosphere superparameterization, which is defined as at least 1 km grid spacing in both horizontal and vertical directions. The second is to achieve weather-resolving resolution in the global atmosphere model, which is defined as 50–25 km average grid spacing in the horizontal directions with ∼1 km grid spacing in the vertical directions—(the resolution of today's global operational forecast models. The third is to achieve an eddy-resolving ocean/ice model, which is defined as a minimum 18 km resolution in equatorial regions, decreasing to 6 km in polar regions to capture the reduction in eddy size with decreasing Rossby radius of deformation with $O(100)$ levels in the vertical. The final aspect is to have the model throughput needed to perform the simulation campaign of the challenge problem in the course of 1 calendar year on the exascale Frontier system. The team's minimum requirement (10 member ensemble of 100 y simulations) requires the ability to run 1000 simulated years in 1 calendar year, which can be achieved with a throughput rate of 5 simulated-years-per-day (SYPD). Ideally, each ensemble member will run at 5 SYPD, but this throughput can also be achieved if $NX = 5$, where N is the number of ensemble members that can run on the machine simultaneously, and X is equal to the SYPD performance of each ensemble member.

The E3SM-MMF will be evaluated by using the E3SM water cycle metrics package, which is under development by the E3SM project and will measure the ability of the model to simulate extreme storms and coastal inundation. The accuracy requirement will be to obtain accuracy similar to or better than the high-resolution E3SM model while running 50× faster, as measured by the FOM. Because of the large natural variability and chaotic nature of atmospheric and ocean dynamics, a rigorous assessment of these metrics requires large ensembles of century-length runs identified in the challenge problem, which require a large INCITE-class computing allocation. The team's challenge problem is a typical example of a simulation campaign used for Earth system science studies. The demonstration calculation (Table 49) will use much fewer resources and is designed to show that E3SM-MMF can achieve the computational performance and stability needed to complete the challenge problem. The performance will be established with a suite of short

Table 49: E3SM challenge problem details.

Functional requirement	Minimum criteria
Physical phenomena and associated models	E3SM-MMF is an Earth system model focused on simulating the Earth's water cycle. It is made up of physical models of the Earth's atmosphere, ocean, and land and sea ice.
Numerical approach, algorithms	Finite elements, finite volumes, and finite difference running on unstructured grids with multiscale coupling and an extensive suite of subgrid parameterizations.
Simulation details: problem size, complexity, geometry, and so on	Weather-resolving atmosphere with a cloud-resolving superparameterization coupled to eddy-resolving ocean and ice components with the necessary throughput for 10–100 ensembles of 100 year simulations.
Demonstration calculation requirements	Computational performance metrics require strong scaling benchmarks out to the full machine size, using short simulations (5 simulated days, $\sim O(5000)$ time steps).
Resource requirements to run demonstration calculation	The full exascale machine for 12 h and 20% of the machine for 20 d.

5 d strong-scaling benchmark calculations, and the stability of the model will be established with a single multiyear simulation.

5.5.2 E3SM-MMF: KPP Stretch Goal

The stretch goal is to develop an Earth system model with a fully cloud-resolving 3 km atmosphere component, as well as an eddy-resolving ocean and ice components, all while obtaining 1.0 SYPD for a single ensemble member on Frontier or Aurora. For the stretch goal, the team will use an E3SM configuration in which the full atmosphere—not just the superparameterization—is run at a global 3 km cloud-resolving resolution. This E3SM configuration currently serves as both a baseline to evaluate a cloud-resolving capability and as a risk mitigation strategy if they are unable to obtain some aspects of a cloud-resolving model with the superparameterization approach in the E3SM-MMF configuration.

5.5.3 E3SM-MMF: Figure of Merit

The E3SM-MMF's project FOM is the throughput of a cloud-resolving Earth system model measured in SYPD.

For the baseline, the team compares against the traditional E3SM model running at a global 3 km resolution. Because it is currently impossible to run this resolution on Titan, benchmarks of the E3SM high-resolution configuration (28 km) that run on 20% of Titan are used, and then these results are scaled to 100% of Titan and to 3 km resolution. For the FOM speedup, the team will compare the baseline FOM to the performance of the E3SM-MMF model that runs with a cloud-resolving convective parameterization on Summit with GPU acceleration. This FOM speedup combines an algorithmic speedup from the MMF approach with GPU acceleration. The cloud-resolving convective parameterization will be run at a resolution of at least 3 km and potentially as fine as 1 km.

Throughput was measured in SYPD without I/O. I/O was excluded to simplify the benchmarking procedures. The amount of I/O is very problem-dependent, and it typically varies from 10 to 50% of the total cost of the model; thus, including I/O would not substantially impact the FOM. The team has tasks focused on improving the I/O infrastructure, and these tasks measure their performance through I/O rate benchmarks.

To estimate the throughput of the full Earth system model, two standardized benchmarks of simpler configurations are used: an "F compset" and a "G compset." The F compset isolates the performance of the atmosphere and land components, and the G compset isolates the performance of the ocean and ice

Table 50: E3SM-MMF FOM data

Model	Machine	Nodes	FOM (SYPD)	Speedup
Baseline	Titan 2018/12	18 700	0.011	1
E3SM-MMF	Titan 2018/9	2700	0.095	8.6
E3SM-MMF	Summit 2019/2	1024	0.395	35
E3SM-MMF	Summit 2020/6	1024	0.81	73

components. Benchmarking these components separately makes it much easier to collect strong-scaling data. It is difficult to collect strong-scaling data for the full coupled system as for a given number of nodes; optimal load balancing and processor layouts must be constructed. This is not difficult, but it is nontrivial and time-consuming. Based on the performance data collected in F and G compsets, the team can get an excellent estimate of the performance of the coupled system via

$$\text{time-to-solution} = 1.2 \times \max(\text{ocean time}, \text{atmosphere} + \text{ice time}) \,. \tag{17}$$

This formula is based on current data, which show that coupling between components adds 20% to the overall cost, and on component concurrency (that atmosphere and ice model run sequentially with respect to each other on the same nodes, while the ocean model runs concurrently on a different set of nodes).

FOM Update

Our baseline FOM was obtained on Titan. Our first benchmarks resulted in an FOM of 0.005 SYPD, which was obtained on March 30, 2018. These were further improved on December 21, 2018, resulting in our final baseline FOM of 0.011 SYPD. These were computed as described previously. In the G compset, the ice component model obtained 6.88 SYPD, and the ocean model obtained 3.46 SYPD. For the atmosphere, we ran an F compset at 27 km resolution (2.74 SYPD) on 5400 Titan nodes. This number was scaled to the full Titan and to 3 km resolution, resulting in 0.013 SYPD. The end result was our baseline-coupled model FOM of 0.011.

For the E3SM-MMF results, we have been collecting F compset performance numbers as we improve the GPU acceleration and port to new hardware. By computing the FOM via these E3SM-MMF F compset benchmarks and the existing G compset benchmarks, we obtained the data displayed in Table 50.

5.5.4 E3SM-MMF: Progress on Early and Pre-Exascale Hardware

Performance on Summit

Our performance work is focused on the atmosphere and MPAS ocean/ice components of the E3SM. For the MPAS components, we are porting the original Fortran code to GPUs by using a Fortran/openACC model. In 2020, we completed the port of several subcomponents and much of the necessary framework refactoring. For the atmosphere component, we are only porting the key computational subcomponents, which include SAM, the cloud-resolving super-parameterization, and RRTMGP, the radiation package. F-case atmosphere simulations have been running well on Summit since its debut with our initial Fortran/openACC port of SAM. In 2020, we started working to transition away from openACC to support both Frontier and Aurora. We explored Fortran/OpenMP and a C++ approach. The OpenMP work has had many difficulties due to the immaturity of OpenMP Fortran support. We obtained our best results by porting our computational kernels into C++. For this work, we explored by using Kokkos but settled on YAKL, an internally written parallel for loop abstraction layer that supports Fortran style arrays, making it easier to port existing Fortran code. We completed the port of SAM into SAM++ and the port of RRTMGP into RRTMGP++ and integrated SAM++ into E3SM-MMF. Our best results on Summit are obtained with the full E3SM-MMF atmosphere model (F compset) running in a configuration with SAM++ via a 3D grid. This configuration has a relatively small cost of radiation and can thus obtain excellent GPU speedups, even though we have not yet integrated RRTMGP++ into E3SM-MMF. This model is currently being used for our science simulation campaign supported by our 2020 INCITE allocation.

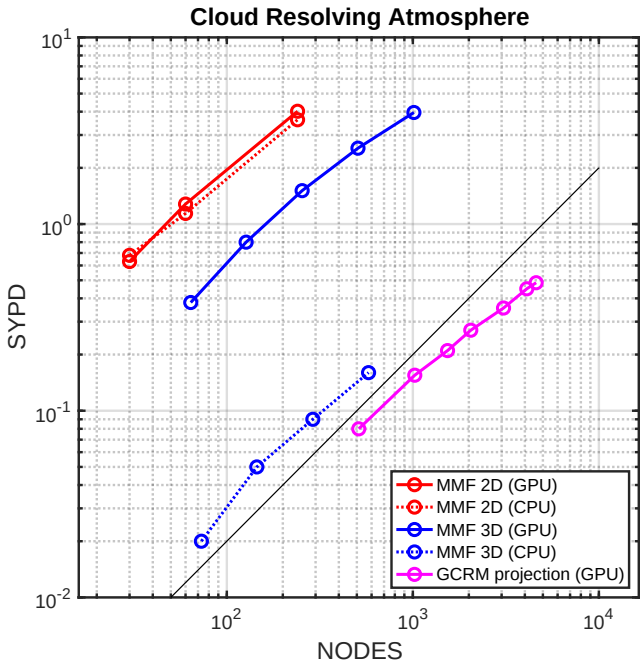


Figure 60: Strong scaling of several configurations of the E3SM. The atmosphere component performance, as measured in simulated-years-per-day, is plotted as a function of the number of Summit nodes used. The MMF configurations are run with a 75 km global mesh and 1 km superparameterization with either a 3D superparameterization (blue) or 2D superparameterization (red). We also include a projection of the non-superparameterized model running at a global 3 km resolution (purple).

Strong scaling of the E3SM-MMF is shown in Fig. 60. The result shows several configurations of the E3SM atmosphere running on Summit. Our main result for the E3SM-MMF with the 3D SAM++ is shown in blue, the dashed lines show the performance of the model on the Summit CPUs, and the solid line show the performance of the model when using the GPUs. The model obtains good scaling with a GPU speedup of $\sim 15\times$ per node, comparing six GPUS with two Power9s. When using a 2D SAM++, other aspects of the mode dominate the performance, and we obtain minimal GPU acceleration (red curve). The purple curve shows a projection of our baseline model, running with a global 3 km resolution. This projection is based on the GPU port of the dynamical core. Based on those results, we project that the model will make excellent use of the Summit GPUs but remain prohibitively expensive, requiring all of Summit to obtain 0.5 SYPD.

6. DATA ANALYTICS AND OPTIMIZATION APPLICATIONS

End State: Deliver comprehensive data-driven and science-based computational applications able to provide, through effective exploitation of exascale HPC technologies, breakthrough solutions that yield high-confidence insights and answers to challenges in a selected variety of DOE and non-DOE US government agencies.

The data analytics and optimization (DAO) L3 area (Table 51) includes applications that do not rely on approximate solutions to equations that state fundamental physical principles or reduced semiempirical models. Instead, DAO applications exploit predictive capabilities that exploit modern data analysis and ML techniques, complex combinatorial models of data interactions, and constrained, nonlinear algebraic representations of complex systems. These applications target the mission space of DOE and other relevant

Table 51: Summary of supported DAO L4 projects.

WBS number	Short name	Project short description	KPP-X
2.2.4.02	ExaSGD	Reliable and efficient planning of the power grid	KPP-2
2.2.4.03	CANDLE	Accelerate and translate cancer research	KPP-1
2.2.4.04	ExaBiome	Improve understanding of the microbiome	KPP-2
2.2.4.05	ExaFEL	Light source-enabled analysis of molecular structure	KPP-2

federal agencies—such as the NIH, the NSF, NASA, and NOAA—identified as high priority per the National Strategic Computing Initiative (NSCI). They include a broad range of application areas and techniques, some of which are only recently coming into maturity in the context of high-end simulation. As such, they represent greater risk but also significant potential for new discovery.

The principal goal of this activity is to develop ECP applications capable of delivering demonstrable solutions to specific DAO challenge problems on the exascale systems. These applications must target science and energy challenge problems within the mission space of the relevant agency. Another objective is to educate, train, and inform DAO staff on the best practice approaches for exascale application development, including an integration objective of demonstrating, for the DAO area, the vital role and return on investment provided by predictive modeling in delivering on DAO strategic goals. An additional objective is to gather requirements from the DAO applications to help guide the ST and hardware and integration (HI) R&D activities.

6.1 ExaSGD

Energy delivery systems, such as the US national power grid, operate by maintaining balance between energy supply and demand. Attacks via physical or cyber means and hazards on the grid can create an imbalance between supply and demand, which can result in drops in voltage or frequency that can permanently damage large and expensive components. To ensure safety and reliability, the grid must operate within narrow voltage and frequency ranges.

Recovering from generation-load imbalance can be achieved by shedding load (i.e., deliberately allowing some load to go unserved and creating a partial blackout) to preserve the functionality of the remainder of the power grid. However, emerging technologies make power grids more complex and more controllable. Examples include the increasing prevalence of cyber-enabled control and sensing, variable generation renewables (e.g., wind and solar), plug-in storage devices (e.g., electric vehicles), smart meters that can control load at a fine granularity (e.g., throttling home appliances at times of peak demand), and other sensored elements that can be controlled remotely. Today's practice focuses primarily on a conventional load-shedding approach and as such might miss more efficient strategies for dynamically achieving balance via a broader spectrum of grid control techniques. The capability to discover more optimal configurations to recover from generation-load imbalance will improve the national readiness to recover from a variety of hazards to the power grid.

6.1.1 ExaSGD: Science Challenge Problem Description

The ExaSGD challenge problem is to optimize the grid's short-term response (e.g., 30 min per NERC operating standards) to many potential under-frequency hazards due to physical and cyber threat scenarios by using a high-fidelity grid model that includes generation, transmission, load, and cyber/control elements. The ExaSGD team will compare the frequency recovery performance of a complex grid for different control responses that involve smart devices, renewables, and advanced demand response technologies. This will require complex high-fidelity analysis with many coupled optimal power flow computations. Each optimal power flow calculation corresponds to a specific contingency/scenario. The solution of the overall problem—stochastic security constrained optimal power flow—is the optimal operating point for the entire grid given

Table 52: ExaSGD challenge problem details.

Functional requirement	Minimum criteria
Models	A nonlinear optimization model describing physical and control systems for power grid operation.
Numerical methods	Nonlinear optimization with equality and inequality constraints.
Problem size and complexity	10^5 optimization parameters. 10^3–10^5 "N − 1" and "N − k" contingencies. 10^3–10^5 stochastic scenarios. Individual contingencies and stochastic scenarios are coupled through the base problem (i.e., deterministic with no faults occurring) but not directly to each other. One power grid model evaluation (e.g., Eastern Interconnection model) is sufficiently sized to exhaust high-end GPU resources, and the computation increases by adding contingencies and stochastic scenarios. For 10^3 contingencies and 10^3 stochastic scenarios, assuming a single grid contingency-scenario instance exhausts a 10 TFlops/s GPU device and leads to a $10^3 \times 10^3 \times 10\,\text{TFLOPS} = 10\,\text{EFLOPS}$ computation.
Demonstration calculation requirements	The software stack implementation depends on the availability of a linear solver that can handle very ill-conditioned problems and achieve high GPU utilization simultaneously. The team has identified solvers from the Magma library as suitable candidates. To create appropriate test cases, power grid models with more than 20,000 buses are needed. Several years of wind and solar data are needed to generate stochastic scenarios. Additional data could be required to create cyber scenarios.
Resource requirements to run demonstration calculation	Production runs for the challenge problem at the highest scale will be ∼1 day on full exascale systems. • Frontier: 1 day on full system. • Summit: 1 day on full system.

the likelihood and the impact of each contingency-scenario combination. Challenge problem details are shown in Table 52.

6.1.2 ExaSGD: KPP Stretch Goal

The KPP stretch goal is a multiperiod security-constrained AC optimal power flow problem with frequency recovery and restoration. The objective is to determine control settings on generators, loads, and renewable energy resources so that the system frequency is restored after the occurrence of the contingency quickly enough to avoid losing stability and to prevent possible cascading outages. The stretch goal includes two analyses.

- *Addition of storage devices, EV charging stations, and controllable loads in the set of resources available to mitigate frequency deviations in addition to conventional generators.* The many physical and virtual storage options further increases the complexity of the grid, not only in terms of having more optimization parameters and scenarios to consider but also in terms of qualitative changes in power grid behavior, such as possible transmission scale backflow.
- *Using multiperiod constraints.* Generator ramping constraints span and couple multiple periods in optimal power flow analysis and require nontrivial modifications to the model and the solving strategy. Correctly capturing ramping and other multiperiod constraints shows not only that a desired solution

exists but also that the control action can bring the system to that solution. Multiperiod analysis is a critical capability for frequency restoration problems and for improving power grid resilience in general.

The stretch goal will stress test ExaSGD algorithms and libraries by using a class of problems that have not been extensively studied, even on sub-exascale benchmarks. If successful, then the stretch goal will demonstrate the breadth of applications for which ExaSGD technology can be deployed.

6.1.3 ExaSGD: Capability Plan

The ExaSGD capability plan was significantly revised in FY20 because the computational methods outlined in the original project plan appeared to be unsuitable for deployment on GPU-based hardware platforms.

- ExaSGD planned to use the PIPS optimization solver [36], which depends on a robust linear solver that can handle ill-conditioned sparse symmetric indefinite linear systems. To date, no such solver can run efficiently on GPUs [37].
- The ExaSGD power grid modeling framework was originally implemented by using Julia language, which is not supported on pre-exascale or exascale platforms. Additionally, the Julia-based modeling framework did not address the potential performance bottlenecks on GPUs. Power grid models are very sparse and highly irregular, so model evaluations on GPUs must be customized to avoid warp divergence and poor memory coalescence.

These issues were identified as significant project risks and required the entire computational approach for the ExaSGD project to be redesigned.

Optimization Engine

The main challenge was to devise a new optimization approach that does not depend on sparse linear solvers. Rather than handle the power grid contingency analysis as one large optimization problem, the new approach partitions the problem via a two-stage optimization method (e.g., Chakrabarti, Kraning, Chu, Baldick, and Boyd [38]). Each partition (i.e., subproblem) is then compressed by using custom-developed dense-sparse linear algebra to a smaller dense subproblem. This allows more mature dense linear solvers to be used that better use GPUs, such as those from Magma library [39]. This methodology was developed within HiOp library [40, 41]. The two-stage optimization is illustrated in Fig. 61.

Modeling Framework

The ExaGO library [42] was developed in FY20 to address power grid modeling challenges on GPU-based platforms. The library design is modular; it separates modeling the front end from different hardware back ends to enable the development of portable power grid applications. The library uses standard power system input formats to specify the problem and thus provides domain experts with a familiar environment. During the model setup, the GPU back end of the ExaGO library performs a one-time copy of the model data to data structures tailored for fine-grain parallelism. This improves data coalescence and reduces warp divergence in subsequent computations. This appears to be the first power system modeling framework designed for hardware accelerator computations.

Programming Model

ExaSGD adopted a programming model based on C++ and portability libraries RAJA [43] and Umpire [44] for key software stack components—HiOp and ExaGO. Using portability libraries greatly increased the speed of ExaSGD software development and reduced the development cost. The libraries allowed for the effective separation of responsibilities between domain experts and computational scientists. Furthermore, this programming model enabled staged development and debugging in which 90 % of the bugs were fixed by using CPU execution policies.

Mileposts

ExaSGD mileposts are organized into two tracks: (1) stochastic security constrained optimal power flow analysis and (2) multiperiod analysis. The former will focus more on development because there is a clear technical path to reach the exascale milestone. The latter requires some research to incorporate time

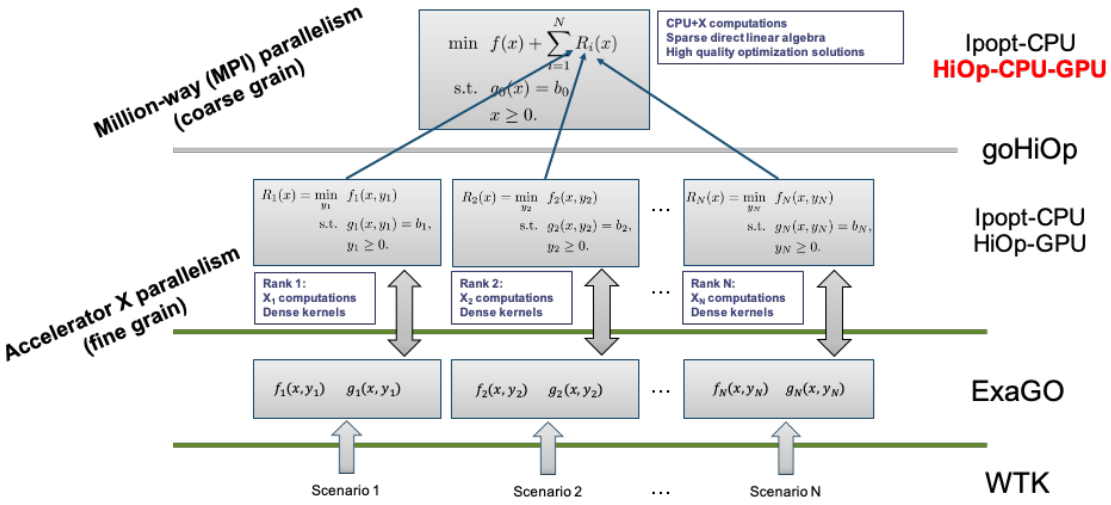


Figure 61: Schematic description of the two-stage optimization method implemented in ExaSGD software stack. Wind toolkit (WTK) generates stochastic input for each power grid instance (i.e., contingency) implemented in the ExaGO modeling framework. Security-constrained optimal power flow is computed across all contingencies by using the two-stage approach with the HiOp engine solving second-stage problems on GPU and instance of optimization solver tuned for high accuracy (HiOp or Ipopt) solving the master problem.

decomposition within the global optimization scheme and some prototyping in Julia before the implementation candidate is designed.

FY20: Milepost 1: HiOp second-stage optimization convergence on GPU on Summit.
Ensure that the GPU-friendly optimization algorithm converges. Run Formulation 3 by using HiOp with goLLNLp modeling framework and an MPI engine on 500–1000 MPI ranks on Summit. Use a 2000–10,000 bus grid model and 1000–5000 contingencies as the test case. Expected to occupy 3–5 PFlops resources. Expected peak performance is ∼1 PFlops. This demonstrates that the optimization algorithm modified to produce GPU-friendly linear problems converges when run within a two-stage optimization framework.

Milepost 2: 1 PFlops ExaSGD software stack performance on GPU (second stage only) on Summit.
Run only the second stage of the Formulation 3 by using the WTK-ExaGO-HiOp software stack as embarrassingly parallel 500–1000 MPI processes on Summit. All computations inside the second stage optimization loops run on GPU. Use a 2000–10,000 bus grid model as the test case. Expected to occupy 3–5 PFlops resources. Expected peak performance is ∼1 PFlops. This demonstrates the performance of the software stack key components within a two-stage optimization method.

FY21: Milepost 3: 10 PFlops full-stack run and method convergence on Summit.
Run Formulation 3 by using the full ExaSGD software stack with HiOp MPI engine and ExaGO modeling framework on Summit. Use a 10,000–20,000 bus grid model with 5+ scenarios and 2000–10,000 contingencies. All computations inside the optimization loops run on GPU. Target is 10 PFlops performance.

Milepost 4: Small-scale full-stack run and method convergence on Tulip.
Run Formulation 3 by using the full ExaSGD software stack on Tulip. Use a 1000–2000 bus grid model with 5+ scenarios and 200–1000 contingencies. All computations inside the optimization loops run on AMD GPUs.

Milepost 5: Convergence of a small scale two-period optimization on Tulip.
Run Formulation 4 by using the full ExaSGD software stack on Tulip and/or Summit. Use a 1000 bus grid model with 5+ scenarios and 100–500 contingencies and two periods. All computations

inside optimization loops run on GPU.

FY22: Milepost 6: 10–100 PFlops performance and method convergence on Frontier.
Run Formulation 3 by using the full ExaSGD software stack on Frontier. Use a 30,000 bus grid model with 10+ scenarios and 10,000+ contingencies. All computations inside the optimization loops run on GPU. Target is 10–100 PFlops performance.

Milepost 7: Convergence of a large-scale multiperiod optimization on Summit.
Run Formulation 4 by using the full ExaSGD software stack on Summit. Use a 10,000 bus grid model with 10+ scenarios, 5000+ contingencies, and 5–10 time periods. All computations inside the optimization loops run on GPU.

Milepost 8: Small-scale frequency restoration on Summit.
Run Formulation 5 by using the full ExaSGD software stack on Summit. Use a 1000 bus grid model with 5+ scenarios, 200+ contingencies, and 5–10 time periods. All computations inside the optimization loops run on GPU. Optionally, also run an emergency planning test case based on Formulation 4.

FY23: Milepost 9: Exascale demonstration on Frontier.
Run Formulation 3 by using the full ExaSGD software stack with the HiOp MPI engine and ExaGO modeling framework on Frontier. Use a 70,000 bus grid model with 10+ scenarios and 10,000+ contingencies. The analysis converges.

Milepost 10. Frequency restoration medium-scale demonstration on Frontier.
Run Formulation 5 by using the full ExaSGD software stack on Frontier. Use a 10,000 bus grid model with 5+ scenarios, 1000+ contingencies, and 5–10 periods. Analysis converges, and all computations inside the optimization loops run on GPU. Optionally, run a multiperiod emergency planning demonstration based on Formulation 4.

6.1.4 ExaSGD: Progress on Early and Pre-Exascale Hardware

Performance on Summit

The FY20 mileposts were essentially large-scale preintegration tests for the redesigned software stack. The achieved results exceeded expectations for both mileposts.

Milepost 1: This milepost verified that the optimization algorithm designed for execution on heterogeneous hardware platforms will converge at large scale. Although optimization-based decomposition was tested at smaller scale during the ARPA-E Grid Optimization competition [45], the mixed dense-sparse linear algebra [46] was tested for the first time within a heterogeneous parallel environment. With this new approach, a security-constrained optimal power flow analysis for a 20,000 bus grid and 12,500 contingencies was performed on 1440 MPI ranks within 10 min. For comparison, for a grid of comparable size, current transmission grid operators can run less than 100 contingencies for a security constrained power flow analysis with standard industry tools. This means that because of present tools constraints, grid operators look for any secure solution without trying to optimize it. Although still under development, the proposed technology could dramatically improve the way power grids are operated today.

Preliminary scaling results that illustrate the operation of the two-stage optimization approach are shown in Fig. 62. The figure shows the number of contingencies (second-stage subproblem optimization runs) evaluated in time. First, the evaluation of the base-level problem occurs on a single device—in this case, a CPU—and runs for about 100 s. Then, second-level problems (i.e., contingencies) are evaluated, and the results are communicated to and from the base-level problem through an asynchronous MPI update. During that period, the computation scales well. When converging to the solution of the full problem, most of the subproblems will have converged, so many MPI ranks become underutilized. This shows as plateauing in curves for 2880 and 1440 MPI ranks in Fig. 62. Work to improve the asynchronous MPI engine for the two-stage optimization method is underway.

In these computations, the model evaluation was performed entirely on CPU by using a mature and well-tested modeling framework to isolate the optimization algorithm from potential bugs in the GPU model

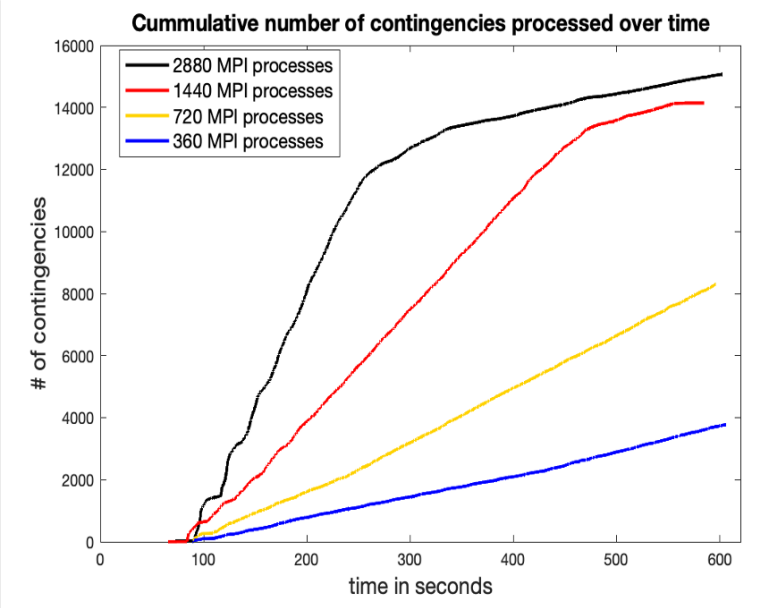


Figure 62: Milepost 1 test results.

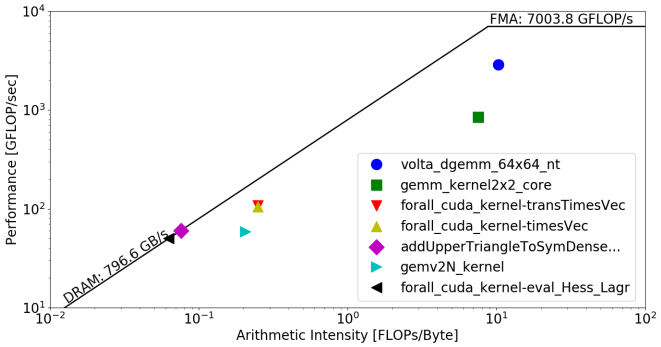


Figure 63: Roofline plot for key HiOp kernels.

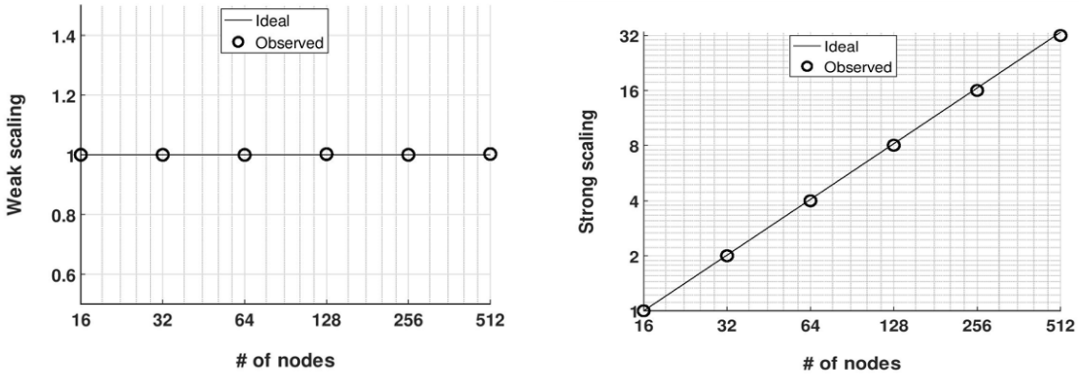
implementation. Such computation was obviously suboptimal, and future milestones will provide even better performance.

The performance of the HiOp solver was analyzed by using a roofline analysis (i.e., by looking at the performance of individual kernels given their arithmetic intensity). The roofline plot for HiOp execution is shown in Fig. 63. The line in the plot represents the base roofline. The closer a point is to the roofline, the more optimized it is with respect to performance that can be achieved for the given arithmetic intensity. Only the top few kernels of HiOp are shown here, and others are omitted because they consume less than 0.5 % of the total execution time. The sloped line labeled "DRAM" represents the limits of routines that are memory-bound. Key kernels are fairly well optimized, with the possible exception of `gemm_kernel2x2_core` and `gemv2N_kernel`. These two kernels are actually called by the linear solver from the Magma library and are not part of the HiOp code.

Milepost 2: This milepost verified the integration of the modeling framework ExaGO and the optimization engine HiOp, and it assessed their performance on GPU. In this exercise, an embarrassingly parallel computation for many contingencies was run to emulate the second stage of the two-stage optimization method. The entire computation—including model evaluation, optimization, and underlying linear solver—

Table 53: Comparison of CPU and GPU execution time for the solve stage on Newell node using HiOp's mixed dense-sparse optimization solver

Grid		GPU run (s)	CPU run (s)	Speedup
Illinois	(200 bus)	2.9	0.6	0.2
Carolina	(500 bus)	8.0	14.9	1.9
Texas	(2000 bus)	60.2	1624	27



(a) Weak scaling.

(b) Strong scaling.

Figure 64: Milepost 2 test results. Near-ideal scaling of the second stage of the two-stage optimization approach suggests no computational bottlenecks.

was run on GPUs. Table 53 shows the preliminary performance results for synthetic grid models [47]. When running computations on GPU, the speedup improves as the size of the problem increases.

As expected, the computation shows near-ideal weak and strong scaling (Fig. 62). This verifies that the implementation is correct and that compute time does not vary significantly from one contingency (i.e., subproblem) to another. This is a verification that there are no computational bottlenecks in the GPU implementation of the modeling framework ahead of the further development of the MPI engine for the two-stage optimization.

Next Steps

The immediate next step is to complete software stack integration after performing the successful preintegration tests specified in Mileposts 1 and 2. This will require improvements in the asynchronous MPI engine for the two-stage optimization, mainly in terms of its scalability.

Once the software stack is well-tested on Summit, the team will port it to Tulip platform, profile its performance, and compare it with profiling results on Summit (Milepost 4). In parallel to this activity, the team will perform several scaling studies at Summit to try to achieve robust performance of the optimization method for grids with 20,000+ buses, 10,000+ contingencies, and 5+ stochastic scenarios (Milepost 3). Finally, the team will prototype multiperiod optimization techniques and select an implementation candidate for FY22 (Milepost 5).

6.2 CANDLE

DOE has entered into a partnership with the National Cancer Institute (NCI) of the NIH and has identified three key challenges that the combined resources of DOE and NCI can accelerate. The first challenge, called the *drug response problem*, is to develop predictive models for drug response that can be used to optimize preclinical drug screening and drive precision medicine-based treatments for cancer patients. The second challenge, called the *RAS pathway problem*, is to understand the molecular basis of key protein interactions in the RAS/RAF pathway that are present in 30 % of cancers. The third challenge, called the *treatment*

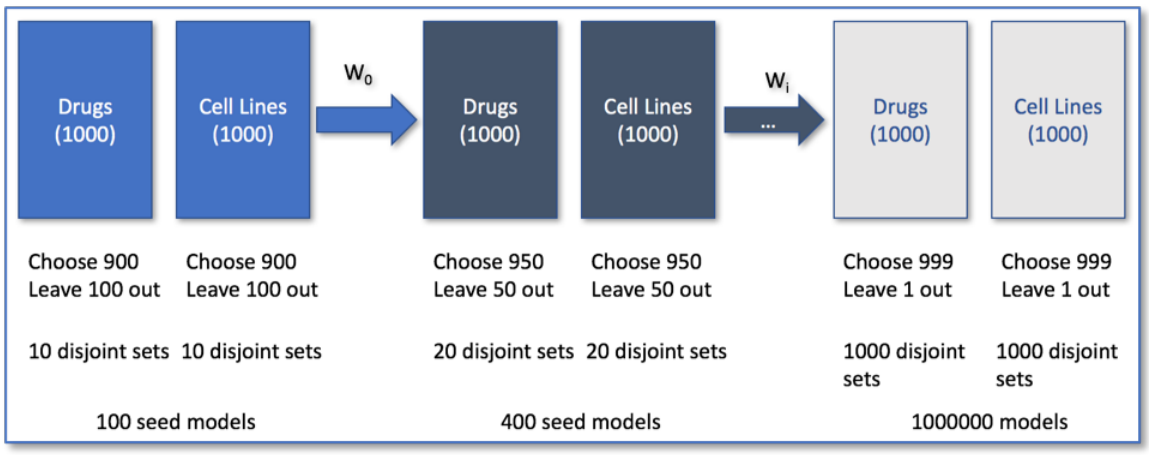


Figure 65: A transfer learning design for training 1 million models to predict the effect of one drug on one cancer cell line.

strategy problem, is to automate the analysis and extraction of information from millions of cancer patient records to determine optimal cancer treatment strategies across a range of patient lifestyles, environmental exposures, cancer types, and healthcare systems. Although each of these three challenges is at a different scale and has specific scientific teams collaborating on the data acquisition, data analysis, model formulation, and scientific runs of simulations, they each also share several common threads. The ECP project called the Exascale Deep Learning and Simulation Enabled Precision Medicine for Cancer focuses on the ML aspect of the three challenges and, in particular, builds on a single scalable deep neural network (DNN) code called Cancer Distributed Learning Environment (CANDLE).

6.2.1 CANDLE: Science Challenge Problem Description

The CANDLE challenge problem is to solve large-scale ML problems for two cancer-related pilot applications: (1) predicting drug interactions and (2) predicting cancer phenotypes and treatment trajectories from patient documents. The CANDLE pilot application that involves predicting the state of MD simulations is treated as a stretch goal. The CANDLE project has specific strategies to address these challenges. For the drug response problem, unsupervised ML methods are used to capture the complex, nonlinear relationships between the properties of drugs and the properties of the tumors to predict response to treatment and therefore develop a model that can provide treatment recommendations for a given tumor. For the treatment strategy problem, semisupervised ML is used to automatically read and encode millions of clinical reports into a form that can be computed upon. Each problem requires a different approach to the embedded learning problem, all of which are supported with the same scalable deep learning code in CANDLE.

The challenge for exascale manifests in the need to train large numbers of models. One need inherent to each of the pilot applications is producing high-resolution models that cover the space of specific predictions individualized in the precision medicine sense. For example, consider training a model that is specific to a certain drug and individual cancer. Starting with 1000 different cancer cell lines and 1000 different drugs, a leave-one-out strategy to create a high-resolution model for all drug-by-cancers requires approximately 1 million models. Yet, these models are similar enough that using a transfer learning strategy in which weights are shared during training in a way that avoids information leakage can significantly reduce the time needed to train a large set of models. Figure 65 shows a general strategy of this weight-sharing approach to training many models in which a transfer of weights delineates stages in the workflow.

In practice, speedup related to weight sharing can be discussed in the context of the challenge problem in terms of work actually done and the naive work done. Consider work actually done W_D as being the number of jobs J multiplied by the actual number of epochs E trained for all stages s:

$$W_D = \sum_{1}^{s} JE\,. \tag{18}$$

A stage is a discrete and naive work done W_N being the number of jobs in the last stage multiplied by the number of epochs needed for a model to converge E_c,:

$$W_n = J_s E_c \, . \tag{19}$$

The team can then talk about speedup as being the ratio W_N/W_D.

Several parameters exist when considering accelerated model training via the transfer of weights. These include how many transfer events, how to partition the input data, and how many epochs before a transfer occurs. Additional considerations include what weights to transfer and whether to allow those weights to be updated in subsequent models. The requirements associated with the science challenge problem of training many high-resolution models are outlined in Table 54.

6.2.2 CANDLE: KPP Stretch Goal

CANDLE will consider two KPP stretch goals.

1. For the RAS pathway problem, Pilot2, multiscale MD runs are guided through a large-scale state-space search by using unsupervised learning to determine the scope and scale of the next series of simulations based on the history of previous simulations. CANDLE has demonstrated a prototype DNN that performs unsupervised feature learning on MD simulation neighborhoods. This stretch goal will consider alternative DNN formulations to improve the predictive performance of Pilot2 models. The final result will be the demonstration of the CANDLE FOM that includes a training rate for the revised Pilot2 models.

2. CANDLE will develop the first full US population level precision oncology model. This model will leverage the work in the DOE-NCI Pilot3 to apply NLP to the national cancer registries to extract structured terms from pathology reports to identify the type, stage and location of tumors, DOE-NCI Pilot1's models to predict tumor drug response to single drug and drug combinations, the NCI Genomic Data Commons repository of tumors and genomic profiles for more than 15,000 patients, and the ALCF's CANDLE Early Science Data Science project. The goal is to develop a nationwide population-level estimate of the potential benefit of precision medicine applied to cancer. The team will use the SEERs database processed with MT-HCAN to generate a national estimate of the emerging cancer patient population profile (approximately 1.6 million new cancer cases per year) for the next 10 years. This patient profile will be combined with the GDC database to generate a national-level tumor population estimate, including tumor type, genomic, and transcript profile. The resulting "digital twin" database of 16 million cancer patients will then be processed by using the drug response models to evaluate optimal drug treatment strategies for each virtual patient. The team will use the "Uno-MT-UQ" drug response model to predict the response for each of the 700 drugs in the current set of standard of care (SoC) drugs and drugs in clinical development. UQ methods will be used to provide confidence intervals for each of the predictions. Additionally, the team will predict drug response for approximately 5000 SoC drug combinations by using our Combo-UQ model. The result will be the first national-scale predictive oncology "benefit" map that outlines which cancers, regions, and population groups would most benefit from a comprehensive implementation of precision medicine.

6.2.3 CANDLE: Figure of Merit

The CANDLE FOM is the average rate of model training to convergence. On a given system, it can be demonstrated that n instances of a pilot model (P_i) can be trained to convergence in a given time t, thus producing a rate x_i equal to n/t. Models are defined as part of each of the three pilot applications. Because each pilot application will focus on different DNN-based models and because rates are being measured for each pilot model, the total FOM will be the harmonic mean (H) of the rates across the models from the three pilot applications, where x is the rate of n^{th} model trained to convergence:

$$H = \frac{n}{\frac{1}{x_1} + \frac{1}{x_2} + \cdots + \frac{1}{x_n}} \, . \tag{20}$$

Table 54: CANDLE challenge problem details.

Functional requirement	Minimum criteria
Models	Deep learning neural networks for cancer: feed-forward, auto-encoder, recurrent neural networks.
Numerical methods	Gradient descent of model parameters, optimization of loss function, network activation function; regularization, and learning rate scaling methods.
Problem size and complexity	KPP-1: Large-scale ML solutions will be computed for the three cancer pilots. • Pilot1: Leave-one-out cross validation of roughly 1000 drugs by 1000 cell lines. This involves roughly 1 million models. Partition the drugs and cell lines into n sets and train with those for e epochs, then transfer the weights w to the next set of models, expanding the number of models in each iteration. Each of the models at iteration i can be safely (i.e., avoiding information leakage) used to seed models for iteration $i+1$, where the set of drugs and cell lines in the $i+1$ validation set was not in the training set of the model at iteration i. • Pilot2: State the identification and classification of one or more RAS proteins binding to a lipid membrane; prediction over time of clustering behavior of key lipid populations that leads to RAS protein binding. RAS proteins are represented in sufficient resolution to model all pairwise interactions within and between proteins. Lipid membranes are represented as continuous density fields of tens of species of lipid concentration. Predictions are trained on the cross product of thousands of simulations, each of which is thousands of time steps, over multiple protein configurations, and performed for a large range of different concentrations. • Pilot3: Predicting cancer phenotypes and patient treatment trajectories from millions of cancer registry documents. Thousands of multitask phenotype classification models will be built from defined combinations of descriptive terms extracted from 10,000 curated text training sets. To accelerate model training, the team will use a transfer learning scheme with weight sharing during training.
Demonstration calculation requirements	The computation performed at scale will be standard neural network computations, matrix multiplies, 2D convolutions, pooling, and so on. These will be specifically defined by the models chosen to demonstrate transfer learning. The computations performed at scale will require weight sharing.
Resource requirements to run demonstration calculation	For each pilot, each pilot problem is estimated to require up to 12 h at full system.

Table 55: Current CANDLE FOMs computed on Summit.

CANDLE pilot model	Full Summit estimates
Pilot1 contribution Uno	931.04 models per hour
Pilot2 contribution C1-TDA	N/A
Pilot3 contribution P3B4	511.2 models per hour
Total	660.01 models per hour

The rate of model training to convergence is model-specific, and the time varies accordingly with the number of epochs needed to train the model to convergence. For this reason, the team must choose a few numbers of models, ideally one per pilot project, then fix the input data and number of epochs associated with each model so that repeated measurements of the FOM can be compared with previous measurements. A weighted harmonic mean (H_w) will be used that considers the number of epochs required to train to convergence for the different models. In cases in which computational resources or training to convergence exceed standard queue policies and if the time per epoch can be assumed to be constant, then H_w remains a valid choice. The weighted harmonic mean of the rate of model training can be represented as

$$H_w = \frac{n}{\frac{e_{c1}}{e_1 x_1} + \frac{e_{c2}}{e_2 x_2} + \cdots + \frac{e_{cn}}{e_n x_n}}, \tag{21}$$

or

$$H_W = \frac{n}{\sum_{i=1}^{n} \frac{e_{ci}}{e_i x_i}}, \tag{22}$$

where H_w is now a weighted harmonic mean, e_{ci} is the number of epochs needed to train the i^{th} model to convergence, and e_i is the number of epochs actually run for the i^{th} model. This assumes that the time per epoch is constant, which holds true for many of the pilot application's models.

The CANDLE FOM will be computed by using training benchmarks for Pilot1 and Pilot3 models. Integrating benchmarks for Pilot2 models is part of the CANDLE stretch goal.

FOM Update

The CANDLE team reported an improvement of the CANDLE KPP from 23.99 to 81.99 in 2020. This reflects the current FOM on Summit relative to the FOM on Titan. The Pilot1 benchmark (Uno) model and Pilot3 benchmark (P3B4) model are the kernels of the CANDLE challenge problems. These are the focus of our FOM. Over the last year, numerous improvements to model training were accomplished. Table 55 provides the current CANDLE FOM calculated on Summit.

6.2.4 CANDLE: Progress on Early and Pre-Exascale Hardware

Performance on Summit

Pilot1: We implemented an on-memory mode in the data loader that expedites the data input process when the dataset size can fit into the host memory. We observed a 4 % improvement due to the software stack update (from ibm-wml-ce/1.6.2-3 to ibm-wml-ce/1.7.0-3) and a 41 % improvement when the on-memory loader is compared with the previous loader (on-disk mode). The input features were down-selected by the method developed in Milestone 16. While keeping the model accuracy equivalent, we achieved about a 10 % speedup with the selected features. We have configured our framework to use all six GPUs per node, which increased the training throughput about five fold.

Pilot3: The P3B4 benchmark was updated to take advantage of mixed precision arithmetic on Summit's V100 GPUs for some operations. We established that FP16 can be used for certain operations without impacting the F1 scores by which the model accuracy is measured. To better align with our current challenge problem, rather than training models over the current dataset, we are partitioning the dataset into ∼20 subgroups to develop more targeted models. This results in more models being trained on smaller portions of the current dataset. For each of the 20 subgroups into which the dataset is partitioned, we are training 100 models for 30 epochs. For each model, we randomly sample 25 % of the subset of data associated with that

subgroup. For the purposes of the FOM computation, this is equivalent to having trained 100 models on 25 % of the current dataset (814,230 reports).

Next Steps

CANDLE has two upcoming milestones due in January 2021. With regards to our FOM, our next steps will focus on the CANDLE milestone "Inference: Model Compression." This milestone aims to compress deep learning models to accelerate the performance of inference and related tasks without significantly compromising accuracy. As part of considering methods to compress a trained model, quantization-aware training will be explored from the perspective of the impact on model accuracy and on the CANDLE FOM.

6.3 ExaBiome

Metagenomics involves the application of high-throughput genome sequencing technologies to DNA extracted from microbial communities, which is a powerful and general method for studying microbial diversity, integration, and dynamics. Since the introduction of metagenomics over a decade ago, it has become an essential and routine tool. Assembly and comparative analyses of metagenomic datasets are among the most computationally demanding tasks in bioinformatics. The scale and rate of growth of these datasets will require exascale resources to process (i.e., assemble) and interpret through annotation and comparative analysis. The ExaBiome project aims to provide scalable tools for three core computational problems in metagenomics: (1) metagenome assembly, which takes raw sequence data and produces long genome sequences for each species; (2) protein clustering and annotation, which finds families of closely related proteins and annotates them with functional properties; and (3) signature-based approaches to enable scalable and efficient comparative metagenome analysis, which might show variability of an environmental community over time or from changes in precipitation, fires, temperature, and more.

The ExaBiome team developed a metagenome assembler, MetaHipMer, which scales well on thousands of compute nodes. MetaHipMer has already been used to assemble large environmental datasets that were impossible with previous tools. Over the last year, MetaHipMer was completely rewritten in a single programming model, UPC++. The resulting code is faster and more scalable, uses less memory, and is easier to install and maintain. The team also had a significant effort in GPU optimizations for key kernels and has integrated GPU-optimized alignment into the code. MetaHipMer is designed for short read (Illumina) data, but a second assembler called diBELLA [48] is also under development for long read technology. diBELLA shows even higher computational intensity than MetaHipMer, making it a good fit for exascale systems. A second thrust of ExaBiome involves protein clustering and annotation. ExaBiome's HipMCL [49] and PASTIS [50] code—the latter developed jointly with ExaGraph—provide a scalable protein clustering pipeline, whereas a new prototype deep learning framework [51] shows promising results for functional annotation. HipMCL runs on thousands of nodes and effectively uses GPUs. Finally, to support comparative analysis across metagenomes, the team recently built a scalable k-mer analysis tool, KmerProf, leveraging modules in MetaHipMer to compute k-mer profiles to quickly approximate and compare metagenomes by using various distance metrics on the profiles. The ExaBiome challenge problem focuses on the first metagenome assembly problems, but that capability will enable exascale feasibility for other bioinformatics problems in the other ExaBiome thrusts and more broadly.

6.3.1 ExaBiome: Science Challenge Problem Description

ExaBiome's challenge problem is to demonstrate a high-quality assembly on at least 50 TB of environmental data (i.e., reads) that effectively use an exascale machine. The intent is to use a scientifically interesting environmental sample that might include multiple temporal or spatial samples; the goal is to perform single assembly by using complete sequence data. In contrast, current state-of-the-art assembly pipelines are forced to use subsampling with large datasets, which limits the ability to assemble rare, low-coverage species and produces confusing duplications of genomes. The team recently published a paper that shows that the combination of larger datasets and more computing produces superior quality assemblies [52]. Addressing this challenge problem will demonstrate a first-in-class science capability by using the power of exascale computing on a non-traditional scientific workload. MetaHipMer is being used by science projects at the Joint Genome Institute (JGI), and it has inspired the genomics community to collect larger datasets knowing that they can be analyzed. There are many potential beneficial science impacts, such as enhancing the

Table 56: ExaBiome challenge problem details.

Functional requirement	Minimum criteria
Models	Given a set of genome fragments of DNA with a given range of length and error rate, (100–200 base pairs with <0.1% errors or 10,000+ base pairs with up to 18% errors), a genomic assembly is an arrangement of these fragments to form large contiguous sequences from which genes and species can be identified. Standard quality metrics are used to assess the accuracy of these sequences: the average length of output strings (10,000+ base pairs for short reads) and percentage of the input reads that map to the output (>90% is reasonable).
Numerical Methods	De Brujin graphs construction and analysis, dynamic programming for string alignment, Bloom filters, k-mer counting/analysis, and distributed hash tables.
Problem size and complexity	50 TB of real or synthetic environmental data using short reads, long reads, or a combination of the two.
Demonstration calculation requirements	To demonstrate the challenge problem, the team must run a complete assembly because all the stages must be executed to truly test the scaling. The MetaHipMer and diBELLA pipelines can be run in separate stages with data output to a file system, if desired. Thus, before performing a full-scale assembly, the scalability of each stage will be tested to ensure that it is progressing at a reasonable rate. The team will also use intermediate problem and machine sizes to validate scaling assumptions.
Resource requirements to run demonstration calculation	Total work required and memory/operation resource requirements can be estimated for the full dataset by extrapolating from current scaling performance. It required roughly 100 node-hours per terabyte of input data in a weak-scaling study of a tropical soil dataset with the largest run being 7.7 TB in 1.4 h on 512 Summit nodes with 84 TB of aggregate memory. Assuming improved GPU effectiveness will compensate for any scaling inefficiency, we estimate that 50 TB will require over 0.5 PB of memory and run in 5 h on 1000 Frontier nodes. Because of uncertainties in the sample complexity, which can substantially increase memory use and running time, we set a baseline goal of 20 wallclock hours for the 50 TB assembly on Frontier or Aurora.

understanding of microbial functions that can aid in environmental remediation, food production, and medical research. Given the growth of genomic data, a scientifically interesting 50 TB environmental sample should be available by 2022 and is expected to be large enough to fully use an exascale machine. However, the challenge problem could also use synthetic data with environmental characteristics or an ensemble assembly of multiple independent environmental datasets. It could use short reads, long reads, or a hybrid of the two. Challenge problem specifications for ExaBiome are listed in Table 56.

6.3.2 ExaBiome: KPP Stretch Goal

The stretch goal of ExaBiome is to develop and use exascale tools to analyze at least 1 PB of microbial or plant data. The ExaBiome project is developing exascale tools for single genome and metagenome assembly, specifically targeting environmental large microbial communities but also applicable to complex plant genomes and possibly pan-genome studies that co-assemble a family of related genomes. The JGI currently has over 8 PB of genomic data, and this is expected to grow. For this goal, it could also be combined with other datasets, including the NIH Sequence Read Archive (SRA) and the newly formed

National Microbiome Data Collaboratory. The ExaBiome team is also building protein analysis tools based on traditional unsupervised methods (e.g.,Markov chain clustering) and deep learning methods that operate on large community databases. The clustering pipeline of PASTIS + HipMCL has been used in production at JGI. Comparative metagenomics tools based on k-mer profiles, alignment against databases, or ML can operate on assembled or unassembled data. Realization of this goal could also involve support for new long-read sequencing technologies or high-quality extension of those techniques, which might require additional algorithmic work and software. The team will work toward this 1 PB goal with intermediate scale problems beyond the 50 TB in the assembly baseline goal. The higher level scientific objective is to build more complete assemblies and improve understanding of the relationship between different species by using environmental samples collected over space or time for DOE applications in energy and the environment.

6.3.3 ExaBiome: Capability Plan

The KPP-2 goal and prior mileposts are for the assembly process. Although scaling estimates are based on metagenome assemblies with MetaHipMer, the goals are broader for large-scale assemblies, which could use diBELLA or pan-genome plant assemblies. Protein clustering mileposts with PASTIS plus HipMCL provide a set of future goals for the higher level analysis.

All the assembly mileposts consist of increasing dataset sizes on larger machines as they become available. Achieving these mileposts will require addressing any scalability or performance issues that come to light and any software bugs that are exposed.

FY19/Q4: 3 TB assembly on NERSC Cori.

- The goal was completed on schedule. It performed a "push-button" assembly that scales in all stages of the pipeline.
- To achieve good scaling on this dataset size, the team switched to the newer, more scalable scaffolding algorithm cgraph and transitioned to an interoperable version of MetaHipMer, which included UPC and UPC++ in a single binary.

FY20/Q4: 3 TB assembly on Summit or Perlmutter.

- The goal was completed on schedule. It demonstrated that the MetaHipMer code runs efficiently and strong-scales well on a different architecture than Cori.
- This goal required adequate GPU support in MetaHipMer. It used the new MetaHipMer2 pipeline, which was entirely rewritten in UPC++ and the GPU-optimized aligner (ADEPT) [53].

FY21/Q4: 16 TB assembly on Summit or Perlmutter.

- Demonstrate that MetaHipMer can assemble a larger dataset on Summit or NERSC9.
- This milepost will depend on the previous year and on the optimized use of GPUs for key computations, including alignment, k-mer analysis, and local assembly.

FY21/Q4: 1.5 billion proteins clustered on Summit.

- HipMCL accepts a large network file that represents protein-protein sequence similarities, which will be generated offline by LAST, a third party tool.
- This dataset required approximately 4 h using the CPU version of HipMCL on 2000 nodes of Cori/KNL.

FY22/Q4: 30 TB assembly on Summit, Perlmutter, or available exascale machine at scale.

- This will require an improved overlap of communication and computation and an overlap between the GPU executions with CPU executions (e.g., run alignment on the GPUs while running distributed hash table retrieval on the CPUs).
- It will also require a port of the GPU kernels to the exascale GPU programming models for Aurora and Frontier if those systems are used.

FY22/Q4: 1.5 billion proteins clustered de novo using PASTIS + HipMCL on Perlmutter.

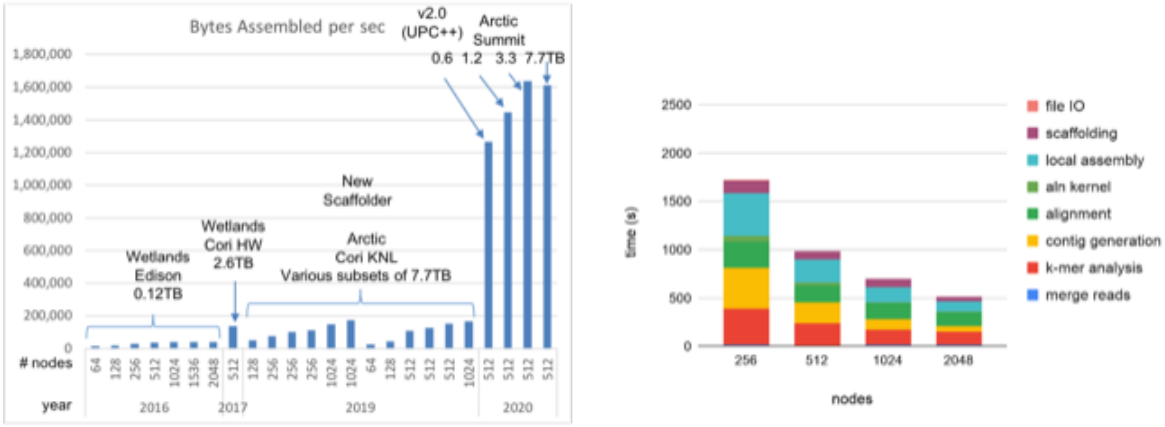


Figure 66: (Left) Performance of various versions of MetaHipMer on different node counts, problem sizes, and systems over time, showing a 44× speedup on a constant 512 nodes since 2016. (Right) Performance breakdown of the latest MetaHipMer.

- The input is raw protein sequences, and this will be a combined PASTIS + HipMCL pipeline that generates protein-protein sequence similarity network on the fly and that feeds to HipMCL.

FY23: 50 TB assembly on an exascale system (KPP-2 goal).

- This milepost will depend on the success of the previous years' efforts, and it requires collaboration with the PAGODA team on memory scaling an efficient use of the Slingshot network on Aurora and Frontier.

FY23: 1.5 billion proteins clustered de novo using PASTIS + HipMCL on Frontier.

- The input is a required port of the protein alignment code in PASTIS (joint with ExaGraph) for the AMD GPUs and use of optimized sparse matrix libraries for that architecture.

6.3.4 ExaBiome: Progress on Early and Pre-Exascale Hardware

Performance on Summit

The MetaHipMer application performance has improved from the availability of faster processors (i.e., CPU and GPU nodes), networks, and memory systems, as well as better algorithms and optimized software. Figure 66 shows the rate at which input data are assembled (bytes assembled per second, which is roughly twice the number of base pairs assembled per second). This metric not perfect because the complexity of a dataset (e.g., number of species, amount of coverage of each) can significantly change the cost of assembly. However, this graph shows assemblies performed on subsets of two environmental datasets with comparable complexity: a 2.6 TB wetlands dataset and a 7.7 TB arctic soil dataset. The wetlands data were used in 2016 but were only fully assembled as a single dataset in 2017. This was done by using 512 Cori Haswell nodes for most of the assembly and two large memory nodes for scaffolding, which had a nonscalable algorithm due to a large connected component in an intermediate stage. The scaffolder was entirely rebuilt in 2018 with a more scalable algorithm, cgraph, and in 2019, there were several attempts to assemble parts of the 7.7 TB arctic dataset on Cori's KNL nodes. The improved strong scaling with node counts is evident in those 2019 data points relative to 2017, but the largest assembly was still 2.5 TB. In 2019, the remainder of MetaHipMer was rewritten in UPC++ to match the new scaffolder, the GPU alignment was added, and the code was run on 512 Summit nodes. The last four bars show the rate at which successively larger problems were assembled, which was roughly 100 node-hours per TB, which gives us an estimate on future large-scale computations. This latest version of MetaHipMer (v2) has significantly improved performance, and it has improved 44× on 512 nodes since 2016.

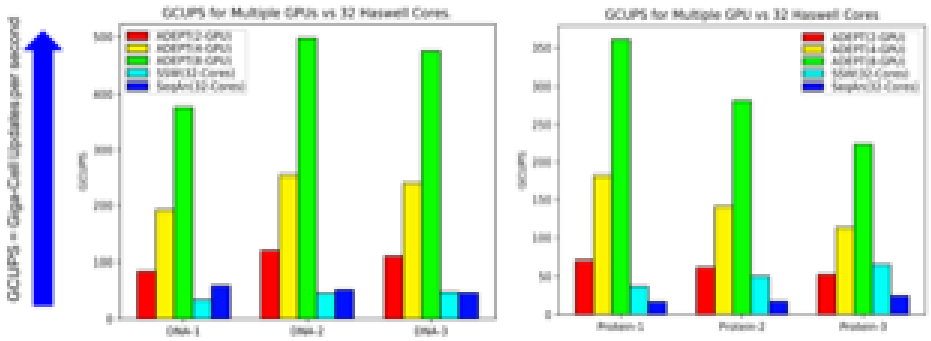


Figure 67: Performance of ADEPT alignment kernel on CPUs vs. GPUs.

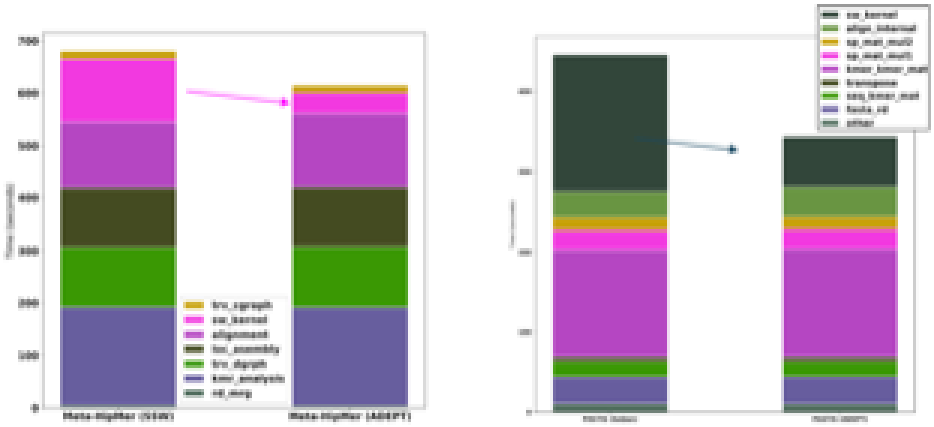


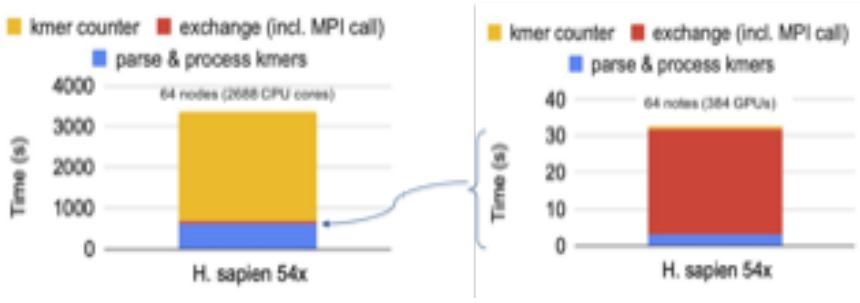
Figure 68: Performance gain from adding GPU-optimized ADEPT alignment code into ExaBiome applications.

Across the ExaBiome applications are some key computational kernels that dominate the workload, including string alignment, k-mer counting/hashing, and sparse matrix multiplication. All of these come in variations, such as for DNA and proteins, and depend on the length of strings, the information stored with k-mers, and whether a matrix is being multiplied by a dense vector, a set of dense vectors, or another sparse matrix. Figure 67 shows the performance of the new ADEPT GPU alignment algorithm as giga-cell updates per execution. ADEPT scales well and significantly outperforms the CPU implementations (SWW and SeqAn) on a 32 core Intel Haswell node. The left-hand graph shows the results on DNA, which is used in both assembly and some comparative analysis approaches. The right-hand graph shows results on proteins used in PASTIS.

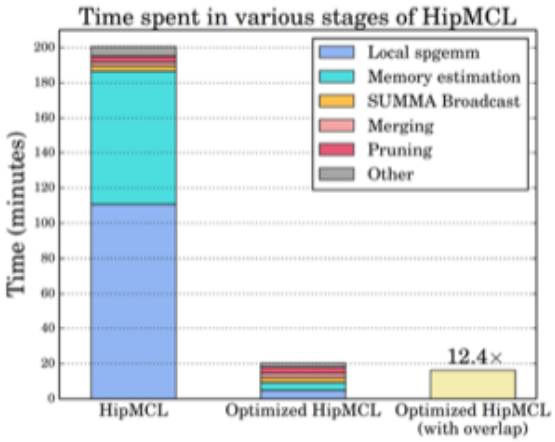
Figure 68 shows the impact on overall application performance for MetaHipMer (left) and PASTIS (right) running on a Cori CPU vs. GPU node with eight v100s from integrating the ADEPT GPU aligner. The GPU alignment work was all performed in collaboration with HI-funded staff at NERSC.

K-mer counting has multiple uses in all three application areas of the project and was more recently adapted to GPUs. More recent work shows the speedup of a stand-alone k-mer counting on multiple nodes. Figure 69a shows CPU (left) and GPU (right) performance on 64 Summit nodes. For the different y-axes, the all-to-all communication time in red is essentially unchanged between the two.

The HipMCL running time is dominated by a sparse matrix multiplication operation with application-specific features (e.g., cutoff) to reduce memory use. It was optimized for GPUs, as shown in Fig. 69b. HipMCL is based on a single node code, MCL, which previously took several days for large datasets and sometimes ran out of memory. With the new high-performance HipMCL, the clustering bottleneck began producing the input, which requires a similarity computation based on protein alignment. This similarity



(a)



(b)

Figure 69: (a) K-mer counter on Summit. (b) HipMCL on Summit.

scoring process is now captured in the PASTIS code, a distributed memory code build by ExaBiome and ExaGraph.

Next Steps

Over the next few months, the ExaBiome team will be performing additional experiments on the GPU k-mer counter, including porting to HIP and integrating into the MetaHipMer and KmerProf (comparative analysis profiling) applications. More work is ongoing on the GPU aligners ADEPT and LOGAN, including some additional optimizations, porting, and integration into PASTIS and diBELLA. A new GPU code for local assembly, one of the stages in MetaHipMer, is also underway and shows some promising initial results. This has proven to be a challenging effort because it required several data structures from the Standard Template Library to be rewritten in CUDA. For sparse matrix algorithms, the plan is to rely on vendor-optimized libraries as much as possible, except where specialized versions are required, such as in HipMCL.

Aside from these single node optimizations, ExaBiome is exploring options to improve locality and reduce the number of messages in MetaHipMer and ways of hiding communication in diBELLA. They are also focused on the large assemblies to continue meeting their milepost. Partnership with the PAGODA project has been critical at many stages to ensure scalability in running time and memory use, and we expect this to continue. Similarly, the ongoing collaboration with ExaGraph is essential to the PASTIS/HipMCL work. The team has also benefited from working closely with the NERSC NESAP team and plans to expand with other similar teams at the leadership computing facilities.

6.4 ExaFEL

The overarching goal of the ExaFEL project is to substantially reduce, from weeks to minutes, the time to analyze molecular structure x-ray diffraction data generated by the SLAC Linac Coherent Light Source (LCLS) facility. Near-real-time interpretation of molecular structure revealed by x-ray diffraction will require computational intensities of unprecedented scales coupled to a data path of unprecedented bandwidth. Detector data rates at light sources are advancing exponentially; LCLS will increase its data throughput by three orders of magnitude by 2025 with the LCLS-II-HE upgrade.

LCLS users require an integrated combination of data processing and scientific interpretation in which both aspects demand intensive computational analysis. The ultrafast x-ray pulses are used like flashes from a high-speed strobe light that produce stop-action movies of atoms and molecules. The analysis must be carried out quickly to allow users to iterate their experiments and extract the most value from scarce beam time. Enabling new photon science from the LCLS will require near-real-time analysis ($\sim$10 min) of data bursts, which require commensurate bursts of exascale-class computational intensities.

The high repetition rate and ultrahigh brightness of the LCLS make it possible to determine the structure of individual molecules, mapping out their natural variation in conformation and flexibility. Structural dynamics and heterogeneities—such as changes in the size and shape of nanoparticles or conformational flexibility in macromolecules—are at the basis of understanding, predicting, and eventually engineering functional properties in biology, material, and energy sciences. The ability to image these structural dynamics and heterogeneities by using noncrystalline-based diffractive imaging, including single-particle imaging and fluctuation x-ray scattering, has been one of the driving forces of the development of x-ray free-electron lasers. However, in experiments with large biological macromolecules, the time-dependent dynamic changes of greatest interest could involve only a few atoms out of tens of thousands, and thus the x-ray diffraction difference effects will only be on the order of 1–2 % of the total diffraction. To visualize important structural changes on this scale, very large datasets are required (10^7 diffraction patterns from randomly oriented samples), as well as new computational algorithms for more accurate analysis.

6.4.1 ExaFEL: Science Challenge Problem Description

The ExaFEL challenge problem is to create an automated analysis pipeline for serial femtosecond crystallography (SFX), also known as *nanocrystallography*. Although the traditional data analysis pipeline quantifies the diffracted Bragg spots by summation-integration (a Bragg spot typically covers more than one pixel), the envisioned exascale algorithm will model each pixel on the image and thus push the overall accuracy to the desired level.

Table 57: ExaFEL challenge problem details.

Functional requirement	Minimum criteria
Models	Iterative estimation of crystallographic structure factors from diffraction images using the size, shape, and intensity profile of Bragg spots.
Numerical methods	FFT, Quasi-Newton parameter estimation (limited-memory Fletcher-Boyden-Goldfarb-Shanno); Bayesian estimation.
Problem size and complexity	**Data ingest:** Ability to ingest diffraction images at 1 TB/s. **Memory:** Ability to store between 0.1 and 10 PB of events data in memory; each calculation will require between 10^7 (one run) and 10^9 (one experiment) diffraction images, and the size of an image will be $O(10\,\text{MB})$. **Workflow:** Ability to ingest data while the calculation is ongoing, ability to delegate data across multiple nodes for analysis, ability to exchange/average the parameter estimates across nodes, ability to off-load the most computing intensive tasks (e.g., x-ray tracing modeling step with nanoBragg) to GPU accelerators.
Demonstration calculation requirements	Run SFX against at least $O(10^7)$ images. If resources (e.g., memory) are available, then a full demonstration calculation will be performed by running SFX against $O(10^9)$ images.
Resource requirements to run demonstration calculation	The team expects to need roughly half of the exascale machine for 20 min to run the demonstration calculation for $O(10^9)$ images.

The basic workflow is envisioned as a parameter-optimization inverse problem. Traditional crystallographic data analysis is used to determine approximate starting values for the structure factors and geometric factors. The nanoBragg/CCTBX software is used to forward-simulate the diffraction images by using a parametric model of pixel intensity with which we can estimate the posterior probability of the model within a Bayesian framework. We then employ an iterative first derivative-based method to compute better parameter estimates and repeat the cycle until convergence at the maximum posterior probability.

Rapid feedback is crucial for tuning sample concentrations to achieve a sufficient crystal hit rate, ensuring that adequate data are collected and steering the experiment. The availability of exascale computing resources and a HPC workflow that can handle incremental bursts of data in the analysis will allow one to perform data analysis on the fly, providing immediate feedback on the quality of the experimental data while determining the 3D structure of the molecule simultaneously.

To show the scalability of the analysis pipeline, we plan to progressively increase the fraction of the machine used for reconstruction while keeping the number of diffraction images distributed across multiple nodes constant. The goal is to distribute the images over an increasing number of nodes while reducing the overall reconstruction time up to the point where the analysis can keep up with the data collection rates (5 kHz). Table 57 summarizes the details of the ExaFEL challenge problem.

6.4.2 ExaFEL: KPP Stretch Goal

The team proposes two high-risk, high-reward stretch goals for ExaFEL in the areas of single particle imaging and resource orchestration.

Single Particle Imaging

This stretch goal will aim at scaling SPI reconstruction with the Multi-Tiered Iterative Phasing (M-TIP) algorithm to keep up with LCLS-II data collection rates and possibly be able to run M-TIP at scale against actual experimental data. The main risk associated with being able to run against experimental data is that the ExaFEL team does not have control over which experiments will be performed at the LCLS during the lifetime of the ECP.

In SPI, diffraction images are collected from individual particles and are used to determine molecular

or atomic structure, even from multiple conformational states or nonidentical particles under operating conditions. Determining structures from SPI experiments is challenging since orientations and states of imaged particles are unknown and images are highly contaminated with noise. Furthermore, the number of useful images is often limited by achievable single-particle hit rates, which is currently $\ll$1. The M-TIP algorithm introduces an iterative projection framework to simultaneously determine orientations, states, and molecular structure from limited single-particle data by leveraging structural constraints throughout the reconstruction, offering a potential pathway to increasing the amount of information that can be extracted from single-particle diffraction.

Resource Orchestration

This stretch goal will aim at scaling resource orchestration capabilities to allow: (1) the science data to be streamed from the beamline to the supercomputer at LCLS-II throughputs, and (2) the analysis to be started within seconds from the start of data collection.

The main risk associated with this stretch goal is related to the fact that the ExaFEL team does not have complete control over these capabilities, which will require engagement with the computing facilities.

6.4.3 ExaFEL: Capability Plan

Recently Delivered

- **NanoBragg integration:** The goal was to develop GPU kernels to solve the nanoBragg inverse problem and solve for physics parameters. This capability was demonstrated on Summit using $O(10^5 - 10^6)$ diffraction patterns of randomly oriented microcrystals simulated in an XFEL beam under defined conditions.
- **Realistic simulation of SPI data:** We leveraged the GPUs available on Summit to generate $O(10^6)$ diffraction patterns of multiple conformations (>5) of a protein sample accounting for beam fluctuations, parasitic beamline scattering, and detector noise. This milepost recognizes the need for simulated SPI data to test and measure the performance of the M-TIP algorithm.
- **M-TIP workflow acceleration:** We identified steps in M-TIP that are suitable for off-loading to GPUs. In particular, we investigated acceleration along three candidate calculations critical for the new, scalable, Cartesian basis version of M-TIP: orientation matching, FFTs, and nonuniform FFTs. This work, which was performed on Summit, represents the first step toward defining the SPI acceleration strategy.
- **Resource orchestration:** The ExaFEL project aims at developing the capabilities required for streaming the science data from SLAC to the computing facility, starting the analysis job on the supercomputer, and reporting the results of the analysis back to the experimenters in quasi-real time. This milepost designs and develops these capabilities and integrates them with the LCLS data management system. This work was performed on Cori in collaboration with the NERSC and ESnet teams.

FY21

- **SFX merging with diffuse scattering:** We will merge the analysis and simulation workflows for Bragg reflections (CCTBX) and diffuse scattering (LUNUS), demonstrating the ability to concurrently process the Bragg signal (structure) and diffuse signal (dynamics) and the ability to generate simulated diffraction patterns that include the Bragg and diffuse intensity. This work will be performed on Summit.
- **Inverse modeling of the structure factors:** We will implement the modeling cycle (forward simulation with nanoBragg, parameter estimation with diffBragg) on Summit/Perlmutter. We will implement this new capability to support: (1) highly performant execution with GPU kernels for

diffBragg, (2) demonstration with experimental data, (3) modeling of multipanel detectors like the Jungfrau 4M and ePix10K, and (4) if possible, Bayesian error estimates for the structure factors. We will design for platform portability where possible (HIP).

- **SDN data path expansion to last mile at SLAC and NERSC:** We will demonstrate operational support and deployment at scale for reserving SLAC, NERSC, and network resources to enable real-time end-to-end workflows from SLAC to NERSC. The goal is to provide network performance predictability for the transfer and saturate the capacity of the physical link between the LCLS beamlines and Cori.

FY22

- **Code scaling on exascale demonstrator machine:** The SFX workflow will be finalized and ported to the exascale demonstrator machine. We will assess the scalability of SFX and identify performance bottlenecks.
- **Performance optimization of M-TIP:** The M-TIP application will be finalized and ported to the exascale demonstrator machine. We will assess the scalability of M-TIP and identify performance bottlenecks.
- **Workflow automation:** We will demonstrate the ability to automate the coordination of workflow resources to execute end-to-end workflows from SLAC to NERSC. This automation will coordinate resources (e.g., reservation, real-time queues), the data flow (e.g., streaming, SDN path), and the pipeline (i.e., the various steps in the reconstruction).

FY23

- **SFX at scale on exascale demonstrator machine:** We will finalize performance optimizations for SFX and execute the SFX demonstration calculation on the exascale demonstrator machine by running on roughly half of the exascale machine.
- **SPI at scale on exascale demonstrator machine (stretch goal):** We will finalize performance optimization for M-TIP and execute the SPI demonstration calculation on the exascale demonstrator machine by running on roughly half of the exascale machine.

6.4.4 ExaFEL: Progress on Early and Pre-Exascale Hardware

Performance on Summit

We evaluated weak scaling for nanoBragg on Summit by processing 420 images per node and changing the number of nodes from 1 to 500. Figure 70 shows how weak scaling extends in this range and how string scaling within a node performs when going from one to six accelerators.

Strong scaling was measured by calculating the wall time taken by the nanoBragg simulation of 100,000 patterns on Summit while varying the number of nodes. Figure 71 shows how the forward simulation scales with minimal performance degradation up to at least 300 nodes.

7. NATIONAL SECURITY APPLICATIONS

End State: Deliver comprehensive science-based computational weapons applications able to provide, through the effective exploitation of exascale HPC technologies, breakthrough modeling and simulation solutions that yield high-confidence insights into at least three currently infeasible problems of interest to the NNSA Stockpile Stewardship Program (SSP).

The National Security Applications within the ECP include projects (Table 58) centered on the stewardship of the US nuclear stockpile and related physics and engineering modeling and scientific inquiries consistent with that mission space. The projects are part of the ATDM program element of the DOE NNSA ASC

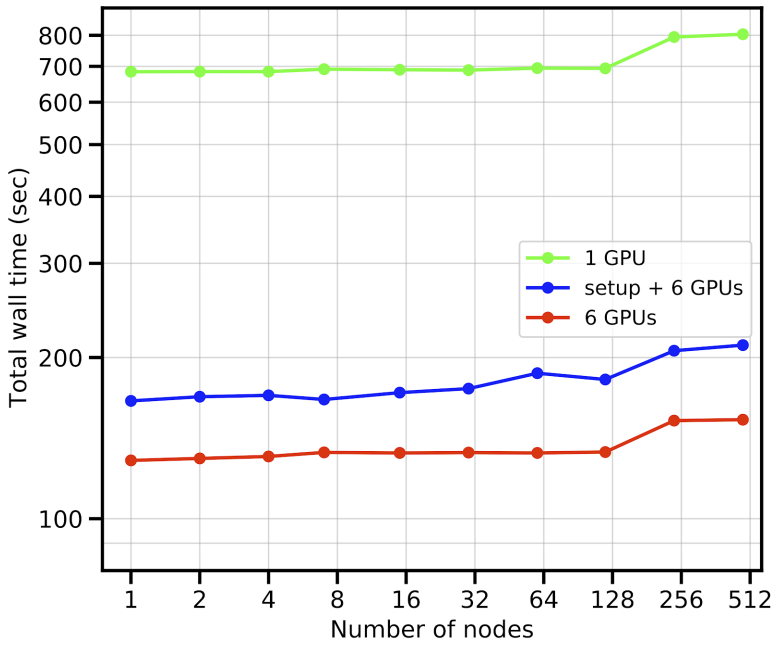


Figure 70: Weak scaling extends from 1 to 500 nodes, except for slightly ragged task completion times above 200 nodes. Overall, we observe good strong scaling within a node with little performance erosion going from one to six accelerators.

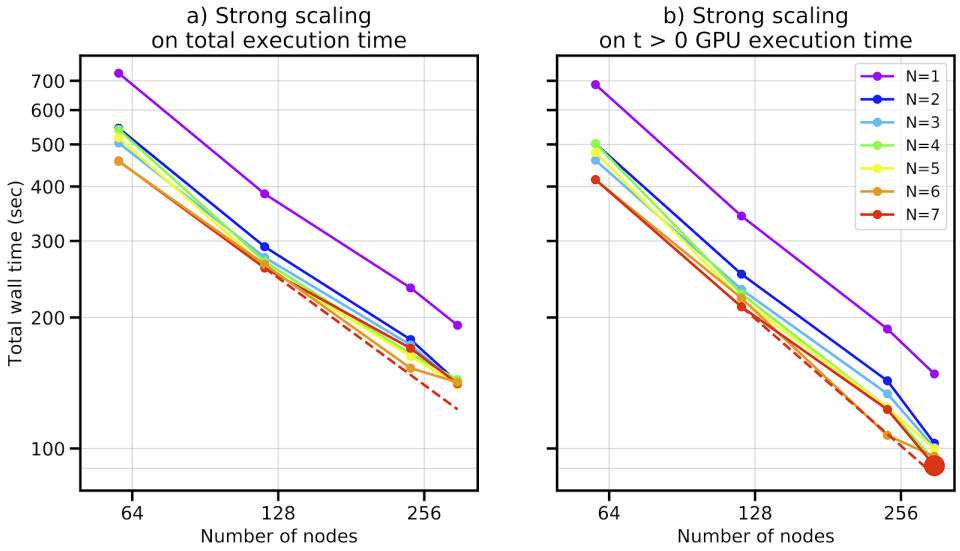


Figure 71: Wall time for nanoBragg simulation of 100,000 patterns vs. the number of nodes. N is the number of ranks (cores) that share one GPU device.

Table 58: Summary of supported NNSA L4 projects.

WBS number	Short name	Project short description	KPP-X
2.2.5.01	Ristra	ATDM LANL application	KPP-2
2.2.5.02	MAPP	ATDM LLNL application	KPP-2
2.2.5.03	SPARC	ATDM SNL application	KPP-2
2.2.5.03	EMPIRE	ATDM SNL applications	KPP-2

program and are implemented at each of the three DOE NNSA laboratories. Each project has a specific challenge problem target, namely a currently intractable 3D problem of interest. The goal of both Lawrence Livermore National Laboratory's (LLNL)'s Multiphysics on Advanced Platforms Project (MAPP) and Los Alamos National Laboratory's (LANL)'s Ristra project is to develop several integrated multiphysics codes for the simulation of mission-relevant problems. The two projects from Sandia National Laboratories (SNL) are developing the multiphysics codes EMPIRE to simulate the effect of electromagnetic pulses and SPARC to simulate aerodynamic flows and associated impacts. For all four projects, scalability has been a design priority from the beginning, and the ability of the new codes to scale to some significant fraction of future machines is an objective. High confidence in physics fidelity has also been an objective from the beginning, with development of novel multiscale and high-order algorithms and integrated verification and validation. All four applications must exhibit efficient use of one or more architectures of the ASC Advanced Technology Systems (ATS) located at the NNSA laboratories, as well as portability and high confidence in physics fidelity. The current two NNSA ATS platforms are Trinity (KNL) and Sierra (V100 GPU), and the next two ATS platforms are Crossroads and El Capitan; El Capitan is the NNSA-designated exascale computer. The outcomes and products of the ATDM activity will be incorporated into the next generation of integrated and high-performance ASC codes on advanced (i.e., next decade) architectures in support of NNSA mission scope. The LANL approach is to concurrently develop a flexible framework, code infrastructure, and physics components with multiscale algorithms. LLNL is developing new high-order algorithms to minimize data motion relative to computation, which are then incorporated into production codes built on modular, interoperable software layers. The SNL approach is built upon agile components as part of a comprehensive tool kit, which includes a data model, an abstraction layer, discretization techniques, and high-quality solvers implemented with performance-portable abstractions. The ECP NNSA applications work closely with ECP software technologies team at each laboratory. Together, the three NNSA laboratories aim to deliver applications that can address currently infeasible 3D problems of interest. These different approaches are complementary, providing both peer review and risk mitigation. As part of the FY20 milestone, each project team was instructed to perform a series of mission-relevant calculations in 3D using up to and including at least 25% Sierra and either 50% of Trinity's KNL partition or 100% of Astra—an advanced architecture prototype with ThunderX2—and report on performance and portability and comparison with existing codes performance on Commodity Technology Systems (CTS).

7.1 Ristra

The property and behavior of various materials under a wide variety of extreme conditions is central to many applications within the realm of national security. Such modeling requires multiple length scales and timescales and drive requirements for exascale computing. LANL is developing a next-generation multiphysics code for national security applications that focuses on 3D multiphysics, mesoscale insight for extreme condition materials, and high-energy density physics simulations.

7.1.1 Ristra: Science Challenge Problem Description

Ristra, LANL's next-generation code project, has taken a high-risk path that enables the exploration of next-generation physics methods implemented by using novel computer science technologies with co-designed abstractions that enforce a separation of concerns between the two disciplines. The benefit of this path is greater agility, both in the code's ability to incorporate new algorithms in physics and computer science and respond to the anticipated increased diversity in application requirements.

The name Ristra—which means "string" in Spanish, but in a New Mexican context means a string of chilis—was chosen to emphasize that the project's aim was not to develop a single monolithic code but rather a tool kit that connects a consistent suite of codes for different application domains.

The ultimate goal of the Ristra project is to create a set of codes that must:

1. solve multiphysics problems by using computational methods with characteristics of those required for national security problems;
2. solve multiphysics problems efficiently on emerging HPC architectures leading to exascale;
3. provide a flexible, extensible, productive programming environment that features a separation of concerns between complex physics expression and underlying HPC technologies, thereby enabling agile response to future drivers from mission needs and computing technology; and
4. provide a realistic test bed for the evaluation of novel programming models and data management technologies.

Computer science technologies that allow for the efficient use of emerging HPC architectures suggest a need for physics algorithms that permit increased concurrency at many scales. This motivates a fresh look at the numerical decisions made throughout the simulation process, from setup through analysis. With this in mind, Ristra is casting a wide net across available physics algorithms for multiphysics simulation in addition to an exploration of programming models for emerging architectures.

Ristra's focus is on two application domains, both of which feature multiscale methods that will be an important component of extreme-scale multiphysics simulations of the future.

- High-Energy Density Physics for Inertial Confinement Fusion. Ristra's Symphony code is an unstructured multimaterial radiation hydrodynamics application that features a multiscale algorithm for the radiation solve. A fully coupled low-order radiation hydrodynamics system is updated by a high-order radiation solver that could be executed asynchronously.
- *Multiscale Hydrodynamics of Materials in Extreme Conditions.* Ristra's FUEL code is an unstructured multimaterial arbitrary Lagrangian-Eulerian (ALE) hydrodynamics code that can be coupled to complex material models to take account of mesoscale physics, such as grain structure, in the dynamic response of materials. Mesoscale modeling is computationally intensive, and the multiscale approach could effectively use exascale-class systems and provide a promising target for data-driven ML techniques.

Two key challenges on the path to multiphysics computing at exascale and beyond are: (1) abstracting details of underlying hardware and systems software from multiphysics code development and (2) solving mission-relevant problems at multiple physical scales.

To address the first challenge, Ristra is developing a computer science interface called FleCSI that limits the impact of disruptive computer technology on the development of multiphysics codes. FleCSI enables the adoption of novel programming models and data management methods to address the challenges and diversity of new technology.

Simultaneously, Ristra is exploring the use of multiscale numerical methods that offer improved physics fidelity and computing efficiency in high-energy density (e.g., inertial confinement fusion) and low-energy density (e.g., advanced materials and solid dynamics) regimes relevant to NNSA's mission of stockpile stewardship. Figure 72 provides an example of physics and methods that Ristra is undertaking.

In a multiphysics code, users must address a variety of discretization types (e.g., structured mesh, unstructured mesh, and particle-based), as well as the need to operate them together. Therefore, an important part of the project is Portage, a remap and link library that allows physical geometry and data to be

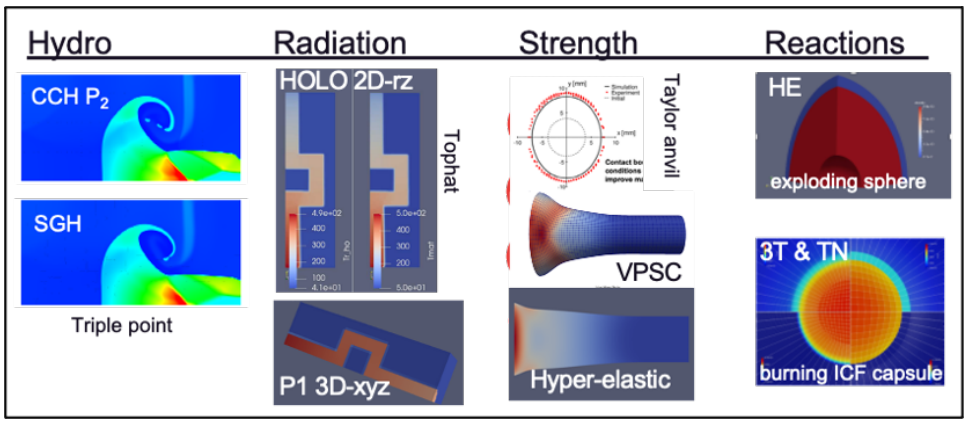


Figure 72: Example of the multiple physics and computational methods that Ristra is undertaking.

mapped between discrete representations. Portage has been developed as a stand-alone library that can be readily extended to provide link capabilities between arbitrary computational physics codes but is also sufficiently lightweight to serve as an inline remap within Ristra codes (e.g., for ALE in several of our reference hydrodynamics codes).

Ristra has adopted a software architecture that emphasizes a separation of concerns between the computational physicists who must respond to a diversity in questions of interest and computer scientists who must adapt to changing software and hardware computing technologies. That separation of concerns is implemented through an open-source abstraction layer called FleCSI. FleCSI is a tool kit for building discretizations and physics operators to support physics expression over an abstract data and execution model that can target a variety of underlying parallel runtimes ranging from traditional MPI implementations to more capable modern runtimes, such as the Legion programming system that originated at Stanford University and is being actively developed at NVIDIA and LANL.

The FleCSI execution model is an asynchronous task-based programming model that is well-suited to highly heterogeneous computer architectures and explorations of new, more asynchronous algorithmic techniques. Specifically, the Ristra project is exploring novel multiphysics couplings in its codes, including several multiscale algorithms that break the conventional operator split implementations of current multiphysics codes into a more concurrent model that could benefit execution on extreme-scale computer platforms.

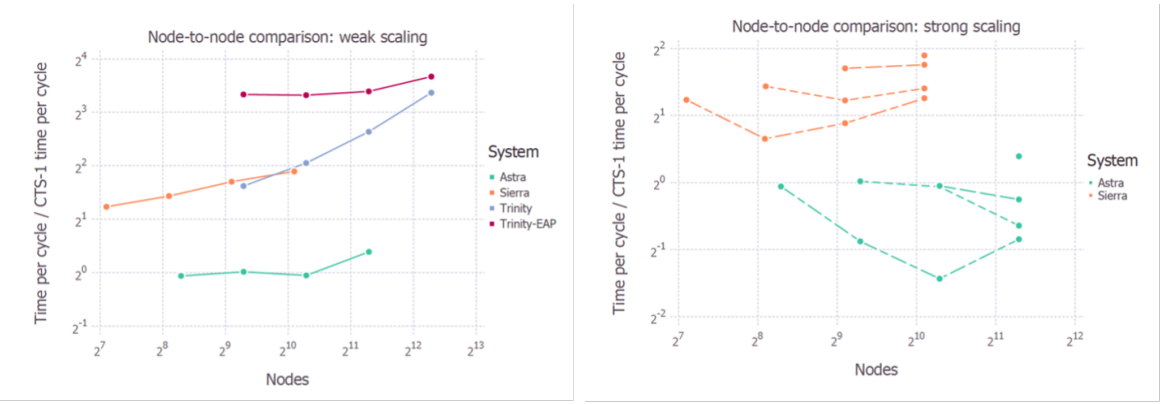


Figure 73: (Left) Weak scaling of the demonstration problem, including runs at the target scales of 50% Trinity KNL, 25% Sierra, and 100% Astra. (Right) Strong scaling for the demonstration problem on Sierra and Astra. Numbers are normalized to CTS-1 performance at 625 nodes.

FleCSI introduces a functional programming model with control, execution, and data abstractions that are

consistent with MPI and state-of-the-art, task-based runtimes, such as Legion and Charm++. The abstraction layer insulates developers from the underlying runtime while allowing support for multiple runtime systems, including conventional models such as asynchronous MPI. The intent is to provide developers with a concrete set of user-friendly programming tools that can be used currently while allowing flexibility in choosing runtime implementations and optimizations that can be applied to future architectures and runtimes. FleCSI's control and execution models provide formal nomenclature for describing poorly understood concepts, such as kernels and tasks. FleCSI's data model provides a low buy-in approach that makes it an attractive option for many application projects because developers are not locked into particular layouts or data structure representations.

For the FY20 milestone, Symphony was run on Sierra, Trinity, and Astra, and its performance and memory usage were documented. A successful run of a demonstration multiphysics problem with Symphony was run on 25% Sierra, 50% Trinity KNL, and 100% Astra. Symphony was also demonstrated with MPI and Legion runtimes. Scaling results are shown in Fig. 73. Symphony is portable to MPI and Legion parallel back ends and has run with both on a variety of platforms, including Sierra. In Fig. 74, MPI vs. Legion strong-scaling behavior is shown for the demonstration problem on a smaller mesh.

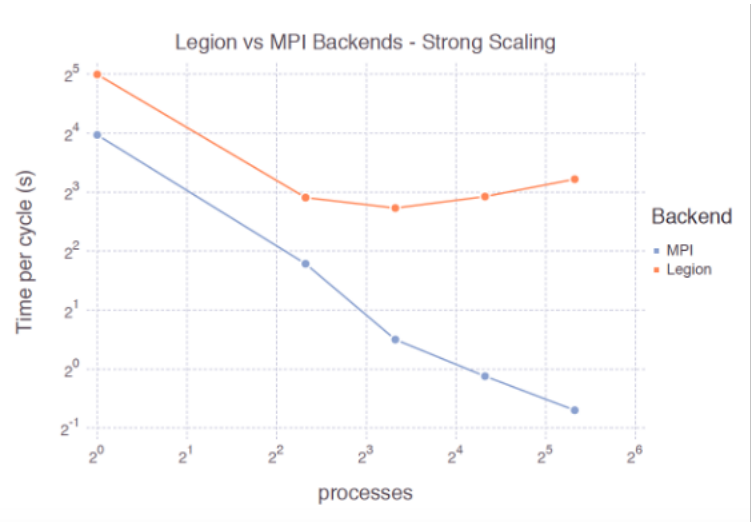


Figure 74: Legion vs. MPI strong scaling of the demonstration problem on a small mesh on Sierra.

FleCSI generally imposes negligible overhead relative to the underlying runtime. As anecdotal evidence, when using a code with both FLeCSI/MPI and stand-alone MPI running a realistic five-material compression problem on CTS-1, no discernible difference in runtime is observed. For Legion and the simple test case, Legion can often be a factor of 2 slower in an initial implementation. Progress toward resolving this through appropriate task granularity and grouping is underway. Hierarchical tasks are a potential path toward back-end agnostic performance of the tasking model are available for us to explore in FleCSI 2.0 in the future.

7.2 MAPP - Multiphysics on Advanced Platforms Project

The ATDM effort at LLNL has embraced two key themes: the use of high-order numerical methods and a modular approach to code development. The LLNL next-generation effort is organized under MAPP. One foundational component of MAPP is the Axom computer science tool kit, which provides infrastructure for the development of modular, performance portable, multiphysics application codes. MARBL is a next-generation application code built on the Axom base to address the modeling needs of the high-energy density physics (HEDP) community for simulating high-explosive, magnetic, or laser-driven experiments—such as inertial confinement fusion (ICF), pulsed-power magnetohydrodynamics (MHD), EOS, and material strength studies—as part of the NNSA's SSP (Fig. 75).

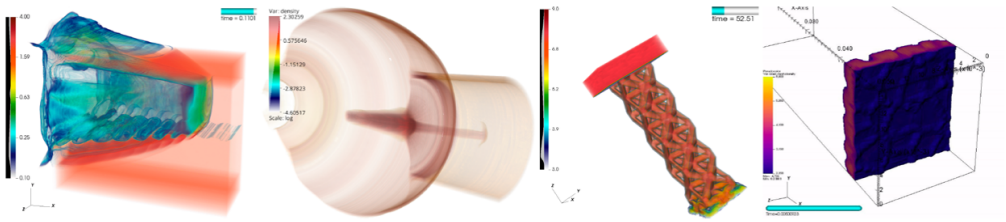


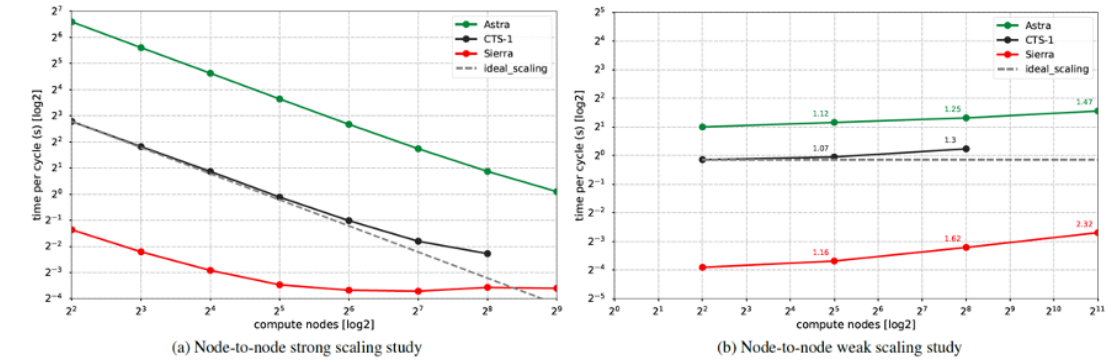
Figure 75: Example MARBL high-order 3D simulations. From left to right: radiation-driven Kevin-Helmholtz instability experiment, BRL81a shaped charge, octet-truss high-velocity impact, and radiation-driven high-density carbon micro-structure shock experiment.

MARBL was designed from inception to support multiple diverse algorithms, including ALE and direct Eulerian methods for solving the conservation laws associated with its various physics packages. One distinguishing feature of MARBL is the use of advanced, high-order numerical discretizations, such as high-order finite element ALE and high-order finite difference Eulerian methods. This algorithmic diversity encompasses the ECP simulation motifs of unstructured and structured AMR. High-order numerical methods were chosen because they have higher resolution and accuracy per unknown compared with standard low-order finite volume schemes and because they have computational characteristics that play to the strengths of current and emerging HPC architectures. Specifically, they have higher Flops/B ratios, meaning that more floating-point operations are performed for each piece of data retrieved from memory. This leads to improved strong parallel scalability, better throughput on GPU platforms, and increased computational efficiency.

One key goal for MARBL is enhanced end-user productivity, including improved workflow for problem setup and meshing, simulation robustness, support for uncertainty quantification and optimization driven ensembles, and in situ data visualization and analysis. High-order ALE and Eulerian schemes have proven to be more robust and should significantly improve the overall analysis workflow for users. The advanced simulation capabilities provided by MARBL will improve user throughput along two axes: faster turnaround for multiphysics simulations on advanced architectures and less manual user intervention.

The success of MAPP will ultimately be determined by the degree of adoption of its simulation tools by the LLNL user community and beyond. To this end, emphasis at this relatively early stage of development has been placed on adding physics and capabilities to meet the current state of the art that users demand from today's petascale production simulation codes. In the case of MARBL, this includes coupled multimaterial radiation-magneto-hydrodynamics, thermonuclear burn for ICF fusion calculations, general EOS, material opacities and electrical conductivities, simulation diagnostics and queries, in situ analytics and rendering, and parallel computational and file I/O performance at a massive scale. Additionally, the performance of the new codes on advanced architectures, such as the GPU-based Sierra and El Capitan systems, at LLNL is critical. The portability of the software stack and long-term maintainability are also critical, placing stringent demands on the integration and interoperability of high-quality production-level software libraries and tools. Finally, MARBL will be the first demonstration of the viability of advanced high-order numerical approaches for production multiphysics simulation at scale in the NNSA, and it has already produced first-of-a-kind simulation results via such methods.

The LLNL ATDM code project represents a massive software development effort, incorporating multiple physics, mathematics, and computer science packages into the overall integrated code. The project collaborates with multiple ST projects to integrate these production-quality capabilities, including software developed internally at LLNL, externally from the ECP, and in the broader open-source community. To facilitate this, much time and effort were invested in developing a robust build, test, and documentation system for the MARBL project, including preliminary use of the Spack package manager. Additionally, the project employs the extensive use of continuous integration and nightly regression testing across multiple platforms via multiple compilers to foster a culture of rigorously tested software developer contributions from a multidisciplinary, geographically distributed team. As a result of this initial investment in software infrastructure and our focus on maintaining a collaborative culture, MAPP has successfully integrated multiple packages and capabilities into the code, including RAJA performance portability abstraction and Umpire host/device



(a) Node-to-node strong scaling study

(b) Node-to-node weak scaling study

Figure 76: Node-to-node (a) strong-scaling and (b) weak-scaling studies for a Lagrangian Triple-Point-3D problem on an unstructured NURBS mesh. Our weak-scaling study scales out to 2048 compute nodes, comprising half of Sierra and more than 80% of Astra. The annotations by each data point indicate the weak-scaling efficiency with respect to the first data point in the series.

memory management, Ascent library integrated via Conduit and Axom datastore, HDF5 binary parallel checkpoint/restart enabled via Conduit and Axom datastore, Jupyter notebook interface prototype enabled using Conduit and Axom datastore, and performance monitoring with Caliper and SPOT tools. Additionally, the project relies heavily on the MFEM and Hypre libraries for high-order finite element capabilities and internode and intranode parallel performance.

The project has focused heavily on achieving high performance on GPU platforms by using a combination of software abstractions and efficient "matrix-free" algorithms in close collaboration with the Center for Efficient Exascale Discretization (CEED) CD center. The multiple C++-based packages in MARBL use RAJA for parallel loop execution, and the high-order Eulerian Fortran-based package uses OpenMP. MARBL uses Umpire for host/device memory management, including memory pools shared between packages. The new MFEM 4.0 GPU abstraction developed under CEED was co-designed with our project to enable interoperability with RAJA and Umpire for host/device execution and memory management. This co-design process was essential for achieving GPU performance in the integrated MARBL code.

For the FY20 tri-lab milestone, LLNL's MAPP team chose to run its scaling simulations on Sierra, Astra, and CTS-1. The team has performed first-of-a-kind multiphysics simulations at large scale on Sierra, taking full advantage of GPU acceleration. They successfully scaled high-order ALE hydro to half of Sierra and to all of Astra. An example of one of the scaling study results is shown in Fig. 76 for a hydrodynamics triple point problem.

This achievement also included:

- successfully integrating multiple GPU-capable libraries to enable performance portability, including MFEM v4.0, RAJA, Umpire, and Caliper;
- achieving full code stack ports to Sierra and Astra platforms;
- achieving full GPU ports of high-order ALE and high-order Eulerian hydrodynamics;
- achieving >15× GPU node vs. CPU node speedup for high-order ALE hydro;
- achieving >10× GPU node vs. CPU node speedup for high-order direct Eulerian hydro;
- establishing detailed performance monitoring and tracking ability using Caliper and SPOT and automated performance testing; and
- developing a new scaling study capability for generating strong, weak, and throughput scaling data for CTS and ATS platforms.

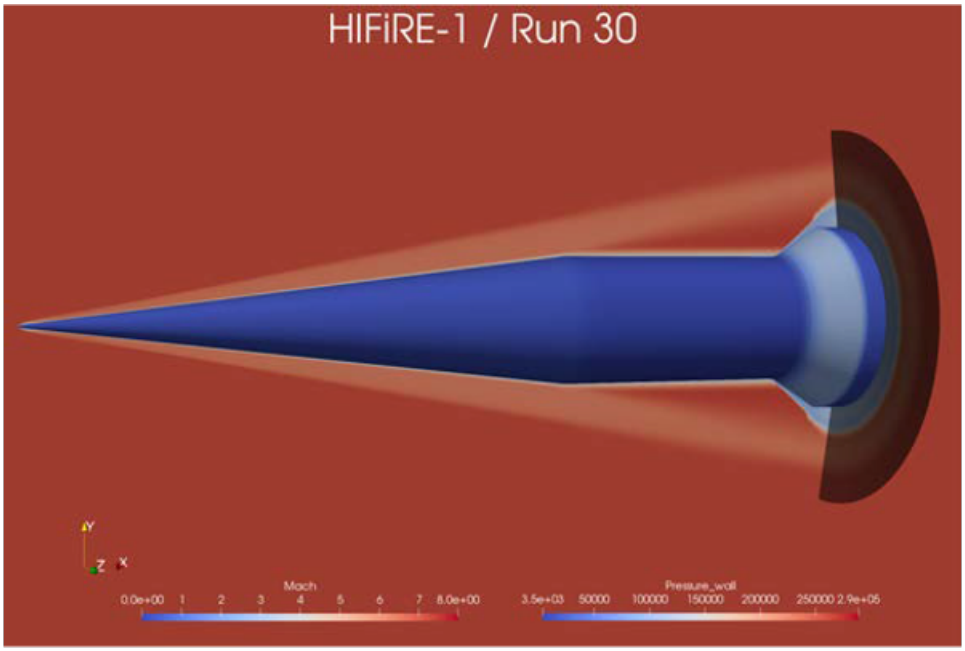


Figure 77: HIFiRE-1, a prototypical reentry vehicle geometry.

7.3 SPARC

The Sandia Parallel Aerodynamics and Reentry Code (SPARC) application represents a revolutionary hypersonic reentry simulation capability that captures the random vibration and thermal environments created by the reentry of a vehicle into the earth's atmosphere. SPARC incorporates the innovative approaches of ATDM projects on several fronts, including the effective harnessing of advanced and heterogeneous computing architectures via Kokkos, exascale-ready parallel scalability, uncertainty quantification, the implementation of state-of-the-art reentry physics and multiscale models, the use of advanced verification and validation methods, and the enabling of improved workflows for users.

The science challenge problem for SPARC is to perform a virtual flight test of a reentry vehicle in its entirety to predict the structural and thermal response of the vehicle's components under simulated reentry environments. Performing this analysis includes simulating the flow field around the vehicle, including the aft end and its wake, by using a turbulence model suited for hypersonic, unsteady turbulent fluid dynamics. The thermal loads generated from the computational fluid dynamics simulation will be used to predict the ablation and thermal response of the vehicle's thermal protection system and internal components. The structural loads generated from pressure and shear stress fluctuation predictions by the turbulence models will be used to analyze the vibrational response of the vehicle and its internal components.

To meet mission needs, SPARC must be able to simulate reentry phenomena with different fidelities. These different fidelities necessitate different turbulence modeling approaches and solver approaches, and they could benefit from different numerical discretization approaches. Furthermore, SPARC must be applicable in transonic, supersonic, and hypersonic speed regimes. This drives the need for a diverse set of physical models.

SPARC has supported multiple spatial discretization approaches since its inception. The original approach for handling multiple discretization changed in favor of better support for next-generation platforms and to have a more sustainable code design. However, using multiple discretizations is still viewed as critical to support the many physical fidelities that the code is targeted to predict. SPARC implements structured and unstructured cell-centered finite volume approaches, structured high-order finite difference, and unstructured discontinuous Galerkin methods that satisfy the summation-by-parts (SBP) property and a continuous Galerkin finite element method. Once the governing partial differential equations have been discretized in space and time, a coupled set of nonlinear equations must be solved. SPARC uses different solvers from the Trilinos package.

SPARC uses a block-tridiagonal line relaxation-based solver from Ifpack2 for the solution of linear

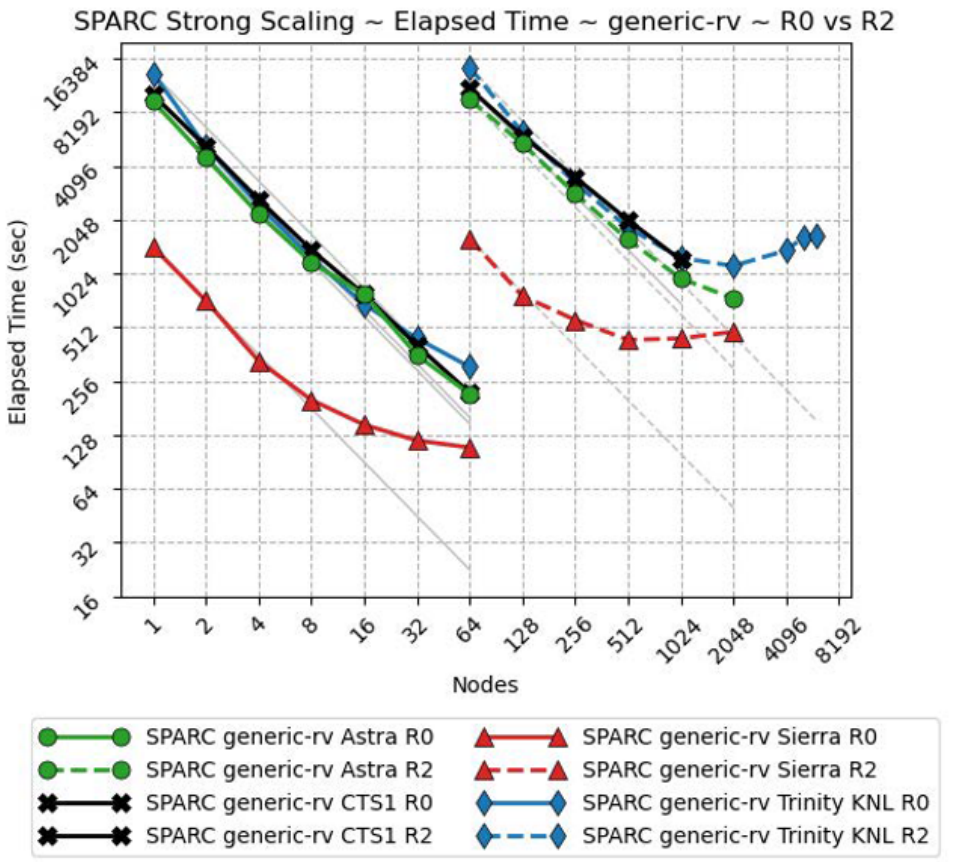


Figure 78: Strong scaling for a generic reentry vehicle simulation.

systems [54]. The core computational kernels of this solver are implemented in KokkosKernels, and the data migration and other ancillary capabilities reside in Ifpack2. Optimizations of the linear solve on GPUs include team-based tridiagonal factorization/solve, improving data layout, norm kernel fusion, and atomic updates in the block SpMV. Through these improvements, as much as a 2× improvement was demonstrated in the linear solver on the ATS-2 platform on up to 256 GPUs. In support of the ATDM L1 milestone, a synchronization bug was diagnosed that was causing segmentation faults and parallel-serial inconsistency problems in Ifpack2/KokkosKernels, which manifested on the SNL ATS-2 proxy platform Vortex. Additionally, an improved data packing algorithm was implemented by using CUDA multiple streams to prepare data for MPI communication in Ifpack2.

In addition to supporting the ATDM L1 milestone simulations, the solvers team developed a new semi-coarsening algebraic multigrid (AMG) linear solver variant for realistic problems (e.g., HIFiRE-1) that is much more robust than previous AMG options. The key idea behind the new solver is to employ a combination of structured semi-coarsening (i.e., avoids coarsening orthogonal to boundary layer) and line solves. It resulted in 2–3× runtime improvements in the solution phase for realistic SPARC HIFiRE-1 problems (Fig. 77) over other options that use Belos and point sparse formats. The SPARC team has worked with the Kokkos team to identify and remove numerous unnecessary device synchronizations and exhaustively profiled MPI usage to verify expected MPI use.

The SPARC code leverages Sacado, an automatic differentiation package in Trilinos, primarily for analytic parameter sensitivity computations in support of sensitivity analysis, optimization, and uncertainty quantification. The entire block-structured, cell-centered, finite volume residual computation and quantity-of-interest postprocessors in SPARC are differentiated by Sacado to provide analytic sensitivities for model parameters, such as free-stream boundary conditions and gas model coefficients. Sacado is interoperable with Kokkos, enabling efficient and thread-scalable differentiations of Kokkos-based computational kernels that

are portable across diverse CPU and GPU architectures.

The portability strategy for SPARC is to use the Kokkos programming model to implement domain-specific data structures and abstractions for common mesh traversal patterns, then implement all physics code by using those data structures and abstractions. This allows all the physics code to be platform-agnostic while enabling excellent cross-platform performance by tuning a small amount of platform-specific code in the core data structures and mesh traversal abstractions. Overall, 96.1% of the SPARC source is platform-agnostic as measured via the Code Base Investigator tool. Additionally, SPARC leverages Trilinos components to provide performance-portable linear solvers. Using this strategy, SPARC has successfully demonstrated portability across CTS-1, Trinity, Sierra, and Astra and is an excellent example of the possibility of writing single-source portably performant applications using Kokkos.

The choice of portability strategy for SPARC enables a few key platform-specific optimizations in ways that are minimally intrusive to the physics code. Of particular importance is the ability to choose the threading strategy, data layouts, and mesh iteration patterns per platform. For example, on Trinity, the optimal approach for SPARC's matrix assembly kernels is to use graph coloring to avoid race conditions between threads, whereas on Sierra, an approach that leverages the high-performance hardware atomics to maximize the number of appropriately interleaved memory accesses is optimal. Supporting these alternative approaches requires no changes to SPARC's physics code due to the portability strategy. Leveraging Trilinos for linear solvers is also key. The Trilinos team has been able to use SIMD types inside solver implementations to maximize performance on CPU platforms such as Trinity with long vector lengths while also maintaining alternative implementations for Sierra that maximize GPU performance. Because this platform-specific code is in Trilinos that SPARC accesses through a common interface, it enables sharing the cost of multiple implementations across multiple projects.

Performance results for a generic reentry vehicle geometry are shown in Fig. 78 and Fig. 79 for strong scaling and weak scaling, respectively. For strong scaling, with CTS-1 as baseline, Trinity is 0.8–1.2×, Astra 1.2–1.4× and Sierra is 2–8× faster. SPARC's block tridiagonal linear solver is the primary limiter of strong scaling. Some increase in elapsed time at the largest scales for Trinity and Sierra was observed. For weak scaling, good scaling is observed for Trinity and Astra with sufficient work per node, and significantly more work per node is required for good weak scaling on Sierra.

7.4 EMPIRE

The EMPIRE application is an advanced electromagnetic and plasma physics capability that will support the analysis of electromagnetic pulse (EMP) phenomena. Validated computational simulation tools are critical because many of these plasma environments must be extrapolated from what can be realized with test facilities. The phenomena encountered in these environments require models of extremely complicated gas chemistry, plasmas, and electromagnetic (EM) fields over a wide range of conditions. EMPIRE accordingly includes particle (kinetic), fluid (continuum), and hybrid particle-fluid plasma representations to efficiently represent a broad range of plasma densities and interactions.

EMPIRE has embraced SNL's ATDM components vision to build on foundational capabilities developed and deployed by other teams, particularly the discretization and linear solver technology deployed in Trilinos. EMPIRE also incorporates the innovative approaches of ATDM projects on several fronts, including the effective harnessing of heterogeneous compute nodes using Kokkos, the exascale-ready parallel scalability, the implementation of state-of-the-art plasma physics, and the use of advanced verification and validation methods. In doing so, EMPIRE has achieved the required portability across the three target ATDM computing platforms—Sierra/ATS-2, Trinity/ATS-1, and Astra/Vanguard—with no platform-specific code in the EMPIRE code base.

The science challenge problem for EMPIRE is to perform large-scale kinetic plasma simulations of an experiment fielded at the National Ignition Facility (NIF) and Z. In this experiment, x-rays generated by NIF or Z interact with surfaces to generate a plasma by the photoelectric effect. This simulation will demonstrate the particle-based plasma representation capability in EMPIRE at large scale and builds upon the progress made in the earlier validation of a simpler diagnostic fielded on NIF and Z. Models for surface emission, space-charge limited emission, neutral blow-off, and particle collisions will be used, and the ability of EMPIRE to scale to billions of elements and hundreds of billions of particles will be shown to achieve a resolution fidelity beyond what is possible with the current plasma simulation capability.

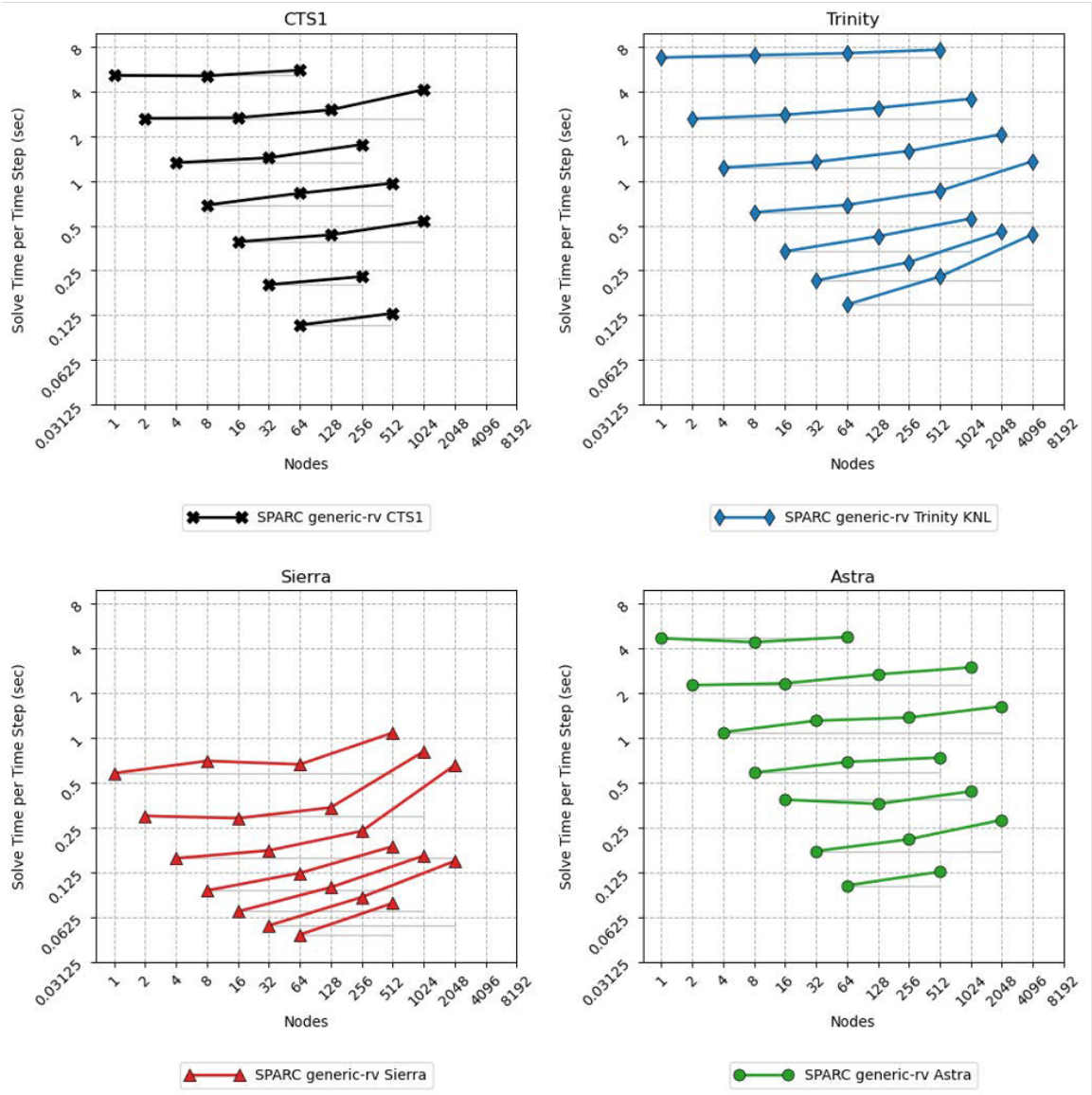


Figure 79: Weak scaling for a generic reentry vehicle simulation.

The EMPIRE simulations employ a combination of linear solvers and preconditioners from the Trilinos Tpetra stack to solve Maxwell's equations. The sparse linear algebra is provided by Tpetra (e.g., maps, vectors, matrices, and data movement). The Belos library provides the Krylov solvers, specifically the conjugate gradient (CG) method. The main preconditioner is a multilevel solver in MueLu that is specifically designed for Maxwell's equations. The fine-level smoother is a Chebyshev polynomial smoother whose computational kernel is a sparse matrix-vector multiply. Two inner multigrid hierarchies are applied in an additive fashion. For the milestone simulations, the field solve operators are constant over the course of the simulation, so the preconditioner can be set up once and used repeatedly.

The ATS-2 GPU-based architecture presents several challenges to delivering scalable solvers for EMPIRE. To fully exploit GPUs, algorithms must supply a much larger workload than CPU-based architectures. As a result, the team has needed to consider significantly different data rebalancing requirements than with CPUs. Another challenge has been surrounding the effective use of profiling and debugging tools. Profiling and debugging tools are being developed and refined concurrently with the algorithms that can use those tools, which has made addressing transient errors challenging. The solvers and KokkosKernels team put forth considerable effort in identifying what was ultimately found to be a CUDA bug. A reproducer and bug report were provided to NVIDIA's compiler team.

The fluid and hybrid capabilities in EMPIRE leverage Sacado for computing analytic Jacobians of the discretized finite element residual in support of Newton-based nonlinear solvers for implicit time stepping. EMPIRE's finite element assembly computations for each physics operator are differentiated by Sacado to provide the corresponding Jacobian contribution, which are then assembled in the global Jacobian matrix. Integrating Sacado with Kokkos enables portable performance of these derivative computations across architectures. Furthermore, a hierarchical parallelism capability was developed for Sacado and Kokkos, allowing fine-grained (i.e., warp-level) parallelism to be mapped across Sacado derivative computations, resulting in substantially improved speedup over flat parallelism on NVIDIA GPUs for several key EMPIRE physics discretization kernels relevant to hybrid simulations.

EMPIRE was started with performance-portable code in mind. The goal of performant-portable software is to have code that is near in performance to an architecture-specific implementation (performant) with little to no architecture-specific code (portable). EMPIRE achieves this through the Kokkos programming model, which abstracts the hardware-dependent portions behind a consistent API. Addressing the notion of portability is therefore quite simple because there is no architecture-specific code in EMPIRE directly, yet the code will run on the different platforms simply by recompiling the code appropriately. All architecture-specific features appear either in Kokkos, such as scatter views that switch between replication and atomics based on hardware capabilities, or in solver parameter defaults that are tuned specifically for the multilevel solver to optimize performance on each platform. In terms of platform-specific performance optimizations, several algorithms in EMPIRE were changed to improve GPU computation. These were found to incur no overhead on other platforms so are not considered platform-specific.

EMPIRE was tested across a wide range of platforms to gather performance and scaling data. On all platforms, the MPI+Kokkos implementation was found to outperform the pure-MPI approach, and relative scaling between platforms behaved as expected. A comparison of performance on all three target platforms for the milestone target problem, presented in Fig. 80 and with CTS-1 included as a reference baseline, shows that Sierra is much faster on a per-node basis (on the order of 8×) than the other platforms. On the other hand, performance on Trinity/KNL lags significantly (on the order of half the expected performance) primarily because EMPIRE cannot take advantage of vectorization on that platform due to its unstructured mesh. Kokkos does not provide this level of optimization, which is a challenge that must be balanced against portability because platform-specific code would be required to achieve such gains. Also, effort has not been expended to optimize performance on Trinity/KNL because the team's attention has moved to the even greater challenge of performance on Sierra. There is a difference here between portability, which ensures that the same code runs on each architecture, and performance, which is not guaranteed and typically requires much more effort to achieve.

Figure 81 shows strong and weak scaling for each platform (CTS-1 for reference, Trinity/KNL, Sierra, and Astra) divided by the two main components of each time step. These are the particle update—which includes weighting the fields, accelerating the particles, moving the particles, and weighting the resulting currents to the mesh—and the linear solve, which involves solving a linear system for the electric and magnetic fields. Across all the machines, the particle update is two to four times more expensive than the linear solve but

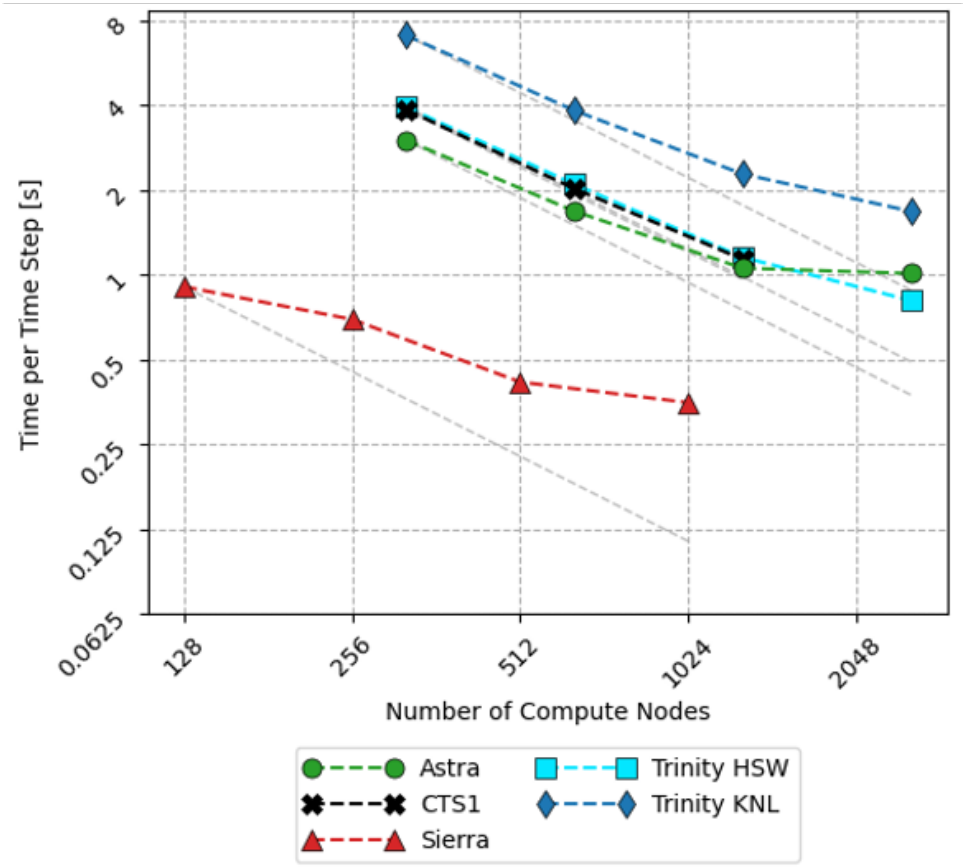


Figure 80: Strong scaling of the milestone target problem at 200 million elements resolution compared across the milestone platforms.

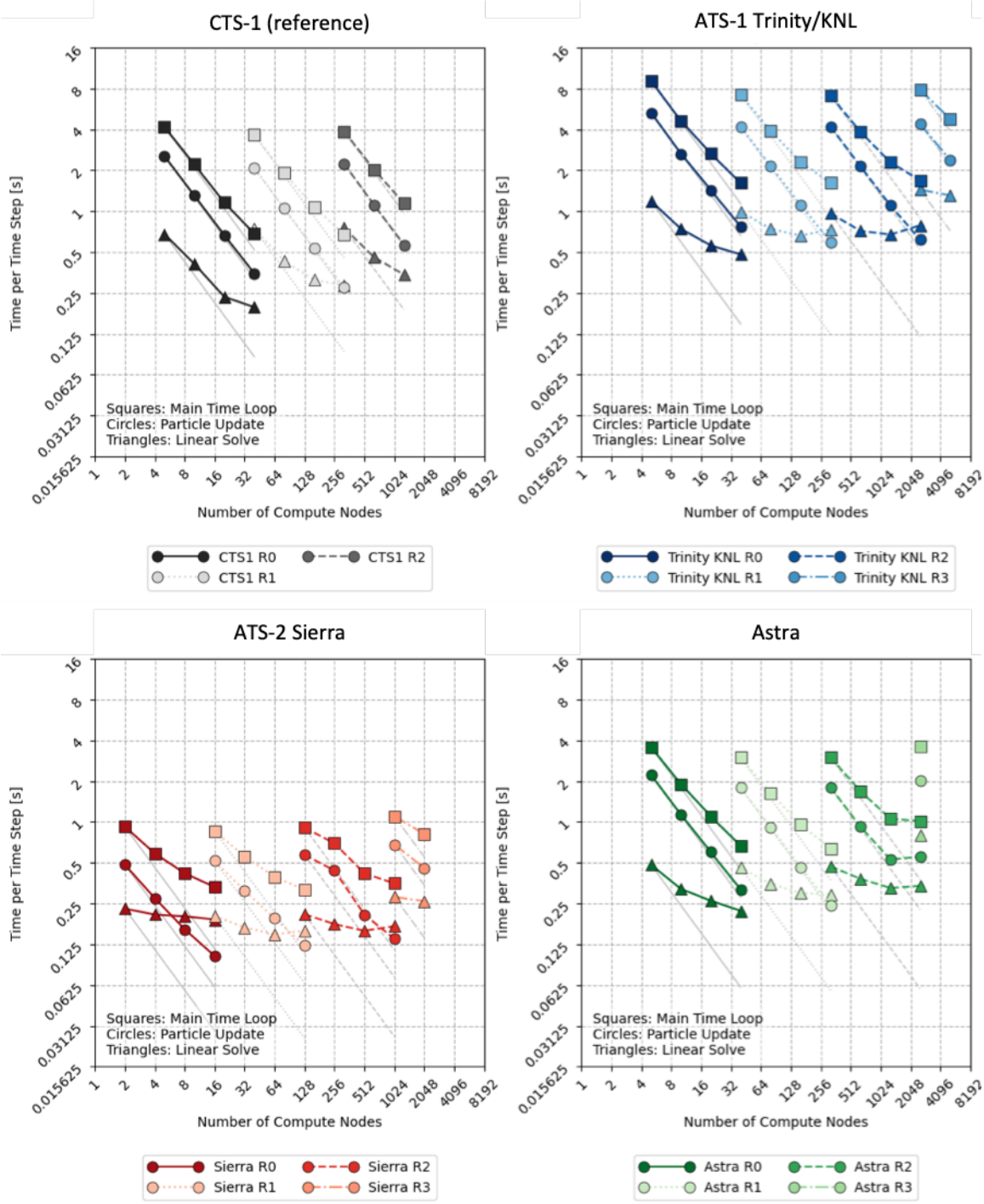


Figure 81: Strong- and weak-scaling results for the milestone target problem on each platform with the algorithmic breakdown between particle update (circles) and linear solve (triangles) shown for each run. Data were collected on up to 2,048 nodes on Sierra (47% of the total nodes), 5,120 nodes on Trinity/KNL (52%), and 2560 nodes on Astra (99%).

strong-scales nearly perfectly across all meshes and hardware architectures. On the other hand, the linear solve does not strong-scale as well. Much work was done to improve the solver performance and identify efficient preconditioning parameters on each system. This resulted in massive reductions in actual solve times, approaching our goal of 10 solves per second at scale in several cases. However, the overhead of communication and multigrid reductions is now more apparent in the solve as we near and exceed the scaling limit for these problems. In production simulations, the balance between particle work and linear solver work would be carefully considered to maximize total throughput, which is impossible in the context of these scaling studies in which resolution and refinement ratios are fixed.

Looking beyond the deployment of ATS-2, the Kokkos-based approach that EMPIRE has pursued provides a clear path to portability on the upcoming ATS-3 (Crossroads) and ATS-4 (El Capitan) systems. Initial work is already underway on Kokkos to develop the infrastructure to support those systems. We anticipate starting work to test EMPIRE on these systems by early FY22, depending on the finalization of the hardware specifications.

8. CO-DESIGN

End State: Develop cross-cutting, motif-based, co-designed software technologies and integrate them into applications, providing them with the potential to fully use exascale hardware technologies and achieve their challenge problem capabilities. Direct HPC vendors and R&D staff on the key application characteristics that must inform the co-design of exascale software and hardware technologies through proxy application software.

The CD activity includes six CD centers focused on specific computational motifs. These target cross-cutting algorithmic methods that capture the most common patterns of computation and communication, known as *motifs*, in the ECP applications. The current list of motifs includes structured and unstructured grids with AMR, dense and sparse linear algebra, spectral methods, particle methods, MC methods, backtrack/branch-and-bound, combinatorial logic, dynamic programming, finite state machine, graphical models, graph traversal, and map reduce. All of these motifs and others that might emerge over the lifetime of the ECP—including ML methods, the focus of a recently added CD center—will be considered within the CD activity with the exception of dense and sparse linear algebra, which is supported within the ST focus area.

Each of the six funded CD centers (Table 59) focuses on a unique collection of algorithmic motifs needed by two or more applications. The top collections of motifs (based on application requirements) were initially targeted, resulting in corresponding CD centers. The goal of the CD activity is to integrate the rapidly developing software stack with emerging hardware technologies while developing software components that embody the most common application motifs. These co-designed components will then be integrated into the respective application software environments for testing, use, and requirements feedback. This process must balance application requirements with constraints imposed by the hardware and what is feasible in the software stack to facilitate performant exascale applications.

In addition to the six CD centers, a Proxy Applications project is managed within the CD activity whose mission is to improve the quality of ECP proxy applications, or "apps,") and maximize the benefit received from their use, such as by maintaining and distributing the ECP Proxy App suite. Proxy apps are tools to explore algorithms, data structures/layouts, optimizations, and so on, as well as the associated trade-offs on different architectures. Success is measured by identifiable lessons learned that are translated either directly into parent production application codes or into libraries with a demonstrated performance improvement. An application assessment project, which concluded in FY20, conducted unbiased evaluations of the capability, performance and scaling, and performance portability of ECP application codes. This integration activity assessed application predictive maturity (e.g., as guided by the SNL Predictive Capability Maturity Model) and the integration of exascale technology through regular, independent application assessments.

8.1 Proxy Applications

The purpose of the ECP Proxy Applications project is to improve the quality of proxies created by the ECP and maximize the benefit received from their use. To accomplish this goal, the project maintains a catalog

Table 59: Summary of supported CD L4 centers.

WBS number	Short name	Project short description	KPP-X
2.2.6.01	Proxy Apps	ECP proxy applications	N/A
2.2.6.02	Apps Assessment	ECP applications assessment	N/A
2.2.6.03	CODAR	CD Center for online data analysis and reduction at the exascale	KPP-3
2.2.6.04	CoPA	CD Center for particle applications	KPP-3
2.2.6.05	AMReX	Block-structured AMR CD Center	KPP-3
2.2.6.06	CEED	Center for efficient exascale discretizations	KPP-3
2.2.6.07	ExaGraph	GraphEx CD center	KPP-3
2.2.6.08	ExaLearn	CD Center for exascale ML technologies	KPP-3

of proxy applications[4] and advises vendors and researchers in the selection and use of proxy applications. The project also has a significant effort to understand whether the selected proxies accurately represent the intended characteristics of their parent applications (e.g., memory use, computation, communication).

8.1.1 ProxyApps: The ECP Proxy App Suite

Release 4.0 of the ECP Proxy App suite includes 14 proxies that cover a wide variety of modeling and numeric motifs as well as communication and I/O operations that span the breadth of ECP. The project website also includes pointers to a set of representative problem inputs and parameters to help investigators run proxy apps with inputs that represent production use cases.

The Proxy Apps team is also continuing its efforts to identify and highlight proxy apps for scientific ML workloads. ML proxies are different from traditional proxies in other important ways. One obvious difference stems from the fact that ML workloads often rely on a fairly complex set of third-party dependencies, such as PyTorch or TensorFlow. This is contrary to the usual principle of creating simple-to-build proxies by avoiding dependencies. Because such dependencies are practically unavoidable in the ML space, it is fortunate that tools such as Spack are making it much easier to manage dependencies. Another difference comes from the rapid pace of innovation and discovery in ML. Algorithms and methods that are popular today can be easily discarded tomorrow, and the search for better training methods or model representations is a significant focus of effort in the field. Hence, good proxies must be nimble to adapt to such changes, participate in the innovation process, and help understand how such changes impact science workflows.

To date, five ML proxies have been identified that support hardware and software CD, programming model exploration and innovation, and the development of numerical methods and algorithms. These proxies are also likely to be useful for optimization, benchmarking, and education. More information about these proxies is available on the project website.

8.1.2 ProxyApps: Assessment of Proxy/Parent Similarity

In FY20, cosine similarity was applied to evaluate node and memory behavior and communication (i.e., MPI) patterns for a suite of 10 ECP applications and proxies. This work on applying cosine similarity to understand communication differences between proxy and parent applications was published previously [55].

Cosine Similarity

[4] https://proxyapps.exascaleproject.org/ecp-proxy-apps-suite

Cosine similarity is an ML technique that is heavily used in AI applications, such as automatic document comparison and identification [56–58]. Within the ECP, it is used to quantitatively determine:

- whether a proxy represents its parent with respect to node and memory behavior,
- a minimal set of proxy applications that represent the composite behavior of the application space, and
- a minimal set of proxy applications that represent the behavior of the application space for specific components of the architecture, such as memory, cache, and branching.

This can be done for essentially any system, but the initial FY20 application is for two Intel platforms.

Cosine similarity takes two vectors as input and calculates the angle (cosine) between them by using properties of the dot product:

$$\cos(\theta) = \frac{\mathbf{a} \cdot \mathbf{b}}{|\mathbf{a}||\mathbf{b}|}. \tag{23}$$

The angle characterizes the distance between the two vectors, which can then be used to indicate the similarity of the two vectors. Hardware performance counter data are used to form an application vector or signature. For each system, the entire set of available and reliable performance events is used. An event is deemed reliable by testing and inspecting results. Each application signature for each platform comprises hundreds of performance counter events.

We look at cosine similarity between the the members of our suite by using all available and reliable performance events as a signature, and we categorize these events based on component or function so that we can look at application similarity based on each primary system component. Categorizing events into performance groups provides an understanding of which proxies/parents comprise a representative workload for each system component. For example, we can pick a minimal set of proxy and/or parent applications that represent the cache behavior seen across the entire suite. Through this sort of analysis, we can also identify which applications we should recommend to vendors and other researchers for exploring architectural innovations for cache, memory, virtual memory, branch prediction, and so on.

Cosine similarity can be used to understand performance differences across many system parameters. For example, one could use it to explore performance difference across operating system versions, compilers, compiler optimizations, or application optimizations. We are currently working on comparisons across architectures by creating a full performance vector for each application for each architecture and using that as input to cosine similarity. For example, we would have an ExaMiniMD Broadwell vector and an ExaMiniMD Skylake vector and compute the cosine similarity between them. This would indicate whether the overall behavior on the two systems is different. If it is not, then no feature on the newer architecture changed the performance as compared with the older architecture. If there is performance difference (i.e., little similarity) between the two systems, we could look at the cosine similarity by using application signatures from the various performance groups to identify which components affect the overall performance change.

Methodology

To understand proxy representativeness, we used the suite of proxy/parent pairs listed in Table 60. We also included two proxies, *pennant* and *snap*, without their respective parent applications. Table 60 shows the applications and the problems and input sizes used. We adjusted problem sizes to use around 40–50% of the node memory for each system. All applications were compiled with the Intel 19 compiler by using vectorization flags at optimization level O3.

We executed all applications by using MPI on only 128 ranks. This scale was chosen because it is large enough to observe important communication patterns but not so large that jobs are forced to wait for days in a scheduling queue to execute. To date, we have collected and analyzed data for two systems: Intel Broadwell and Intel Skylake. We are currently working on the infrastructure for collecting data on IBM Power9 and NVIDIA and AMD GPUs.

We used the Lightweight Distributed Metric System (LDMS) monitoring infrastructure from SNL to collect all performance data (e.g., MPI/communication, node, memory, network). We used the PAPI and Aries plug-in samplers to collect respective node and network performance data. In addition to computing cosine similarity, we computed dozens of metrics from the performance counter data to help us understand and validate the cosine similarity results. We also performed a bottleneck analysis by using the Intel Topdown

Table 60: Applications and input problems.

Proxy/parent	Domain	Input line	Input file
ExaMiniMD/	MDs	-il in.snap.Ta.mod –comm-type MPI –kokkos-threads=1	in.snap.Ta.mod
LAMMPS		-i in.snap.Ta.mod	in.snap.Ta.mod
miniQMC/ QMCPack	QMC	-r 0.99 -g '2 2 2' NiO-example.in.xml	N/A NiO-example.in.xml; NiO-fcc-supertwist111-supershift000-S64.h5
sw4lite/ SW4	Seismology	new_gh.in	new_gh.inz
SWFFT/ HACC	Cosmology	15 2048 params4x32 -n	N/A params4x32 -n
pennant	Unstructured mesh hydro	leblancx128.pnt	leblancx128.pnt
snap	Neutron particle transport	inh0004t1a out4a	inh0004t1a

	ExaMiniMD	LAMMPS	MiniQMC	QMCPack	sw4lite	sw4	SWFFT	HACC	pennant	snap
ExaMiniMD	0.00	10.24	84.61	83.55	61.94	64.17	86.71	85.58	75.88	44.50
LAMMPS	10.24	0.00	75.12	73.95	53.63	56.50	79.66	78.51	70.97	34.97
MiniQMC	84.61	75.12	0.00	5.97	42.91	47.75	51.57	51.28	66.16	43.41
QMCPack	83.55	73.95	5.97	0.00	37.71	42.28	45.85	45.52	60.31	40.89
sw4lite	61.94	53.63	42.91	37.71	0.00	6.47	27.99	26.86	30.17	24.55
sw4	64.17	56.50	47.75	42.28	6.47	0.00	23.59	22.42	23.83	29.89
SWFFT	86.71	79.66	51.57	45.85	27.99	23.59	0.00	1.22	18.65	51.79
HACC	85.58	78.51	51.28	45.52	26.86	22.42	1.22	0.00	18.14	50.70
pennant	75.88	70.97	66.16	60.31	30.17	23.83	18.65	18.14	0.00	51.63
snap	44.50	34.97	43.41	40.89	24.55	29.89	51.79	50.70	51.63	0.00

Figure 82: Broadwell cosine similarity in degrees, full vector.

Microarchitecture Analysis (TMA), which we implemented within LDMS. These data are not reported here for brevity, but they will be made available in the future on the ECP Proxy App project website.

Results

We define an application vector or signature as the performance data collected for a specific application on a particular architecture. On each architecture, a full application signature comprises around 400 performance event counts. Figure 82 shows the cosine similarity (COS) results for the Intel Broadwell system using all available and reliable performance event counts (full vector) in the application signature. In this figure and all of the following COS figures, colors are interpreted as follows.

- Green indicates good similarity. As the similarity decreases (i.e., the angle between applications becomes larger), the shade of green becomes lighter.
- Yellow indicates some similarity but also some difference. Whether this is acceptable is based on the intended use of either the proxy or the parent application.
- Orange indicates poor similarity. The two applications are more different than they are similar. The shade of orange darkens as the similarity decreases (i.e., difference increases).
- Red indicates almost no similarity. These applications exhibit very different behavior on this architecture or for this component.

The scale here was somewhat arbitrarily assigned. In researching methods for defining such a scale, no rigorous quantitative methods were found. Validating and improving this assignment scheme is a focus of ongoing work.

	ExaMiniMD	LAMMPS	MiniQMC	QMCPack	sw4lite	sw4	SWFFT	HACC	pennant	snap
ExaMiniMD	0.00	8.97	81.95	68.83	38.66	39.55	28.51	37.76	43.58	22.20
LAMMPS	8.97	0.00	81.38	68.47	38.60	39.33	29.50	38.49	42.40	20.45
MiniQMC	81.95	81.38	0.00	16.35	47.28	47.63	58.78	49.85	46.58	65.55
QMCPack	68.83	68.47	16.35	0.00	36.05	36.40	46.19	37.82	36.33	53.30
sw4lite	38.66	38.60	47.28	36.05	0.00	4.05	20.56	17.09	12.89	21.69
sw4	39.55	39.33	47.63	36.40	4.05	0.00	19.82	15.87	11.91	22.79
SWFFT	28.51	29.50	58.78	46.19	20.56	19.82	0.00	10.33	24.49	21.44
HACC	37.76	38.49	49.85	37.82	17.09	15.87	10.33	0.00	19.92	26.67
pennant	43.58	42.40	46.58	36.33	12.89	11.91	24.49	19.92	0.00	25.00
snap	22.20	20.45	65.55	53.30	21.69	22.79	21.44	26.67	25.00	0.00

Figure 83: Skylake cosine similarity in degrees, full vector.

In Fig. 82, the diagonal reflects the self-similarity of each application (e.g., the similarity between ExaMiniMD and ExaMiniMD) in that the angle between them is zero. The next observation is that all of the proxy/parent pairs have good similarity—they are all some shade of green. SWFTT and HACC have the highest similarity. ExaMiniMD and LAMMPS are similar to each other but are very different compared with any other application. This is also the case for miniQMC and QMCPack. SW4 and sw4lite have some similarity with SWFFT, HACC, and snap (yellow entries). We refer to this as *redundancy* because if we choose one of these applications and discard the other two, we would still represent all of the unique behavior of the entire suite. Pennant and snap are dissimilar from each other, as expected.

Looking at the COS data from the Skylake system in Fig. 83, we see that the similarity between the proxy and parent pairs remains good (green), although the magnitude of the angle between the application signatures changes. For instance, the angle between ExaMiniMD and LAMMPS is smaller (more similar), that between miniQMC and QMCPack and SWFFT and HACC is significantly larger, and that between sw4lite and SW4 is essentially the same. Also, sw4lite, SW4, SWFFT, HACC, pennant, and snap are less divergent from ExaMiniMD and LAMMPS than they were on the Broadwell architecture; this area in the matrix is characterized more by yellow and orange blocks than by red blocks. MiniQMC remains a distinct outlier compared with the other applications, but here QMCPack has more similarity than before—although it is still more different than similar—to sw4lite, SW4, HACC, and pennant. We also can see that on Skylake vs. Broadwell, snap is now much more similar to SWFTT, HACC, and pennant, transitioning from red to green and yellow. These changes in similarity between application signatures (i.e., behavior) are all in response to changing architectural constraints when going from Broadwell to Skylake.

Next Steps

Release 4.0 of the ECP Proxy App suite provides a high-quality set of proxy apps to represent ECP workloads. The proxy apps team is continuing to expand and curate the catalog of proxies to ensure that future CD efforts will have a rich set of proxies to draw from. They have also demonstrated the use of cosine similarity to quantitatively compare proxies and parents to help answer the question of how well proxy represent the true nature of their parents.

8.2 CODAR

The growing disparity between simulation speeds and I/O rates makes it increasingly infeasible for high-performance applications to save all results for offline analysis. By 2024, computers are expected to compute at 10^{18} operations per second but write to disk at only 10^{12} B/s: a compute-to-output ratio 200 times worse than on the first petascale systems. In this new world, applications must increasingly perform online data analysis and reduction—tasks that introduce algorithmic, implementation, and programming model challenges that are unfamiliar to many scientists and that have significant implications for the design of various elements of exascale systems.

CODAR, a CD center focused on online data analysis and reduction at the exascale, addresses this issue. Working closely with ECP applications, CODAR is undertaking a focused process that targets common data

analysis and reduction methods (e.g., anomaly detection, feature tracking, and compression) and methods specific to particular data types and domains (e.g., particle and structured finite-element methods). The team engages directly with providers of the ECP system software, programming models, data analysis, and reduction algorithms, as well as applications, to understand and guide trade-offs in the development of applications and software frameworks given constraints relating to AD costs, application fidelity, performance portability, scalability, and power efficiency.

The goals of CODAR are to (1) reduce the development risk for the ECP application teams by investigating crucial performance trade-offs related to the treatment of scientific results created by scientific models, (2) produce high-performance implementations of data analysis and reduction methods, (3) enable the easy and efficient integration of those methods with applications, and (4) contribute to the CD of effective exascale applications and software. To accomplish these goals, the team produces infrastructure for online data analysis and reduction; provides valuable abstractions for implementing and customizing data analysis and reduction methods; imports, integrates, and develops essential libraries implemented by using these abstractions; incorporates the libraries into scientific applications and quantifies accuracy and performance; releases software artifacts; constructs application-oriented case studies; documents success stories and the process applied to obtain them; and reports on CD trade-off investigations.

8.2.1 CODAR: Algorithms and Software Objectives

Software is partitioned into two pieces: (1) infrastructure for orchestrating online data analysis and reduction workflows and running, collating, and analyzing CD experiments and (2) the development of high-performance implementations of online data analysis and reduction methods that are inspired by unique data access requirements and unfulfilled application needs.

Infrastructure: Cheetah, Savanna, Chimbuko

Cheetah enables CD experiments for improving performance and functionality of online analytics and reducing exascale science. Cheetah is the interface for defining the CD experiments. Specifically, the Cheetah component works to define a set of conventions and reusable scripts for conducting parameter sweep experiments on different science application scenarios that are necessary for CD studies.

Savanna is the runtime that can launch and manage the individual CD experiments on current and future extreme-scale systems. The role of the Savanna component is to isolate the definition of the set of online workflows from the inaction and enforcement of those workflows and their policies. The transitions between different scheduler systems, parameters set on command line vs. environment variables, and so on are managed through the uniform service interface.

Chimbuko works with data provided by TAU to identify performance anomalies and save relevant information in a window around the anomaly for further analysis. In particular, Chimbuko is capable of capturing, analyzing and visualizing performance metrics for complex scientific workflows that include online data analysis and reduction and relating these metrics to the context of their execution on extreme-scale machines. This tool enables empirical studies of workflow performance during the initial application development phase or when porting to a new computational environment.

Online Methods: Z-Checker, Feature Tracking Kit, MGARD

Z-checker is designed to assess lossy compression comprehensively offline and online in parallel for scientific datasets. Because of the vast volume of data being produced by today's scientific simulations and experiments, lossy data compression that allows user-controlled loss of accuracy during the compression is a relevant solution for significantly reducing the data size. However, lossy compressor developers and users do not have a tool to explore the features of scientific datasets and understand the data alteration after compression in a systematic and reliable way. Z-checker is an open-source community tool developed to fill this gap.

The Feature Tracking Kit (FTK) is designed to robustly extract, track, and visualize features as they evolve in large-scale simulations. This kit fills a gap in the production visualization tools being developed in the ECP software technologies area to associate detected features in adjacent time steps. This work was inspired by the WDMApp project that must detect and track blobs and streamers in 5D gyrokinetic tokamak simulations, such as XGC.

MGARD is a technique for the multigrid adaptive reduction of data. Special attention is given to the

case of tensor product grids in which the team's approach permits the use of nonuniformly spaced grids in each direction, which can be problematic for many types of data reduction methods. one important feature of CODAR's approach is the provision of guaranteed, computable bounds on the loss incurred by the data reduction and the preservation of quantities of interest and statistics.

8.2.2 CODAR: Performance Objectives

CODAR's focus on the online data analysis and reduction motif means that a CODAR application typically comprises one or more application components, analysis components, and reduction components that all run simultaneously on the same or different nodes. Performance challenges can occur within any component and/or as a result of communication among components. Decisions that must be made related to performance include placing components across the cores within the nodes (heterogenous node usage) or running the components in separate nodes (homogeneous node usage); mapping components to use CPU, GPU, and other resources; choosing the types of memory to use within the nodes and across nodes for the different components (e.g., NVRAM); and selecting online data analysis and reduction components (e.g., the lossy compression routine to apply to ensure user-controlled accuracy is maintained) and the frequency with which to apply them. The overarching approach to meeting these challenges is as follows.

1. **Overall architecture**: The application composition Savanna architecture allows us to configure and experiment with alternative mechanisms and configurations on different platforms—including component placements and memory type—and the performance data collection and analysis system (e.g., TAU and Chimbuko) that provides performance feedback.

2. **Application components**: The performance of individual application components is out of scope for CODAR. The team can document when individual components perform badly and communicate that information to application developers, but addressing such performance problems is not in the CODAR charter.

3. **Data analysis and reduction components**: Whenever possible, existing high-performance implementations of methods (e.g., SZ and ZFP for compression) are integrated. The team can document when these components perform badly and communicate that information to the software technology team, but addressing such performance problems is not in the CODAR charter. For the online data analysis and reduction components that the team develops in its tool kit, the work to optimize these components (e.g., MGARD) on the specific machines is undertaken.

4. **Inter-component communication**: The team leverage the ADIOS communication library for communication between components. ADIOS uses specialized mechanisms to enable high-speed data transfer between parallel components, such as shared memory mechanisms when processes are running on the same node.

The Cheetah experiment management system allows an investigator to perform, collate, and analyze experiments to identify performance bottlenecks, which can then either be addressed directly for internally developed software or relayed to appropriate AD and ST teams.

The team's metrics of success are application-dependent and include the ability to use online data analysis and reduction methods to write information of sufficient quality to the file system at a rate that leads to the application idling while waiting for I/O to complete. Additionally, adding the online data analysis and reduction methods must fit within resource restrictions (e.g., cores and impact on application performance) set by the AD teams. If the team can simultaneously satisfy all these resource restrictions, then it will be able to provide the correct information at the correct time and place to accelerate scientific discovery.

8.2.3 CODAR: Co-Design Engagements and Integration Points

CODAR's engagement with AD and ST teams consists of three activities: infrastructure integration and CD studies for online data analysis and reduction; the development of application-inspired data analysis and reduction methods; and investigations of the runtime support required for task parallel computation.

Application integration activities have so far focused on the WDMApp (§ 4.5), NWChemEx (§ 3.2), and CANDLE (§ 6.2) projects.

Table 61: CODAR KPP-3 goals and metrics.

Passing value	Stretch value	Tentative present value
6	12	1

Application-Inspired Methods

In addition to the infrastructure integration and CD studies, the team also engages with application teams to identify their needs for novel data analysis and reduction methods. These are focused activities between a small number of team members from CODAR and the application to develop methods that either fill gaps in methods available from existing software technologies, cover different types of data (e.g., structured, unstructured, or particle data), or exhibit unique data access patterns (e.g., multigrid or hierarchical access).

The application-inspired methods pipeline includes methods that are being developed into a production capability, including FTK, which was inspired by WDMApp; the MGARD multigrid data reduction method, which was inspired by combustion; and the performance anomaly detection methods, which were inspired by NWChemEx included in Chimbuko. Additionally, activities in the exploration phase include developing improved statistical metrics for compression quality analysis, which were inspired by ExaSky and WDMApp and will be include in Z-checker, and numerical optimization-based compression, which was inspired by the EXAALT project (§ 3.4). Two activities were stopped after prototyping: the functional data analysis topic, which was inspired by WDMApp, and the hierarchical data analysis topic, which was inspired by ExaSky (§ 5.2). Further development of the hierarchical data analysis methods for computing halo centers could be undertaken by the ExaSky project and implemented in its CosmoTools package.

As part of the online data analysis and reduction method development, the team will provide benchmarks and other artifacts to software technology projects. This community service includes the Scientific Data Reductions Benchmarks [59] developed as part of the Z-checker activity.

Runtime Support for Task-Parallel Computation

CODAR worked to design and prototype a new MPI mechanism, `MPI_Comm_launch`, which allows a child MPI application to be launched inside the resources originally held by processes of a parent MPI application. Two important aspects of `MPI_Comm_launch` are that it pauses the calling process and runs the child processes on the parent's CPU cores but in an isolated manner with respect to memory. CODAR scientists used this prototype to experiment with the use of the new construct and demonstrate its value for applications that must run multiple components simultaneously in such a way that terminating one component does not cause a computation to fail. This exercise thus constitutes a valuable CD exercise for exascale system software.

Work with CANDLE on DeepDriveMD

The CANDLE DeepDriveMD code [60] implements a sampling approach to study potential energy surfaces of biological macromolecules. An ensemble of simulations are started, and the results are passed to a neural network that discerns the optimal parts of phase space to explore next. As more results are obtained, the neural network is periodically retrained. The current DeepDriveMD code couples the simulations, neural network training, and simulation scoring tasks via the file system, which leads to significant inefficiencies. The CODAR team is working with CANDLE to adapt the DeepDriveMD code to couple the various components via ADIOS communications. Early results are promising; when running 96 simulations concurrently, order-of-magnitude performance improvements are achieved relative to an implementation that retrains the neural network after each batch of simulations. Current work is looking to understand scalability to much larger sizes and to understand trade-offs between the staleness of neural network predictions and the quality of the results obtained.

8.2.4 CODAR: Progress on Early and Pre-Exascale Hardware

Performance on Summit

We conducted numerous CD studies on Summit, some of which involved test codes and complete

Table 62: CODAR code base.

Package name	LOC	Target exascale challenge problems	Computational motifs
Cheetah	∼2,000	Launch and manage the execution of a set of CD experiments on supercomputer platforms, as specified by CD specification file.	Performance portability
Chimbuko	N/A	Performance metrics collection of scientific workflows through performance tools and correlating performance information collected for different workflow components.	Performance portability
FTK	10,000	Feature tracking and extraction (to be used for WDMApp; potential use for climate).	Data analysis
MGARD	N/A	Data reduction while preserving quantities of interest and statistics; already used to compress data for WDMApp; to be used for combustion.	Data reduction
Savanna	N/A	Run a multicomponent application (e.g., 1+ application modules, analysis modules, reduction models) on a supercomputer platform.	Performance portability
Z-checker	40,000	Data reduction error analysis (already used to assess compression errors for ExaSky, ExaFEL, EXAALT, GAMESS; will also be used for NWChemEx, QMCPACK).	Data reduction

applications. The work with DeepDriveMD was already mentioned, as well as online data analysis that enables large speedups on a code that won a 2020 Gordon Bell award. Publications detail investigations of compression quality trade-offs [61], work with Cheetah [62], and work with EFFIS [63]. Work with Chimbuko that applied NWChem on Summit won the best paper award at the 2020 In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization conference [64].

Performance tuning of FTK continues with a focus on GPU performance. Strong scalability on CPU behaves as expected, but because GPU kernel functions are 100× faster than their CPU counterparts, the non-GPU code, including trajectory reconstruction, becomes the bottleneck. Work continues to address this bottleneck for high scalability with GPUs on Summit.

Work on MGARD also continues with a focus on using GPUs to accelerate the "data refactoring" operations associated with MGARD compression. The MGARD compression pipeline comprises data refactoring, quantization and lossless compression steps. Running solely on a CPU, these steps have been shown to run at roughly 80 MB/s, 17 GB/s, and 100 MB/s, respectively. On a V100 GPU on Summit, the first two run at 10 GB/s and 250 GB/s. Current work will use a GPU-enabled lossless compressor (`nvcomp`) to enable at least 10 GB/s throughput.

Next Steps

Our plans for the next year will focus on further developments of CODAR software and continued engagements with applications. Our milestones are as follows.

- December: CD reports, Chimbuko release, Z-checker release.
- March: FTK release.
- June: Cheetah/Savanna release.
- September: MGARD release.

8.3 CoPA

The ECP Co-Design Center for Particle Applications (CoPA) is addressing challenges for particle-based applications to run on upcoming exascale computing architectures. This scope is partitioned into four "sub-motifs": short-range particle-particle interactions (e.g., those which often dominate MD and SPH methods), long-range particle-particle (e.g., electrostatic and gravitational) interactions, PIC methods, and linear-scaling electronic structure and quantum molecular dynamics (QMD) algorithms. The crosscutting co-designed technologies fall into two types: proxy apps (e.g., ExaMiniMD, ExaSP2, CabanaMD, and CabanaPIC) and libraries. Libraries are modular instantiations that multiple applications can use or be built upon and include the Cabana particle library, PROGRESS/BML matrix library packages for QMD, and the SWFFT and fftMPI parallel FFT libraries. Success is measured by the adoption by existing or newly developed production codes with both productivity and performance benefits.

8.3.1 CoPA: Algorithms and Software Objectives

PROGRESS/BML

The QMD capabilities are included in a computational framework that aims to foster developments in computational chemistry and electronic structure packages. This framework consists of two libraries, PROGRESS and BML, in which the computational developments are performed. The Parallel, Rapid $O(N)$, and Graph-based Recursive Electronic Structure Solvers (PROGRESS) library is a FORTRAN library that can be used for general purpose quantum chemistry calculations. The basic matrix library (BML) package provides a common application programming interface (API) for linear algebra and matrix functions in C and Fortran, targeting those operations and use cases that commonly arise in quantum chemistry codes. Linear-scaling electronic structure applications rely on sparse linear algebra and require hand-tuned implementations of sparse matrix operations. Since existing libraries for sparse linear algebra—such as MKL, ACL, and NVIDIA's CUDA sparse Matrix Library (cuSPARSE)—are limited and lacking in performance, BML was developed to addresses this challenge by offering high-level abstractions for matrix operations independent of the underlying data structures and algorithms. The BML API is matrix format independent. Dense, ELLPACK-R sparse, and CSR sparse matrix data types are available, each with different implementations, and a new blocked format, ELLBLOCK, is currently under development. PROGRESS relies entirely on BML for algebraic operations, so although quantum chemistry and electronic structure algorithms and calculations are outlined in PROGRESS, the underlying mathematical manipulations are all performed in BML. At the application level, multiple codes will benefit from the new solvers (i.e., techniques) developed within PROGRESS. The current focus is on low-level BML library implementations on accelerated architectures (e.g., GPUs).

Cabana

The CoPA particle tool kit Cabana is a collection of software packages that will allow scientific software developers targeting exascale machines to develop scalable and portable particle-based algorithms and applications. This tool kit provides an open-source implementation of a variety of basic particle-based algorithms and data structures applicable to a wide range of application types, such as PIC and its derivatives, MD, and SPH codes. Cabana is written in C++ and is usable by application codes written in C++, C, and Fortran.

Cabana provides native particle data structures, parallel programming APIs, and algorithm implementations by using those data structures. These algorithms will span the space of particle operations needed to support each relevant application type. This includes intranode (i.e., local and threaded) operations on particles and internode (i.e., communication between nodes) operations to form a hybrid parallel capability. The initial set of algorithms sort particle build lists of neighboring particles, exchange particles distributed across a set of nodes (i.e., a halo exchange), interpolate between a set of particles and an underlying mesh or vice versa, and provide a variety of long-range solvers. Efforts are currently underway to add functionality, including load balancing. From the perspective of users, these data structures and algorithms can then be integrated into an application as building blocks for a wide variety of particle-based physics algorithms.

Cabana algorithms are built on parallel loop constructs, which ensure code-it-once performance portability on pre-exascale and anticipated exascale platforms. The tool kit will be interoperable (e.g., easily permits

linking and allows for simultaneous use) with other ECP scientific computing libraries that the user may leverage for other needed services not provided by or in the scope of the Cabana tool kit. Using Cabana with other ECP software technologies should facilitate the composition of scalable particle-based application codes on these new architectures.

ExaMiniMD/CabanaMD

ExaMiniMD/CabanaMD are extensible MD proxy apps with a modular design to enable experimentation with different interatomic potentials, time integrators, neighbor/cell list algorithms, and diagnostic computations, including new parallel algorithms at either the internode (MPI) or intranode (primarily Kokkos) level. The parent LAMMPS MD code, used in the EXAALT ECP project (§ 3.4), has a similar modular structure and MPI-level parallelism based on a spatial decomposition of the simulation domain. To accurately model tungsten-based fusion materials, EXAALT requires a performance-portable implementation of the spectral neighbor analysis potential (SNAP). Although ExaMiniMD was the initial MD proxy app, the newer CabanaMD provides full integration with the Cabana particle tool kit. Both provide useful platforms for development and a compact expression of use cases that can be used for vendor engagement and experimentation. Code developed in ExaMiniMD was ported directly into the parent applications code (LAMMPS) and also into the Cabana library where it is tested with CabanaMD (e.g., neighbor list construction algorithms).

CabanaPIC

CabanaPIC is an extensible PIC proxy app built upon the Cabana library and designed to enable experimentation with computational algorithms for particle-grid interactions and long-range Coulomb forces. Additionally, CabanaPIC is a scalable computational test bed for the implementation of new solution algorithms for nonlinear plasma physics.

SWFFT/fftMPI

Electrostatic and gravitational interactions arise in many particle-based applications. Since a direct N^2 calculation is neither practical nor necessary, long-range (e.g., Poisson) solvers are required. Although several methods, including particle-particle particle-mesh (PPPM, or P3M) and particle-mesh Ewald (PME), use FFTs, others, such as the fast multipole method (FMM), do not. As an initial step toward a general long-range solver library that enables the user to readily switch between and evaluate different algorithms for their particular use cases, the team began by extracting custom parallel FFT implementations from applications partners and evaluating their use in other applications. SWFFT is a 3D complex-to-complex FFT library based on the distributed FFT originally developed for the cosmology code HACC. It was developed to run very large 3D complex-to-complex FFTs (e.g., of order 10^{12} with a relatively low memory overhead, excellent scalability, and good performance). Using SWFFT makes it easy for other applications to also use advanced Poisson solver techniques tailored to their particular needs. Similarly, the fftMPI library was created by extracting the FFT kernels from the LAMMPS code and repackaging them as a stand-alone library with supporting test harnesses and documentation.

8.3.2 CoPA: Performance Objectives

PROGRESS/BML

The challenge is to develop libraries that are portable across the range of available multicore, manycore, and hybrid accelerated architectures while still supporting a wide range of compilers (e.g., GNU, Intel, IBM, PGI, Clang, ...), low-level linear algebra implementations (e.g., BLAS, MKL, ESSL) and accelerator programming options (e.g., MAGMA, OpenMP off-load, cuSPARSE/cuBLAS, SLATE). This is exacerbated by the varying maturity of compliers and the underlying hardware (e.g., for OpenMP off-load). The general strategy has been to keep up with the current state of maturity and invest in similar technologies that can be readily converted rather than committing to any single specific programming model or math library.

Cabana

One fundamental challenge of developing a particle library is the programming of loops over complex particle data layouts in a flexible and performance portable fashion, recognizing the fact that most particle-

pushing algorithms are memory-bound. The aim is to allow users to program with a pool of kernels to compose different algorithms or the same algorithm with different flavors. The implementation strategy is to take advantage of modern C++ functional and metaprogramming techniques to create a flexible design of the software library that exhibits performance on a variety of architectures. In the team's implementation, the data structures and particle loop composition build on the Kokkos library to enable performance portability across different devices. This strategy allows us to explore and develop different particle data layouts, threading models, and code that might vectorize more easily on modern architectures and allows us to make these developments accessible to library users. As a result, users can compose their application with the Cabana library and compile it with cross-platform support, creating a single implementation that can both execute and perform on expected exascale platforms. Simple kernels and representative mini-applications (e.g., for MD and PIC) built on Cabana are used to track performance and identify and address any gaps that arise.

ExaMiniMD/CabanaMD

ExaMiniMD and CabanaMD are being used to experiment with new hardware capabilities, such as NVIDIA's shared memory (SHMEM). MD communication kernels are being reimplemented with an SHMEM option to test performance against conventional MPI. The development of new algorithmic options for MD that have superior performance on highly threaded hardware that does not support fast atomics (e.g., Intel KNLs) is also being tested. Although this work is driven by LAMMPS, it mostly involves internal Kokkos code and will thus benefit other Kokkos-based applications within the ECP.

CabanaPIC

CabanaPIC is being used to explore the implementation of new solution algorithms for nonlinear plasma physics. These equations are inherently stiff, and new solution algorithms are required to attain the timescales required by the WDMApp application.

SWFFT/fftMPI

SWFFT and fftMPI are designed for large distributed 3D FFTs. Slab-decomposed parallel FFTs are not scalable to very large MPI ranks but require only one "all-to-all" communication. On the other hand, data partitioning across a 2D subgrid ("pencil" decomposition) are scalable but require multiple "all-to-all" communication steps. Additions to fftMPI include: (1) an option to perform data rearrangement as a one-step pencil-to-pencil transpose (LAMMPS style) vs. a two-step pencil-to-block, block-to-pencil (HACC style) operation and (2) a method to auto-tune for optimal performance on a particular FFT size and processor count by scanning a set of available algorithmic and parameter choices. These will be back-ported to LAMMPS once further CoPA benchmarking is complete. Within CoPA, SWFFT and fftMPI are used by the Cabana-based long-range solver effort. Because they are currently CPU-only, new options are being examined, namely heterogeneous node support using both CPU and GPU and threading 1D FFTs for GPUs and multicore CPUs.

8.3.3 CoPA: Co-Design Engagements and Integration Points

Applications

The closest engagements are with four applications with whom personnel are shared. These AD projects and relevant codes are: ExaSky (HACC) (§ 5.2), EXAALT (LAMMPS and LATTE) (§ 3.4), WDMAPP (XGC) (§ 4.5), and ExaAM (ExaMPM) (§ 3.5). The PROGRESS/BML libraries were integrated into LATTE, and the team works closely with EXAALT members to co-design new capabilities and modifications, as needed. As discussed previously, libraries and proxies were extracted from HACC, LAMMPS, and LATTE (ExaSP2), and the PIC-related algorithm kernels developed in the Cabana library are designed to be usable by XGC, WarpX, and ExaMPM. The interactions with these ECP applications have been ongoing since the library design phase and continue through the current development and eventual deployment. ExaMPM is a new Material Point Method application code that is being co-designed in parallel with Cabana. WarpX has motivated the development of a mini-PIC app based on Cabana. XGC is a legacy code that is being used to guide the Fortran interoperability design strategy; the successful performance portability already

obtained by using Cabana for the electron push bottleneck of XGC has motivated their team to begin using the XGC/Cabana version for production runs on Summit. AD interactions span inputs on the design of data structures and programming interfaces to algorithmic content and integration with mini-apps and kernels representative of the applications. The EXAALT team has provided algorithmic and interface input to Cabana development for MD applications, provided performance numbers for relevant computational kernels, and is in the process of investigating Cabana performance by using the ExaMiniMD proxy. The availability of enhanced FFTs in LAMMPS will be useful for the EXAALT project in conjunction with the LATTE DFTB code, which has its own Coulombic solver, and materials modeling for charged systems (UO_2 fission fuel).

In a collaboration across CD centers, the potential integration of the Cabana and AMReX libraries for block-structured AMR with particles has been discussed. Although a direct coupling has not been achieved, the MFIX-Exa (§ 4.4) project, which was built on AMReX, has adopted Cabana algorithms to construct neighbor lists on GPUs. In another collaboration with AMReX, the team has integrated SWFFT to solve the discrete Poisson equation on a single level of refinement and demonstrated that Nyx, the ExaSky cosmology code built on AMReX, can build and run by using the SWFFT solver. The fftMPI library is being used by the WarpX (§ 4.6) application (Rob Ryne) for an initial stage in their modeling workflow.

ExaMiniMD and SWFFT are part of the ECP Proxy Applications suite and were used by the ECP Proxy Applications project (§ 8.1) in their first quantitative performance assessment by comparing their computation and memory behavior with their parent applications: LAMMPS and HACC, respectively. ExaSP2 is also in the ECP Proxy Applications catalog, and in response to multiple requests, the team has also contributed a lightweight PIC proxy, CabanaPIC, as an early test of the Cabana library.

Benchmark/Bake-Off Problems

The successful engagement with XGC has driven the need to incorporate benchmark and bake-off problems into the CoPA CD process. In this case, a benchmark/bake-off problem is an application-specific problem that isolates the section in the application code (subroutine/algorithm) that is the focus of the CD engagement. In this way, an apples-to-apples comparison can be made before and after code refactoring or adoption.

Software Technologies

The MAGMA library was successfully integrated into BML for node-level dense matrix operations on GPUs. In the move to a more distributed approach, the SLATE library will be explored for dense distributed matrices. The team is aware of the performance evaluation of the current version of SLATE for EXAALT/LATTE and will learn from the experience. Performance of BML was assessed by using the Roofline/Advisor performance tools. This evaluation will be revisited periodically as more capability is added. In the move to the matrix operations to run on accelerated architectures, there will be more interaction with the OpenMP project for the off-load capability. The team will also explore the new capabilities offered by the MPI project in its distributed approach.

The Kokkos library is a core dependency of ExaMiniMD and the Cabana library. Kokkos provides the basic Cabana memory allocation utilities, some data structures, portable performant parallel loop constructs, and data layout options for different hardware types that Cabana uses to represent particle data and implement algorithms and operations on particle data. Cabana developers have engaged and continue to engage extensively with the Kokkos development team, including the PI, with Cabana developers, providing feedback and requesting Kokkos changes, as well as Kokkos developers providing code changes for Cabana. Additionally, other ECP application and software teams using Kokkos are being engaged to develop a broader knowledge base and set of best practices for using the library.

fftMPI is currently being analyzed and benchmarked by the SLATE software project, and discussions with both of the new ECP FFT projects (HEFFTE and FFTX) have also occurred. They will use fftMPI as a baseline in their benchmarking and new algorithm development.

Vendors

The team members have participated in Intel, AMD, and Cray deep dives/hackathons, using the BML library and ExaSP2 and ExaMiniMD proxy apps with their simulators. The team participated in the National Energy Research Scientific Computing Center (NERSC) GPU hackathon using the Cabana library. Team members have participated in the NVIDIA Summit on Summit/Sierra (SoSS) meetings and biweekly conference calls per GPU issues on Summit and Sierra. The team held videoconferences and in-person

Table 63: CoPA KPP-3 goals and metrics.

Passing value	Stretch value	Tentative present value
7	14	8

Table 64: CoPA code base.

Package name	LOC	Target exascale challenge problems	Computational motifs
PROGRESS/BML	70,900	EXAALT (LATTE)	Sparse/dense linear algebra
ExaSP2	3400	EXAALT (LATTE)	Sparse/dense linear algebra
Cabana	10,000	ExaAM (ExaMPM), WDMAPP (XGC)	Particles, Long-Range Solvers
ExaMiniMD	6200	EXAALT (LAMMPS)	Particles
CabanaMD	6500	EXAALT (LAMMPS)	Particles
CabanaPIC	3000	WDMAPP, WarpX	Particles, Particle-Grid Interactions
SWFFT	3600	ExaSky (HACC)	FFT
fftMPI	6400	WarpX	FFT

meetings to discuss potential use cases for new PathForward technologies. Team members have participated in the Aurora workshop and online, as well as the Frontier workshop, in preparation for porting libraries and proxy applications to the exascale architectures. Although the team anticipates that the Kokkos library will be transferable, the choice of AMD for the Frontier GPU will require a comparable deep engagement with AMD comparable with SoSS with NVIDIA.

8.3.4 CoPA: Progress on Early and Pre-Exascale Hardware

Performance on Summit

PROGRESS/BML

New capabilities that were added to PROGRESS/BML include the following.

1. New sparse formats were added for use by application partners and performance. CSR and ELLBLOCK formats and methods were added to BML. They are now available on the CPU by using OpenMP with GPU methods to be added in FY21. ELLBLOCK provides for better performance by allowing the Hamiltonian matrix to be split into multiple variable size blocks.

2. A parameterized model Hamiltonian benchmark capability was added that allows for the creation of model Hamiltonian Matrices for metals, semiconductors, and biosystems/soft matter for benchmarking BML matrix formats for matrices of increasing size. The required parameters are four coupling parameters, four on-site energies, a decaying exponential parameter, and a randomization factor.

3. Performance for dense eigensolver calculations on GPU was improved by using the faster cuSOLVER (NVIDIA) instead of MAGMA's diagonalization. In Fig. 84, a comparison of cuSOLVER for diagonalization and dense SP2 using MAGMA on a Summit node for Hamiltonian matrices of increasing size is shown along with GPU performance and utilization. The optimal algorithm depends on the architecture and matrix size.

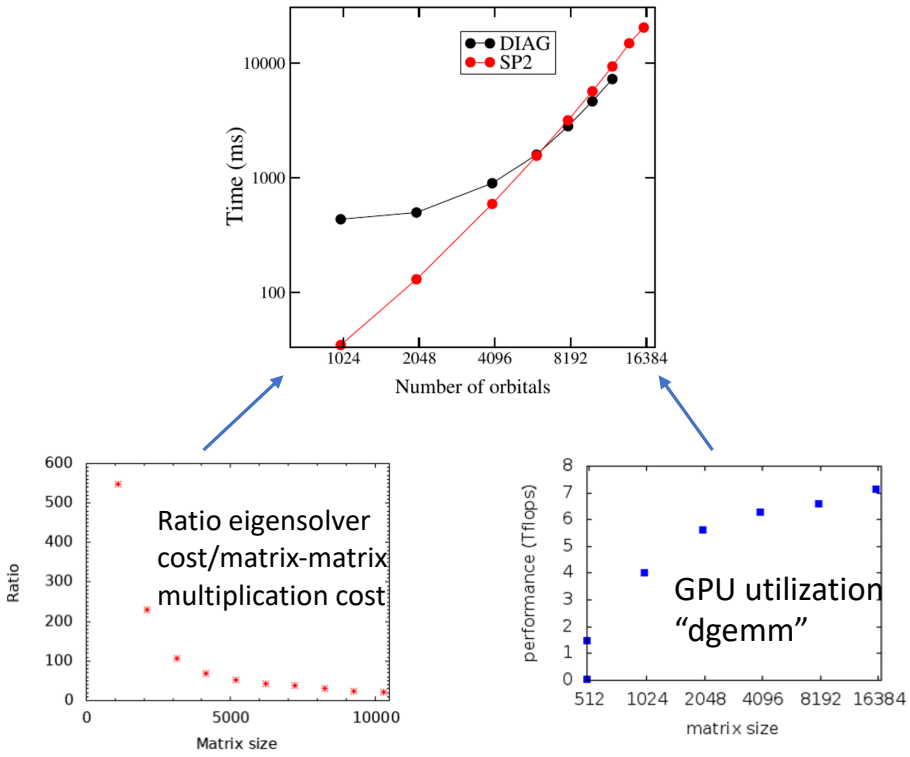


Figure 84: GPU performance and utilization for dense eigensolver vs. dense SP2 using MAGMA on a Summit node. The dense eigensolver uses NVIDIA's cuSOLVER. SP2 uses MAGMA for matrix-matrix multiplications.

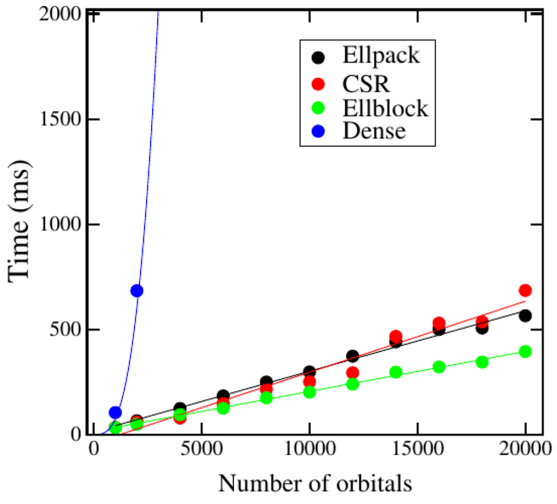


Figure 85: Performance of sparse formats for SP2 on Summit Power9 CPU. CSR and ELLPACK have similar performance for biosystems of increasing size, whereas ELLBLOCK shows better performance for larger matrix sizes.

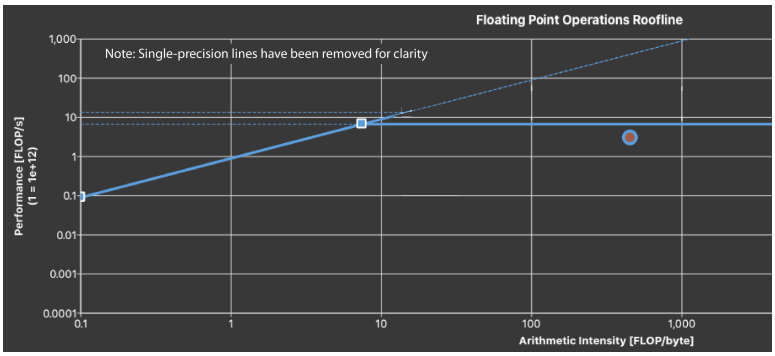


Figure 86: Roofline analysis of the XGC electron push kernel on Summit.

Table 65: Speedup using PROGRESS/BML for biomolecular simulation performance on Summit node.

SARS-CoV-2 system	Number atoms	Number orbitals	No PBML time (s)	PBML time (s)	PBML speedup
NPS3 Macrodomain	2557	6393	57	6	9.5
RBD	3018	7711	99	9	11
RBD + Water Layer	6038	13 743	553	36	15

4. Efficient $O(N)$ sparse solver performance on Summit a Power9 CPU is shown in Fig. 85. Based on bio-system Hamiltonians of increasing size, improved CSR and ELLPACK show similar performance, and ELLBLOCK performs best.

5. The BML GPU off-load strategy is designed for exascale. An OpenMP target with native kernels is being used for sparse formats for performance. Third- party libraries are being used for dense formats (e.g., MAGMA, cuSOLVER, cuBLAS). Options when building allow for either CPU or GPU, where data are local.

6. A distributed memory strategy was designed to leverage serial BML capabilities as follows. Implementation and testing are planned for FY21. Two approaches are being considered. The first, distributed linear algebra, will use 2D square submatrices, the SUMMA algorithm for matrix-matrix multiplications, and will rely on SLATE or ELPA for dense eigensolver calculations. At the submatrix level, matrix operations will be performed by using BML. The second approach, density matrix divide and conquer, will divide the system into overlapping submatrices by using graph partitioning or another method and combine for the final result.

The EXAALT interaction has resulted in PROGRESS/BML adaptation and optimization following bottleneck eliminations and the addition of new features. Exploration of NVIDIA's Multi-Process Service (MPS) for small matrices (<1000) on GPU resulted in maximum oversubscription of seven processes per GPU and 1.8× improvement in throughput.

A PROGRESS/BML COVID milestone effort developed a quantum mechanical/molecular mechanical (QM/MM) capability for long-duration biomolecular simulations. NAMD and LATTE/PROGRESS/BML were coupled for this effort. Simulations of SARS-CoV-2 proteins were performed in support of therapeutic strategies. Using CoPA's PROGRESS/BML libraries with LATTE resulted in a 10× speedup on a single Summit node (CPU/GPU) compared with LATTE without PROGRESS/BML (Table 65).

Cabana

Algorithmic development and performance optimization of the Cabana particle library has primarily focused on the Summit computer system at ORNL. Algorithm development has focused on the following areas: `buffered_parallel_for` to allow for massive particle counts while mitigating CUDA UVM page faults, new

tree-based neighbor list capabilities based on ECP ST library ArborX, adding native periodic boundary support for MD-like neighbor lists and communication, long-range solvers, and improved particle-grid support.

Performance improvements in Cabana have focused on the following areas: MPI pack/unpack performance improvements with NVIDIA with impacts on migration and halo exchange and promoting coalescing with Array of Structures of Arrays (AoSoA), assessment of communication patterns within partner applications, propagating binning/sorting enhancements to the remainder of library and potentially `Kokkos::BinSort`, and deploying neighbor list and sorting/binning performance tests and deploy/improve on Summit. The use of the Kokkos portability layer has proven essential for using the GPUs on Summit.

Other Applications

CoPA's LAMMPS/SNAP contribution included an exploration of rendezvous algorithms for the LAMMPS setup and SNAP improvements. The rendezvous algorithms require decomposition across processors. Improvements resulted in a 4.5× speedup on one node, a 940× speedup on 256 nodes, and 1720× on 48,000 nodes (3 to 4 million MPI tasks). All tested problem sizes ran faster. SNAP improvements included memory layout change between kernels via transpose operations, refactored loop indices and data structures, refactored kernels to avoid thread atomics and the use of global memory, used Kokkos hierarchical parallelism and GPU memory, fused kernels to eliminate data structures and reduced memory use, added new data layout to enforce perfect coalescing and load balancing, symmetrized data layouts to reduce memory overhead and use of thread atomics (CPU/GPU), and precomputed certain parameters. This resulted in a Kokkos/SNAP speedup on Summit of 3.7× in FY20, which is a 21.7× cumulative speedup over the baseline on a single V100 GPU.

Performance enhancements to the HACC code have focused on several areas: the use of DPC++/Sycl for porting CUDA code to pre-Aurora hardware; the exploration of OpenMP Target Off-Load for GPU (OMPT) for portability, although not performant, subgroup concepts within OpenCL; and the exploitation of ArborX for CosmoTools analysis. ArborX is a parallel MPI/Kokkos library dedicated to spatial indexing and geometric search algorithms.

Activity on the XGC code has continued to focus on the C++ port of key computational kernels. The electron push and ion/electron scatter kernels were converted by using Cabana/Kokkos, and the collision kernel uses Kokkos. These conversions resulted in a 12 % overall performance gain on Summit. Roofline analysis is shown in Fig. 86.

Next Steps

PROGRESS/BML

FY21 plans include implementing distributed memory parallelism, continuing to develop and optimize for GPU accelerator architectures, fully integrating one sparse format and performant for BML on GPU, demonstrating the use of BML in MGmol for density matrix divide and conquer, continuing to integrate with LATTE and other partner applications, adapting libraries to pre-exascale and exascale architectures (Aurora and Frontier), and finishing COVID QM/MM NAMD-LATTE/PROGRESS/BML efforts.

Cabana

CoPA/Cabana FY21 milestones are driven by application engagement. Foremost is a focus on Frontier and Aurora platforms for exascale delivery with the performance milestone due on September 30, 2021. Because of the tight engagements, the support of WDMApp and ExaAM simulation capabilities will be emphasized with a library release milestone due on June 30, 2021. MD algorithmic development (EXAALT) will continue with the demonstration of scalable and performant halo exchanges, long-range solvers, tree-based neighbor lists, and ML potentials in CabanaMD with the goal of incorporation into LAMMPS. Support for HACC analysis tools and developing a main compute kernel proxy app based on Cabana will be delivered. CabanaPIC will be enhanced to explore the implementation of advanced algorithms for plasma simulations, which has been a main interest of our growing external collaborations.

Exascale Philosophy/Readiness

Recognizing the importance of data layout and data movement on exascale architectures, both the Cabana and Progress/BML libraries are designed for exascale by optimizing memory usage. Cabana uses

the generalized concept of AoSoA, whereas Progress/BML is optimizing a data layout for sparse matrix multiplication. An AoSoA is a generalization of the more commonly known Struct-of-Arrays (SoA) and Array-of-Structs (AoS) with both being common choices in practice, depending on the type of computation and the architecture on which the simulation is performed. Both libraries are designed for exascale by using highly tuned third-party libraries and optimized portability layers (Kokkos) by separating execution and memory to enable closer mapping to vendor machine road maps and kernel fusing to reduce memory usage.

8.4 AMReX

The goal of this project is to develop a new framework, AMReX, to support the development of block-structured AMR algorithms for solving systems of partial differential equations on exascale architectures. Block-structured AMR provides the basis for the temporal and spatial discretization strategy for many applications relevant to DOE. Seven ECP application projects—in the areas of accelerator design (WarpX, § 4.6), astrophysics (ExaStar, § 5.1), combustion (Pele, § 4.2), cosmology (ExaSky, § 5.2), multiphase flow (MFIX-Exa, § 4.4), wind energy (ExaWind, § 4.1), and additive manufacturing (ExaAM, § 3.5)—are using the exascale AMR capability under development. AMReX provides a unified infrastructure with the functionality needed for these and other AMR applications to be able to use exascale architectures effectively.

AMR reduces the computational cost and memory footprint compared with a uniform mesh while preserving the local descriptions of different physical processes in complex multiphysics algorithms. Fundamental to block-structured AMR algorithms is a hierarchical representation of the solution at multiple levels of resolution. At each level of refinement, the solution is defined on the union of data containers at that resolution, each of which represents the solution over a logically rectangular subregion of the domain. Solution strategies vary from level-by-level approaches (with or without sub-cycling in time) with multilevel synchronization to full-hierarchy approaches, and any combination thereof. AMReX provides data containers and iterators that understand the underlying hierarchical parallelism for field variables on mesh, particle data, and embedded boundary (cut cell) representations of complex geometries. Both particles and embedded boundary representations introduce additional irregularity and complexity in the way data are stored and operated on, requiring special attention in the presence of the dynamically changing hierarchical mesh structure and AMR time stepping approaches.

The AMReX team is working closely with application partners to ensure that the software meets their requirements. The team is also working closely with several ST projects to take advantage of new tools that are being developed. Finally, the team is engaged in a dialogue with hardware vendors to provide them with information about adaptive mesh algorithms and provide feedback on the impact of hardware design decisions on AMR applications.

8.4.1 AMReX: Algorithms and Software Objectives

The goal of the AMReX CD center is to develop a computational infrastructure, AMReX, to support applications that already use or plan to use block-structured AMR at the exascale. For the purposes of this project, block-structured AMR is considered to have the following defining features.

- The mesh covering the computational domain is decomposed spatially into structured patches (grids) that each cover a logically rectangular region of the domain.
- Patches with the same mesh spacing are disjoint; the union of such patches is called a *level*. Only the coarsest level is required to cover the domain, although finer levels can also cover it.
- The complete mesh hierarchy on which field variables are defined is the union of all the levels. Proper nesting is enforced, that is, the union of grids at level $l > 0$ is strictly contained within the union of grids at level $l - 1$.
- The physical region covered by each level can be decomposed into different patches to support particle vs. mesh data.
- The mesh hierarchy can change dynamically throughout a simulation.

Within this broad framework, the applications supported represent a wide range of multiphysics problems that couple a variety of different processes and have different computational requirements. Many of these processes are described by systems of partial differential equations that are discretized on a mesh. Discretization strategies for these processes often use either explicit discretizations that express updates in terms of the local state or implicit discretizations that require the solution of linear systems. In some cases, the problem includes stiff systems of ordinary differential equations that represent single-point processes, such as chemical kinetics or nucleosynthesis. Many AMReX-based applications also use Lagrangian particles to represent some aspect of the solution. Particles play a variety of different roles in different applications, ranging from passively advected quantities used for analysis to playing the dominant role in the overall dynamics. Several applications have a requirement for complex geometries. For these types of applications, the team is developing an efficient EB representation in which the solid boundaries are represented as an interface that cuts through a regular adaptive mesh on which the fluid variables are defined.

At the core of the AMReX software is a flexible set of data structures that can be used to represent block-structured mesh data in a distributed memory environment. Operations supported on these data structures include iterators for operations at a level, a communications layer to handle ghost cell exchange and data distribution, and tools for synchronization between levels. The iterators support logical tiling with OpenMP on CPU-based architectures, as well as kernel launching and effective memory management on hybrid CPU/GPU systems.

This basic framework includes native geometric multigrid solvers with support for solving systems arising from embedded boundary discretizations and interfaces to external solvers, such as those provided by hypre and PETSc. On top of this core functionality, the team is also developing a rich and flexible set of tools for treating Lagrangian particles. These tools allow for different representations of particle data (SoA vs. AoS), particle communications, and support for particle algorithms in an AMR context. For embedded boundary representations of complex geometry, AMReX provides support for creating and using the relevant geometric information. Additionally, AMReX provides tools for regridding operations and load balancing, a fast I/O layer for writing checkpoint/restart and visualization/analysis data, and a rich set of native profiling tools.

The AMReX design allows developers to interact with the software at several different levels of abstraction. In one approach, the developer uses the AMReX data structures and iterators for single- and multilevel operations but retains complete control over the time evolution algorithm (i.e., the ordering of algorithmic components at each level and across levels). In an alternative approach, the developer exploits additional functionality in AMReX that is designed particularly to support traditional sub-cycling-in-time algorithms. In this approach, stubs are provided for the necessary operations, such as advancing the solution on a level, correcting coarse grid fluxes with time- and space-averaged fine grid fluxes, averaging data from fine to coarse, and interpolating from coarse to fine. One guiding principle for the AMReX design is to maintain flexibility in discretizations and time-stepping strategies. The core software components are designed to provide the flexibility to support the exploration, development, and implementation of new algorithms that might generate additional performance gains. Although many core discretizations, such as standard second-order and some fourth-order spatial and temporal discretizations, are provided within the AMReX framework for the convenience of users and developers, AMReX also provides application developers with sufficient access to the underlying data structures to allow them to implement and optimize new discretizations.

8.4.2 AMReX: Performance Objectives

The applications AMReX supports represent a wide range of multiphysics applications with different performance characteristics. Consequently, AMReX must provide a rich set of tools to allow for sufficient flexibility so that performance can be tuned for different situations. Furthermore, as part of the AMReX design, specific language requirements are not imposed on users. Specifically, the project supports application modules written in Fortran, C, C++, or other languages that can be linked to C++.

Hierarchical Parallelism

AMReX is based on a hierarchical parallelism model. At a coarse-grained level, the basic AMReX paradigm is based on the distribution of one or more patches of data to each node with an owner-computes rule to allocate tasks between nodes. For many use cases, a node is divided into a small number of MPI ranks and the coarse-grained distribution is over MPI ranks. For example, on a system with six GPUs per

node, the node would typically have six MPI ranks. The nodes on modern architectures all have hardware for parallel execution within the node, but there is considerable variability in the the details of intranode parallel hardware. For code executing on CPUs, AMReX supports logical tiling for cache reuse by using OpenMP threading. Tile size can be adjusted at runtime to improve cache performance; tile size can also vary between operations. AMReX includes both a standard synchronous strategy for scheduling tiles and an asynchronous scheduling methodology. AMReX also provides extensive support for kernel launching on GPU accelerators (using C++ lambda functions) and for effective mmory management, that allows users to control where their data are stored. Although much of the internal AMReX functionality currently uses CUDA/HIP/DPC++ for maximum performance on current machines, AMReX supports the use of CUDA, HIP, DPC++, OpenMP, or OpenACC in AMReX-based applications. Specific architecture-dependent aspects of the software for GPUs are highly localized, enabling AMReX to easily support other GPU architectures.

To meet the diverse requirements of particle applications, both AoS and SoA representations of particle data are included. Multiple types of particles can coexist in a single application; different types can carry different numbers of real and integer attributes. Operations on particles are controlled by a particle iterator that also uses a similar tiling or kernel launching approach. However, particle tiling differs from grid tiling in that the particle tiling determines the memory layout of the particles, whereas mesh data tiling does not change the layout.

Linear Solvers

Several applications that use AMR require the solution of one or more linear systems at each time step. This has three important implications. First, performant linear solvers are necessary for overall performance. AMReX includes single-level and multilevel native linear solvers for nodal or cell-centered data, as well as interfaces to external solvers, such as those in hypre and PETSc. Second, regardless of the specific solution procedure, the efficient solution of elliptic equations at scale requires attention to efficient global communication. The third implication is that the effectiveness of general task scheduling approaches might be constrained by the synchronization points/barriers imposed by linear solvers within a time step.

The team refactored the AMReX native linear solvers for improved parallel performance and extended them to work on hybrid CPU/GPU systems. As part of this development, the team has implemented agglomeration (i.e., merging boxes in the AMR hierarchy to enable additional coarsening as part of the multigrid algorithm) and consolidation (i.e., reducing the number of ranks to reduce communication costs at coarser multigrid levels) strategies from HPGMG as part of the general-purpose solvers.

Communication

AMReX grids can have a complicated layout, which makes the communication metadata needed for ghost cell exchange and inter-level communication nontrivial to construct. AMReX uses a hash-based algorithm to perform intersections between `Box`es. The hash is constructed in an $O(N)$ operation, where N is the number of global `Box`es, and is cached for later use. It then takes only $O(n)$ operations to construct the list of source and destination `Box`es for communication, where n is the number of local `Box`es. Furthermore, the communication metadata is also cached for reuse.

AMReX supports GPU-aware MPI on GPU machines, if available. To reduce latency, the data that needed to be communicated through MPI are aggregated into communication buffers. For packing the buffer, we need to copy data from slices of multidimensional arrays to the 1D buffer, whereas for unpacking the buffer, data are copied from the 1D buffer to slices of multidimensional array. The straightforward approach of the copying is to launch a GPU kernel for each slice. Unfortunately, this simple approach is very expensive because there are often hundreds of very small GPU kernels. In AMReX, we implemented a fusing mechanism that merges all these small GPU kernels into one kernel, significantly reducing kernel launch overhead.

Performance Characterization

Performance profiling is a crucial part of developing and maintaining a successful scientific code, such as AMReX and its applications. The AMReX community uses a wide variety of compatible profilers, including VTune, CrayPat, ARM Forge, Amrvis, and the Nsight suite of tools. AMReX includes its own lightweight instrumentation-based profiling tool, TinyProfiler. TinyProfiler consists of a few simple macros that insert scoped timers throughout the application. At the end of a simulation, TinyProfiler reports the total time spent in each region, the number of times it was called, and the variation across MPI ranks. This tool is on

by default in many AMReX applications and includes NVTX markers to allow instrumentation when using Nsight.

I/O: In Situ Analysis and Visualization

AMReX provides a native file format for plotfiles that stores the solution at a given time step for visualization. Plotfiles use a well-defined format that is supported by a variety of third-party tools for visualization. AMReX provides functions that directly write plotfiles from mesh and particle objects. Mesh and particle plotfiles are written independently for improved flexibility and performance. AMReX also provides users with the option to use HDF5 for data analysis and visualization.

Writing a plotfile requires coordination between MPI ranks to prevent overwhelming the I/O system with too many simultaneous writes to the file system. AMReX has implemented multiple output methodologies to provide efficient I/O across a variety of applications and simulations. A static output pattern prints in a predetermined pattern that eliminates unnecessary overhead, which is useful for well-balanced or small simulations. A dynamic output pattern improves write efficiency for complex cases by assigning ranks to coordinate the I/O in a task-like fashion. Finally, asynchronous output assigns the writing to a background thread, allowing the computation to continue uninterrupted while the write is completed on a stored copy of the data. The I/O output methodology and other standard features, such as the number of simultaneous writes, can be chosen through runtime and compile-time flags.

Asynchronous I/O is currently the targeted I/O method for exascale systems because it is a portable methodology that substantially reduces I/O impact on total runtime. AMReX's native Async I/O has reduced write time by a factor of around 80 on full-scale Summit simulations. However, asynchronous I/O requires a scalable `MPI_THREAD_MULTIPLE` implementation to achieve the best results.

8.4.3 AMReX: Co-Design Engagements and Integration Points

ECP Applications

Seven ECP application projects in the areas of accelerator design (WarpX), astrophysics (ExaStar), combustion (Pele), cosmology (ExaSky), multiphase flow (MFIX-Exa), wind energy (ExaWind), and additive manufacturing (ExaAM) include codes based on AMReX. All codes use the basic mesh data structures and iterators along with additional capabilities, which are discussed as follows.

- WarpX is a multilevel electromagnetic PIC code for simulating plasma accelerators; electrons are modeled as AMReX particles while the electric and magnetic fields are defined on the hierarchical mesh.
- The ExaStar project is developing the CLASH ecosystem, which includes the FLASH and Castro simulation codes for compressible astrophysical flows and the Sedona code for radiation transport; all three use AMReX. Particles can be used as tracer particles in Castro and FLASH5 and in an MC algorithm in Sedona; linear solvers are used to solve for self-gravity. CVODE can be used to evolve nuclear reaction networks.
- The Nyx N-body plus hydrodynamics code in the ExaSky project is based on AMReX. The particles represent dark matter, linear solvers are used to solve for self-gravity, and CVODE is called to integrate the heating/cooling source terms.
- The MFIX-Exa multiphase modeling code is based on AMReX; the particles represent solid particles within a gas, the EB methodology is used to represent the bounding geometry, and the linear solvers are used for pressure solves in a projection formulation and for the implicit treatment of viscous terms.
- The compressible combustion code, PeleC, and the low-Mach number combustion modeling code, PeleLM, are based on AMReX. Both use the EB methodology to represent the problem geometry and possibly CVODE to evolve the chemical kinetics. PeleLM uses the linear solvers to solve for the dynamic pressure field in a projection formulation and for the semi-implicit treatment of viscous terms. Particles can be used as tracer particles and to represent sprays.

Table 66: AMReX KPP-3 goals and metrics.

Passing value	Stretch value	Tentative present value
6	14	9

Table 67: AMReX code base.

Package name	LOC	Target exascale challenge problems	Computational motifs
AMReX	325,000	N/A	AMR, structured grids, particles, sparse linear algebra, ODEs

- The ExaWind project combines AMR-Wind, an AMReX-based multilevel structured mesh flow solver, with Nalu-Wind, an unstructured mesh flow solver. The flow solvers are coupled by using an overset mesh approach handled by the TIOGA library. Both Nalu-Wind and AMR-Wind solve the incompressible Navier-Stokes equations with additional physics to model the atmospheric boundary layer. In a wind-farm simulation, Nalu-Wind is designed to resolve the complicated geometry and flow near the wind turbine blades, and AMR-Wind solves for the flow in the full domain away from the turbines.
- One of the codes in the ExaAM project, TruchasPBF, uses the multilevel linear solvers in AMReX. TruchasPBF targets the modeling of part-scale process and melt pool physics.

Additionally, the AMReX CD center regularly communicates with the CoPA CD center regarding best practices for particle data layout and operations. There has been no direct use of shared code, but discussions are ongoing.

ECP Software Technologies

The AMReX CD center has interacted with many of the ECP ST projects. The most significant of those interactions have been with the SUNDIALS, HDF5, and ALPINE projects. SUNDIALS integrators are used by the Nyx and Pele codes. Additionally, the AMReX CD center has regular interactions with the following.

- The AMReX CD center interacts with the HDF5 project (2.3.4.08). Both the mesh and particle data in AMReX plotfiles and checkpoint files can now be written in HDF5 format. The Nyx code in the ExaSky project uses this capability extensively.
- The AMReX CD center interacts with the APLINE project (2.3.4.12) to determine the best ways for ALPINE and SENSEI to support the AMReX-based application projects. A publication describing joint work between the ALPINE, MFIX-Exa and AMReX teams on in situ feature analysis, tracking, and data reduction in MFIX-Exa was published in 2020.
- The AMReX team maintains xSDK compatibility and interoperability. AMReX has been part of the last two xSDK releases.

ECP Vendor Interaction

The AMReX vendor liaison continues to interact with several vendors, reading PathForward milestones and participating in PathForward reviews. The liaison regularly provides feedback geared toward improving the sustained performance that future systems will deliver on AMReX-based and similar applications. Summaries of the architectural trends and implications have been discussed with ECCN/RSNDA-cleared personnel within AMReX to ensure that AMReX software development is in line with trends in architecture and system software. In addition to these high-level interactions, several members of the core AMReX development team have regular detailed discussions with AMD and Intel engineers about issues related to using AMReX on test hardware.

```
amrex::ParallelFor(bx,
[=] AMREX_GPU_DEVICE (int i, int j, int k) noexcept
{
    a(i,j,k) = a(i,j,k) + 0.5*b(i,j,k);
});
```

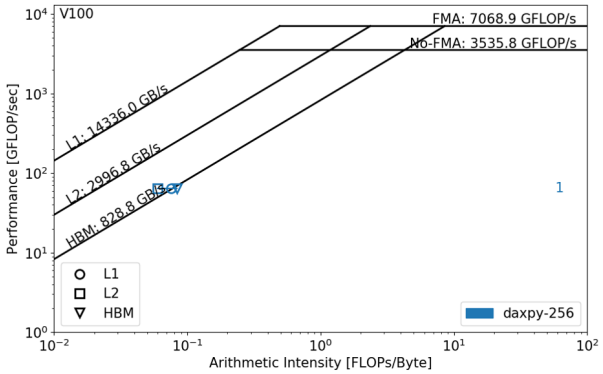


Figure 87: Roofline analysis of memory-bound kernel on Summit.

8.4.4 AMReX: Progress on Early and Pre-Exascale Hardware

Performance on Summit

One main consideration in the design of AMReX is to provide performance portability to applications. AMReX must support a variety of new architectures that are being developed with different capabilities and programming models. Our goal is to isolate applications from any particular architecture and programming model without sacrificing performance. To achieve this goal, we have introduced a lightweight abstraction layer that effectively hides the details of the architecture from the application. This layer provides constructs that allow users to specify what operations they want to perform on a block of data without specifying how those operations are carried out. AMReX then maps those operations onto the hardware at compile time so that the hardware is used effectively. For example, on a manycore node, a stencil operation would be mapped onto a tiled execution model by using OpenMP to guarantee good cache performance, whereas on a different architecture, the same operation might be mapped to a kernel launch appropriate to a particular GPU.

The main looping construct is a `ParallelFor` that provides the basic looping construct for both structured mesh and particle data. On Summit, the `ParallelFor` launches a CUDA kernel on the GPU. Other performance portability constructs provide support for on-node reductions and memory allocation.

To assess the performance of the `ParallelFor`, we performed a roofline analysis on some simple kernels. The first example, shown in Fig. 87, is a simple `DAXPY` loop, which is memory-bound. The figure shows that the performance is limited by HBM bandwidth.

The second kernel is a compute-bound kernel that computes the square root of elements of an array by using a fixed number of Newton iterations. As expected, in this case, the performance is limited by floating point performance, as shown in Fig. 88.

The final roofline example is the construction of the neighbor list for particles. This is a somewhat more complex example that includes multiple kernel launches. The roofline analysis, shown in Fig. 89, shows that these kernels are all memory-bound. The key observation from these different roofline analysis plots is that the AMReX constructs introduced for portability do not sacrifice performance. On Summit, all these examples show that the AMReX implementation provides performance near the capability of the architecture.

Another important performance issue for several AMReX applications is the performance of linear solvers. We tested the performance of the AMReX linear solvers on Summit vs. Cori Haswell. A weak scaling study is shown in Fig. 90. The solution time on a single node of Summit (using only the six GPUs) is six times faster than on a single node of Cori Haswell using 32 CPU cores. Weak scaling results are reasonable up to 4096 Summit nodes with scaling that is consistent with the fact that linear solvers are communication intensive, particularly on GPU architectures. Consolidation and aggregation were also demonstrated to generate an order of magnitude (or more) savings for large numbers of MPI ranks.

Next, we consider two more comprehensive examples. The first example is electromagnetic PIC. This example includes the full PIC loop, including ghost cell exchange, deposition across multiple grids, and

```
amrex::ParallelFor(bx,
[=] AMREX_GPU_DEVICE (int i, int j, int k) noexcept
{
    Real y = a(i,j,k);
    Real x = 1.0;
    for (int n = 0; n < 20; ++n) {
        Real dx = -(x*x-y) / (2.*x);
        x += dx;
    }
    a(i,j,k) = x;
});
```

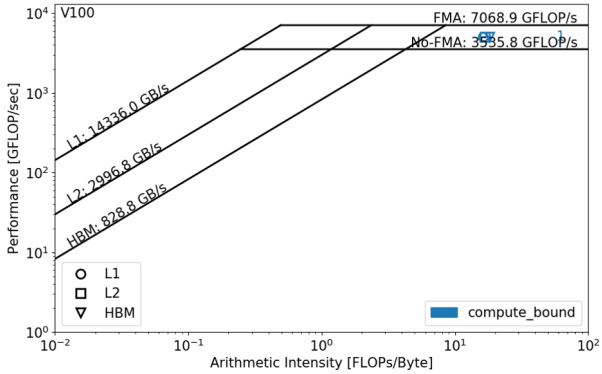


Figure 88: Roofline analysis of compute-bound kernel on Summit.

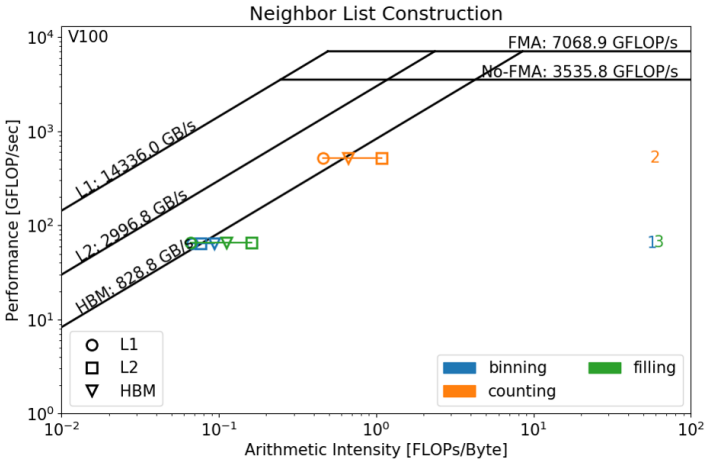


Figure 89: Roofline analysis of particle neighborlist operation.

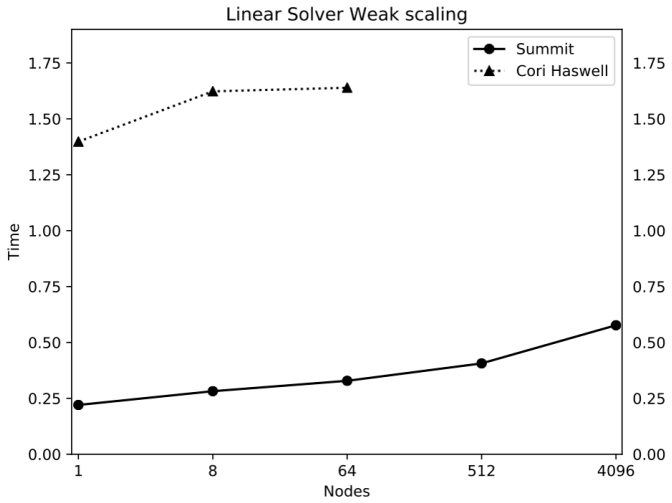


Figure 90: Weak-scaling study of linear solver. There are 256^3 cells per node.

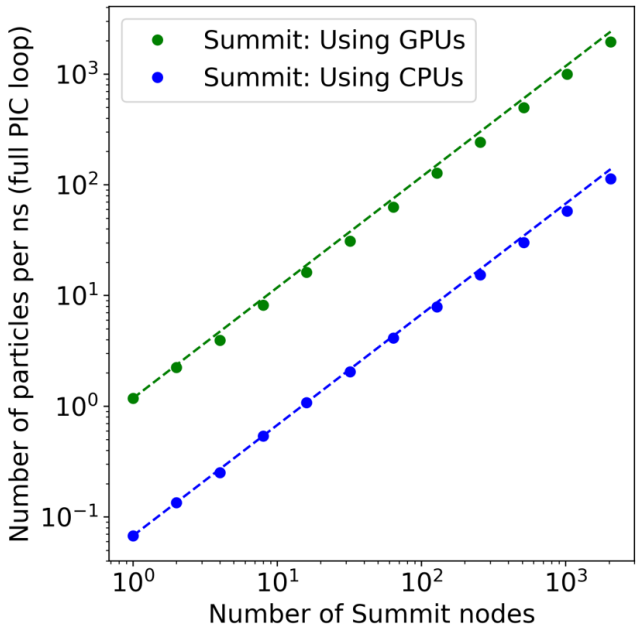


Figure 91: Weak-scaling study of electromagnetic PIC.

particle redistribution. For this example, we compare the performance of 42 CPUs per node vs. six GPUs per node, both on Summit. The GPU runs are approximately 30 times faster than the CPU runs with both showing near-ideal scalability up to 2048 nodes.

As a final example of AMReX performance on Summit, we compare a full three-level AMR simulation of gas dynamics on GPUs vs. CPUs, again with both cases on Summit. On a single node, the six GPUs are approximate 20 times faster than the 42 CPU cores. For this example, we see good scaling to almost the full machine. Regridding in AMReX is a communication-intensive operation. In typical applications, regridding is a much smaller fraction of the total time.

Finally, all the AMReX-based ECP applications except TruchasPBF essentially run fully on the Summit GPUs.

8.5 CEED

The efficient exploitation of exascale architectures requires rethinking the numerical algorithms used in large-scale applications of strategic interest to DOE. These architectures favor algorithms that expose ultrafine-grain parallelism and maximize the ratio of floating-point operations to energy-intensive data movement. Many large-scale applications employ unstructured finite element discretization methods in which practical efficiency is measured by the accuracy achieved per unit of computational time. One of the few viable approaches for achieving high-performance in this case is to use matrix-free high-order finite element methods since these methods can increase the accuracy and/or lower the computational time due to reduced data motion. To achieve this efficiency, high-order methods use mesh elements that are mapped from canonical reference elements (e.g., hexes, wedges, pyramids, tetrahedra) and exploit, where possible, the tensor-product structure of the canonical mesh elements and finite element spaces. Through matrix-free partial assembly, the use of canonical reference elements enables substantial cache efficiency and minimizes extraneous data movement in comparison with traditional low-order approaches.

The CEED CD center is a focused team effort to develop the next-generation discretization software and algorithms that enable a wide range of finite element applications to run efficiently on exascale hardware. High-order methods are the logical choice for this, from a mathematical (higher quality simulations) perspective, and from HPC (better performance) and risk mitigation perspectives (range of orders provides flexibility

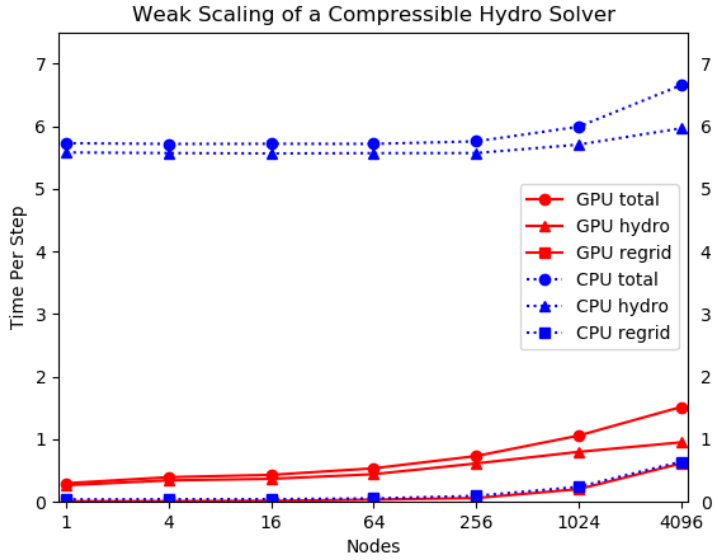


Figure 92: Weak-scaling study for the three-level AMR simulation of inviscid compressible flow.

in the uncertain exascale hardware and software environments). Their efficiency extends to problems with unstructured nonconforming mesh refinement and general curved meshes and includes low-order finite element discretizations as a special case. The team's work covers all of these topics, including the full low- to high-order spectrum of discretizations, allowing software to be easily integrated with low-order applications while enabling such applications to naturally transition from low- to high-order methods.

The team is pursuing a cross-cutting approach that includes working with hardware vendors, software developers, and computational scientists to meet the needs of ECP applications. CEED is developing next-generation finite element discretization libraries to enable unstructured PDE-based applications to take full advantage of exascale resources without the need to reinvent complicated finite element machinery on upcoming hardware. The project team is also delivering CEED mini-apps that combine applications-relevant physics with key high-order kernels that capitalize on matrix-free forms for efficient performance. These mini-apps are being used to inform and influence hardware development. Finally, CEED is working to further the development of more general software technologies, including extensions of dense linear algebra libraries to support fast tensor contractions, performance-portable programming models, and scalable matrix-free linear solvers. These improvements, specifically motivated by finite element applications on exascale hardware, will also benefit the broader scientific computing community.

Current information about the project is available on the website, https://ceed.exascaleproject.org, including recent publications and news items. The software catalog—including MFEM, Nek, the libCEED low-level API library, the CEED benchmarks, the Laghos, Remhos and Nekbone miniapps, the libParanumal set of GPU kernels, and the high-order Field and Mesh Specification (FMS), as well as mirrors of the primary CEED packages—are freely available on GitHub[5].

8.5.1 CEED: Algorithms and Software Objectives

The finite element method is a powerful discretization technique that has been applied to virtually every computational problem involving the solution of differential or integro-differential equations. It has been exhaustively studied, both theoretically and in practice, in the last several decades. Finite elements use mappings to the reference element to evaluate integrals that arise in weak variational formulations of PDE problems. High-order finite elements use higher order polynomials in reference space for approximating

[5] https://github.com/CEED

physics fields and potentially the geometry of mesh elements.

Although they live on unstructured grids and result in globally sparse matrices, finite elements have a dense Cartesian structure locally on each element high-order, thus offering a bridge between the unstructured/structured and sparse/dense worlds, and they can benefit from tools from all of these fields. Spectral elements are a special case of the general high-order finite elements when the degrees of freedom and the quadrature points on the reference element coincide, resulting in a diagonal mass matrix. One can consider spectral elements and general high-order finite elements as two ends of the same spectrum—both use the same finite element machinery, but spectral elements emphasize efficiency, whereas general high-order finite elements emphasize robustness. There is a unique opportunity to combine complementary spectral/finite element R&D efforts within the CEED team to deliver a next-generation high-order discretization portfolio to the ECP applications.

The key to efficient high-order methods for the finite element method is to use factored matrix-free forms with per-grid-point memory demands on par with or lower than standard fully assembled low-order methods and considerable savings in the number of grid points. Matrix-free high-order finite elements offer several important advantages for exascale architectures featuring thread-based nodes with multiple memory hierarchies. First, the order of the methods can be used as a performance-tuning parameter; second, they can handle small on-node memory with just a few elements per core and compute on-the-fly intermediate quantities; and third, their kernels are threadable with localized data, making them well-suited to heterogeneous on-node parallelism, GPUs, and power-efficient computing.

Most of the flops >90 % in matrix-free high-order finite element implementations are in the form of tensor contractions that effect differentiation or interpolation on the reference element. For an element of order p, these contractions require only $n = Ep^3$ memory references for the data and $O(p^2)$ references for the operators, whereas the number of flops scales as $O(np) = O(Ep^4)$. (Here, n is the total number of grid points, E is the number of elements, and p is the approximation order.) Further operations that involve physics or geometry evaluations at the quadrature points require only $O(n)$ work and $O(n)$ memory references. Interprocessor communication on P processors involves only minimal surface data with complexity $O(n/P)^{2/3}$. The stencil is of unit-depth, independent of p.

CEED is focused on developing optimal implementations of finite element operator evaluators and solvers across a variety of applications. One way that these developments are manifest is through a lightweight library, libCEED, and related mini-apps and examples. libCEED is designed to allow users to express central kernels at a low level (e.g., local matrix-vector product) without changing the current discretization. libCEED is an API between front-end apps and back-end kernels that supports efficient operator description, not a global matrix. This description is based on a purely algebraic finite element operator decomposition and thus applies to many applications. The front-end application can use any compatible back end, and any performant high-order code should be able to plug in at that front end now or in the future. The back ends can be added independently of front ends and are divorced from finite element information, allowing computer scientists to optimize the evaluation without domain knowledge. These back ends can be selected for best performance, optimized for order and device (e.g., CPU, GPU), and can use JIT compilation to deliver performance. Each new back end is automatically usable in all compatible front-end applications, giving a broad impact of the optimization efforts.

8.5.2 CEED: Performance Objectives

Petascale finite element codes currently execute with up to 1 million CPU cores by using single-program-multiple-data programming models on distributed-memory architectures. For sufficiently large problems, PDE solvers generally have adequate parallelism to make this approach efficient, even at scale. The principal challenges are having enough work per core to offset communication overhead and linear system solvers that are scalable in terms of communication costs and bounded iteration counts independent of problem size and processor count.

The CEED team has significant experience in developing efficient multilevel codes that have bounded iteration counts at these scales in which even the coarse-grid problem, which has $O(1)$ DOF per core, consists of millions of DOF. For high-order methods, there are two prevailing strategies for preconditioning: (1) exploiting spectral equivalences between low- and high-order operators to reduce the problem of solving a dense but factorizable high-order system to that of solving a sparse low-order system or (2) exploiting the

tensor-product structure of the high-order elements to develop approximate separable operators that can be applied locally within a domain-decomposition context. For the Poisson problem, the former approach is robust and effective if the sparse problem can be solved quickly with algebraic multigrid (AMG), and the latter is potentially faster unless the elements have high aspect ratios. Either approach requires a communication-intensive coarse-grid solve that maps distributed data to a coarse distributed solution. Conquering this coarse-grid communication problem is a significant challenge at exascale, particularly in light of current device-to-host latencies. Presently, one of two approaches is used for the solution of coarse-grid systems of size n_c. For $n_c < 10^5$–10^6, one can project the solution onto a sparse set of vectors $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{nc}\}$ with optimal $\log_2 P$ communication complexity but suboptimal $O(n_c^{5/3}/P)$ work. For larger values of n_c, AMG has proven more effective, despite its $O(\log_2 P)$ communication complexity.

The other principal challenge at exascale is to realize high performance per node in which greater on-chip parallelism makes it imperative to focus on finer-grain parallelism and to exploit significant data reuse. For accelerator-based nodes in particular, there is a need to usefully elevate the flops-to-bytes ratio, and high-order methods are very effective in this respect. Fast tensor-product-based high-order finite element implementations require only $O(n) = O(Ep^3)$ memory references and only $O(Ep^4)$ operations. The factor of p increase in operation count arises from tensor contractions—applications of derivative or interpolation operators on the reference element—which are readily expressed in the form of BLAS3 `dgemm`s.

For CPUs, optimizing the `dgemm`s for small matrices of order $p = 6$–16 (typically) is the standard starting point for performance tuning. However, to further push development and identify the fastest implementations for the commonly occurring high-order kernels, the team has developed a series of benchmarks known as the *bake-off kernels (BKs)* and *bake-off problems (BPs)*. CEED's BK/BPs are designed to test and compare the performance of high-order implementations. The current specifications (BP1-BP6 and BK1-BK6) describe simple mass and stiffness matrix solves without preconditioning with specific choices for high-order DOF and quadrature points. These benchmarks were used to compare Nek (§ 4.3), MFEM, libParanumal, and deal.ii (external to CEED) and have already resulted in improvements across these codes by learning from each other. The idea is that the open-source competition benefits all high-order applications and ensures that no code misses significant optimization opportunities on present and future architectures.

In addition to pushing kernel development, the BPs serve an important role in identifying the strong-scale limits of various implementations and architectures, which is important when the assessing overall performance of a given code-problem coupling on a particular platform.

For GPUs, the CEED development is particularly taking advantage of the OCCA and MAGMA efforts in the project. OCCA provides a portable means of expressing accelerator-directed code to be translated in CUDA OpenCL, or OpenMP code that performs as well as hand-tuned CUDA. MAGMA is directed toward fast tensor contractions and batched `dgemm`s that are appropriate for the size of matrices encountered in high-order finite element codes. Currently, all the mini-apps have GPU versions, and the team is in the process of bringing these technologies to first-wave applications.

High order alone is insufficient to ensure high performance on either CPUs or GPUs. In the case of CPUs, this point is illustrated by the significant variance in the bake-off results between Nek5000, MFEM, and deal.ii. In the case of GPUs, the CEED team demonstrated that it is possible to match roofline performance models on the NVIDIA P100 and V100s, but only after extensive kernel tuning. For high enough order and problem sizes, these kernels have been shown to achieve up to 2 TFlops on a V100 GPU on Summit, as shown in Fig. 93. The tuning steps include mapping intermediate arrays in the tensor contraction steps to shared memory, padding shared memory for order-8 or order-16 tensors to avoid bank conflicts, unrolling the inner contraction loops, reducing thread synchronizations by allocating additional shared memory, and replacing shared memory references with register read/write instructions, where possible. Clearly, even for something as straightforward as tensor contractions, the realization of fast implementations on GPUs and other accelerators requires a deeper understanding of the architecture than most application scientists would desire. One objective of CEED is to make these fast implementations accessible through a convenient interface, libCEED, which supports tuned backends for each of the forthcoming exascale architectures.

One of the goals of the project is to explore and identify the best algorithms (in terms of time to solution) for the full range of discretization spaces: from low-order ($p = 1$) to high-order (e.g., $p = 16$). For that purpose, the benchmark problems described below, which represent the key numerical kernels of several PDE solvers, are used. The computational experiments performed in CEED explore the best implementations for a given architecture, the components that make them perform well, and software expressions that allow these

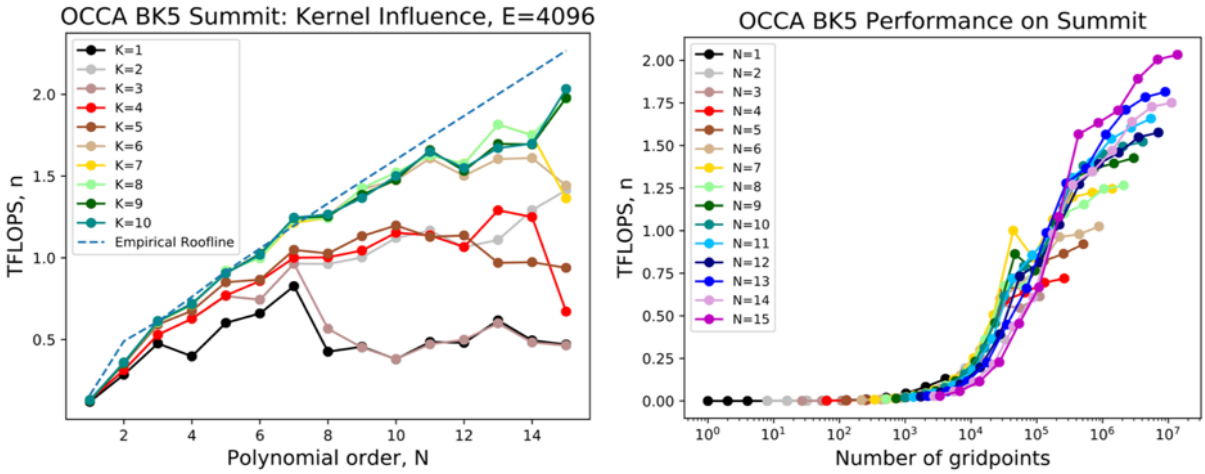


Figure 93: Significant variance in the bake-off results.

implementations to be leveraged on exascale platforms without change to the application-level software.

One example of the performance evaluation and improvement activities is given below, where CEED's BP1 benchmark is run on the Cetus BG/Q machine at ANL (BG/Q is picked for repeatability). All of these runs use 8192 cores in C32 mode and loop over orders $p = 1, \ldots, 16$ with $q = p + 2$ quadrature points. High-order methods clearly show better maximum performance, but more importantly they are also better in the strong-scaling limit (left on the x-axis) where a user may be running on many nodes at an efficiency no lower than 50 %. The problem size per node at which that efficiency is achieved is called $n_{1/2}$ and is a critical value in performance evaluations.

8.5.3 CEED: Co-design Engagements and Integration Points

CEED is delivering discretization libraries for applications, benchmarks, and standards for the high-order community and Miniapps for hardware vendors and ST projects interactions. These deliverables involve close collaboration between four R&D thrusts: Applications, Hardware, Software, and Finite Elements. While each thrust is focused on a specific ECP goal, their work is highly integrated across team members, institutions and deliverables.

Application targets

The CEED co-design team is interested first and foremost in applications. The team has a track record of delivering performant software on leading-edge platforms. The team collectively supports hundreds of users in national laboratories, industry, and academia, and is committed to pushing simulation capabilities to new levels across an ever-widening range of applications. In the ECP the team uses a focused one-on-one interaction with applications facilitated by CEED application liaisons, as well as through one-to-many interactions, based on the development of easy-to-use discretization libraries for high-order finite element methods. The first-wave application targets are the ExaSMR (§ 4.3 application from ORNL and the MARBL application from LLNL (§ 7.2) are already integrated with Nek5000 and MFEM respectively. Additional application targets are the E3SM (§ 5.5), ExaWind (§ 4.1), ExaAM (§ 3.5), SNLApp, GEOS, WDMApp (§ 4.5), and the Combustion projects (§ 4.2). In addition to maintaining a close connection with these high-priority ECP applications, the team is reaching out to lower-priority ECP and non-ECP applications; these interactions are used to derive requirements for CEED's mini-apps and software technologies.

Application engagement comes in two primary forms. Initially, CEED build on top of existing application codes/libraries. The ExaSMR application from ORNL and the MARBL application from LLNL are already integrated with the CEED base codes Nek5000 and MFEM, respectively. Other ECP applications that may engage these codes include ExaWind and ExaAM. Exascale advancements for these applications include all developments directed at Nek5000 and MFEM, including extensions to GPUs and Aurora. Secondly,

CEED reached out to new applications through libCEED. The goal is to provide PDE-based applications with a lightweight and portable interface to highly performant kernels on all of the exascale platforms. Particular candidates include E3SM (climate), SNLApp, GEOS, and WDMApp. In some cases, the general libCEED model need to be tailored to the application kernels. For example, E3SM uses a mixed FE/finite-difference formulation. Its local tensor-product structure, however, is ideally suited to the fast tensor-product factorizations that are at the heart of CEED, such that extensions are very natural. The CEED development group is also reaching out to these applications to assess the potential of libCEED-enabled performance gains.

Benchmarks: Bake-Off Problems and Bake-Off Kernels

CEED's BPs are extensions of the BKs mentioned earlier that are instantiations of the high-order kernels/benchmarks within iterative solvers that are designed to test and compare the performance of high-order codes under realistic data loading and communication patterns. The current specifications (BP1-BP6 and BK1-BK6) describe simple mass and stiffness matrix solves (without preconditioning) with specific choices for high-order degrees for freedom and quadrature points. These benchmarks have been used to compare Nek, MFEM and deal.ii (external to CEED) and have already resulted in improvements from learning from each other, that can benefit all high-order applications.

Community Standards: High-Order Operator Format and Field and Mesh Specification

One of the challenges with high-order methods is that a global sparse matrix is no longer a good representation of a high-order linear operator, both with respect to the FLOPs needed for its evaluation, as well as the memory transfer needed for a matrix-vector multiply (matvec). Thus, high-order methods require a new "format" that still represents a linear (or more generally nonlinear) operator, but not through a sparse matrix. One of the goals of libCEED is to propose such a format, as well as supporting implementations and data structures, that enable efficient operator evaluation on a variety of computational device types (e.g., CPUs, GPUs). This new operator description is based on the algebraically factored form given by the finite element decomposition above, which is easy to incorporate in a wide variety of applications, without significant refactoring of their own discretization infrastructure.

Another challenge for the practical use of high-order methods is the lack of common description of high-order simulation data (both meshes and fields) which hampers data exchange between applications as well as high-order visualization. CEED's Field and Mesh Specification (FMS) is a newly proposed high-order interface, that allows a wide variety of applications and visualization tools to represent unstructured high-order meshes with general high-order finite element fields defined on them. FMS is intended as a lightweight format and API that can represent general finite elements within a common, easy to use framework. This includes high-order solutions and meshes as well as non-standard finite elements, such as Nedelec and Raviart-Thomas elements.

Miniapps: Nekbone, Laghos, Remhos, libParanumal

Nekbone is a lightweight subset of Nek5000 that solves a standard Poisson equation; weak-scaled to 6 million MPI ranks; currently supports OpenACC / CUDA-based GPU variants. Laghos is a CEED-developed miniapp that for the first time provides a proxy for high-order discretizations of the Euler equations of compressible gas dynamics, as solved by the BLAST code at LLNL (the ALE component of ECP's MARBL application). Laghos features moving (high-order) curved meshes, (high-order) explicit time integration, AMR version, and OCCA and RAJA versions targeting GPUs. Both Nekbone and Laghos are procurement benchmarks for CORAL2 and ECP 1.0 proxy apps. Remhos is a new CEED mini-app that complements Laghos. It reflects the Lagrangian remap phase that is used in the BLAST code. libParanumal (formerly Holmes) is a CEED-developed experimental test bed for multilevel parallel implementations of high-order finite element computations; under development.

Collaboration with Hardware Vendors and Software Technology Projects

The CEED team is working closely with several of the ECP vendors, most notably: Intel, CRAY, AMD, IBM and ARM on hardware optimizations, miniapp evaluation, the Aurora architecture, GPU performance and more. The team built a two-way (pull-and-push) collaboration with the vendors, where the team develops hardware-aware technologies (pull) to understand performance bottlenecks and take advantage of inevitable hardware trends, and vendor interactions to seek (push) impact and improve hardware designs within the

Table 68: CEED KPP-3 goals and metrics.

Passing value	Stretch value	Tentative present value
6	12	6

Table 69: CEED code base.

Package name	LOC	Target exascale challenge problems	Computational motifs
MFEM	260k C++	MARBL, ExaAM, GEOS, SNLApp	High-order finite elements, unstructured AMR, computation, matrix-free computation, scalable solvers
NekRS	74k C/C++/Fortran	ExaSMR, ExaWind, E3SM	High-order methods, matrix-free computation, scalable solvers
Nek5000	200k Fortran/C	ExaSMR, ExaWind, E3SM	High-order methods, matrix-free computation, scalable solvers
libCEED	67k C	CEED, SNLApp, E3SM, GEOS	Low-level API library for efficient high-order operator evaluation
libParanumal	53k C++/OKL	CEED	Prototype GPU accelerated algorithms for high-order finite element methods
OCCA	69k C++	CEED	Portable many-core programming platform
MAGMA	346k C/C++	CEED	Numerical linear algebra batched linear algebra, tensor contractions, dense and sparse matrix computations
PUMI	93k C++	CEED	Parallel, unstructured, mesh infrastructure

ECP scope. CEED is also collaborating with a number of projects in ECP's software technologies focus area, including MPICH, STRUMPACK, PETSc, Spack, SUNDIALS, ALPINE/VTK-m, KokkosKernels and ZFP. In addition, CEED packages are also part of the FASTMath institute in SciDAC, the xSDK and the OpenHPC distribution.

8.5.4 CEED: Progress on Early and Pre-Exascale Hardware

Performance on Summit

All of the CEED software components have been ported to NVIDIA GPUs and have run on Summit, Lassen or similar V100 machines. Two examples of this work are the NekRS collaborations with ExaSMR and the MFEM collaborations with MARBL. While not reported here, the CEED team has also collaborated with ExaWind and ExaAM in their Summit runs.

NekRS Applications—ExaSMR and ExaWind

NekRS is a new GPU-oriented version of Nek5000 written in OCCA, which is a C-based concurrent compute abstraction that provides portability across a variety of backends, including CUDA, HIP, OpenMP, and OpenCL. The principal kernels in NekRS come from the high-order finite-element library, libParanumal, out of Virginia Tech. These kernels typically realize >90 % of the bandwidth-limited peak and sustain 1–2 TFlops/s (FP64) on a single V100, depending on the kernel and polynomial order. Extensive testing has

Table 70: ExaSMR: NekRS strong and weak scaling performed on Summit, using 6 GPUs per node, for simulating turbulent flow in the 17×17 rod-bundle, with $Re_D = 5000$. Time per step in seconds (t_{step}), velocity iteration count (v_i), and pressure iteration count (p_i), are all averaged over 100 steps. R is the ratio of t_{step} of 1810 nodes to that of others for strong scaling and t_{step} of 87 nodes to that of others for weak scaling, provided with the ideal ratio, R_{ideal} and the parallel efficiency, P_{eff}.

ExaSMR application performance: 17×17 fuel rods simulation												
case	node	gpu	E	N	E/gpu	n/gpu	v_i	p_i	$t_{step}(s)$	R	R_{ideal}	P_{eff} (%)
strong	1810	10 860	175 618 000	7	16 171	5.5M	4	2	1.855e-01	1.00	1.00	100
	2536	15 216	175 618 000	7	11 542	3.9M	4	2	1.517e-01	1.22	1.40	87
	3620	21 720	175 618 000	7	8085	2.7M	4	2	1.120e-01	1.65	2.00	82
	4180	25 080	175 618 000	7	7002	2.4M	4	2	1.128e-01	1.64	2.30	71
	4608	27 648	175 618 000	7	6351	2.1M	4	2	1.038e-01	1.78	2.54	70
case	node	gpu	E	N	E/gpu	n/gpu	v_i	p_i	$t_{step}(s)$	R	R_{ideal}	P_{eff} (%)
weak	87	522	3 324 000	7	6367	2.1M	4	2	8.57e-02	1.00	1.00	100
	320	1920	12 188 000	7	6347	2.1M	4	2	8.67e-02	0.98	1.00	98
	800	4800	30 470 000	7	6347	2.1M	4	2	9.11e-02	0.94	1.00	94
	1600	9600	60 940 000	7	6347	2.1M	4	2	9.33e-02	0.91	1.00	91
	3200	19 200	121 880 000	7	6347	2.1M	4	2	9.71e-02	0.88	1.00	88
	4608	27 648	175 618 000	7	6351	2.1M	4	2	1.03e-01	0.83	1.00	83

established that the strong-scale limit for NekRS (where performance is 80 % of its saturated peak) is ∼2.5M points per GPU, for both the NVIDIA V100 and A100. Using more GPUs (and thus fewer points per GPU) beyond this point does not yield significant speedup. A strong-scaling study, see Table 70, for a 60 billion grid point 17×17 rod-bundle configuration for ExaSMR realizes 70 % parallel efficiency on all of Summit (27,648 NVIDIA V100s, 2.1 million points per GPU), and 92 % efficiency on 21,720 V100s (2.7 million points per GPU). A weak-scaling study for the same configuration, using 2.1 million points per GPU, shows only a 20 % increase in time-per-step as the number of nodes is increased from $P = 87$ to 4608 (full-core). The factor of 42 increase in compute power for a 50 fold increase in number of GPUs is remarkably good, given that the chosen problem size per V100 is relatively small and thus influenced by communication overhead. Finally, for a fixed-size problem at the strong-scale limit, NekRS on Summit outperforms Nek5000 on Mira by about a factor of three, which is notable because the strong-scale limit on Mira is ∼4000 points per core, a factor of 600 smaller than the limit on the V100s. The 3× gain on Summit comes from a combination of kernel performance, communication hiding, and algorithmic improvements.

MFEM Applications—MARBL and ExaAM

MARBL is a next-gen multiphysics simulation code being developed at LLNL. The code provides multi-material radiation-magneto-hydrodynamics with applications in ICF, pulsed power and equation of state/material strength experiments as part of the NNSA ATDM program. One of the central features of MARBL is an ALE formulation based on the MFEM-enabled BLAST package, which solves conservation laws of mass, momentum, and energy in a moving material frame. The BLAST package utilizes high-order finite element discretizations of physical processes on a high-order (curved) moving mesh. The GPU port of BLAST makes extensive use of on the partial assembly (PA) technology from CEED and GPU support via MFEM. Below we provide specifics about the major GPU kernels in BLAST, and the impact of the CEED project in these GPU development efforts.

Memory management: Since MARBL/BLAST is based on MFEM, it directly uses the high-level memory management interface for reading and writing device data. In addition, the BLAST team has enhanced the MFEM's memory manager capabilities by introducing the Umpire memory manager providing access to memory pools. This approach enables the following benefits: substantially reduces slowdowns caused by `cudaMalloc` performance; sharing of device memory buffers inside BLAST to reduce the total device usage; and sharing overall temporary memory between other external packages in MARBL that use Umpire.

Lagrangian phase: In this phase, the multi-material compressible Euler equations are solved on a moving curved mesh. The algorithm contains four major components, namely, inversion of a global CG mass matrix, computation of physics and material specific quadrature point data, computation of artificial viscosity coefficients, and application of a force operator.

The optimization of the mass and force operators has been aided by the matrix-free methods that were introduced by the CEED-developed Laghos miniapp, which models the main computational kernels of Lagrangian hydrodynamics, without the additional complication of physics-specific code. The GPU kernels for these methods were implemented by the BLAST team and reside in the BLAST code. The latest Laghos GPU implementations of these kernels give an alternative that might be used in the future, based on performance tests. A key CEED benefit provided to MARBL is the ability to drop in replacements for these expensive kernels as they become available.

Remesh phase: The mesh optimization phase of BLAST has several user selectable options; but the method of choice is based on the Target-Matrix Optimization Paradigm (TMOP), where the mesh optimization problem is posed as a variational minimization of a nonlinear functional. The development of the GPU port was performed in MFEM's mesh optimization miniapp, and then directly ported to BLAST, as both codes use the same core TMOP algorithms.

Remap phase: The remap algorithm in BLAST has two main components, namely, velocity remap, which is solved by a continuous Galerkin (CG) advection discretization, and remap of other fields, which is modeled by flux-limited discontinuous Galerkin (DG) advection.

The DG method is nonlinear, involving three separate components, namely, a high-order (HO) method, a low-order (LO) method, and a nonlinear flux-corrected transport (FCT) procedure. Currently, all of these three components require sparse matrices. Using the MFEM infrastructure, the MARBL developers have developed custom GPU code to populate the sparsity of the advection sparse matrix, thus achieving LO and HO implementations on the GPU. It is expected that this approach will be improved significantly by the future work in the CEED-developed Remhos mini-app, as it contains matrix-free methods to obtain LO and HO solutions.

The CG advection solve is also fully GPU ported, which includes the computation of quadrature data and application of the CG mass and advection matrices. Similarly to the CG mass matrix inversion in the Lagrangian phase, the remap GPU code is implemented inside MARBL, and the alternative to switching to the optimized MFEM kernels will be explored.

Next Steps

We will continue to explore platform- and node-dependent tuning. For example, significant performance gains in NekRS and libParanumal derive from covering communication with useful work. On Summit, reduced precision in the preconditioners is beneficial because it reduces pressure on injection bandwidth. On other platforms where more NICs are available, the precision reduction has minimal impact. Ultimately, implementations will need to be flexible and runtime configurable to maximize performance for large-scale applications. The CEED library development will continue to provide expanded support for such optimizations as dictated by forthcoming architectures and use-models.

8.6 ExaGraph

Combinatorial algorithms in general and graph algorithms in particular play a critical enabling role in numerous scientific applications. The irregular memory access nature of these algorithms makes them one of the hardest algorithmic kernels to implement on parallel systems. The team therefore proposes to develop methods and techniques for efficient implementation of key combinatorial (graph) algorithms chosen from a set of exascale applications.

There are three dimensions to the work: (i) exascale applications that drive the selection of combinatorial kernels and integration of software tools developed, such as computational biology, computational chemistry, and climate science; (ii) combinatorial (graph) kernels that play a crucial enabling role in the chosen application areas, such as graph traversals, graph matching, graph coloring, and graph clustering; and (iii) software

framework for efficient implementation on hierarchical distributed-memory architectures representative of potential exascale platforms, such as Zoltan2, KokkosKernels, CombBLAS, Vite and Ripples.

8.6.1 ExaGraph: Algorithms and Software Objectives

Through engagements with AD projects, work in the first year had focused on algorithmic development and scaling of several key graph algorithms. The algorithms were selected from the needs of the following ECP applications: ExaBiome (§ 6.3), NWChemEx (§ 3.2), ATDM, ExaWind (§ 4.1), ExaSGD (§ 6.1) and SuperLU/STRUMPACK. Key contributions from fiscal year 2020 were in the development of scalable protein similarity network construction, GPU-enabled graph partitioning, distributed multi-GPU implementation of graph clustering and influence maximization algorithms. To increase further engagement with the broader science community, the ExaGraph team collaborated with non-ECP Application Projects and included novel applications in computational biology and algebraic multigrid solvers, with goals to enhance these collaborations further in fiscal year 2021.

As demonstrated in the discussion below, significant strides were made in porting and scaling the algorithms on Summit for several graph problems. While we aim to continue optimization of the tools on Summit, we aim to focus on the porting of tools to exascale test beds in fiscal year 2021 along with integration with exascale applications.

8.6.2 ExaGraph: Performance Objectives

- Single-node performance: Develop efficient implementations for optimal performance on a single-node of future exascale architectures and target shared-memory parallelism using OpenMP and single GPU performance using CUDA and other technologies.
- Distributed performance: Design and develop distributed-memory implementations using MPI and OpenMP for the graph algorithms selected in this project.
- Multi-GPU implementation: Develop distributed multi-GPU implementations for graph clustering, graph partitioning and influence maximization algorithms.

8.6.3 ExaGraph: Co-design Engagements and Integration Points

- ExaBiome: Through a close working relationship with ExaBiome (§ 6.3), the ExaGraph team has developed new sparse matrix-matrix multiplication (SpGEMM) codes as well as a new distributed connected component algorithm (LACC) that together accelerate HipMCL (ExaBiome's protein clustering application). We also developed a distributed algorithm for generating protein similarity graphs [65], which is the input for HipMCL. This algorithm is packaged as part of the PASTIS software[6]. With ExaBiome, we have a joint milestone on end-to-end GPU acceleration of this protein clustering pipeline.
- ATDM/SNL. We have worked with the Empire team to analyze their load balance needs. We have implemented a graph partitioning model for the mesh. Preliminary results show a slight performance improvement using our graph partitioning instead of their previous geometric approach.
- ATDM/SNL math libraries: The MueLu algebraic multigrid (AMG) solver is used in several ECP applications, such as ExaWind and ATDM (Sparc, Empire). The team is working with the MueLu team to parallelize the AMG setup phase. Graph coloring can be used to find independent sets for coarsening; parallel shared-memory coloring has been developed in KokkosKernels for this purpose.
- ExaWind: The team is working with the ExaWind (§ 4.1) team at Sandia to design a parallel scheme for finite element assembly for unstructured (mixed) meshes. The current approach uses atomics and locks, but an alternative is graph coloring. ExaGraph is advising the application team on how to use Zoltan2 and KokkosKernels coloring, and to identify new application needs.

[6] https://github.com/PASSIONLab/PASTIS

Table 71: ExaGraph KPP-3 goals and metrics.

Passing value	Stretch value	Tentative present value
2	8	4

Table 72: ExaGraph code base.

Package name	**LOC**	**Target exascale challenge problems**	**Computational motifs**
CombBLAS	50,000	Protein clustering, factorization-based sparse solvers	Graphs implemented in sparse matrix algebra
Vite	12,534	Model reduction in power grid simulations, computational biology	Graph clustering
Matchbox	7081	Task assignment, algebraic multigrids, sparse direct solvers	Graph matching
Zoltan2/Sphynx	2000	Mesh and matrix partitioning, load balancing	Graph partitioning
PASTIS	13,000	Protein similarity construction	Sparse GEMM
KokkosKernels		FEM assembly, AMG setup	Graph coloring
Ripples	15,927	Computational biology and epidemic intervention	Influence maximization

- SuperLU/STRUMPACK: In addition to existing accomplishments on developing a heavy-weight perfect matching (HWPM) algorithm in collaboration [66], ExaGraph is working with the factorization-based sparse solvers team on two new fronts—a nested-dissection based sparse matrix ordering capability and a parallel symbolic sparse matrix factorization algorithm using GraphBLAS primitives that can run on GPUs.
- NWChemEx: The team is working with the NWChemEx team on a developing a novel load balancing algorithm based on submodular matching. The goal is to impact the Fock matrix construction step in NWChemx.
- Software Technologies: ExaGraph worked with the Pagoda team on developing a PGAS based implementation of half-approximate matching using UPC++ and demonstrated better performance and programmer productivity. ExaGraph also worked with the Metall team to integrate persistent memory tools in miniVite proxy application. The team has also started initial discussions with the SuperLU/STRUMPACK (Factorization-based preconditioners) ECP-ST project regarding parallel sparse matrix ordering techniques necessary for several applications. As an independent thrust, the team is also working with the same project to improve the symbolic factorization step using GraphBLAS primitives and accelerators.
- Benchmarking/Vendors: ExaGraph is working with vendors to port and optimize the proxy application on graph clustering targeting accelerator (GPU) platforms and memory systems.

8.6.4 ExaGraph: Progress on Early and Pre-Exascale Hardware

Performance on Summit

Multiple efforts were initiated by the ExaGraph team to design and develop efficient multi-GPU algorithms targeting Summit. Significant speedups were observed for many of a majority of the efforts. We summarize the progress on Summit in the following narrative.

- **Protein Network Construction:** Given a large collection of proteins, the objective is to identify groups of similar proteins by constructing a similarity network over protein sequences and the clustering

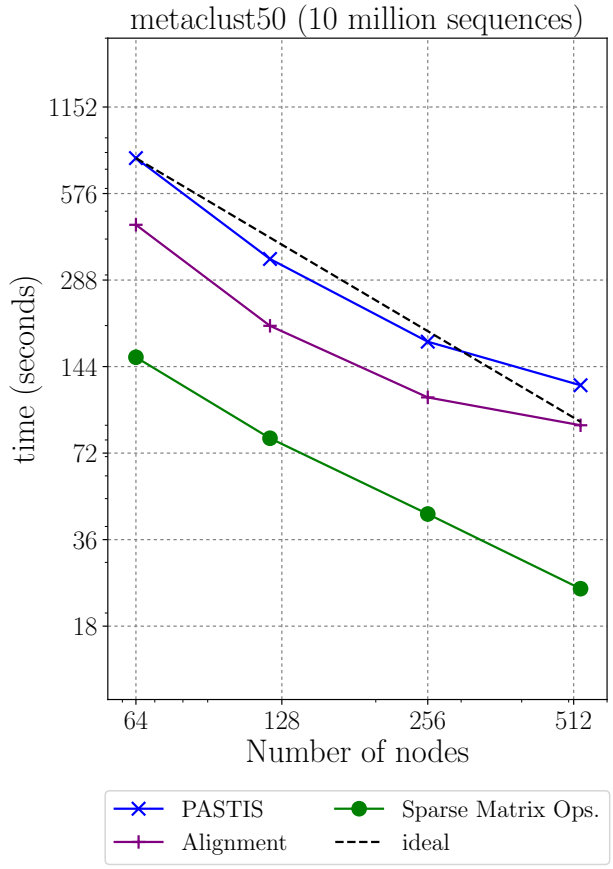


Figure 94: Strong scaling of GPU-accelerated PASTIS on Summit, running on the 10 million subset of metaclust50 dataset.

this network to discover possible protein families. We developed PASTIS (Protein Alignment via Sparse Matrices), a fully distributed protein similarity construction library using CombBLAS sparse matrix support. The current design will enable the replacement of sparse linear algebra kernels using vendor-optimized libraries such as cuSPARSE, ROCm and MKL. The preliminary scalability and performance of GPU-enabled PASTIS is shown in Figure 94.

- **Graph Partitioning:** Graph partitioning is a successful approach to load balancing that is essential to numerous scientific computing tasks. Given the absence of GPU-enabled tools for graph partitioning, ExaGraph developed Sphynx [67] as a distributed-memory accelerator-enabled, and portable tool. Sphynx uses a spectral partitioning method (Fig. 96). It builds on linear-algebra kernels towards this end and consequently benefits from the well-established and portable libraries such as Trilinos. We demonstrated good scalability on up to 96 GPUs as well as competitive performance relative to ParMetis, a popular graph partitioning tool (which only runs on CPU). We observed up to 970× speedup for Sphynx on GPU versus ParMetis on CPU on irregular graphs.

- **Influence Maximization:** The objective of the Influence Maximization problem is to identify the k-most influential vertices in a graph for a given diffusion model. Since the problem is a submodular optimization problem, efficient greedy algorithms are available that guarantees approximate solutions. ExaGraph team has developed shared-memory, distributed-memory and multi-GPU implementations for influence maximization using state-of-the-art algorithms. We have demonstrated speedups of up to 790× over a state-of-the-art serial implementation for a distributed multi-GPU library, cuRipples, on Summit [68]. An illustration is provided in Fig. 95. We applied the algorithm to design intervention strategies to minimize spread in epidemic outbreaks [69].

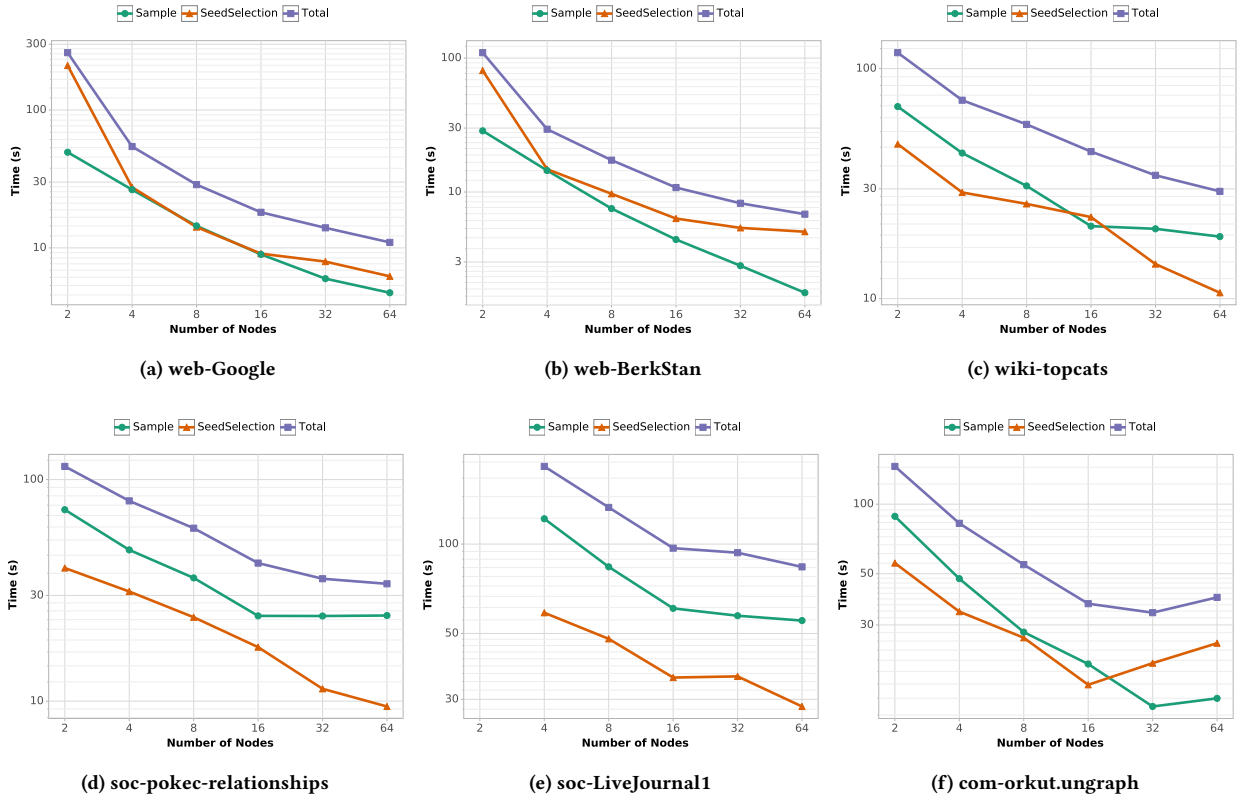


Figure 95: Scaling on Summit with the IC Model. Parameters: $\epsilon = 0.13$, $k = 100$.

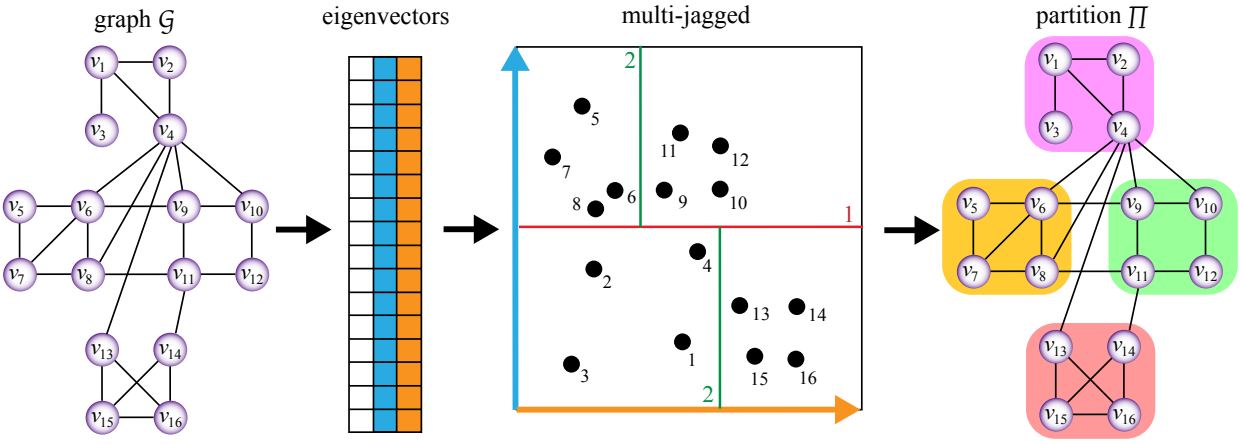


Figure 96: Flow of the spectral algorithm in Sphynx. This example partitions 16 vertices into four parts.

- **Graph Clustering:** Given a graph, graph clustering partitions the vertex set into clusters that are densely connected within a cluster, but sparsely so to the rest of the graph. While it is similar to the graph partitioning problem in concept, it is a relatively simpler optimization problem, and efficient heuristics exist. Using the concept of modularity optimization through the Louvain algorithm as the serial template, the ExaGraph team has developed shared-memory, distributed-memory and multi-GPU implementations for graph clustering. We demonstrate speedup of up to 19× relative to state-of-the-art single-GPU implementation from NVIDIA (cuGraph).

8.7 ExaLearn

The ExaLearn co-design center was funded as of FY18/Q4, and is designed to leverage the revolution in what is variously termed machine learning, statistical learning, computational learning, and artificial intelligence (henceforth referred to as ML). New ML technologies can have profound implications for computational and experimental science and engineering and thus for the exascale computing systems that DOE is developing to support those disciplines. Not only do these learning technologies open up exciting opportunities for scientific discovery on exascale systems, they also appear poised to have important implications for the design and use of the same exascale computers themselves: HPC for ML and ML for HPC.

ExaLearn has provided exascale machine learning software for use by the ECP Applications projects, other ECP co-design centers and DOE experimental facilities and leadership class computing facilities. Working closely with ECP applications, ExaLearn has undertaken a focused co-design process that targets learning methods common across these applications. This includes deep neural networks of various types (RNNs, CNNs, GANs,VAEs, etc.), kernel and tensor methods, decision trees, ensemble methods, graphical models and reinforcement learning methods. ExaLearn has engaged directly with developers of the ECP hardware, system software, programming models, learning algorithms, and applications to understand and guide tradeoffs in the development of exascale systems, applications, and software frameworks,

ExaLearn has identified the fundamental ML challenges associated with ECP and concentrated efforts on the development of scalable ML technologies for the analysis of data generated by exascale applications and DOE user facilities as well as to guide the optimal selection and steering of (1) complex computer simulations (e.g., current exascale application projects), and (2) experiments (e.g., at DOE facilities including light sources and accelerators). Key to success in this endeavor has been a deliberate focus on verification and validation and uncertainty quantification with a solid determination of generalization errors. A unifying principle is that of using exascale ML to improve the efficiency and effectiveness both of DOE computing resources and experimental facilities.

The technical goals for ExaLearn are fourfold: (1) reduce the development risk of ML software for the ECP application teams by investigating crucial performance tradeoffs related to implementation and application of learning methods in science and engineering, (2) produce high-performance implementations of learning methods, (3) enable easy and efficient integration of those methods with applications, and (4) contribute to the co-design of effective exascale applications, software, and hardware.

The team envisions ExaLearn as the intersection of applications, learning methods, and exascale platforms, advancing understanding of the constraints, mappings, and configuration choices that determine their interactions. The success of ExaLearn will ultimately be evaluated with respect to four metrics: (1) use of ExaLearn tools and technologies by the ECP applications and DOE experimental facilities; (2) efficiency of learning methods on exascale computers; (3) improvements in scientific deliverables of applications; and (4) support of ExaLearn tools and technologies by hardware vendors.

8.7.1 ExaLearn: Algorithms and Software Objectives

The ExaLearn project is delivering state of the art machine learning and deep learning techniques available from the open-source community, as well as optimizing ones specific to the needs of the application pillars. To date, the team has leveraged industry standard frameworks, such as Tensorflow, PyTorch, LBANN, and artificial intelligence (AI) gym, and extended LBANN to support distributed 3D convolutions for the ExaSky (§ 5.2) application.

ExaLearn is leveraging an engagement with the CANDLE project to adapt the CANDLE workflow to a broader range of projects. Specifically, ExaLearn is using both the supervisor framework for hyper-parameter optimization and the general CANDLE model description framework for model portability. ExaLearn will

Table 73: ExaLearn KPP-3 goals and metrics.

Project	Passing value	Stretch value	Tentative present value
ExaSky	2	3	1
CANDLE	1	2	0
ExaAM	0	0	0
NWChemEx	0	0	0
E3SM	0	0	0
Pele-Combustion	0	0	0
Total	3	5	1

continue to deliver a series of software releases that combine multiple open-source packages that span the ML/DL (deep learning) spectrum to address the needs of the application pillars. These software releases will capture a coherent set of tools used to deliver scientific output for each milestone, including ExaLearn-specific optimizations that address particular scalability impediments. Upcoming ExaLearn releases will include: LBANN—a scalable deep learning toolkit for the ExaSky project; PyTorch—rapid prototyping of neural network models for Control, Design, and Inverse; Ai-Gym—reinforcement learning for control.

8.7.2 ExaLearn: Performance Objectives

The performance objectives of the ExaLearn project are to apply state of the machine learning and deep learning techniques and tools to problems of interest across the DOE. The performance objectives have been tailored to each application within the four application pillars: surrogates, control, inverse, and design. Within each of these applications areas the goal has been to enable new applications to take advantage of leadership class computing and eventually exascale computing to advance the state of art within the discipline. Broadly the goals for each engagement are to accelerate the development of new ML/DL models, targeting the productivity of the domain scientist. As such, the most consistent metric across applications will be the number of models that can be trained per unit of time, enabling broader and more rapid use of model exploration. For example, in the surrogate application pillar, the team has accelerated the training of the cosmoflow application using the LBANN framework to drive down the per-network training time, while increasing the data input size on which it is trained.

8.7.3 ExaLearn: Co-design Engagements and Integration Points

Initial engagements with AD are primarily with two projects: ExaSky (§ 5.2) and CANDLE (§ 6.2). Surrogate model development for cosmology is the focus with ExaSky. Application targets in this current year include: E3SM (§ 5.5), Pele-Combustion (§ 4.2), NWChemEx (§ 3.2), and ExaAM (§ 3.5). Additional work will be targeting RL methods for tokamak control.

8.7.4 ExaLearn: Progress on Early and Pre-Exascale Hardware

Performance on Summit (or Summit equivalent)

ExaLearn has demonstrated the ability to scale the training of deep neural networks on leadership class systems for three key applications: a molecular generator model for developing small molecule therapeutics for COVID-19, the CosmoFlow regression model from ExaSky, and a 3-D U-Net architecture that will have direct use for additional ExaSky applications. All of the work detailed here has been written up in the ExaLearn Software Toolkit v2.0 report and the companion Applications and COVID reports. Additionally, this work is highlighted in a pair of peer-reviewed publications.

As part of the ExaLearn pivot to COVID-19 driven research, we scaled the training of a character-based Wasserstein AutoEncoder (cWAE) to all of the Sierra supercomputer at LLNL. Using the LBANN scalable deep learning toolkit and investments from ExaLearn we were able to achieve a peak performance of 318 PFlops/s in mixed precision FP16-FP32 (accumulate), which corresponded to 17.1 % of the system

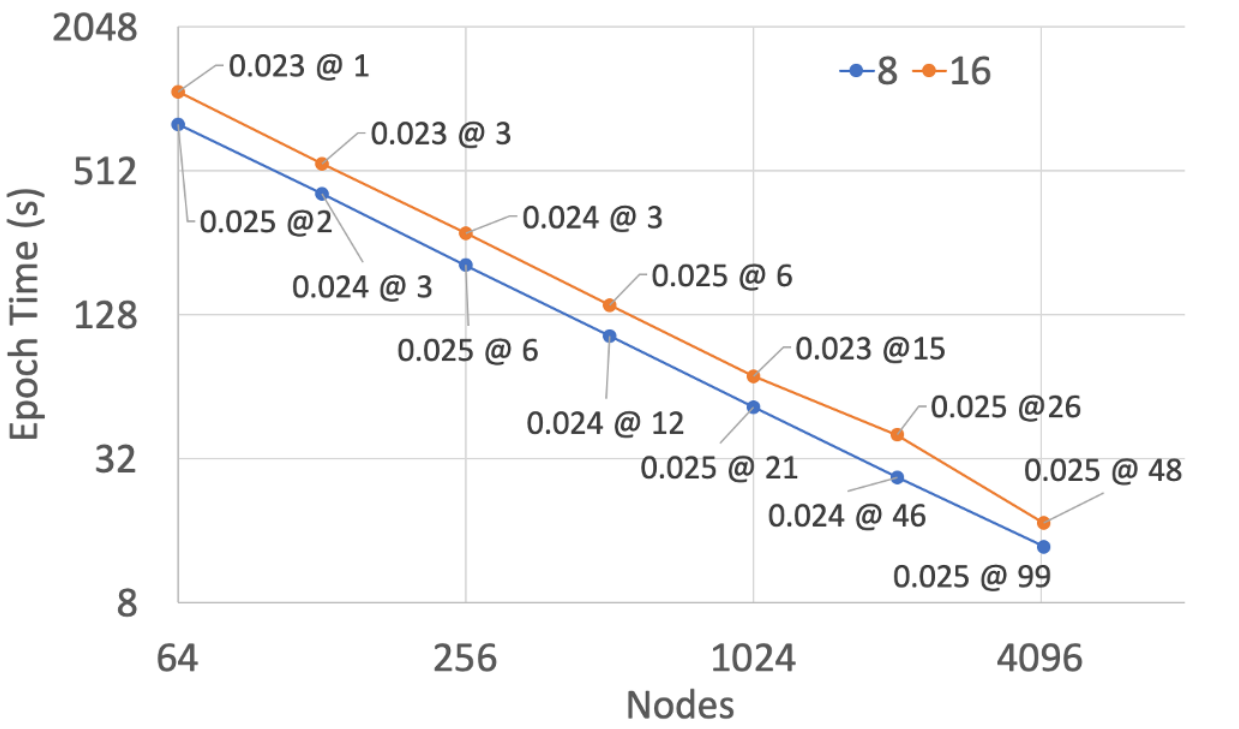


Figure 97: Strong scaling performance with eight and 16 GPUs per trainer and a mini-batch size of 4096. Data points are labeled with the number of epochs required to achieve an accuracy of 0.025 or lower.

peak. Additionally, we were able strong scale the training time and produce a trained model on 1.613 billion compounds in just 23 minutes, using 16,640 GPUs versus 55 minutes on 256 GPUs, a 2.39× speedup in time to solution. Furthermore, this represent a massive breakthrough when compared to the prior state of the art that required more than 24 hours to train a Junction-Tree Variational AutoEncoder model on only 1 million chemical compounds. This work was highlighted as a finalist for the ACM Gordon Bell Special Prize for High Performance Computing-Based COVID-19 Research.

To achieve these results we optimized multiple aspects of scalable training, primarily focusing on: single GPU performance, per-trainer GPU mini-batch size, number of GPUs per trainer, and the total number of trainers. These optimizations focused on reducing the impact of GPU kernel launch times, tuning the number of samples per GPU based on the limitations of the GPU device memory, and using mixed-precision training to avoid learning instabilities that appeared with native FP16 training. Overall, the collection of optimizations enabled us to accelerate the time to insight when working with complex generative models and massive data sets, and use Scalable HPC DL training to make neural network architecture design and debugging possible at human time scales. Figure 97 shows the reduction in training time to a fixed model accuracy as the number of compute resources is increased.

Additional work on creating the capability to train neural networks at scale and on data sample sizes that is previously unobtainable is showcased using the CosmoFlow model and a 3-D U-Net model. For both of these architectures we have developed techniques for spatial partitioning of the convolutional filters within the model, allowing us to strong scale the training using multiple GPUs per sample. Not only has this led to performance results such as a 147.3× speedup using 2048 GPUs versus 8 GPUs for the CosmoFlow model on the Sierra system, but also a 2× improvement in the quality of model as measured by the mean squared error by allowing the model to learn on entire 512^3 data samples rather than subsets of the data as shown in Figure 98. This work was also applied to a 3-D U-Net model that is able to perform reconstruction and segmentation tasks and is the next step for the engagement with ExaSky. For the 3-D U-Net, our work was able to achieve a 28.4× speedup using 1024 GPUs versus 32 GPUx for a model with a sample size of 256^3. This work also required the development of a scalable data ingestion pipeline and distributed in-memory

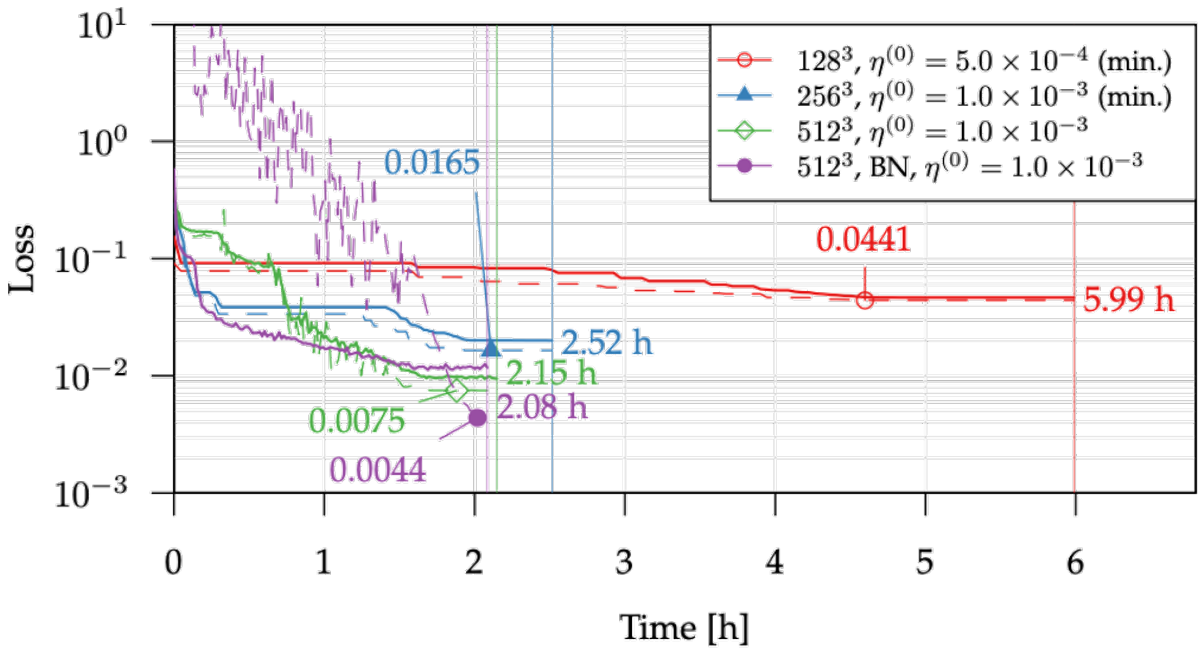


Figure 98: Strong scaling performance of the CosmoFlow network with 256 GPUs on Sierra with different data cube sizes.

data store that matched the spatial partitioning of the neural network architecture. Without these advances the model training is completely I/O bound, and with them, it is completely compute bound.

Similar work was also performed on a smaller version of the CosmoFlow model, that lacked spatial partitioning and worked only with smaller, down-sampled, data samples. This version reflected the newly adopted version of CosmoFlow that is now in the MLPerf-HPC benchmark suite. ExaLearn tuned the data ingestion of this version of the model to use a shared data ingestion on Summit to completely hide the data ingestion.

ACKNOWLEDGMENTS

This research was supported by the Exascale Computing Project, Project Number: 17-SC-20-SC, a collaborative effort of two DOE organizations—the Office of Science and the National Nuclear Security Administration—responsible for the planning and preparation of a capable exascale ecosystem—including software, applications, hardware, advanced system engineering, and early testbed platforms—to support the nation's exascale computing imperative.

REFERENCES

[1] A. Siegel, E. Draeger, J. Deslippe, A. Dubey, T. Evans, T. Germann, and W. Hart, "Early Application Results on Pre-exascale Architecture with Analysis of Performance Challenges and Projections," en, Exascale Computing Project, Tech. Rep. WBS 2.2, Milestone PM-AD-1080, Mar. 2020, p. 186. [Online]. Available: https://www.exascaleproject.org/reports.

[2] A. J. Woss, J. J. Dudek, R. G. Edwards, C. E. Thomas, and D. J. Wilson, "Decays of an exotic 1^{-+} hybrid meson resonance in QCD," Sep. 2020. arXiv: 2009.10034 [hep-lat].

[3] R. C. Brower, M. Clark, D. Howarth, and E. S. Weinberg, "Multigrid for Chiral Lattice Fermions: Domain Wall," Apr. 2020. arXiv: 2004.07732 [hep-lat].

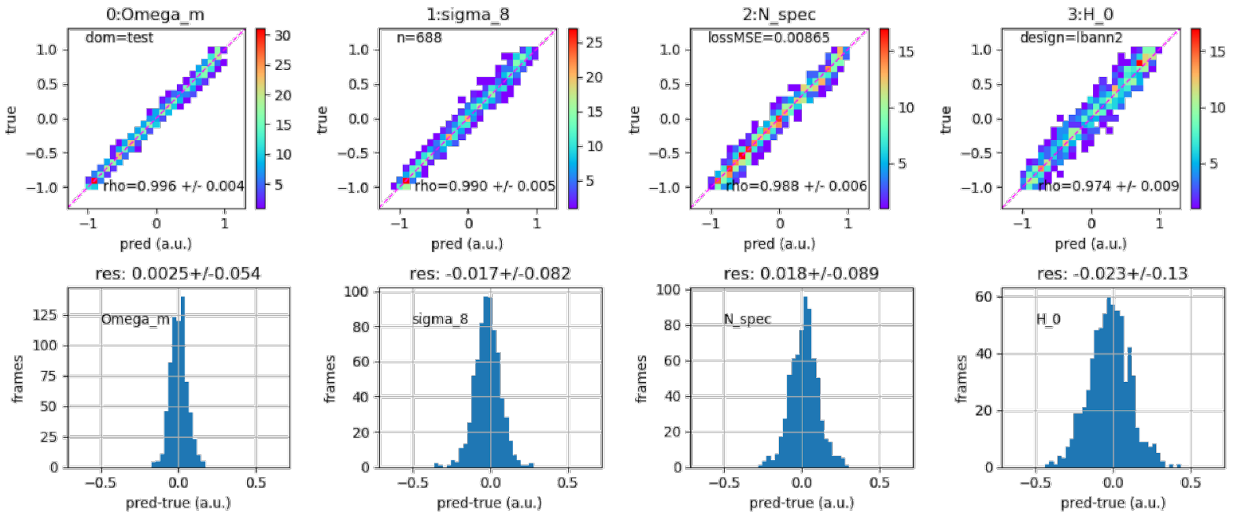


Figure 99: Results of training the CosmoFlow network using a new multiscale method of compressing the samples to work with only data parallel training algorithms.

[4] Y. Zhi, H. Shi, L. Mu, Y. Liu, D. Mei, D. M. Camaioni, and J. A. Lercher, "Dehydration pathways of 1-propanol on H-ZSM-5 in the presence and absence of water," *J. Am. Chem. Soc.*, vol. 137, pp. 15 781–15 794, 2015.

[5] R. Gayatri, S. Moore, E. Weinberg, N. Lubbers, S. Anderson, J. Deslippe, D. Perez, and A. P. Thompson, *Rapid exploration of optimization strategies on advanced architectures using TestSNAP and LAMMPS*, 2020. arXiv: 2011.12875 [cs.DC].

[6] A. J. Trainer, C. K. Newman, and M. M. Francois, "Overview of the Tusas Code for Simulation of Dendritic Solidification," en, Los Alamos National Laboratory, Tech. Rep. LA–UR-16-20078, 1237211, Jan. 2016. [Online]. Available: http://www.osti.gov/servlets/purl/1237211/ (visited on 12/31/2016).

[7] S. Ghosh, C. K. Newman, and M. M. Francois, "Tusas: A fully implicit parallel approach for coupled nonlinear equations," *arXiv*, arXiv–2006, 2020.

[8] *The Trilinos Project*, 2020. [Online]. Available: https://trilinos.github.io.

[9] D. Pekurovsky, "P3DFFT: A Framework for Parallel Computations of Fourier Transforms in Three Dimensions," en, *SIAM Journal on Scientific Computing*, vol. 34, no. 4, pp. C192–C209, Jan. 2012, ISSN: 1064-8275, 1095-7197. DOI: 10.1137/11082748X. (visited on 12/31/2016).

[10] M. Frigo and S. G. Johnson, "The design and implementation of FFTW3," *Proceedings of the IEEE*, vol. 93, no. 2, pp. 216–231, 2005, Special issue on "Program Generation, Optimization, and Platform Adaptation".

[11] A. Ayala, S. Tomov, A. Haidar, and J. Dongarra, "heFFTe: Highly efficient FFT for exascale," in *Computational Science—ICCS 2020*, V. V. Krzhizhanovskaya, G. Závodszky, M. H. Lees, J. J. Dongarra, P. M. A. Sloot, S. Brissos, and J. Teixeira, Eds., Cham: Springer International Publishing, 2020, pp. 262–275, ISBN: 978-3-030-50371-0.

[12] H. C. Edwards, C. R. Trott, and D. Sunderland, "Kokkos: Enabling manycore performance portability through polymorphic memory access patterns," *Journal of Parallel and Distributed Computing*, vol. 74, no. 12, pp. 3202–3216, 2014, Special Issue: Domain-Specific Languages and High-Level Frameworks for High-Performance Computing, ISSN: 0743-7315. DOI: 10.1016/j.jpdc.2014.07.003.

[13] A. Gholami, J. Hill, D. Malhotra, and G. Biros, "AccFFT: A library for distributed-memory FFT on CPU and GPU architectures," *CoRR*, vol. abs/1506.07933, 2015. arXiv: 1506.07933. [Online]. Available: http://arxiv.org/abs/1506.07933.

[14] *Nvidia cuda toolkit v11.0.3*, 2020. [Online]. Available: https://docs.nvidia.com/cuda/cuda-toolkit-release-notes/.

[15] N. R. Barton, R. A. Carson, S. R. Wopschall, and U. N. N. S. Administration, *ECMech*, Dec. 2018. DOI: 10.11578/dc.20190809.2. [Online]. Available: https://github.com/LLNL/ExaCMech.

[16] R. A. Carson, S. R. Wopschall, and J. A. Bramwell, *ExaConstit*, Aug. 2019. DOI: 10.11578/dc.20191024.2. [Online]. Available: https://github.com/LLNL/ExaConstit.

[17] *MFEM: Modular finite element methods library*. DOI: 10.11578/dc.20171025.1248. [Online]. Available: http://mfem.org/.

[18] R. D. Hornung and J. A. Keasler, "The RAJA Portability Layer: Overview and Status," en, Lawrence Livermore National Laboratory, Tech. Rep. LLNL-TR–661403, 1169830, Sep. 2014. [Online]. Available: http://www.osti.gov/servlets/purl/1169830/ (visited on 12/31/2016).

[19] A. K. Gupta and B. Mohraz, "A method of computing numerically integrated stiffness matrices," *International Journal for Numerical Methods in Engineering*, vol. 5, no. 1, pp. 83–89, Sep. 1972, ISSN: 1097-0207. DOI: 10.1002/nme.1620050108.

[20] A. K. Gupta, "Efficient numerical integration of element stiffness matrices," *International Journal for Numerical Methods in Engineering*, vol. 19, no. 9, pp. 1410–1413, Sep. 1983, ISSN: 1097-0207. DOI: 10.1002/nme.1620190910.

[21] R. Anderson, J. Andrej, A. Barker, J. Bramwell, J.-S. Camier, J. Cerveny, V. Dobrev, Y. Dudouit, A. Fisher, T. Kolev, and et al., "Mfem: A modular finite element methods library," *Computers 'I&' Mathematics with Applications*, Jul. 2020, ISSN: 0898-1221. DOI: 10.1016/j.camwa.2020.06.009. [Online]. Available: http://dx.doi.org/10.1016/j.camwa.2020.06.009.

[22] K. Saritas, W. Ming, M.-H. Du, and F. A. Reboredo, "Excitation energies of localized correlated defects via quantum Monte Carlo: A case study of Mn^{4+}-doped phosphors," *The Journal of Physical Chemistry Letters*, vol. 10, no. 1, pp. 67–74, 2019. DOI: 10.1021/acs.jpclett.8b03015. eprint: https://doi.org/10.1021/acs.jpclett.8b03015. [Online]. Available: https://doi.org/10.1021/acs.jpclett.8b03015.

[23] H. Shin, Y. Luo, P. Ganesh, J. Balachandran, J. T. Krogel, P. R. C. Kent, A. Benali, and O. Heinonen, "Electronic properties of doped and defective NiO: A quantum Monte Carlo study," *Phys. Rev. Materials*, vol. 1, p. 073 603, 7 Dec. 2017. DOI: 10.1103/PhysRevMaterials.1.073603. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevMaterials.1.073603.

[24] *ITER*, 2019. [Online]. Available: https://www.iter.org.

[25] T. Esirkepov, "Exact charge conservation scheme for particle-in-cell simulation with an arbitrary form-factor," *Computer Physics Communications*, vol. 135, no. 2, pp. 144–153, 2001, ISSN: 0010-4655. DOI: 10.1016/S0010-4655(00)00228-9.

[26] D. Merrill and M. Garland, "Single-pass parallel prefix scan with decoupled lookback," NVIDIA Research, Tech. Rep. NVR2016-001, Mar. 2016. [Online]. Available: https://research.nvidia.com/sites/default/files/pubs/2016-03_Single-pass-Parallel-Prefix/nvr-2016-002.pdf.

[27] C. Yang, T. Kurth, and S. Williams, "Hierarchical roofline analysis for GPUs: Accelerating performance optimization for the nersc-9 perlmutter system," *Concurrency and Computation: Practice and Experience*, vol. 32, no. 20, e5547, 2020, e5547 cpe.5547. DOI: 10.1002/cpe.5547.

[28] A. Huebl, R. Lehe, J.-L. Vay, D. P. Grote, I. Sbalzarini, S. Kuschel, D. Sagan, F. Pérez, F. Koller, and M. Bussmann. "openPMD: A meta data standard for particle and mesh based data." (Nov. 2015), [Online]. Available: https://github.com/openPMD (visited on 2020).

[29] T. Sukhbold, S. E. Woosley, and A. Heger, "A high-resolution study of presupernova core structure," *The Astrophysical Journal*, vol. 860, no. 2, p. 93, Jun. 2018, ISSN: 1538-4357. DOI: 10.3847/1538-4357/aac2da. [Online]. Available: http://dx.doi.org/10.3847/1538-4357/aac2da.

[30] A. Menon and A. Heger, "the quest for blue supergiants: Binary merger models for the evolution of the progenitor of sn 1987a," *Monthly Notices of the Royal Astronomical Society*, ISSN: 1365-2966. DOI: 10.1093/mnras/stx818.

[31] A. W. Steiner, M. Hempel, and T. Fischer, "Core-collapse Supernova Equations of State Based on Neutron Star Observations," *The Astrophysical Journal*, vol. 774, 17, p. 17, Sep. 2013. DOI: 10.1088/0004-637X/774/1/17. arXiv: 1207.2184 [astro-ph.SR].

[32] J. D. Emberson, N. Frontiere, S. Habib, K. Heitmann, P. Larsen, H. Finkel, and A. Pope, "The borg cube simulation: Cosmological hydrodynamics with CRK-SPH," *The Astrophysical Journal*, vol. 877, no. 2, p. 85, May 2019, ISSN: 1538-4357. DOI: 10.3847/1538-4357/ab1b31. [Online]. Available: http://dx.doi.org/10.3847/1538-4357/ab1b31.

[33] N. Frontiere, C. D. Raskin, and J. M. Owen, "CRKSPH – a conservative reproducing kernel smoothed particle hydrodynamics scheme," *Journal of Computational Physics*, vol. 332, pp. 160–209, Mar. 2017, ISSN: 0021-9991. DOI: 10.1016/j.jcp.2016.12.004. [Online]. Available: http://dx.doi.org/10.1016/j.jcp.2016.12.004.

[34] S. Wiederhorn, "Subcritical crack growth," in *Concise Encyclopedia of Advanced Ceramic Materials*, R. BROOK, Ed., Oxford: Pergamon, 1991, pp. 461–466, ISBN: 978-0-08-034720-2. DOI: https://doi.org/10.1016/B978-0-08-034720-2.50126-X. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B978008034720250126X.

[35] S. D. Walsh, H. E. Mason, W. L. D. Frane, and S. A. Carroll, "Experimental calibration of a numerical model describing the alteration of cement/caprock interfaces by carbonated brine," *International Journal of Greenhouse Gas Control*, vol. 20, pp. 176–188, 2014), doi = doi:10.1016/j.ijggc.2014.01.004.

[36] M. Schanen, F. Gilbert, C. G. Petra, and M. Anitescu, "Toward multiperiod ac-based contingency constrained optimal power flow at large scale," in *2018 Power Systems Computation Conference (PSCC)*, IEEE, 2018, pp. 1–7.

[37] K. Świrydowicz, E. Darve, J. Maack, S. Regev, M. A. Saunders, S. J. Thomas, and S. Peleš, "Linear solvers for power grid optimization problems: A review of gpu-accelerated linear solvers," *Parallel Computing*, no. submitted, 2020.

[38] S. Chakrabarti, M. Kraning, E. Chu, R. Baldick, and S. Boyd, "Security constrained optimal power flow via proximal message passing," in *2014 Clemson University Power Systems Conference*, IEEE, 2014, pp. 1–8.

[39] S. Tomov, J. Dongarra, and M. Baboulin, "Towards dense linear algebra for hybrid GPU accelerated manycore systems," *Parallel Computing*, vol. 36, no. 5-6, pp. 232–240, Jun. 2010, ISSN: 0167-8191. DOI: 10.1016/j.parco.2009.12.005.

[40] C. G. Petra, "A memory-distributed quasi-newton solver for nonlinear programming problems with a small number of general constraints," *Journal of Parallel and Distributed Computing*, vol. 133, pp. 337–348, 2019.

[41] ——, "Hiop user guide," Lawrence Livermore National Laboratory, Tech. Rep. LLNL-SM-743591, 2020.

[42] S. G. Abhyankar, S. Peles, R. Rutherford, and A. Mancinelli, "Evaluation of ac optimal power flow on graphical processing units," in *Proceedings of IEEE Power and Energy Society General Meeting 2021*.

[43] D. A. Beckingsale, J. Burmark, R. Hornung, H. Jones, W. Killian, A. J. Kunen, O. Pearce, P. Robinson, B. S. Ryujin, and T. R. Scogland, "Raja: Portable performance for large-scale scientific applications," in *2019 IEEE/ACM International Workshop on Performance, Portability and Productivity in HPC (P3HPC)*, IEEE, 2019, pp. 71–81.

[44] D. A. Beckingsale, M. J. Mcfadden, J. P. Dahm, R. Pankajakshan, and R. D. Hornung, "Umpire: Application-focused management and coordination of complex hierarchical memory," *IBM Journal of Research and Development*, vol. 64, no. 3/4, pp. 00–1, 2019.

[45] ARPA-E. "Grid optimization competition." (2020), [Online]. Available: https://gocompetition.energy.gov/.

[46] S. Peleš, K. Perumalla, M. Alam, A. J. Mancinelli, R. C. Rutherford, J. Ryan, and C. G. Petra, "Porting the nonlinear optimization library hiop to accelerator-based hardware architectures," *Parallel Computing*, no. submitted, 2020.

[47] A. B. Birchfield, T. Xu, and T. J. Overbye, *Electric grid test case repository*, https://electricgrids.engr.tamu.edu/electric-grid-test-cases/, 2016.

[48] G. Guidi, O. Selvitopi, M. Ellis, L. Oliker, K. Yelick, and A. Buluc, "Parallel string graph construction and transitive reduction for de novo genome assembly," *arXiv preprint arXiv:2010.10055*, 2020.

[49] O. Selvitopi, M. T. Hussain, A. Azad, and A. Buluç, "Optimizing high performance markov clustering for pre-exascale architectures," *arXiv preprint arXiv:2002.10083*, 2020.

[50] A. Zeni, G. Guidi, M. Ellis, N. Ding, M. D. Santambrogio, S. Hofmeyr, A. Buluç, L. Oliker, and K. Yelick, "Logan: High-performance gpu-based x-drop long-read alignment," *arXiv preprint arXiv:2002.05200*, 2020.

[51] N. Swenson, A. S. Krishnapriyan, A. Buluc, D. Morozov, and K. Yelick, "Persgnn: Applying topological data analysis and geometric deep learning to structure-based protein function prediction," *arXiv preprint arXiv:2010.16027*, 2020.

[52] S. Hofmeyr, R. Egan, E. Georganas, A. C. Copeland, R. Riley, A. Clum, E. Eloe-Fadrosh, S. Roux, E. Goltsman, A. Buluç, D. Rokhsar, L. Oliker, and K. Yelick, "Terabase-scale metagenome coassembly with metahipmer," *Scientific reports*, vol. 10, no. 1, pp. 1–11, 2020.

[53] M. G. Awan, J. Deslippe, A. Buluc, O. Selvitopi, S. Hofmeyr, L. Oliker, and K. Yelick, "Adept: A domain independent sequence alignment strategy for gpu architectures," *BMC bioinformatics*, vol. 21, no. 1, pp. 1–29, 2020.

[54] M. Howard, A. Bradley, S. W. Bova, J. Overfelt, R. Wagnild, D. Dinzl, M. Hoemmen, and A. Klinvex, "Towards performance portability in a compressible CFD code," in *23rd AIAA Computational Fluid Dynamics Conference*. 2017. DOI: 10.2514/6.2017-4407. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.2017-4407.

[55] O. Aaziz, C. Vaughan, J. Cook, J. Cook, J. Kuehn, and D. Richards, "Fine-grained analysis of communication similarity between real and proxy applications," in *2019 IEEE/ACM Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS)*, 2019, pp. 93–102.

[56] R. Wagh and D. Anand, "Application of citation network analysis for improved similarity index estimation of legal case documents : A study," in *2017 IEEE International Conference on Current Trends in Advanced Computing (ICCTAC)*, 2017, pp. 1–5.

[57] M. D. Cao and X. Gao, "Combining contents and citations for scientific document classification," in *AI 2005: Advances in Artificial Intelligence*, S. Zhang and R. Jarvis, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 143–152, ISBN: 978-3-540-31652-7.

[58] R. Nallapati, D. McFarland, and C. Manning, "Topicflow model: Unsupervised learning of topic-specific influences of hyperlinked documents," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 2011, pp. 543–551.

[59] *Scientific data reductions benchmarks*, 2019. [Online]. Available: https://sdrbench.github.io.

[60] H. Lee, M. Turilli, S. Jha, D. Bhowmik, H. Ma, and A. Ramanathan, "DeepDriveMD: Deep-learning driven adaptive molecular simulations for protein folding," in *3rd Workshop on Deep Learning on Supercomputers*, IEEE, 2019, pp. 12–19.

[61] I. Yakushin, K. Mehta, J. Chen, M. Wolf, I. Foster, S. Klasky, and T. Munson, "Feature-preserving lossy compression for in situ data analysis," in *49th International Conference on Parallel Processing: Workshops*, Edmonton, AB, Canada: Association for Computing Machinery, 2020, pp. 1–9.

[62] K. Mehta, B. Allen, M. Wolf, J. Logan, E. Suchyta, J. Choi, K. Takahashi, I. Yakushin, T. Munson, I. Foster, and S. Klasky, "A codesign framework for online data analysis and reduction," in *IEEE/ACM Workflows in Support of Large-Scale Science*, 2019, pp. 11–20.

[63] E. Suchyta, S. Klasky, N. Podhoeszki, M. Wolf, A. Adesoji, C. S. Chang, J. Choi, P. E. Davis, J. Dominski, S. Éthier, I. Foster, K. Germaschewski, B. Geveci, K. A. Huck, C. Johnson, Q. Liu, J. Logan, K. Mehta, G. Merlo, S. V. Moore, T. Munson, M. Parashar, D. Pugmire, M. S. Shephard, P. Subedi, C. W. Smith, L. Wan, and S. Zhang, "The Exascale Framework for High Fidelity coupled Simulations (EFFIS): Enabling whole device modeling in fusion science," *to appear*, 2020.

[64] C. Kelly, S. Ha, K. Huck, H. Van Dam, L. Pouchard, G. Matyasfalvi, L. Tang, N. D'Imperio, W. Xu, S. Yoo, *et al.*, "Chimbuko: A workflow-level scalable performance trace analysis tool," in *ISAV'20 In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization*, 2020, pp. 15–19.

[65] O. Selvitopi, S. Ekanayake, G. Guidi, G. Pavlopoulos, A. Azad, and A. Buluç, "Distributed many-to-many protein sequence alignment using sparse matrices," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, 2020.

[66] A. Azad, A. Buluç, X. S. Li, X. Wang, and J. Langguth, "A distributed-memory algorithm for computing a heavy-weight perfect matching on bipartite graphs," *SIAM Journal on Scientific Computing*, vol. 42, no. 4, pp. C143–C168, 2020.

[67] S. Acer, E. G. Boman, and S. Rajamanickam, "SPHYNX: Spectral partitioning for hybrid and axelerator-enabled systems," in *2020 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, 2020, pp. 440–449.

[68] M. Minutoli, M. Drocco, M. Halappanavar, A. Tumeo, and A. Kalyanaraman, "Curipples: Influence maximization on multi-gpu systems," in *Proceedings of the 34th ACM International Conference on Supercomputing*, ser. ICS '20, Barcelona, Spain: Association for Computing Machinery, 2020, ISBN: 9781450379830. DOI: 10.1145/3392717.3392750. [Online]. Available: https://doi.org/10.1145/3392717.3392750.

[69] M. Minutoli, P. Sambaturu, M. Halappanavar, A. Tumeo, A. Kalyanaraman, and A. K. Vullikanti, "Preempt: Scalable epidemic interventions using submodular optimization on multi-gpu systems," in *2020 SC20: International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, Los Alamitos, CA, USA: IEEE Computer Society, Nov. 2020, pp. 765–779. DOI: 10.1109/SC41405.2020.00059. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/SC41405.2020.00059.

A. APPLICATION CODE SUMMARY

Table A.1 gives a brief summary of the codes used by AD Application projects, the primary languages and their strategy for utilizing the GPUs.

Table A.1: AD application codes.

Application project	Code	Main language	GPU programming model
ExaStar	FLASH	Fortran	OpenMP
ExaStar	CASTRO	Fortran, C++	OpenMP, OpenACC
EQSIM	SW4	C++	RAJA
ExaSky	HACC	C++	CUDA, OpenCL
ExaSky	CRK-HACC	C++	CUDA, OpenCL
ExaSky	Nyx	C++	AMReX
Subsurface	Chombo-Crunch	C++	PROTO, UPC++
Subsurface	GEOSX	C++	RAJA
E3SM-MMF	E3SM	Fortran	OpenACC, moving to OpenMP
Combustion-PELE	PeleC	C++	CUDA, OpenACC
Combustion-PELE	PeleLM	C++	CUDA, OpenACC
WarpX	WarpX + PICSAR	C++	AMReX abstractions
ExaSMR	Nek5000	Fortran	OpenACC
ExaSMR	NekRS	C++	libParanumal (OCCA)
ExaSMR	OpenMC	C++	OpenMP, OpenCL or SYCL
ExaSMR	Shift	C++	CUDA
WDMApp	GENE	Fortran	OpenMP
WDMApp	GEM	Fortran	OpenACC
WDMApp	XGC	C++	OpenMP, OpenACC
MFIX-Exa	MFIX-Exa	C++	AMReX abstractions
ExaWind	Nalu-Wind	C++	Kokkos
ExaWind	AMR-Wind	C++	AMReX abstractions
ExaWind	OpenFAST	Fortran 90	N/A
ExaBiome	MetaHipMer	C++	UPC++
ExaBiome	GOTTCHA	C++	OpenMP, HIP, SYCL
ExaBiome	HipMCL	C++	OpenMP, HIP, SYCL
ExaFEL	M-TIP	C++	CUDA, HIP, OpenCL
ExaFEL	PSANA	C++	Legion
CANDLE	CANDLE	Python	TensorFlow, PyTorch
ExaSGD	GridPACK	C++	
ExaSGD	PIPS	C++	RAJA or Kokkos
ExaSGD	StructJuMP	Julia	
QMCPACK	QMCPACK	C++	OpenMP
ExaAM	MEUMAPPS-SS	Fortran	OpenMP, OpenACC
ExaAM	ExaConstit	C++	MFEM
ExaAM	TruchasPBF	Fortran	AMReX
ExaAM	Diablo	Fortran	OpenMP
ExaAM	ExaCA	C++	Kokkos
NWChemEx	NWChemEx	C++	CUDA, Kokkos
LatticeQCD	Chroma	C++	Kokkos
LatticeQCD	CPS	C++	GRID library
LatticeQCD	MILC	C	GRID library
GAMESS	GAMESS	Fortran	libcchem, libaccint
GAMESS	libcchem	C++	libaccint
EXAALT	ParSplice	C++	N/A
EXAALT	LAMMPS	C++	Kokkos
EXAALT	SNAP	C++	Kokkos