Early Application Results on Pre-exascale Architecture with Analysis of Performance Challenges and Projections

WBS 2.2, Milestone PM-AD-1080

Andrew Siegel[1], Erik Draeger[2], Jack Deslippe[3], Anshu Dubey[1], Tom Evans[4], Tim Germann[5], 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 25, 2020

ECP Milestone Report
Early Application Results on Pre-exascale Architecture with Analysis of Performance Challenges and Projections
WBS 2.2, Milestone PM-AD-1080

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 25, 2020

REVISION LOG

Version	Creation Date	Description	Approval Date
1.0	March 25, 2020	Original	

EXECUTIVE SUMMARY

This Exascale Computing Project (ECP) Milestone Report summarizes the status of all 30 ECP Applications Development (AD) sub-projects at the end of FY19. In August and September of 2019, a comprehensive assessment of AD projects was conducted jointly by the ECP leadership and a team of external subject matter experts. Reviews took place in person over five days—two at the National Renewable Energy Laboratory and three at Argonne National Laboratory and the University of Chicago. The review committees were tasked with evaluating each sub-project's progress in porting their code(s) to current multi-GPU architectures considered precursors to 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 sub-project's current status, but also represents an unprecedentedly broad account of experiences porting large scientific applications to next-generation HPC architectures.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

AD Applications Development

ADT Alternating Digital Tree

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

CAAR Center for Accelerated Application Readiness

CC Coupled Cluster

CD Co-Design

CFD Computational Fluid Dynamics

CLR Chemical Looping Reactor

CM Chemistry and Materials Applications

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 Simulation

DOD Department of Defense

DOE Department of Energy

DOF Degrees of Freedom

EA Energy Applications

ECP Exascale Computing Project

EOS Equation of State

ESS Earth and Space Science Applications

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
JIT	Just-in-Time
KPP	Key Performance Parameter
LCLS	Linac Coherent Light Source
LES	Large Eddy Simulation
M&S	Modeling and Simulation
MC	Monte Carlo
MD	Molecular Dynamics
MHD	Magneto Hydrodynamics
ML	Machine Learning
MSN	Mesoporous Silica Nanoparticles
NASA	National Aeronautics and Space Administration
NCI	National Cancer Institute
NERSC	National Energy Research Scientific Computing Center
NESAP	NERSC Exascale Science Applications Program
NIF	National Ignition Facility
NIH	National Institutes of Health
NIST	National Institute of Standards and Technology
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
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

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 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 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 (science, energy, national security) and the missions of other agencies such as the National Institutes of Health (NIH), the National Science Foundation (NSF), the 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 one of partnership through provision of selected staff to development teams in need of expertise in computer and computational science, applied mathematics, and HPC.

To achieve these goals, AD includes six L3 Work Breakdown Structure (WBS) elements, each with multiple subprojects at L4. These are described in the following sections. Two KPP's, 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. The meaning and requirements for each KPP are given in Table 1.

Table 1: Key performance parameters 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 Figure of Merit improvement ≥ 50	100% of selected applications achieve Figure of Merit 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 2019 AD External Review

For the 2019 AD External Review the committee Subject Matter Experts (SMEs) were asked to evaluate each applications project based on the following *Charge Questions*:

1. Has the project adequately quantified "base" and "stretch" passing criteria for their challenge problem? For Key Performance Parameter (KPP)-1 applications, this should include the role of algorithmic vs. hardware improvements in achieving the Figure of Merit (FOM). For KPP-2 applications, this should include a discussion of plans to quantitatively demonstrate "efficient" use of an exascale resource.

2. Has the project made progress on pre-exascale hardware, such as Summit, Sierra, or any small accelerator based test systems? Projects should address the following issues:

   (a) What parts of the code specifically were ported to the accelerator, and what fraction of overall performance do they account for in the challenge problem?

(b) What programming model(s) was used?

(c) What single node speedup (if any) was achieved relative to the best performance on other classes of systems?

(d) What are the key bottlenecks, if any, to improving on-node performance, including plans for how to address them? For example, will there be a need to explore risky, fundamentally new algorithmic approaches, different mathematical formulations, or more fine tune for specific hardware features?

3. For KPP-1 projects, did the project sufficiently explain/justify their current FOM value in light of the answers above? For KPP-2 projects, did the project adequately explain/justify their current progress relative to their project mileposts?

4. Has the project adequately formulated 2020 milestones consistent with the discussion above, including key risks and plans for how to maintain flexibility to address them?

5. Has the project clearly identified expected critical dependencies on Software Technology (ST) and Co-Design (CD) developments, including the practices and processes used for collaboration and any contingency plans?

Alternatively, each co-design project was evaluated on the following:

1. Has the project adequately defined their KPP-3 integration metric "base" and "stretch" passing criteria?

2. Did the project sufficiently explain/justify their self-assessment of their current KPP-3 metric?

3. Has the project made progress on pre-exascale hardware, such as Summit, Sierra, or any small accelerator based test systems? Do they have a clear and realistic strategy for providing performance portability across multiple exascale architectures?

4. Is the project appropriately leveraging HPC software from ST and the broader community, and have they clearly identified any expected critical dependencies, including the practices and processes used for collaboration and any contingency plans?

5. Evaluate the software architecture relative to usability across multiple applications, both the targeted ECP applications and the broader set of potential applications for which this motif appears prominently. Does the software implement an appropriate set of APIs and workflows that facilitate integration into such applications? Do they have an appropriate process for engaging their ECP applications stakeholders?

6. Has the project adequately formulated 2020 milestones consistent with the discussion above, including key risks and plans for how to maintain flexibility to address them?

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 is the use of 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 both 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 2) and applications targeting KPP-2 (Table 3) are required to provide a detailed milestone plan that outlines all needed work to enable 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.

Table 2: 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 3: 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 Multi-Physics Code	LLNL	DOE NNSA
SNL ATDM	SPARC for Virtual Flight Testing and EMPIRE for Electromagnetic Plasma Physics	SNL	DOE NNSA

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 targeting 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 both internally and externally (to 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, a key concept is the performance baseline, which is a quantitative measure of an application FOM using 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 the OLCF such as Mira, Theta, Cori, and Titan—systems in the 10–20 PFLOP/s 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 targeting 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 are reviewed annually by AD leadership and external subject matter experts (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.

Because an exascale machine promises approximately 50× the theoretical FLOP/s rate as the fastest currently deployed machine, the ECP sets the minimum FOM improvement aggressively at a factor of 50. The ECP supports complex multi-physics codes that put great demand on various aspects of the system: I/O capacity and bandwidth, memory bandwidth, memory latency, i.e., not just floating-point instruction capacity and throughput. Many applications are not based on algorithms that can make perfect use of specialized hardware features. A key focus of ECP is not only improved use of hardware but also the development of innovative algorithms that can achieve the same accuracy more efficiently. In cases where 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 ECP in the computational science community. While risky, any projects that are successful in this approach have the possibility of 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 sub-project leads and iterated upon and vetted over a two-year period. This includes a panel of external subject matter experts, technical ECP leadership across the entire management team, experts at each of the three computing facilities (ALCF, NERSC, OLCF), as well as in ECP-wide meetings and previous project reviews. Furthermore, the KPP-1 definition, including threshold and objective targets, were modified from 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 of 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 of 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—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 require many key pieces falling into place, but they stand as critical definitions of 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. ECP, therefore, 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 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 targeting KPP-2 are required to define an exascale challenge problem that represents a significant capability advance in its area of interest to the DOE. These challenge problem targets are reviewed both 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 targeting KPP-1 is the amount of new capability that must be developed to enable execution of 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 mere prototypes). Without a well-defined starting point at the 10–20 PFLOP/s 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 targeting KPP-2 (Table 3) are required to provide a detailed milestone plan that outlines all needed work to enable 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.

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 may be limited and performance optimization may 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 make use of specialized exascale hardware features, such as accelerators; and (3) completion of all necessary physics and algorithmic capabilities to successfully carry out 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 three NNSA ECP application projects (WBS 2.2.5.01, 2.2.5.02, 2.2.5.03) therefore target the ECP's KPP-2 metric. Because the national security nature of the NNSA challenge problems requires 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–FY23 for the necessary physics and algorithmic capabilities needed to execute their exascale challenge problems. The rigor of this process ensures that successful completion of 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 targeting KPP-2 are working toward a significant advance in simulation capability (physics and numerical fidelity) in a relatively short time. As such, it is judged to be 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 concurred with by the ECP's DOE sponsors). Because the review and assessment criteria are slightly different for the NNSA applications, 2 out of the 3 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 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 both 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 their impact and use on their 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 the number of application teams that are relying on their software products. Two co-design centers are considered high-impact (AMReX and CEED) and will be assigned a weight of 2, one is considered medium impact (CoPA) and assigned a weight of 1, and three are considered lower impact (CODAR, ExaLearn and ExaGraph) and assigned a weight of 1/2. This goal is explicitly tracked and reported for satisfying KPP-3 requirements.

Verification of 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 where 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, both a passing score and a stretch goal on the number of applications that they will be integrated into have been defined. The KPP-3 threshold is defined to be 50% of the products meet or exceed 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. KPP-3 objective is 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 4) 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.

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,

Table 4: 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

which is one of the four fundamental forces of nature (strong, weak, electromagnetic, 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 is 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 past three decades, QCD computations have been a driver of and benefited from the spectacular advances in HPC. Computing at the exascale is essential to reach two decadal challenges of central importance to nuclear and high-energy physics, which are the challenge problems of focus for this project (LatticeQCD).

The advance to exascale capability over the coming decade offers exciting opportunities for groundbreaking discoveries in high-energy and nuclear physics. Exascale computing has the potential to realistically simulate the atomic nucleus and to discover the first harbingers of new laws of nature, revealing a deeper theory which 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 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 carried out with that gauge configuration.

HISQ : This benchmark (Table 5) carries out calculations needed to measure meson masses and decay constants. The benchmark problem measures the rate of generating a new gauge configuration using a molecular dynamics algorithm and the rate of making measurements on the gauge-field configuration.

DWF : As with the HISQ action, two figures of merit for the domain wall fermion component have been 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 using this ensemble. See Table 6 for problem specifications.

Table 5: 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	Molecular dynamics, sparse matrix solution, deflation.
Simulation details: problem size, complexity, geometry, etc.	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 1,000 molecular-dynamics 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, 1,500,000 socket-hours on the exascale machine are needed, where a performance of 100 GFLOP/s per socket is assumed. To run the demonstration calculation of only two molecular-dynamics time units, approximately 10 socket hours for gauge-field generation are required. For demonstration of a single measurement, approximately 100 socket hours are required.

Wilson-clover : The Clover benchmark has two components (Table 7). The first is the rate at which dynamical Clover fermion lattices can be generated using a molecular dynamics 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: Figure of Merit*

The base FOM will be the geometric mean of the three component FOMs, with the components corresponding to the three fermion types (HISQ, DWF, Clover-Wilson) described above. Each component FOM is calculated as the geometric mean of the gauge-generation FOM and the analysis FOM for that fermion type. Each of those 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, F represent the baseline and final system; a, b represent baseline and target problems; and t, f, and n represent the wall time, fraction of the system used (assuming benchmark run is part of a large ensemble), and complexity of the problem (FLOPs). After computing the FOM for each fermion type (HISQ, DWF, Clover) on Aurora and Frontier, the team will choose the better result for each fermion type and take the geometric mean of those three numbers to obtain the *base* KPP goal.

The baseline FOM is 9.56 per hour measured on up to 32k nodes on Mira. The current FOM measurement is 71.22 per hour on up to 128 nodes on Summit. These measurements yield an FOM of 7.45. The full set of node measurements are listed in Table 8.

3.1.3 *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×.

Table 6: LatticeQCD DWF challenge problem details.

Functional requirement	Minimum criteria
Physical phenomena and associated models	Study the decays of K, D and B mesons. Examine both simple single-hadron final states and more complex processes involving 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 bi-linear combinations of composite operators are renormalized non-perturbatively. Requires Lanzcos eigenvectors, deflation, all-2-all propagators, all-mode-averaging, open boundary conditions and Fourier acceleration.
Simulation details: problem size, complexity, geometry, etc.	The target lattice volume is $96^3 \times 384$ with a lattice spacing of $a = 0.055\,\text{fm}$. The Wilson gauge action and Mobius DWF would be used.
Demonstration calculation requirements	• Monte Carlo evolution for 5 time units of the physical mass, $96^3 \times 384$, $a = 0.55\,\text{fm}$ 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 hours for evolution and for 10 hours for measurements.

Table 7: 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 minimum grid size of $64^3 \times 128$ and lattice spacing of $0.091\,\text{fm}$. Use the hybrid Monte Carlo molecular dynamics algorithm and sparse matrix solutions. Analysis methods will use multiple right-hand side solvers.
Simulation details: problem size, complexity, geometry, etc.	Generate part of an isotropic Clover ensemble with 2 light physical quark masses and 1 strange quark on a lattice size of $96^3 \times 256$, and a lattice spacing of $0.06\,\text{fm}$. Specifically, will run 10 trajectories (Monte Carlo time units) to observe behavior and stability of the algorithm.
Demonstration calculation requirements	A fully equilibrated lattice is needed. This will require about 1,000 Monte Carlo 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 Monte Carlo time units are required, needing $\sim$3,000 Summit node hours. Will test on a range of machine sizes, but minimum is 256 Summit nodes. For the measurement tests, a range of partitions will be tested on and can scale to 2,500 Summit nodes. This test will require approximately 1,000 Summit node hours.

Table 8: Node counts per component for baseline and current FOM measurements.

	Baseline		Current	
Component	Machine	Node count	Machine	Node count
DWF gauge generation	Mira	8192	Summit	128
DWF analysis	Mira	32768	Mira	32768
HISQ gauge generation	Mira	12288	Summit	96
HISQ analysis	Mira	12288	Summit	108
Clover gauge generation	Titan	1024	Summit	32
Clover analysis	Titan	128	Summit	16

3.1.4 LatticeQCD: Progress Toward Advanced Architectures

GPU Strategy

Our challenge problems and our FOM represent the three Lattice QCD code bases in wide spread use, namely Chroma (Jefferson Laboratory), CPS (Columbia and Brookhaven), and MILC (MILC collaboration). The codes specialize in three of the most common treatments of quarks, namely the Wilson-clover formulation (Chroma), the domain-wall formulation (CPS), and the staggered formulation (MILC). We also support two minor Lattice QCD code bases, namely HotQCD (HotQCD collaboration) and QEX (Argonne). The HotQCD code is optimized for QCD thermodynamics and the QEX code is intended for exploration of a variety of lattice field theories besides QCD. The three major code bases support the QUDA library (NVIDIA) and some elements of the Grid code (Edinburgh University). The principal authors of QUDA are our NVIDIA colleagues.

Currently, QUDA runs only on NVIDIA GPUs. For a few years, now, QUDA has provided the performance-critical modules needed for our three major code bases to run efficiently on NVIDIA GPUs. The most critical modules are the sparse matrix solvers. Others include the calculation of forces needed in the hybrid molecular dynamics algorithm for generating gauge-field configurations. Particularly important recent developments are mulitgrid solvers for both Wilson-clover-quark and staggered-quark formulations.

Grid, on the other hand, was originally developed with the KNL-type multicore architecture in mind. However, it has been extensively reworked so that it now supports both NVIDIA GPUs and multicore architectures. For efficient running on Aurora and Frontier, we intend to port QUDA and Grid to their planned accelerators.

Progress to Date

Prior to the availability of Summit, We were already running efficiently on GPUs . When Summit first became available, we measured the performance of the QUDA multimass solvers for both domain-wall (CPS) and staggered (MILC) quark formulations. Results are plotted in Figs. 1 and 2. They compare CPU-only running with CPU-plus-GPU running.

When these performance measurements were made, the GPU-direct feature was unavailable, and several subsequent code optimizations had not been introduced. Thus, we were quite pleased with the GPU-plus-CPU speedup of a factor of 10, node-for-node, over the CPU-only calculation. The GPU speedup is approximately 10×.

The Chroma code gauge-configuration generation benchmark improved dramatically in going from Titan to Summit, thanks partly to the improved hardware, but mostly to algorithmic improvements, namely, the introduction of the multigrid solver and related tuning.

Our FOM gives an indication of progress on Summit. The FOM has two components for each of the three code bases, or six altogether. For each code the components benchmark the generation of a gauge-field configuration and a set of measurements performed on a gauge-field configuration. Our first FOM compares the Summit benchmark with the same benchmarks on Mira or Titan. The composite Summit FOM represents a factor of 7.45 greater rate of scientific output than the Mira/Titan FOM. Our goal for Aurora and Frontier is an improvement factor of more than 50.

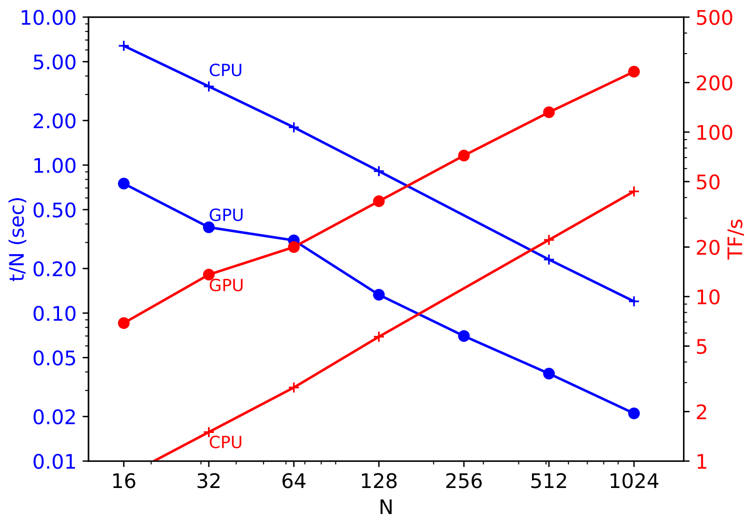


Figure 1: Weak scaling performance of the MILC/QUDA multimass HISQ solver showing (left scale, blue) time to solution divided by node count and (right scale, red) total teraflops/s vs. the node count with local volume 32^4 per GPU. Circles show GPU + CPU performance (6 MPI ranks per node, 7 CPU threads per rank) and plusses, CPU only (same ranks and threading). The GPU speedup is approximately 10×.

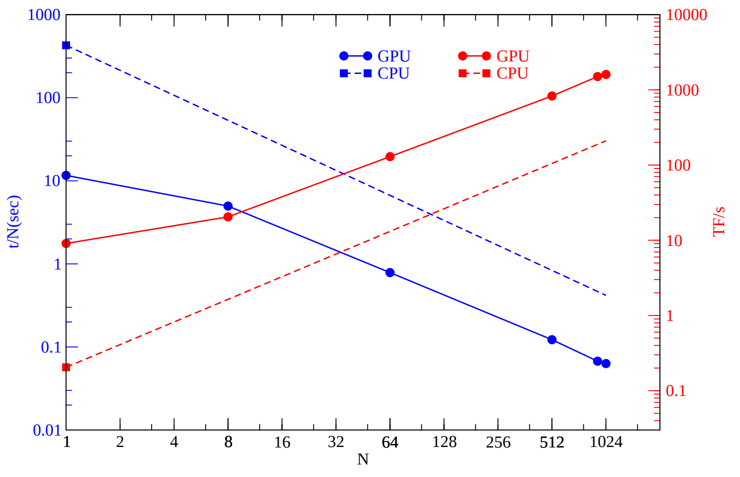


Figure 2: Weak scaling performance of the CPS/QUDA DWF solver for local volume $16 \times 12^3 \times 12$ per GPU, following the same plot layout as for MILC/QUDA above. Solid lines show GPU + CPU performance (6 MPI ranks per node, 28 threads per rank) and dashed lines, CPU only (same ranks and threading).

Since we ran this FOM in Spring 2019, there have been further improvements to QUDA and our code support for QUDA, including out ability to support mixed-precision solutions, which reduces communication loads. Further improvements include support for shared memory (NVSHMEM) and kernel fusion, which eliminates some of the MPI overhead and kernel latencies.

Next Steps

Our software P6 Activity this year is primarily devoted to porting QUDA and Grid to the planned Intel and AMD accelerators. If these efforts are successful, we expect both the CPS and MILC codes will then run reasonably efficiently. The developers of the Chroma code are following a different path, exploiting the existing Just-in-Time (JIT) capability of the Chroma code. The JIT kernels can either be converted to Kokkos, which will have SyCL and HIP back ends or they can be compiled to LLVM, which appears to be a common denominator code for NVIDIA, Intel Gen, and AMD accelerators.

We are experimenting with a variety of offloading methods, including SyCL, OpenMP-5.0, and Heterogeneous-compute Interface for Portability (HIP). For both QUDA and Grid, as with NVIDIA GPUs, we depend crucially on unified shared memory to ensure that C++ objects are offloaded properly without the need for dissecting them and naming by hand all the class members. Thus we hope the necessary compiler and hardware support is provided at the outset.

QUDA is currently being restructured to make it more amenable to porting. The QUDA port to AMD is expected to be reasonably straightforward. The port to the Intel Gen architecture via SyCL and data-parallel C++ is expected to be more involved. It will benefit from an effort to convert CUDA constructs to data-parallel C++, which will reduce the needed number of CUDA replacements.

The recent Grid port to NVIDIA GPUs uses code constructs that are intended to make it easy to adapt it to the Intel Gen and AMD accelerators. However, compiler deficiencies at present are preventing us from making progress with the full code, so we are carrying out offloading tests on simplified examples of the same coding strategy.

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 three 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, and the design of new functional materials.

3.2.1 NWChemEx: Science Challenge Problem Description

To guide the development of NWChemEx, the NWChemEx project is focusing on two interrelated target science problems 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 9.

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 3,500 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 9: 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: • In FY20, 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. These calculations are representative of Science Challenge #1 and will define an interim KPP-1 and FOM. • In FY23, 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 in the zeolite cavity. These calculations are representative of Science Challenge #2 and will define the final KPP-1 and FOM.
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: 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 as the benchmark molecular system for assessing the performance of NWChemEx on ORNL's pre-exascale Summit computer system. 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, explicitly correlated calculations on ubiquitin, a 1,231-atom molecule, it will not be feasible to run canonical CC calculations on this molecule. Since one of the goals in the first phase of the NWChemEx project is to validate the localized (DLPNO), explicitly correlated (R12/F12) 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.

In the latter stages of the NWChemEx project, the physical models implemented in the first phase will be optimized for the exascale computers expected to become available in FY22–FY23 and new physical models will be implemented to describe chemical reactions and molecular excited states.

The protocol for calculating the FOM is as follows:

- 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 previous canonical coupled cluster calculations with NWChem, these results will be used to define the Base FOM for NWChemEx. The Base FOM will largely represent the advances made in redesigning NWChemEx for exascale computers.

- After computing the Base FOM, a DLPNO-based, NWChemEx R12/F12 coupled cluster calculation will be run on the target molecule—currently, a minimum size of ubiquitin. Comparing the timing for

Table 10: Computational costs of the steps in targeted science challenge calculations.

Computational method	Computational costs
DFT	$a_{\mathrm{DFT}} n_{\mathrm{it}} N_{\mathrm{mo}}^4$
Integral transformation	$a_{\mathrm{int}} (N_{\mathrm{occ}} + N_{\mathrm{virt}})^5$
CCSD	$a_{\mathrm{CCSD}} n'_{\mathrm{it}} N_{\mathrm{occ}}^2 N_{\mathrm{virt}}^4$
(T) of CCSD(T)	$a_{(T)} N_{\mathrm{occ}}^3 N_{\mathrm{virt}}^4$

this calculation with that from the canonical CCSD(T) calculation will define an enhancement factor that quantifies the impact of the algorithmic improvements made in NWChemEx. Combining the two results yields an Enhanced FOM.

The ECP will be provided with both sets of numbers—the first as the Base FOM and the second as the Enhanced FOM.

There are two different series of benchmark molecules that the team proposes to use in the verification stages of the FOM. These molecules have different characteristics—protein molecules like ubiquitin are relatively compact molecules, while many catalytic molecules, like zeolites, are lattices with open pores. A comparison of these two types of molecules will lead to a better understanding of (1) the approximations made in the DLPNO coupled cluster calculations and (2) the dependence of the calculations on the size of the molecules in the two series.

BER Benchmark (Base) : As noted above, ubiquitin with 1,231 atoms is too big for a canonical coupled cluster calculation on even an exascale computer. To address this issue, a series of ubiquitin fragments have been identified that start with the molecule used to assess the performance of NWChem (DGRTL, 79 atoms) and end with ubiquitin.

BES Benchmark (Stretch) : A series of zeolite fragments that increase in size, finally terminating in the $Si_{96}O_{192}$ unit cell of H-ZSM-5 has also been identified. These fragments range from $Si_{10}O_{30}H_{20}$ to the unit cell (with or without terminating hydrogen atoms).

Details on the benchmark calculations and scaling : The CCSD(T) calculations on the targeted science challenges require multiple steps, each with different characteristics. The impact of these variations on the KPPs is reviewed in more detail below. A consequence of this variation is that the KPPs depend on a few parameters that relate to the specific details of the target calculations. These details will not be known until the final year of the project. Hence, there are some uncertainties associated with the calculation of the KPPs and therefore of the FOM. The estimated computational costs of the various components are summarized below.

In these expressions the factors a are prefactors that encapsulate cost factors that result from the specific implementations of the particular components as shown in Table 10. The factors n_{it} and n'_{it} refer to the number of iterations needed to converge the DFT and CCSD calculations, respectively. The variables N_{mo}, N_{occ}, and N_{virt} refer to the total number of orbitals, the number of occupied orbitals, and the number of virtual orbitals, respectively. These orbitals are those included in the CC calculations.

In practice all the variables depend on the particulars of the calculation. However, for a given molecular system and a given partitioning, the variables N_{mo}, N_{occ}, and N_{virt} are well defined. The other variables depend on the methods used to solve the equations. This causes some variability in the cost of a calculation even when the molecular system is fixed. This variability may be exacerbated by differences in the scalability of the different steps in the computation. Hence, if the KPP is defined as

$$\mathrm{KPP} = \frac{a_{\mathrm{DFT}} n_{\mathrm{it}} N_{\mathrm{mo}}^4 + a_{\mathrm{int}} \left(N_{\mathrm{occ}} + N_{\mathrm{virt}}\right)^5 + a_{\mathrm{CCSD}} n'_{\mathrm{it}} N_{\mathrm{occ}}^2 N_{\mathrm{virt}}^4 + a_{(T)} N_{\mathrm{occ}}^3 N_{\mathrm{virt}}^4}{t}, \quad (2)$$

where t is the time to solution, the overall result may be non-trivially affected by the variations in the different factors. The headline cost is determined by that of the (T) correction of the CCSD(T) method, although the

Table 11: 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
$\mathrm{KPP} \times t \times \left(N_{occ}^3 N_{virt}^4\right)$	1.9×10^{16}	4.5×10^{24}	2.4×10^{8}

Table 12: Titan timings for DGRTL for the CCSD part of the calculation.

Nodes	Cores	Time to solution (s)
128	2 048	6 653.7
256	4 096	4 528.6
512	8 192	3 919.9

CCSD must be performant due to the iterative nature of the computation. The team proposes to compute the FOM based on the CCSD(T) calculation:

$$\mathrm{KPP} \approx \frac{a_{\mathrm{int}}\left(N_{\mathrm{occ}}+N_{\mathrm{virt}}\right)^5 + a_{\mathrm{CCSD}}n'_{\mathrm{it}}N_{\mathrm{occ}}^2N_{\mathrm{virt}}^4 + a_{(T)}N_{\mathrm{occ}}^3N_{\mathrm{virt}}^4}{t}, \tag{3}$$

and

$$\mathrm{FOM} = \frac{\mathrm{KPP}_{\mathrm{NWChemEx}}}{\mathrm{KPP}_{\mathrm{NWChem}}}. \tag{4}$$

As noted above, one of the molecular benchmark series involves the protein ubiquitin. Since ubiquitin is far too large for canonical CCSD(T) calculations with NWChem on Titan (and even canonical NWChemEx on Summit), 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. Based on the relationships detailed in the previous subsection, 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 11, the DGRTL molecule is a 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.

Timings on DGRTL for the CCSD part of the CCSD(T) calculation on different fractions of Titan, ranging from 1/146 to 1/37 of the whole machine are show in Table 12. Extrapolating to the full machine (18,688 nodes) will require additional scaling information on the CCSD since the scaling is not simple for this computation.

CPU timings on DGRTL for the (T) part of the CCSD(T) calculation on different fractions of Titan, ranging from 1/16 to 1/4 of the whole machine, as well as the estimated timing for the full machine (18,688 nodes) are shown in Table 13. GPU Timings on DGRTL for the (T) part of the CCSD(T) calculation on different fractions of Titan are shown in Table 14.

The timings of the (T) correction for the CCSD(T) calculations on DGRTL as run on Titan are listed above. As the code achieves a super linear speed up from 37,376 to 74,752 cores for the CPU computations, there is no indication that the code is experiencing limitations in its scalability. Therefore, it seems reasonable

Table 13: Titan timings for DGRTL for the (T) part of the calculation.

Nodes	Cores	Time to solution (s)	
1 168	18 688	9 171.2	
2 336	37 376	5 210.5	
4 672	74 752	2 270.8	
18 688	299 008	567.7	(estimated)

Table 14: Titan timings using GPUs for DGRTL for the (T) part of the calculation.

Nodes	Cores/GPUs	Time to solution (s)
1 004	16 064/1 004	4 062
18 688	299 008/18 688	218 (estimated)

to extrapolate to the full machine, resulting in an estimated time to solution of 567.7 seconds. Combined with the volume of work of 1.9×10^{16}, this gives $\text{KPP}_{\text{NWChem}} = 3.3 \times 10^{13}/\text{s}$. Assuming that (1) the same code was used to calculate the (T) correction for ubiquitin on an exascale computer and (2) an FOM of 50 was achieved, 2.7×10^9 s or more than 86 years would be required to calculate this correction. The GPU extrapolation of the (T) correction for DGRTL on the full Titan machine only decreases the time by a factor of 2.6. This clearly shows that the targeted science challenge is of a complexity that requires exascale computing, and it also underscores the need for reduced order implementations of the CCSD(T) method to bring solutions within reach even on an exascale machine. This is the reason that reduced order implementations are one of the development targets of the NWChemEx project.

3.2.3 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 et al. [1] 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 $O(10^2\text{–}10^3)$ atoms using the high accuracy coupled cluster CCSD(T) method embedded in a DFT description of the larger zeolite environment of $O(10^4\text{–}10^5)$ 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.

Table 15: Timing for the (T) calculation of the DGRTL fragment using the 6-31G basis set using NWChem on Titan (with GPUs).

Nodes	Cores	Time to solution (s)
1 004	16 064	4 062
18 688	299 008	218 (estimated)

3.2.4 NWChemEx: Progress Towards Advanced Architectures

GPU Strategy

The Tensor Algebra for Many-body Methods (TAMM) computational infrastructure is NWChemEx's primary approach to obtaining high scientific productivity with portable performance. We anticipate that the implementation of many of the physical models will eventually be based on TAMM. TAMM takes high-level expressions describing computations on block-sparse tensors, decomposes these into a graph of dependent tasks that is then passed to a backend for scheduling and execution. High performance is obtained in the backend by focusing upon a small number (several tens) of kernels that are extensively optimized by the vendor or other providers (such as cuBLAS or TAL-SH), or by code generation plus auto-tuning, or by hand tuning. Currently these codes are written in CUDA, but future work will focus on models such as SYCL and HIP to enable the code on Aurora and Frontier. The TAMM API includes high-level expressions, including those for distributed, block-sparse tensors that will be used by the chemistry algorithms, thus ensuring that changes in TAMM will propagate throughout the NWChemEx code.

One class of algorithms that do not map as well to TAMM is the DFT module. There are two components to the DFT method: (1) the density fitting approximation for Coulomb operator and local density fitting approximation for the exact exchange operator; and (2) the numerical integration of the density functional. The numerical integration includes grid generation, weight computation, basis function evaluation, density (and its gradients) calculation, density functional evaluation through LibXC, and the XC potential assembly. All parts of the integration have been ported to GPU code using CUDA with the exception of the LibXC functional evaluations which are still in progress. However, the LibXC evaluation has the potential to make the biggest impact on the chemistry community since the LibXC project has agreed to work with the NWChemEx project to ensure that the GPU changes from the NWChemEx project will be incorporated into the public version of LibXC. OpenMP offloading of kernels is currently limited to a single target region and is too restrictive for our application needs.

While NWChemEx will use appropriate ST project software as delineated in the dependency maps for the project, the critical software technology for NWChemEx are BLAS, MPI, LAPACK, ScaLAPACK and SLATE (when it is available).

Progress to Date

The canonical CCSD and (T) computations have been implemented using TAMM and have been ported to the Summit GPUs using CUDA as the underlying GPU model. Multiple tests have been run using this code to examine correctness, performance and scalability. The cases below are representative of the testing to date.

Tables 15 and 16 and Fig. 3 show initial data to produce an FOM value of 27 using the DGRTL fragment that is representative of the proton transfer in membrane science challenge for the canonical (T) calculation. Table 15 shows data from the use of NWChem on Titan using the GPUs with an estimate of the time to solution for the full Titan machine. Table 16 shows the same calculation using NWChemEx on Summit using GPUs—again with a projection of the calculation to the full machine. Figure 3 plots shows additional data for the (T) using NWChemEx on Summit in a graphical form. We currently observe non-linear scaling partly due to the relatively small system size and also due to the fact that the new fused TensorGen kernels reduce the compute time significantly and the communication costs starts to affect the linear scaling property of the (T) calculation. With good communication overlap strategy, we expect to see closer to linear scaling with a large enough system size.

Table 16: Timing for the (T) calculation of the DGRTL fragment using the 6-31G basis set using NWChemEx on Summit (with GPUs).

Nodes	Cores	Time to solution (s)
220	9 680	112
4 600	202 400	8 (projected)

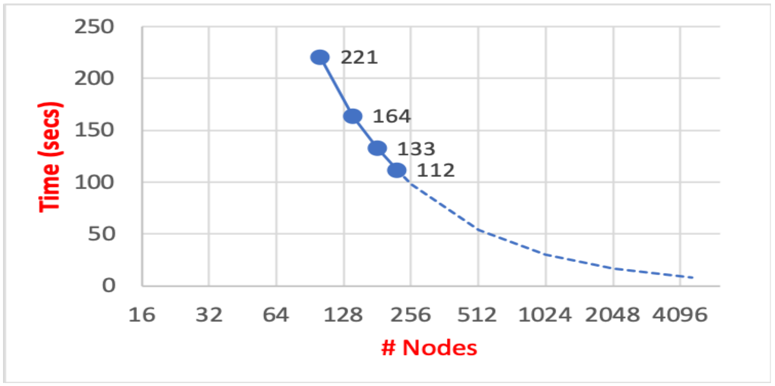


Figure 3: Time to solution versus number of nodes for the (T) calculation of the DGRTL fragment using the 6-31G basis set using NWChemEx on Summit (with GPUs).

Figure 4 shows the improvement of the canonical CCSD code on one node using improved kernel-level and communication optimizations for the DGRTL molecule using an STO-3G basis set on one node of Summit. The CPU code is using all CPUs on the node, while the GPU code is using the 6 GPUs. There is a 50 % speedup of the calculation using the GPUs. While this is fairly modest, the initial profiling has shown multiple pathways to improved performance.

Next Steps

Much of the next year will be devoted to developing the reduced scaling algorithms for the science challenge. In particular, we will be implementing the domain local, pair natural orbital (DLPNO) versions of the second order Moller-Plesset method, the coupled cluster cd single and double (CCSD) method and the approximate triples (T). However, the team will also be developing these algorithms on the systems available to us (Summit and JLSE) while planning for the future Frontier and Aurora architectures. Since these methods will rely on the TAMM infrastructure, porting and tuning activities associated with TAMM will benefit significant portions of the code.

The NWChemEx project has been integrating staff at the Oak Ridge Leadership Computing Facility (OLCF), Argonne Leadership Computing Facility (ALCF) and National Energy Research Scientific Computing Center (NERSC) into the project team to ensure that the software is portable and performant across multiple architectures. In addition, we work closely with the vendors to make sure our planning efforts are as informed as possible. The NWChemEx project recently held a meeting at Intel to help prepare for the Aurora software stack using the JSLE system. As part of this meeting, strategies for porting CUDA to SYCL were discussed and initial testing was undertaken. In addition, two members of the NWChemEx team attended the Frontier kick-off workshop to ensure that the development path of the project is consistent with the directions for the new architecture and associated software path forward.

3.3 GAMESS

Heterogeneous catalysis and the design of new catalysts is a grand challenge problem that will require the availability of exascale computers. In order to take full advantage of exascale architectures, it is critical

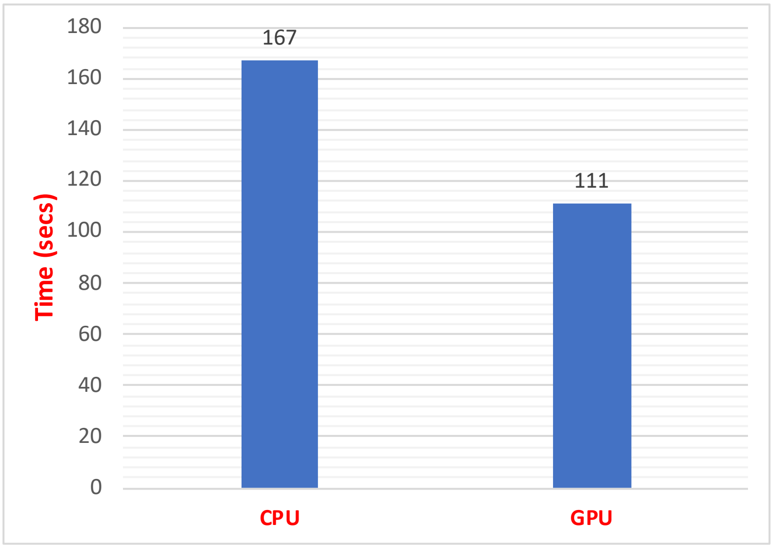


Figure 4: Single node performance using all CPUs versus 6 GPUs on Summit for a CCSD calculation of DGRTL with the STO-3G basis.

that application software be developed that is capable of exploiting multiple layers of parallelism and takes advantage of emerging low-power architectures that dramatically lower the energy/power cost without 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 quantum Monte Carlo. In order to attain exascale performance, GAMESS will be refactored to take advantage of modern computer hardware and software, and the capabilities of the C++ libcchem code that is co-developed with GAMESS will be greatly expanded. Concurrently, performance analyses will be done 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 developed will be brought to bear on the heterogeneous catalysis problem, specifically using Mesoporous Silica Nanoparticles (MSN), requiring thousands of atoms, as a template.

MSN 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 A to enter the pore, keeping undesirable species B from entering the pore. The presence of a solvent further complicates the problem. Accurate electronic structure calculations are needed to deduce the reaction mechanism(s), 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 like this with a small model, in the hope 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 both qualitatively and quantitatively.

3.3.1 *GAMESS: Science Challenge Problem Description*

The team will compute both energetics and dynamics on a model reaction with a representative MSN. An adequate representation of the MSN pore requires thousands of atoms with an appropriate basis set. For example, 5,000 heavy atoms with the aug-cc-pVTZ basis set requires 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 17.

The energy surface will be mapped via GAMESS calculations using the EFMO + RI-MP2 methodology, with refined calculations using the EFMO+CR-CC (2,3) coupled cluster approach or GAMESS EFMO +

Table 17: 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 either with the same level of theory or with EFP. Final energies will be captured using multi-level EFMO calculations, with either coupled cluster or QMC calculations for the reaction region and RI-MP2 elsewhere.
Numerical approach, algorithms	Configurations can be computed concurrently; each configuration will utilize 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 → Coupled Cluster / Quantum Monte Carlo (QMC).
Simulation details: problem size, complexity, geometry, etc.	At least 10,000 atoms, comprising the MSN pore, reactants, and solvents. An estimated one 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 hours for each energy + gradient RI-MP2 calculation.

QMCPACK Quantum Monte Carlo (QMC) approach (stretch goals) for more accurate reaction rates.

The pore selectivity dynamics will be computed with a Molecular Dynamics (MD) approach requiring approximately 10,000 energetics type calculations utilizing the GAMESS + FMO code.

3.3.2 GAMESS: KPP Stretch Goal

The base goal will be a system comprised of 1,738 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 both include 20–40 points (energy + gradient) on the potential energy surface.

In addition, 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: Progress Towards Advanced Architectures

GPU Strategy

GAMESS is a multi-functional electronic structure code. The main components for the ECP Challenge Problem are HF, second order perturbation theory (MP2), the resolution of the identity (RI) version of MP2, and CC theory. The workhorse is expected to be RI-MP2, with the possibility of using CC for very high accuracy in the “reaction region”. Both RI-MP2 and CC require HF as a starting point. We are pursuing two GPU strategies: (a) the continuing development of the libcchem C++ library which already has HF and RI-MP2 capability for both CPU and GPU and (b) direct offloading of GAMESS HF and RI-MP2 onto GPUs. Both are being actively pursued and both so far are promising. Since a central component of our overall ECP strategy is to use HF, RI-MP2 and CC within fragmentation methods (e.g., FMO and EFMO), path (a) will be enhanced by the new libfrag generalized fragmentation library, while path (b) will make direct use of the existing GAMESS FMO and EFMO codes.

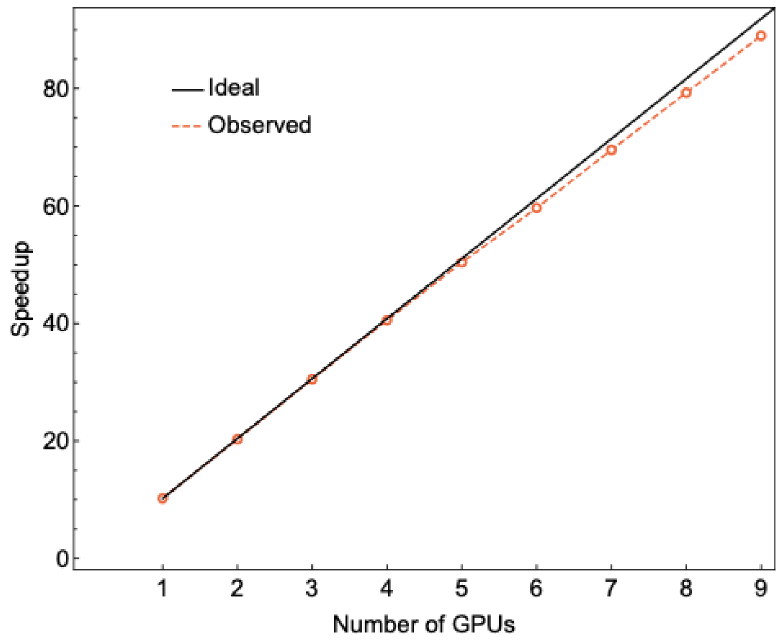


Figure 5: Speedup for LibAccInt on Summit.

Progress to Date

Currently implemented in Libcchem for GPUs are:

- One algorithm for computing the two-electron integrals (2-EI) that are a central bottleneck for HF calculations;
- Hartree-Fock and MP2 for closed shell molecules; and
- RI-MP2 energy + gradients.

Now we consider actual results. Shown in Fig. 5 is the single node speedup curve for LibAccInt run on Summit for a system of 150 water molecules. Each calculation used from 1–9 GPUs, each GPU being associated with a unique MPI rank. Speedup for a single GPU is 10× relative to a single CPU calculation using the LibAccInt CPU implementation. Single node speedup is ∼60×. The scaling with respect to the number of GPUs achieves ∼97 % efficiency.

For the generalized Fock build, a similar graph is shown in Fig. 6, based on the same system on Summit. Speedup for a single GPU is ∼2.5× relative to a single core calculation made with a previous version in Libcchem. Single node speedup is ∼14.5×. The overall 9 GPU speedup is ∼21.5×. The scaling with respect to the number of GPUs is ∼98 % of parallel efficiency.

For RI-MP2, an analogous graph is shown in Fig. 7, again on the water system on Summit. The overall speedup for the 66 GPU calculation, relative to a single CPU socket, is ∼798×. The single node speedup is ∼89×.

Next Steps

Work currently planned for GPUs are:

- Improved scaling of the algorithms mentioned above. This includes profiling all codes and developing proxy apps for the most important components.
- Implementation of alternative algorithms for 2-EI in order to optimize the efficiency of 2-EI. This is part of a broader development of a general integral library called LibAccInt and the development of the generalized Fock build (GFB). Improvements in both LibAccInt and GFB are planned.

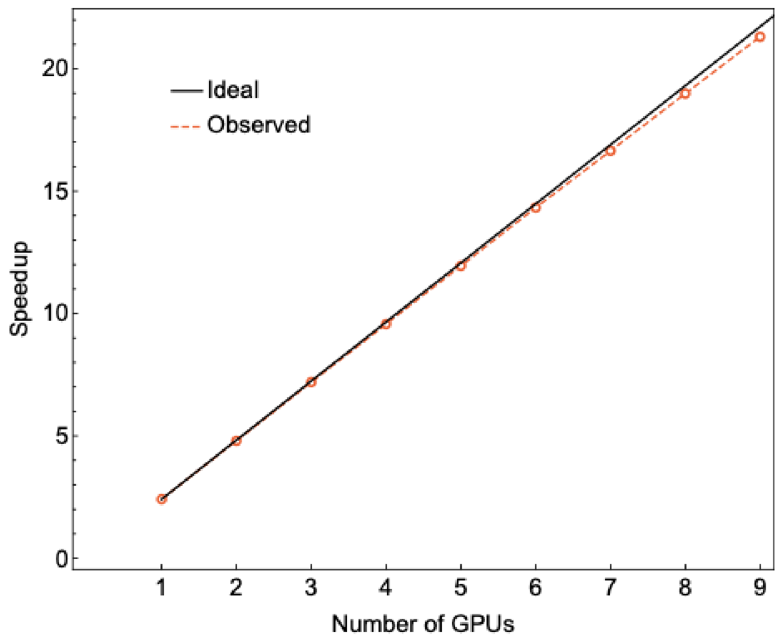


Figure 6: Speedup for a generalized Fock build.

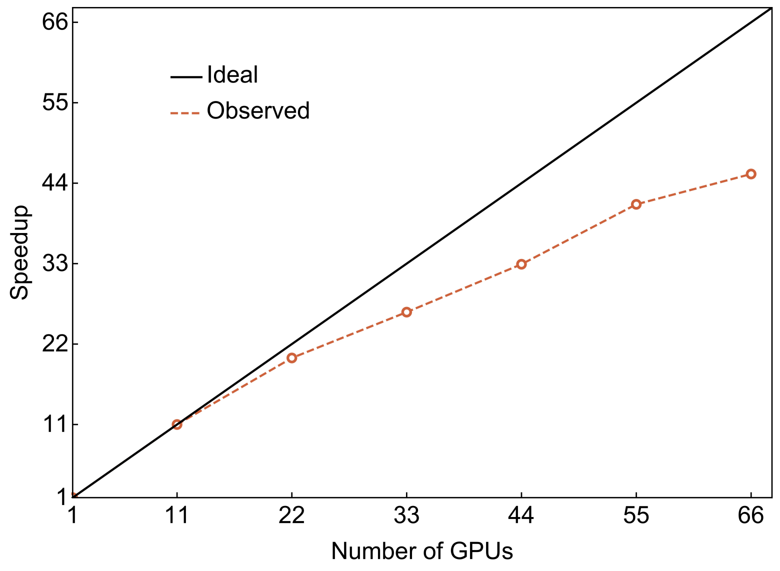


Figure 7: Speedup for RI-MPI2.

The implementation of CC methods and the RI-CC methods. The development of RI-CC methods is particularly important. Because CC methods (especially CR-CC(2,3)), are the highest level of theory that will be used in the challenge problem, and because CC methods are very computationally demanding (in terms of both computer time usage and memory usage) relative to methods like RI-MP2, deriving and implementing RI-CR-CC(2,) is essential since, like RI-MP2, applying the RI approach is expected to lose very little in accuracy while greatly reducing computational time and memory cost.

3.4 EXAALT

MD is a cornerstone of computational sciences. However, over and over again, MD is prevented from achieving complete scientific success by the inability to simultaneously reach the necessary length and time scales while maintaining sufficient accuracy. While 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 (ns) and accuracy (empirical potentials). As 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 be composed of 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 Molecular Dynamics (AMD) 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 large number of 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 potentials.

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 swelling of the fuel, pellet-clad interactions, and increased pressure on the clad. Because current burnup 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, containing up to four U 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+), the agreement with DFT + U calculations for static quantities is significantly improved. Unfortunately, this approach is incompatible with studying dynamics, as holes need to be able to redistribute as the geometry evolves. Further, 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, as this behavior forms the input to higher-level fuel evolution models.

Solving this grand challenge will require a significant advance in the ability to carry out high-accuracy, electronic structure–driven MD simulations on the timescales that are necessary to observe diffusion of defects while accounting for the changing polaron distribution. Given the size of these defects, relatively small systems (<1,000 atoms) are sufficient. However, given the high barriers for U-defect evolution ($\sim$2.5 eV), ms timescales will be required for the defects to move at the temperatures of interest (800–1300 K). This regime is inaccessible on present platforms. Indeed, 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 21st 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 containing helium. Experiments show that nanostructured fuzz, a nanoporous phase with tendrils on the order of tens of nm 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 about 100 eV. Such surface features will impact heat transfer and fuel retention, increase the rates of erosion through both 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 sub-surface 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 He finds its way deep below tendril surfaces. Additional questions arise beyond the obvious question of the fuzz formation mechanism: (i) what factors control the temperature dependence of the He accumulation and transition to fuzz formation? (ii) how does low-energy helium penetrate through a thick tungsten fuzz layer to reach the bulk? and (iii) what mechanisms control the continued growth of fuzz?

While sophisticated efforts leveraging 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. In order 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 ms with (relatively inexpensive) conventional potentials, in order to identify the nature of the mechanisms of roughening and early-stage fuzz growth and (2) 10^5 atoms over ms at higher accuracy (and ∼100–1,000× the computational cost per atom), in order to investigate the mechanisms that allow for transport of He along tendrils, while accurately accounting for competing kinetics (trapping, desorption, agglomeration, and bubble nucleation). The fact that current simulation capabilities fall short of this target by a factor of 1000× explains in great part 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 below.

3.4.1 EXAALT: Science Challenge Problem Description

Two exascale challenge problems are defined:

Fusion problem (Base) : The second target problem requires a dramatic extension of the reach of large-size long-time MD simulations. The team aims at simulating 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) : The first target problem requires a dramatic extension of the reach of long-time, high-accuracy MD simulations in order 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 18.

3.4.2 EXAALT: Figure of Merit

The FOM definition for EXAALT fusion problems is

$$\text{FOM} = \frac{N_t N_{\text{atoms}}}{t} \, , \tag{5}$$

where N_t is the number of timesteps and t is the wall clock time. The current baseline FOM measurement is 8,918,000 extrapolated to 49,000 nodes of Mira. The current FOM measurement is 169,831,000 from a simulation on 1,000 nodes of Summit. These yield a FOM of 781,222,600 extrapolated to 4,600 nodes on Summit.

3.4.3 EXAALT: KPP Stretch Goal

As discussed above, a 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 make efficient use of 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 timesteps/s), but the measure of success will be the deployment of a high quality computational framework. This will involve:

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

2. Making efficient use of 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, as many functions contribute to the run-time in the former case. Further, the relative cost of these different functions varies with the system size. This means that a large number of functions potentially have to be ported to the accelerators in order 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. Indeed, at each timestep, LATTE has to 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 further proceed. The failure rate in serial is very low, but in a ParSplice setting where the number of replicas can be very large (e.g., 10,000), which dramatically amplifies the overall failure rate. The team therefore has to develop extremely robust approaches that can keep the failure rates to extremely low levels.

The second stretch goal encompasses both of the 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, 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, at it requires generating and curating large sets of training and testing configurations, carrying out expensive DFT calculations on these configurations, carrying out 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 has to 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 model parameterization workflow discussed above can be executed at scale, dramatically cutting down 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 where 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.

Table 18: 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 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 using a tight-binding description of UO_2 containing 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, etc.	Describe size/scale/properties of physical system **Fusion problem:** The reference simulation consists of a Sublattice ParSplice simulation of a 10^5-atom system 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, 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 program, such as the new quadratic form or neural network extensions, may allow the team to ultimately achieve this same accuracy with fewer bispectrum components, and hence 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 containing an individual oxygen/uranium vacancy complex at the DFTB + U level of theory, using the LATTE backend. The stretch goal is to develop the computational capability to carry out such a simulation at the exascale 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. In order to obtain an accurate performance benchmark, on the order of fifty thousand timesteps 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 hours on the full exascale machine to demonstrate both challenge problems.

3.4.4 *EXAALT: Progress Towards Advanced Architectures*

GPU Strategy

EXAALT's KPP is defined for our fusion challenge problem which is to simulate a surface of tungsten in conditions typical of plasma-facing materials in fusion reactors. These simulations will use the Parallel Trajectory Splicing technique, as well as a hierarchy of parallelization levels (over coarse domain elements, over replicas, and over fine domain elements). The task management components of the calculations are very light, so will not have to be ported to the GPUs. The overwhelming majority of the flops in the calculations will be consumed carrying out molecular dynamics simulations on each worker process using the LAMMPS MD code and the SNAP model of interatomic interactions. SNAP is a new generation of machine-learned potential that promises high accuracy in exchange for a rather high computational cost. It is therefore critical to efficiently port the SNAP MD kernels to GPUs in order to achieve optimal performance. Our main approach is to rely on the Kokkos programming model, whose development is also supported by ECP. Kokkos promises portable performance over a wide range of architectures, including Aurora and Frontier. The MD code will be fully ported to Kokkos and will therefore be able to efficiently run on GPUs. In addition, in collaboration with the CoPA (§ 8.2) co-design center and the NERSC Exascale Science Applications Program (NESAP) program at the NERSC, we are developing a suite of SNAP proxy apps (TestSNAP) implemented using different programming models, including OpenMP, CUDA, and OpenACC. This will allow us the flexibility to assess the relative merits of the different approaches and insure we have a fall-back solution in place if the deployment of a production-quality Kokkos backend on Aurora and Frontier is delayed. For example, OpenMP will be supported by all upcoming machines and the CUDA version should be convertible to the HIP runtime API relatively easily.

Progress to Date

Rapid progress on the development of a high performance implementation of the SNAP kernels has been made over the last year, in preparation for early access to NERSC/Perlmutter and for upcoming exascale machines Aurora and Frontier. This work resulted from a close collaboration between EXAALT, NERSC (through the NESAP program), and CoPA.

The development of this new version proceeded by the extraction of a CPU SNAP proxy-app (TestSNAP) from the LAMMPS codebase, its rewrite following the discovery of an algorithmic trick that can reduce the number of executed flops, and the restructuring of its memory layout. These improvements yielded an increase in simulation throughput of roughly 2.4× in CPU performance on the P9s of Summit. This version of TestSNAP formed the basis on the new GPU implementation of the SNAP kernels. This new implementation proceeded from scratch, completely independently of the previous Kokkos implementation. Multiple versions were developed by the team, first using OpenACC, then CUDA, and finally OpenMP offload. TestSNAP was re-engineered throughout the year, resulting in a spectacular increase in performance during the summer of 2019, from about 1 katoms-steps/wall-clock second in April to 40 katoms-steps/s in July, as shown in Fig 8. At this point in time, the TestSNAP implementation was ported back to a production version of LAMMPS using Kokkos. This effort yielded an increase of 5.5× in simulation throughput on the V100 of Summit, as compared to the original Kokkos implementation that was used in production at the beginning of the year.

Since then, the Kokkos implementation has fully caught up with the latest CUDA version of TestSNAP, which has provided further performance increases of about 50 %, putting the production version roughly on par with the CUDA version of TestSNAP. As of the end of October 2019, the performance of the production Kokkos SNAP kernel now stands at 43.6 katoms-steps/s per V100 on Summit, and at 4.8 katoms-steps/s per P9 socket of Summit. Extrapolated to the whole machine, the production version of SNAP now delivers a roughly 134× increase in FOM with respect to the baseline on ALCF/Mira obtained at the beginning of the project. Runs at scale on Summit are projected to occur during Q1-Q2 of FY20.

Next Steps

Our strategy to prepare for runs on Aurora and Frontier relies on TestSNAP, our SNAP proxy app. Tests on currently available architectures have shown that the Kokkos implementation of TestSNAP performs at a similar level of performance as the other implementations, taking into account the subset of improvements that are implemented in each. Our strategy is to rely on OpenMP (on Aurora and Frontier) and/or HIP

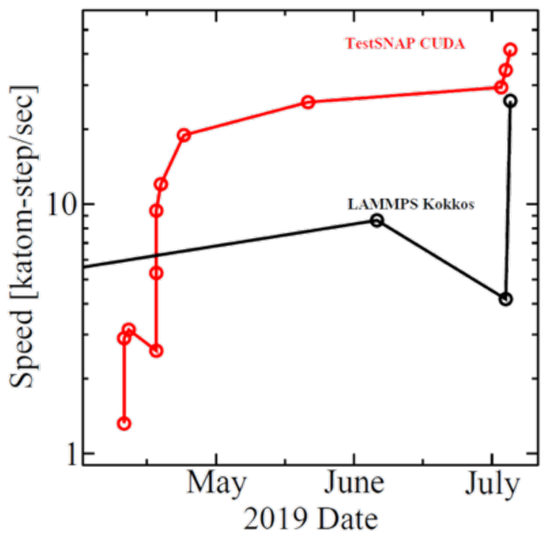


Figure 8: CUDA performance of TestSNAP.

(on Frontier) versions in order to assess the performance of the kernels on the latest hardware and software releases before full Kokkos support is available on these platforms. This will enable us to tweak and tune the kernels and to preemptively implement improvements into the Kokkos version, in anticipation of its full availability on the final hardware. This strategy will enable us to make steady progress towards developing efficient implementations of the main kernels for Aurora and Frontier without having to wait for Kokkos support first. We are working closely with NERSC through the NESAP program to make sure the kernels show high performance on the upcoming pre-exascale machine Perlmutter and are in initial contact with staff at ALCF in order to provide them with an OpenMP version of TestSNAP that can be used to monitor the performance on Aurora testbeds and preliminary software stacks. We will initiate similar contacts at OLCF in order to gradually adapt to Frontier's hardware and software stack.

3.5 ExaAM

ExaAM, the Exascale Additive Manufacturing (AM) project, is developing the Integrated Platform for Additive Manufacturing Simulation (IPAMS), 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 design of AM components with location-specific properties and acceleration of performance certification.

AM is revolutionizing manufacturing, allowing construction of complex parts not readily fabricated by traditional techniques. In addition, 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 (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 not focused on development of high-fidelity predictive models but instead on reduced-order models for industry use. ExaAM represents a unique opportunity to leverage DOE investments to address challenges that require exascale resources and, to the team's knowledge, 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, where each layer is approximately 50 µm. Hence, a part 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 spanning

Table 19: Computational simulation stages in an ExaAM simulation.

Stage	Exascale Simulation	Required computational capability	ExaAM component(s)
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 derived constitutive properties	Diablo

orders of magnitude determines the performance of the final manufactured part. These include heat transfer (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, the diameter, magnitude, speed of the energy source, etc. 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 key 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 past decades. Unfortunately, the simulation tools developed for welding and other similar processes, while calibrated and approaching predictive capability, 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 due to the fact that the process-structure-property-performance relationship is traditionally analyzed in an uncoupled manner, relying on tabular databases unable to adequately capture the implicit, dynamic, non-equilibrium nature of AM processes. During AM processing, the part and the material are built at the same time.

One of the goals of ExaAM is to remove those limitations by coupling high-fidelity mesoscale simulations within continuum process simulations to determine microstructure and properties using local conditions. Typically, thermo-mechanical 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 Monte Carlo, or 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.

A critical observation is that there is no single code able to capture all of the relevant physics required. However, several codes have been developed to simulate phenomena similar to those required for AM, both within the DOE complex and the broader computational science community. ExaAM is leveraging several of those existing capabilities—enhancing and extending as needed and developing new capabilities when necessary. These codes are components in a new software environment referred to as the Integrated Platform for Additive Manufacturing (IPAMS).

A full ExaAM simulation consists of five stages, as listed in Table 19. For the purposes of estimating computational resources, the focus is on only stages 1–3, since the cost of the stages 0 and 4 is relatively small

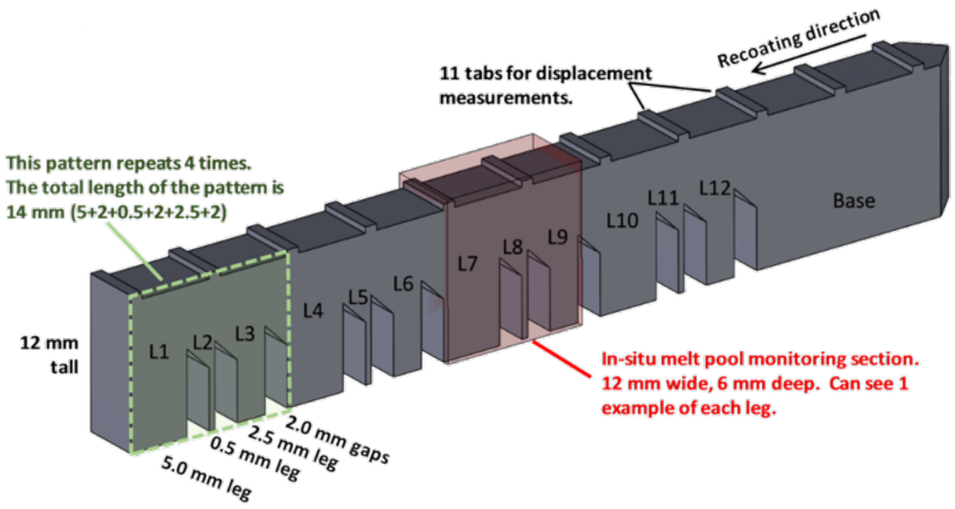

Recoating direction
11 tabs for displacement measurements.
This pattern repeats 4 times. The total length of the pattern is 14 mm (5+2+0.5+2+2.5+2)
12 mm tall
L1
L2
L3
L4
L5
L6
L7
L8
L9
L10
L11
L12
Base
In-situ melt pool monitoring section. 12 mm wide, 6 mm deep. Can see 1 example of each leg.
2.0 mm gaps
2.5 mm leg
0.5 mm leg
5.0 mm leg

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.

and can be performed on capacity HPC computational platforms. Additional capabilities required as either risk mitigation or to provide parameters for models above (e.g., sub-grain scale solidification microstructure evolution via phase field and OpenFOAM for melt pool simulation) are also omitted from the remaining discussion.

3.5.1 ExaAM: Science Challenge Problem Description

ExaAM will develop and deploy a collection of simulation capabilities for performing process-aware performance modeling of additively manufactured parts 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 AM-Bench web site [2].

The threshold (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. Note that stage 2 would be required for other materials such as IN718 or Haynes282, so stage 2 appears as a stretch science goal.

Tables 20 and 21 describe stages one and three in detail for the threshold challenge problem.

Notes:

- 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 (number of layers in Stage 1, number of RVEs in Stage 3, etc.), 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 22.

Table 20: ExaAM challenge problem details, 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, etc.	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 12.5M timesteps (0.5 µs timestep size) Microstructure evolution occurs on the same set of domains, but at 1 µm resolution and a similar timestep 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 hours. Execution on Frontier should require < 1 hour for a similar number of nodes.

Table 21: ExaAM challenge problem details, 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, etc.	Each representative volume element (RVE) is a $100 \times 100 \times 100$ µm domain containing approximately 1000 grains, at 1 µm resolution (1M 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 (RVEs) will be required, resulting in a total of 2000 independent ExaConstit simulations. Based on scoping simulations on Summit, each simulation will require approximately 1 hour on 2 Summit nodes (4000 nodes for 1 hour for all 2000 ExaConstit simulations). Execution on Frontier should require 10–15 minutes on a similar number of nodes.

Table 22: ExaAM science stretch goals.

Description	Motivation	Component(s)
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 AM process simulation	Topology and shape optimization require faster-running models for process simulation	TBD
Detailed solidification simulation (IN625, IN718, and/or Haynes282)	Sub-grain 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 above)	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 cool-down and annealing	ExaConstit
Powder-resolved process simulation	Capture details of melt pool behavior in order to optimize process parameters to minimize porosity due to keyholing, etc.	ExaMPM

3.5.3 ExaAM: Progress Towards Advanced Architectures

As noted earlier, a full ExaAM simulation consists of five stages, as listed in Table 19, with stages 1 and 3 required for the threshold challenge problem.

In the discussion that follows, we group the components into the following three categories:

- Components required for the threshold challenge problem
- Components required for stretch science goals
- Prototype and proxy app components

GPU Strategy

The GPU strategy for each of the ExaAM components is as follows.

Threshold Challenge Problem Components

- **Diablo**: Given that Diablo uses implicit time integration, with the linear solve typically being 60 % or more of the wall clock runtime, the primary strategy is to use GPU-enabled linear solvers. It is currently anticipated that this will be Hypre.
- **TruchasPBF**: Use GPU-enabled Hypre solvers for the implicit time integration preconditioners and linear systems (currently 80–90 % of execution time). Use OpenMP offloading with AMReX-managed GPU memory to form the systems in order to minimize host-GPU memory transfers.
- **ExaCA**: Use of Kokkos for offloading of calculations to GPU.
- **ExaConstit**: The material models within ExaCMech are using RAJA to offload calculations to the GPU. ExaConstit is being ported to the GPU through MFEM's v4.0 support of running the assembly operation using the GPU by formulating the assembly operation as a matrix-free operation using partial assembly. Within MFEM, other matrix multiplication and iterative solver operations are supported through Hypre's GPU capabilities.

Stretch Science Components

- **MEUMAPPS-SS**:
  1. Explore speedup using OpenACC in the current Fortran MPI version of MEUMAPPS-SS. Since this step is already completed and significant speed up has been achieved, we are currently extending this strategy to other MEUMAPPS-SS routines
  2. Explore GPU based Fast Fourier Transforms (FFTs) in the Fortran test problem and investigate potential for additional speed up (or not)
  3. Explore speedup using C++ version of the test problem and using all available GPU options and compare against the speed up achieved in 1 and 2 above, and make a decision whether or not to implement this option on the MEUMAPPS-SS routines.
- **AMPE**: Use CUDA-based Hypre (Struct PFMG) for preconditioning linear solvers ($\sim$60 % of wall clock time). Follow SAMRAI strategy for the rest: RAJA + UMPIRE.

Prototype and proxy app components

- **ExaMPM**: ExaMPM uses the CoPA Cabana toolset to deploy the MPM algorithm. Cabana builds on Kokkos and MPI and has been demonstrated to run a variety of accelerator-based HPC systems including Summit. The entire application is developed in this manner such that all code (if appropriate) is executed on the GPU. In addition, GPU-aware MPI is used for communication of data between GPUs. Linear solves are currently provided through Hypre.

- **Tusas**: In addition to utilizing GPU enabled solvers and preconditioners (Trilinos; Nox, Belos, MueLu) Tusas utilizes a combination Kokkos data structures and parallel dispatch in combination with CUDA for the Jacobian-Free, Newton-Krylov (JFNK) residual and preconditioner fills. The JFNK residual fill is the most compute intensive component in Tusas.

Progress to Date

Acceleration progress to date for ExaAM components is described below.

Threshold Challenge Problem Components

- **Diablo**: Work toward GPU usage within Hypre is ongoing; its current state is that it exists, but does not currently provide much performance improvement.
- **TruchasPBF**: The GPU-enabled Hypre Struct solver has been successfully integrated into the flow model and used for the pressure projection solve and implicit viscous velocity update. However, performance is poor relative to the CPU-based solver. This is most likely due to the relatively small size of the flow problem (melt pool sized), making the GPU a poor match for these tasks.

  Integration of GPU-enabled Hypre BoomerAMG for preconditioning the nonlinear heat transfer time step system has been attempted but is not currently working. The issue appears to be incomplete/missing GPU support for the Hypre semi-structured interface (SStruct) that is being used.
- **ExaCA**: The two most computationally-intensive subroutines- the updating of cell states and the capture of new cells have been ported to GPU. ExaCA with GPU offloading has been run on single nodes of Lassen (LLNL).
- **ExaConstit**: The material models within ExaCMech have been ported over to the GPU with a 16.7× speed-up when comparing 1 Nvidia Volta against 7 IBM Power9 processors given a realistic workload of 350,000 quadrature points. The 7 IBM Power9s to 1 Volta GPU is the recommended division of work for Summit. ExaConstit is still being ported over to the GPU.

Stretch Science Components

- **MUEMAPPS-SS**: Strong and weak scaling of MEUMAPPS-SS has been demonstrated previously. The current goal is to achieve additional in-node scaling using GPUs and OpenACC. A MEUMAPPS-SS test problem that involves the calculation of the elastic energy of a hard inclusion in a matrix was run in Summit with either 42 cores using MPI or 42 cores and 6 GPU using OpenACC with `jsrun` options of $n = 6$, $a = 7$, $c = 7$, and $s = 1$. For portions of the code where parallel loops were accelerated using OpenACC, a maximum speed up of 7.2× was achieved. This includes the routines `eldis.f` and `elener.f`. In Fig. 10, this corresponds to the black curve. Portions of the code requiring FFT calls using P3dFFT where still run using MPI. The overall speed up for the test code including the time taken for P3DFFT calls was 4.73× that corresponds to the blue curve in Fig. 10. The scaling for the `eldis.f` routine continues to scale nicely up to 420^3. This is the routine where converged displacement field is calculated and uses GPUs to the maximum extent. This is the green curve shown in Fig. 10. The speedup comes down significantly when the P3DFFT times are included because the total time spent in the P3DFFT calls in the test problem is roughly twice the time spent in the GPUs. Therefore, the speedup could be significantly reduced by decreasing the time spent in the FFT calls. Two strategies are being explored: (1) Moving all the data to the GPUs and performing FFT calls using GPU based FFT packages (CuFFT, ECP-FFT), and (2) explore efficiencies by introducing OpenMP threads in the P3dFFT calls and assigning 1 MPI task per GPU with multiple threads per MPI task.
- **AMPE**: CUDA-based Hypre for preconditioning linear solvers coupled to AMPE proxy app (PFiSM) and running on Summit (OLCF). RAJA + Umpire is still in development branch within SAMRAI.

Prototype and proxy app components

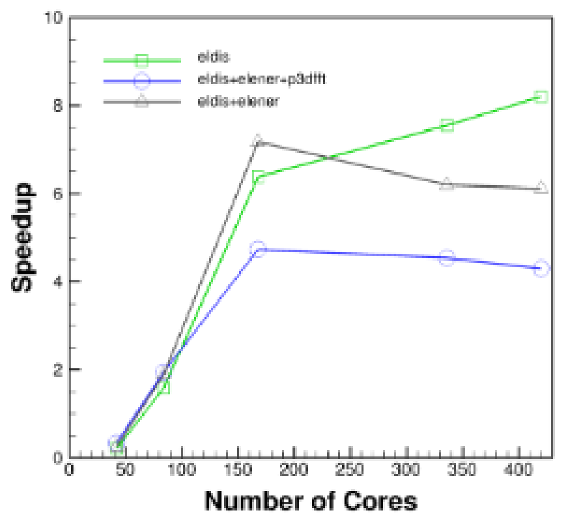


Figure 10: Speedup obtained using OpenACC in MEUMAPPS-SS test problem compared to all-MPI version using 1 code of Summit.

- **ExaMPM**: ExaMPM scaling studies on Summit are ongoing. We are exploring efficient domain decomposition strategy using Cabana when the V100 GPUs are used entirely for the computation. Current results indicate good strong scaling down to at least 300k particles per GPU. Based on this, we expect the strong scaling behavior of the grid-based linear solvers to be more prohibitive than the particle algorithms in the code. We expect this as current results from the solvers community indicate at least 1M grid cells per GPU are needed for performance.
- **Tusas**: Tusas has been ported to Summit and has demonstrated a runtime speedup of 4× per node (6 V100 GPUs versus 42 OpenMP threads or MPI ranks) as shown in Fig. 11. In addition, Fig. 12 shows strong scaling of Tusas on up to 6,144 Summit GPUs (1,024 nodes). We are currently testing CUDA-aware MPI and utilization of the burst buffer.

Next Steps

Next steps for each ExaAM component are shown below.

Threshold Challenge Problem Components

- **Diablo**: Continue monitoring Hypre GPU improvements.
- **TruchasPBF**: Monitor Hypre GPU advances for the SStruct interface. Alternatively, move to the GPU-supported IJ interface (costly).

  Implement OpenMP offloading for computational kernels (system matrix/right-hand-side formation). Shift to AMReX GPU-based multifab data structures.
- **ExaCA**: To achieve the desired speedup, it is likely that the “deep copy” operations performed each CA time step between the GPU and CPU will need to be avoided. The offloading of all calculations outside of initialization, file reading, and output, to GPU using CUDA-aware MPI should be the next step.
- **ExaConstit**: The reformulation of ExaConstit’s assembly operation over to a partial assembly formulation is being worked on with help from the MFEM team. Once this work is complete, strong scaling studies will be conducted in order to compare the CPU only implementation against the CPU/GPU implementation of ExaConstit. Additionally, a miniapp created for ExaCMech will be shared with AMD and Nvidia engineering teams to begin engagements with them on ways that ExaCMech can be improved to run faster on their platforms.

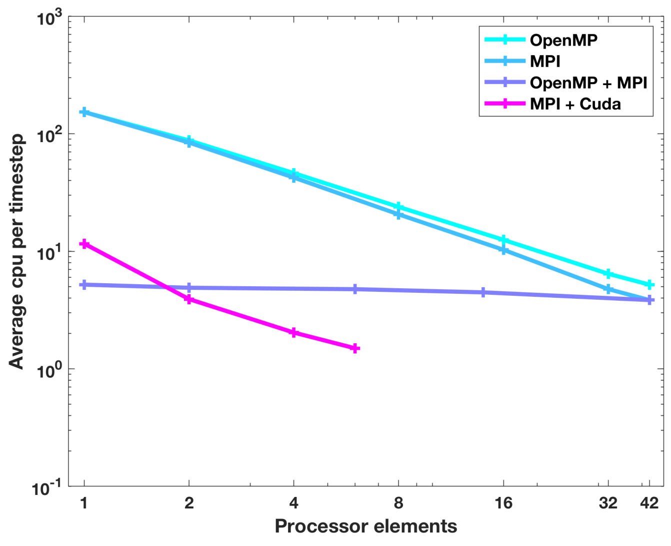


Figure 11: Single node strong and weak scaling for MPI + Kokkos MPI, threads and GPUs on Summit; one MPI rank per GPU for 16M unknowns. Color represents Kokkos (OpenMP) only, MPI only, Kokkos (OpenMP) + MPI and Kokkos (Cuda) on GPUs. The horizontal axis represents the number of processor elements, ie. number of threads, number of MPI tasks or number of V100 GPUs.

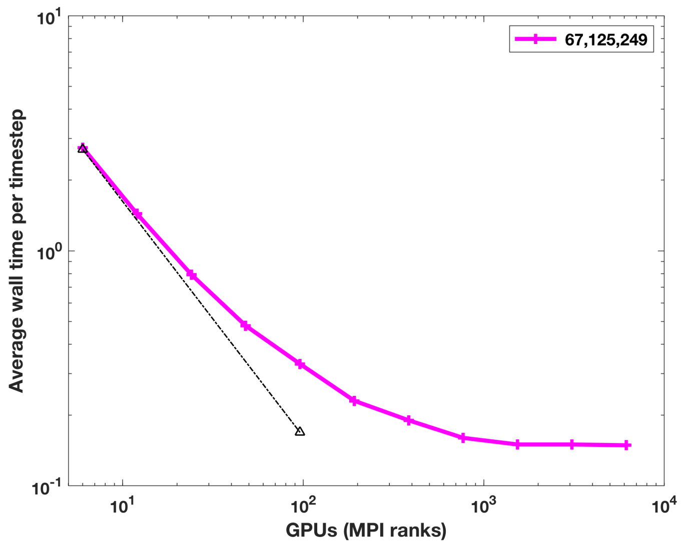


Figure 12: Strong scaling for MPI + Kokkos GPUs on Summit; one MPI rank per GPU. Color represents fixed problem size 67M unknowns on up to 6,144 GPUs (1,024 nodes). The figure confirms that ~2M unknowns per V100 GPU are required to maintain efficiency.

Stretch Science Components

- **MEUMAPPS-SS**: Extend OpenACC to all parallel loops in the MEUMAPPS-SS routines and evaluate speed up compared to all-MPI code using multiple Summit nodes.

  Run OpenACC with other GPU based FFT packages, especially the ECP-FFT package with brick domain decomposition.

  Re-write MEUMAPPS test problem in C++ and explore GPU acceleration using all available options. If the speedup is better than what has been achieved with OpenACC, a decision will be made to extend this option to other routines in MEUMAPPS-SS.

- **AMPE**: Adapt AMPE to latest SAMRAI and Sundials releases to benefits from their recent/ongoing support for GPUs.

Prototype and proxy app components

- **ExaMPM**: Continue Summit scaling studies to further assess the behavior of the domain decomposition algorithm for particles. Assess scaling/performance on Summit in the implicit case using linear solvers for the grid portion of the algorithm.

- **Tusas**: Some further optimizations of Kokkos use within Tusas are needed in terms of memory layout and memory management between the host and device. Tusas currently utilizes a graph coloring approach which avoids shared memory race conditions and the use of atomic operations on threads and GPUs; examination of the use of atomic operations will be initiated in FY20. Tusas will rely on Kokkos for leveraging the AMD GPUs on Frontier; we will examine HIP and ROCm alternatives to our specific CUDA implementations in FY20.

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 (parameter free) and therefore 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 since they robustly deliver highly accurate calculations of complex materials that do not artificially bias solutions of a given character. Significantly, with increased computer power, the few approximations in these methods can be tested and systematically reduced, which is not possible with other first-principles methods. See Ref [3] for a recent study of defects in phosphors.

The trade-off is that the computational demands of the QMC method are quite large. As an example, the use of petascale computers has allowed calculations of the magnetic exchange in a copper oxide important for understanding the mechanism of high-temperature superconductivity. However, this calculation involved a highly symmetric supercell containing only 56 atoms, when a realistic model considering 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, 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 quantum-mechanics-based materials calculations today. This capability will be highly useful across the materials sciences, nanoscience, and physics communities, particularly where experimental data is costly or difficult to obtain.

Table 23: QMCPACK challenge problem details.

Functional requirement	Minimum criteria
Physical phenomena and associated models	Predict cohesive energy of a large supercell of nickel oxide (NiO) using QMCPACK and diffusion Quantum Monte Carlo to an accuracy of 0.010 eV per NiO formula unit. The FOM formula allows calculation of arbitrary supercell size but a 1,024 atom supercell calculation on exascale systems is anticipated. To ensure the chosen target problems are both realistic and representative, trial wavefunctions and pseudopotentials are specified that are the same as in the team's recent publications [4]; i.e., they have 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, etc.	The problem is defined by an 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 hours. The FOM can also be accurately estimated by measuring throughput at full scale of the machine.

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) 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 wall clock time, e.g., < 1 day. The team anticipates a minimum 256 atom supercell up to 1,024 atom supercell, as specified in the team's original proposal, but the current FOM (next section) is more flexibly defined and includes the formal power law scaling of the method with system size. Details on the challenge problem can be found 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 non-empirical predictions by other methods. NiO is also part of the class of materials that is being studied by a US DOE BES's 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.

While 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 wavefunctions, a large range of observables (e.g., electron density, forces, density matrices), a sophisticated trial wavefunction optimization scheme, support for multiple QMC methods, and viable sources of input trial wavefunctions. The design templates established by reaching the FOM should be transferable since these additional capabilities add computational cost and are therefore thought to increase ease of mapping to different architectures. A key stretch goal is to ensure that low symmetry materials can be studied. While these do not change the electron count and therefore formal computational cost, memory requirements are greatly increased. This therefore requires support for localized orbitals to reduce memory usage and/or successful development of latency hiding techniques to enable the use of slower or remote memory.

3.6.2 QMCPACK: Figure of Merit

The FOM for QMCPACK is defined

$$\text{FOM} = \frac{N_{\text{samples}} N_{\text{elec}}}{\text{wall time}} \,. \tag{6}$$

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.

A prototype port of mainline QMCPACK developed in March 2019 to use OpenMP target/offload functionality results in a projected FOM of 37.4×10^{14} on the entirety of Summit, assuming a 95% weak scaling from single node results. This increased FOM results from improved algorithms for the costliest part of the calculation and the capability to run much larger problems (512 atoms) than is feasible on Titan due to the enhanced GPU memory size. However, due to the lack of efficient multiple Monte Carlo walker vectorization/parallelization in this new port, efficiency and performance for smaller problems are poor. A redesign of QMCPACK internals is being performed to support this in a maintainable and portable fashion.

3.6.3 QMCPACK: KPP Stretch Goal

Complete QMC workflow for a QMC calculation of a general low symmetry non-bulk system (e.g. defect or interface) using localized orbitals.

Exact achievable size/complexity will depend on memory of A21 & Frontier, and memory reductions achievable with using the localized RMG orbitals compared to delocalized schemes $\sim 7\times$ for 512 atom NiO, 6.5 Bohr localization)

RMG localized orbital implementation and support in QMCPACK will need to be sufficiently capable, portable.

3.6.4 QMCPACK: Progress Towards Advanced Architectures

GPU Strategy

This project is focused on the open source QMCPACK code, and targets both Aurora and Frontier. To complete the Exascale challenge problem (KPP-1 or stretch), this probAs part of several milestones in previous years, the application and key Quantum Monte Carlo algorithms in it have been extensively profiled. Mini-apps have also been constructed and verified to represent the essential operations of the main algorithms with good fidelity. By study these results as a function of simulated system size for candidate FOM problems, the development team have firmly established which kernels are essential to execute on accelerator platforms. There are around a half dozen flavors of kernel that are relevant. For the largest Exascale challenge problems only the orbital evaluation and matrix updates are essential to accelerate, since they dominate execution time. However, more kernels become relevant for smaller problems due to the accelerator equivalent of Amdahl's law.

QMCPACK has a several years old CUDA implementation, which we term the "legacy CUDA" implementation. This uses explicitly coded CUDA kernels, cuBLAS and cuSOLVER, and explicit memory management. This runs relatively efficiently on current hardware, is part of the CORAL2 procurement, and is used for the project FOM reference from Titan. It is also used for current INCITE science production. However, the project is planning to move away from this implementation since it has a sparse feature matrix and the code hard aborts when unsupported features are used—no fallbacks to CPU were implemented due to (at the time) different data layouts between CPU and GPU, as well as other

The GPU strategy adopted by the project for QMCPACK is therefore to accelerate all the essential kernels needed for a high FOM and to do so using an improved design that enables more flexibility in execution, naturally provides CPU fallbacks, and allows multiple architectures to be more easily supported than present by pushing architectural specific features out to the "leaf nodes" of the application.

Progress to Date

In FY19 and MS3.2, a major effort was spent assessing strategies for obtaining high GPU performance in a portable and maintainable manner. Besides meeting the FOM, a key project goal is to minimize the amount of architecture specific code necessary to obtain high performance. Maintaining multiple architectural implementations is not feasible long term and also adds significantly to the testing and development burden.

Kokkos and OpenMP v4.5 target/offload approaches were investigated via the mini-apps. In brief, both were found to be generally capable but both had issues relating to current maturity and usability. The Kokkos miniapp was demonstrated on a broad variety of platforms including ARM, x86, Power, and with NVIDIA accelerators. While good performance was achieved with OpenMP offload, a key weakness of OpenMP offload relates to the available compilers: only the IBM XL had sufficiently capability on Power and was able to offload sufficiently efficiently to NVIDIA accelerators. It was also possible to retrofit OpenMP offload to mainline QMCPACK and run a large FOM-sized calculation. In this regime, only the splined orbital evaluation and matrix updates need to offload efficiency. **An estimated** 37× **increase in FOM on Summit was projected as part of MS3.2**, using a 512 atom NiO supercell. This implementation is included in QMCPACK v3.8.0 released 23 July 2019.

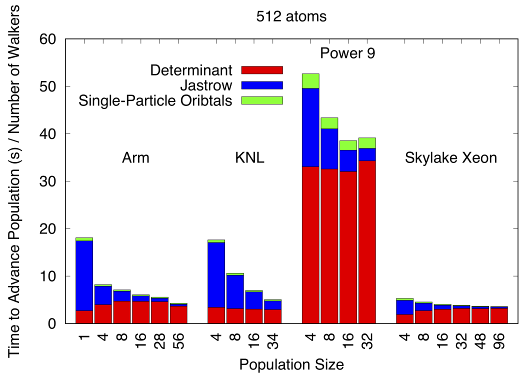
Performance studies performed as part of MS3.2 demonstrated that for the largest problems considered (around 10000 electrons), even on today's NVIDIA V100 GPUs, the application was barely in a regime where all the kernels would have sufficient work to saturate the GPU. Smaller calculations using the "one walker at a time" dispatch followed by the existing CPU algorithms would clearly not saturate the GPU and would run inefficiently. Therefore, multiple walker simultaneous algorithms will be required on current and future GPUs in order to sufficiently occupy the GPUs and run efficiently. We invented a generalization of the algorithms used in the legacy CUDA implementation where individual CPU threads will propagate "crowds" of walkers of variable size, and dispatch work to GPUs. (This is also referred to as "batching".) This adds the flexibility to vary continuously between a "one walker at a time" up to "all walkers simultaneous" style of propagation, varying the work in the dispatched kernels and also increasingly the possibility to propagate walkers asynchronously with different threads. We showed that even on CPU systems, crowds are a useful concept that can result in faster overall execution due to better cache and memory bandwidth utilization. Adding this additional flexibility requires a significant update to the application and is more significant to the achievable performance envelope than picking any particular implementation route.

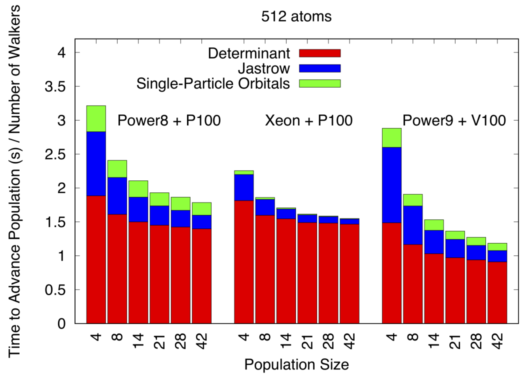
Besides identifying and GPU accelerating the major kernels in the application, the fundamental algorithms used have also been reconsidered and new algorithms adopted in some cases. While most of the changes made to date are small optimizations, a significant new algorithm was introduced for the Slater matrix updates in real space QMC that yield substantial speedups for large problems. This is the most expensive kernel for large problems and results in the formal numerical cubic scaling of QMC with problem size. Instead of using the traditional Sherman-Morrison rank-1 updating, the new "delayed update" algorithm permits changes to these matrices to be deferred for several Monte Carlo moves and then applied en-bloc. Crucially this does not change the sampled path in the Monte Carlo and the quantities that need to be evaluated based on the "updated" matrices can be calculated at low cost. This new algorithm enables much better use of memory bandwidth by allowing uses of BLAS3 operations instead of BLAS1 and BLAS2 for the most expensive part of the calculation. For systems with thousands-tens of thousands of electrons, such as the FOM, speedups are dramatic on CPU like systems (e.g. 10×) and more modest on GPU system (e.g. 2×) due to the different memory bandwidths and relative overall execution efficiencies of CPU code vs GPU kernels and kernel launches. Performance comparisons using Kokkos to provide cross-platform portability between CPU and GPU architectures are shown in Fig. 13.

Next Steps

We are currently finishing the implementation of new QMC drivers able to run crowds/batches of walkers and dispatch their execution flexibly to CPUs or GPUs. Our initial implementation will use OpenMP offload, but we retain the flexibility to write non-universal and potentially vendor specific code (e.g. HIP, CUDA) if needed, or even abandon OpenMP and adopt a different technology such as Kokkos. Our goal is to transition *current production science* to use this new implementation on Summit at the start of the INCITE year 2020. Significant additional work will be required over the remainder of the year to update the rest of the application and to obtain fair performance. This new version will execute from a single code path and enable the legacy CUDA and legacy array-of-structures CPU codes to be excised.

Development and verification of this new implementation is only possible due to the now sizable set of tests that have been built up in the previous years of ECP. New bugs are now usually caught quickly, either by unit or nightly integration tests. We have also begun adding fully deterministic tests to enable rapid





(a) **(b)**

Figure 13: Normalized time to propagate various size populations per walker on a variety of (a) CPU and (b) GPU architectures for the 512 atom NiO cell using the Kokkos implementation of miniqmc.

testing of the stochastic code. While these tests potentially require frequent updating, they have the benefit of being totally reliable and repeatable unlike the existing stochastic tests which have occasional failures. The deterministic tests will be expanded and we seek an automated maintenance strategy for them.

Once the new QMCPACK v4.0.0 version is running on Summit, we will assess and improve performance e.g. by porting additional kernels. We will run this version on additional platforms and with additional compilers as these become available via early access platforms such as the Cray Center of Excellence machines. A key issue of concern is the lack of sufficiently mature OpenMP compilers targeting GPUs. On Summit the capable IBM compiler currently only supports C++14, stopping us from moving to C++17. On AMD, we have found and reported multiple issues with the AOMP OpenMP compiler using our miniapp and will investigate the situation with the main application and latest compilers. The Intel compilers will similarly be critically and urgently assessed.

4. ENERGY APPLICATIONS

End State: Deliver a broad array of science-based computational applications able to provide, through 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 simulation of 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 detailed modeling of complex facilities and multiple coupled physical processes. Their goal is to help overcome obstacles to the efficient and safe delivery of energy.

These applications are highly complex and involve the modeling of intricate geometric detail and the inclusion of a broad range of physical phenomena. A key additional requirement for EA is broader community adoption of the computational models and methods developed in the project, or in some cases the virtual data sets or physical insights that result from simulations carried out in the ECP. Additionally, applications are expected to influence, either directly or indirectly, design choices for both 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-

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 Monte Carlo 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

turbine-interaction effects. Greater use of the nation's abundant wind resources for electric power generation, reaching 30% of US electrical supply, will have a profound societal and economic impact: 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 thermo-electric power generation.

A key challenge to 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 a lack of computational power necessary 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 composed of $O(100)$ multi-MW 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 reliably integrating 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, where 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 and OpenFAST. Nalu-Wind is an acoustically incompressible Computational Fluid Dynamics (CFD) code written in C++, and it is a wind-specific version of the Nalu code, a 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 discretization of the underlying models, and it heavily utilizes 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. Moving meshes are handled 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 (WETO) High-Fidelity-Modeling (HFM) project, Nalu-Wind is being continually appended and improved with wind-specific capabilities, including a new time-step algorithm,

Table 25: ExaWind challenge problem details.

Functional requirement	Minimum criteria
Physical phenomena and associated models	Low-Mach-number (acoustically incompressible) fluid flow with fluid-structure interaction. A hybrid-RANS/LES model with unsteady-RANS model will be used near turbine surfaces, and an LES model will be used in the wake region.
Numerical approach, algorithms	Unstructured-grid and/or structured-grid finite volume solver with overset meshes and implicit pressure projection. Linear and nonlinear structural finite element models.
Simulation details: problem size, complexity, geometry, etc.	Four MW-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 20B points with 100B 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 is representative of 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.

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 square kilometers. 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 a ModSim capability that resolves turbine geometry and utilizes adequate grid resolution (down to micron scales within the blade boundary layers). The resolution must capture the upstream chord-scale atmospheric turbulent eddies, generation of near-blade vorticity, and 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 have been 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 $3\,\text{km} \times 3\,\text{km}$ domain with height of at least 1 km. A hybrid-RANS/LES model will be employed for which an 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) to resolve the system, and near-blade grid spacing will be such that the viscous sub-layer (within the RANS region) is resolved. A successful simulation will require an optimized solver stack that minimizes time-per-timestep. A scientifically meaningful simulation duration will be for at least one domain transit time (about 370 s for the $3\,\text{km} \times 3\,\text{km}$ 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 $4\,\text{km} \times 4\,\text{km}$ fluid domain with complex terrain and with 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 degrees of freedom) to resolve the system, and near-blade grid spacing will be such that the viscous sub-layer (within the RANS region) is resolved. A successful simulation will require an optimized solver stack that minimizes time-per-timestep. A scientifically meaningful simulation duration will be for at least one domain transit time (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: Progress Towards Advanced Architectures*

GPU Strategy

The primary application codes of ExaWind are Nalu-Wind and OpenFAST, which embody the numerical models for the fluid and wind turbines, respectively. Nalu-Wind handles creation of the linear systems associated with discrete operators under a semi-implicit, pressure projection time-update algorithm. The primary simulation costs are the creation and solution of the linear systems for the momentum and pressure equation systems. Due to mesh motions, those linear systems and preconditioners must be recreated at every time step. Nalu-Wind is dependent on several libraries, most notably the Trilinos Sierra Toolkit (STK), which provides an unstructured mesh in-memory, parallel-distributed database. Linear systems and preconditioners can be created and solved with Trilinos, Hypre, or some combination of the two. Moving meshes are handled with the overset approach, for which mesh connectivity and constraints are created with TIOGA (Toplogy Independent Overset Grid Assembly), which is an open-source, third-party library created and maintained by an ExaWind collaborator under subcontract.

In regard to the GPU strategy, the Exawind team has been preparing Nalu-Wind for GPUs through the Kokkos abstraction layer, as are the key ExaWind components of Trilinos, including STK and the solver/preconditioner stack. For solvers and preconditioners, a key strategy is to assemble matrices and create preconditioners on the GPU rather than on the host to minimize the data movement. Efforts to prepare the Hypre solver stack are happening within the ExaWind project and through collaboration with the Hypre team. The Hypre strategy is around hardware-vendor-specific APIs, e.g., CUDA and HIP. TIOGA is also being prepared for GPUs directly using CUDA. For both Hypre and TIOGA, the long-term plan is to use HIP for Frontier.

OpenFAST is written almost entirely in Fortran. While it handles complex turbine multi-physics and controls, it has dramatically less computational cost than Nalu-Wind. For the ExaWind challenge problem OpenFAST is expected to run on the host rather than device.

Progress to Date

In this section we describe progress to date in preparing the Nalu-Wind software stack for GPU machines, focusing on advances with the primary software components. GPU-performance testing has been performed on NREL's Eagle system (2 Intel Skylake CPUs and 2 NVIDIA Volta V100 GPUs per node), OLCF SummitDev (2 IBM Power8 CPUs and 4 NVIDIA Tesla P100 GPUs per node), and OLCF Summit (2 IBM Power9 CPUs and 6 NVIDIA Volta V100 GPUs per node).

Nalu-Wind

The Exawind team has produced initial simulations of the atmospheric boundary layer on hybrid CPU-GPU architectures using the Nalu-Wind computational-fluid-dynamics code. The simulation was the result of a focused effort of redesigning the numerical-discretization operators and linear-system assembly operations using the Kokkos abstraction library within Nalu-Wind. In addition to linear-system assembly, the field-update operations, Krylov solvers, and multi-threaded symmetric Gauss-Seidel (MTGS) preconditioning for the momentum and scalar-variable solves were also performed on the GPU using the Trilinos solver stack.

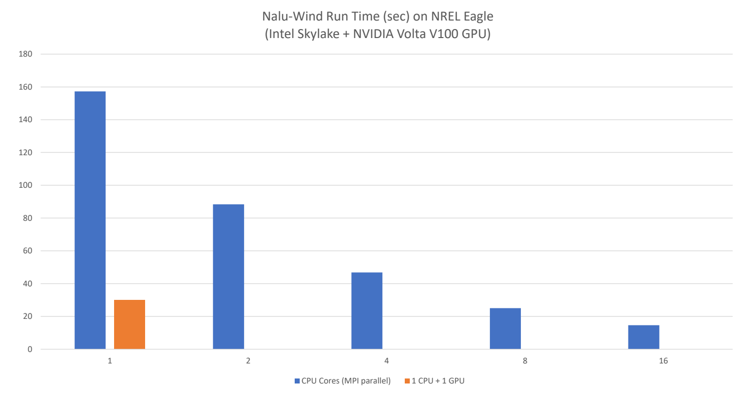


Figure 14: Nalu-Wind simulation times for an atmospheric boundary layer on the NREL Eagle system.

Several other operations were partially offloaded to the GPUs through Trilinos solver and linear algebra libraries (MueLu, Kokkos Kernels, etc.).

Simulations were performed for a 5 km $\times$ 5 km $\times$ 1 km domain at a coarse resolution (40 m cells) yielding 412,776 grid points. Figure 14 shows initial performance comparisons have been performed on the Summit platform, and on NREL's Eagle supercomputer. The Eagle hybrid CPU-GPU simulation took 30.2 seconds to perform 10 timesteps (with 4 Picard iterations per timestep). In comparison, CPU-only (Skylake) flat-MPI simulations took 157.3 s, 25.1 s, and 14.6 s for 1, 8, and 16 MPI ranks respectively (i.e., approximately 400k, 50k, and 25k grid points per MPI rank, respectively). Preliminary performance analysis indicates Eagle there is a 5$\times$ speedup using a single GPU compared to a single CPU core. The ABL precursor simulations were also performed on ORNL Summit system. On both NREL Eagle and ORNL Summit systems, it was observed that a single V100 simulation is comparable to an 8-rank MPI CPU simulation.

Trilinos solver and preconditioners

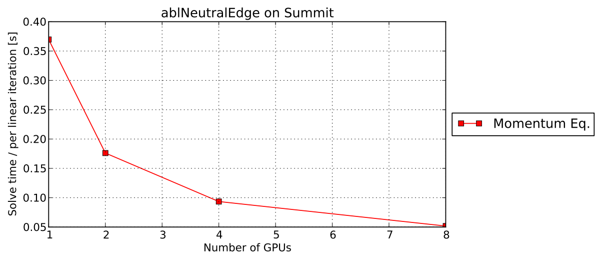
The Trilinos solvers team has re-written the aggregation kernel coarsening phase kernel to use Kokkos and leverage distance-2 coloring algorithm from Kokkos-Kernels. This removes a major impediment to the team running a full simulation on GPUs. An initial end-to-end Nalu-Wind simulation was completed using the Trilinos solver stack on two Summit nodes using 8 GPUs (1 MPI rank per device). Strong scaling results are given in Fig. 15. The plots therein show the time per iteration for the momentum (15a) and continuity (15b) solvers, respectively. Novel GMRES solvers based on s-step and pipelined methods have been developed in the Trilinos Krylov library Belos[1]. Performance of the methods on a 3D model problem is shown in Fig. 16.

Hypre solver and preconditioners

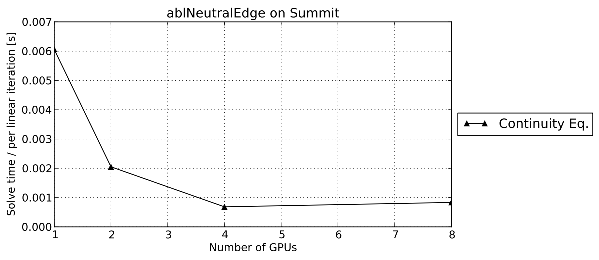
The Exawind team in collaboration with the Hypre team at LLNL has performed initial runs of the McAlister blade wind turbine problem using a multi-MPI multi-GPU implementation of the Nalu-Wind computational fluid dynamics model coupled to the new low-synch GMRES solver. One MPI rank is assigned to a processor coupled to a GPU. The mesh size for this problem is relatively large with over 3 million degrees of freedom.

To prepare for GPU acceleration of the most time-consuming Nalu-Wind model and Hypre solver stack components, a new momentum preconditioner and pressure AMG smoother were introduced into Hypre. These are based on a Jacobi-Richardson type polynomial algorithm. Because the smoother depends upon

[1]Full descriptions are given in the submitted paper, Yamazaki et al. Low-synchronization orthogonalization schemes for s-step and pipelined Krylov solvers in Trilinos



(a) Momentum solve time-per-linear-iteration.



(b) Continuity time-per-linear-iteration.

Figure 15: Strong scaling for Nalu-Wind using Trilinos solver stack on Summit.

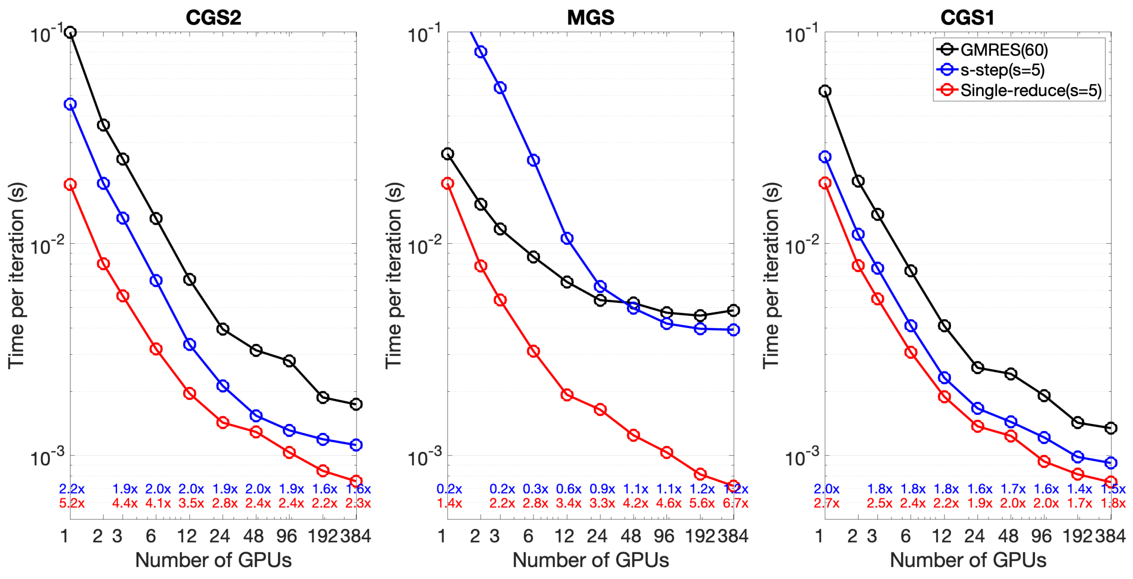


Figure 16: Scaling of s-step and pipelined GMRES in Trilinos/Belos on model problem.

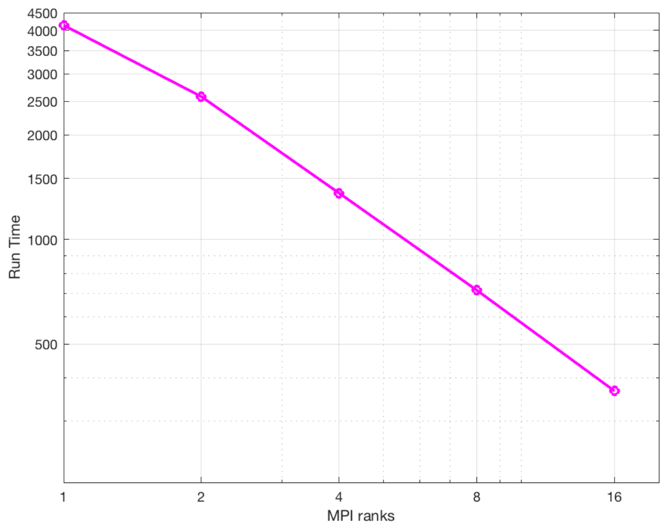


Figure 17: Nalu-Wind + Hypre linear strong scaling for McAlister blade problem on NREL Eagle NVIDIA V100.

matrix-vector multiplication kernels, the resulting performance both on CPU and GPU is optimal. Nalu-Wind now exhibits near perfect linear strong-scaling on Eagle for the CPU version (see Fig. 17). In addition, the multi-MPI multi-GPU code exhibits the same linear scaling pattern. We are able to achieve 3.5 seconds per time step on 160 CPU MPI ranks at the strong scaling limit of 20,000 DOF per MPI rank. Preliminary results indicate the CPU-GPU times are lower.

The model execution time breakdown at the strong scaling limit is now momentum solve 7.3 %, pressure solve: 16.6 %, momentum assemble: 7 %, pressure assemble 5 %, AMG set-up 12.8 %.

TIOGA

The core component of the overset connectivity algorithm in TIOGA is the search for donors cells amongst multiple overlapping meshes within the computational domain. Currently, the search process uses an Alternating Digital Tree (ADT) algorithm within TIOGA. This algorithm has been ported to GPU using native CUDA kernels. In addition, a new algorithm based on inverse-Cartesian map has been also implemented.

The performance of the original ADT algorithm was compared on the ORNL SummitDev system and is shown in Fig. 18. The performance of a single GPU execution (on NVIDIA Tesla P100 GPU) is compared to the a full node CPU run (with 20 MPI ranks and 8 OpenMP thread = 180 processes). The performance studies indicate a 1.7× speed-up for the largest problem studied.

The GPU performance of the inverse Cartesian map algorithm was compared to flat-MPI only simulations on ORNL Summit system. For the comparisons, a single GPU (on NVIDIA Volta V100 GPU) is compared against CPU simulations of the standard TIOGA search algorithm running on 84 MPI ranks (and no OpenMP threads). Compared to the current production capability, the new inverse map algorithm showed a performance improvement of 200×, which can be attributed to two factors:

- Inverse map algorithm is known to be 3–4× faster than ADT-based searches. However, ADT-based searches are more robust owing to the simplicity of the algorithm.
- The ADT algorithm ascertains the containment of a point using an iterative Newton-type algorithm that determines the computational coordinates of a given point against a curved cell. This step is completely avoided in the line based-search that determines containment using intersection checks.

Next Steps

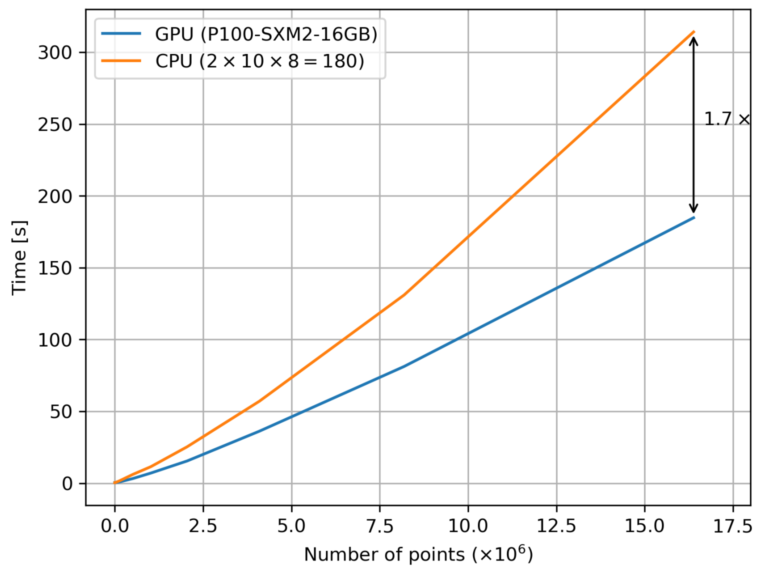


Figure 18: CPU-GPU performance comparison on SummitDev for the original ADT algorithm.

The ExaWind team is primarily focused on simulating its grand-challenge problem on Frontier, though it is hoped that our use of Kokkos will enable simulations on Aurora with minimal changes. The Exawind GPU strategy has relied heavily on the use of the Kokkos performance portability abstraction layer within the Nalu-Wind application code and the Trilinos based libraries. Within those components, direct CUDA API calls have been avoided in our runs on NVIDIA GPUs. Communications with the Kokkos authors indicates that they are working to develop appropriate back-end layers for the AMD GPUs that are in the Frontier platform. This is expected to result in minimal efforts on the Exawind team to adapt to the AMD hardware with the HIP programming model. Within the non-Kokkos software components, CUDA API call will be ported to HIP for Frontier simulations.

In regard to the Nalu-Wind CFD code, the team will perform thorough benchmarking and profiling to identify bottlenecks and improve the Nalu-Wind design, as well as proceed to Phase 2 of the Nalu-Wind transition effort to focus on the kernels necessary for blade-resolved overset simulations. For the Hypre component, the team will perform thorough benchmarking and profiling to identify performance bottlenecks and improve the Nalu-wind coupled to Hypre design. The strategy to further prepare Hypre for GPUs and simulation runs on Summit and Frontier involves focusing on three core algorithms. These are

1. Sparse-matrix vector multiplies in a new Jacobi-Richardson AMG smoother and preconditioner.
2. Acceleration of the sparse matrix assembly in hypre within the function `HYPRE_IJMatrixAddToValues` with appropriate chunk sizes and avoid thread collisions (with LLNL).
3. Continue to accelerate the AMG set-up phase in Hypre with the LLNL Hypre team as this becomes a tall pole at the strong scaling limit.

For the Trilinos stack, the solvers team will continue to benchmark and optimize performance of the solvers using end-to-end simulations. In particular, we will focus on

1. Exploiting multiple right-hand side optimizations possibilities for segregated momentum solves.
2. Amortizing sparse graph initializations across linear solves.
3. Test the new s-step and pipelined Krylov methods in Trilinos/Belos within Nalu-Wind simulations.
4. Seek opportunities to reduce the number of kernel launches necessary.

4.2 Combustion-Pele

Aggressive national goals for significantly reducing petroleum use and greenhouse gas emissions require major improvements in all aspects of the nation's energy use. Combustion processes have historically dominated electrical power production and transportation systems. Despite major 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 need to be optimized for energy efficiency and reduced emissions.

This project is structured around providing a combination of first-principles Direct Numerical Simulation (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 vaporization, low-temperature ignition, flame propagation, and soot/radiation. RCCI is thermodynamically favorable relative to existing engines and hence holds the promise of groundbreaking efficiencies while operating in a regime that limits pollutant formation. The roadmap towards this exascale-era motivating problem includes simulation of a multi-injection low-temperature diesel jet into an open domain with a large alkane fuel undergoing two-stage ignition processes, simulations of dilute spray evaporation and mixing, and simulation of 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 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 utilizing low-temperature combustion (LTC), for which RCCI is the exemplar. The enabled exascale-era simulations will address key scientific questions regarding mixture formation effects, multi-stage ignition of a diesel surrogate fuel, lifted flame stabilization, jet re-entrainment affected by cylinder-wall geometry, and emissions. The simulation will account for 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 (injection of liquid fuel sprays into high-pressure conditions), mixing, and 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 both the compressible and low-Mach limits where, using the machinery of Adaptive Mesh Refinement (AMR), the team will refine to the DNS limit where 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 necessary on several challenging, but tractable, fronts. Existing simulation tools will need to 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 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 non-ideal fluid behavior, sprays, soot, and radiation, as well as non-trivial 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 non-ideal chemistry.

4.2.1 Combustion-Pele: Science Challenge Problem Description

The specific science-based challenge problem is derived from the roadmap 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 multi-pulse injection strategy into engine-relevant geometry. It is a

Table 26: Physical characteristics and computational approaches.

Characteristic/need	Approach
Impulsively started jets with disparate scales between fronts and turbulence. (Outer scales: 10 cm, ms; Inner timescales: µm, ns)	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 multi-stage 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

baseline for a series of simulations that will enable isolating the impacts of effects such as spray evaporation on mixture fraction and temperature, alternative fuels, and design of strategies to control combustion phasing and subsequent combustion rate. The problem will be tractable under a realistic allocation 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.

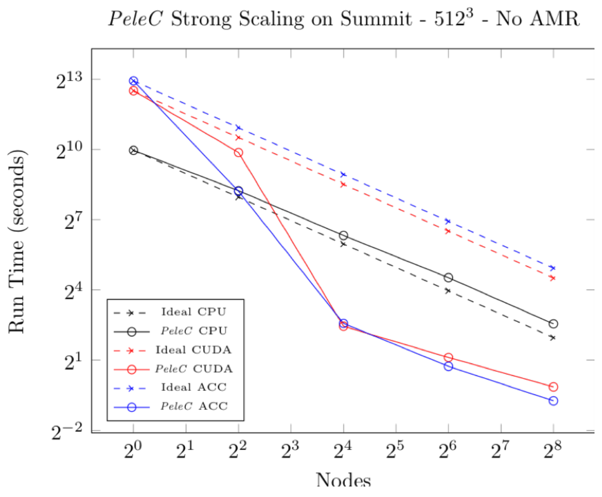
4.2.2 Combustion-Pele: KPP Stretch Goal

As a stretch goal, multi-phase fuel injection and soot emissions will be simulated in Pele to provide more realistic mixture formation (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 multi-pulse injection strategy of our KPP baseline problem. The droplet size distribution resulting from upstream spray atomization processes will be obtained off-line from existing volume-of-fluid DNS of atomization, large-eddy simulation of engine spray combustion and/or engine experiments. 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

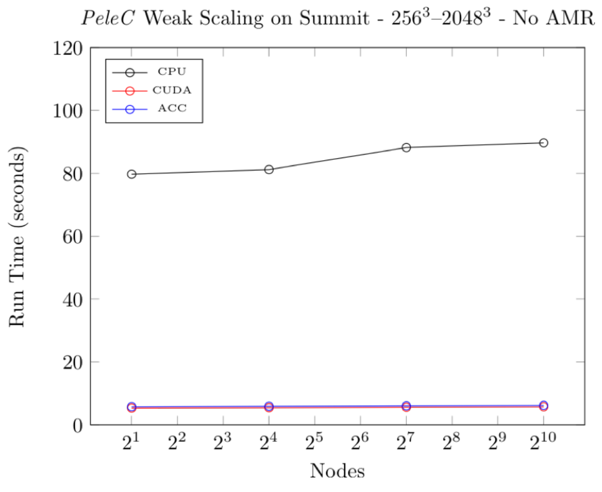
Recent advances in ML/AI software and hardware technologies (such as the deployment of Summit with TPU's/GPU's/CPU's) provide the potential to reframe the role of high-fidelity simulations in enabling progress in combustion research and engineering design. Traditionally the scientific process involves 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. There are two aspects of this workflow where ML/AI can be transformative that we will attempt to demonstrate with our stretch goal. First, AI techniques are currently showing 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 of paramount importance to effectively

Table 27: Combustion-Pele challenge problem details.

Functional requirement	Minimum criteria
Physical phenomena and associated models	Multiscale CFD with reacting flows in both 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 (DNS) treatment where turbulence-chemistry interaction occurs and modeled (LES) treatment to reduce resolution requirements in low heat release portions of the flow.
Simulation details: problem size, complexity, geometry, etc.	Gas-phase simulation of multiple jets (4) 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 including a low-reactivity and high-reactivity fuel capturing cross-mixing and reactions between fuels using ∼30–35 species; 1.5 ms physical simulation time; four levels of hierarchical mesh refinement; finest grid 1.25 µm; realistic environment: >50 bar.
Demonstration calculation requirements	Restart from checkpoint that gives us a realistic development of plume into geometry obtained by running the case with restricted resolution (two levels of mesh refinement), additional refinement added at restart, for a total of four levels, and 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 hours using the anticipated full system.



(a)



(b)

Figure 19: PeleC strong (a) and weak (b) scaling on Summit.

addressing the motivating question of interest. As part of our stretch goal, we will demonstrate the use of ML/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 accuracy of the chemical mechanism, turbulence treatment, multi-physics closure models, number of realizations, etc. We will also explore the ability of surrogate models using ML/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: Progress Towards Advanced Architectures

GPU Strategy

The Pele codes are based on the AMReX (§ 8.3) library for block-structured adaptive mesh refinement. AMReX currently supports the use of CUDA, OpenACC and OpenMP. Internally, AMReX relies on CUDA for NVIDIA accelerators, and is in the process of developing core-level support for AMD accelerators based on HIP. A similar migration strategy will be implemented when appropriate for supporting Intel accelerators. AMReX is currently expecting to support Intel through DPC++, but is waiting for Intel to release test hardware and standards before making a final decision. AMReX also provides support for users to utilize directive-based offloading strategies as an alternative.

Originally, much of the compute-intensive code in the Pele codes, PeleC and PeleLM, was written in Fortran. PeleC was chosen for an initial port to GPU-based systems (during FY19) due to its simplicity (with respect to PeleLM), and a dual-path strategy was selected for the port in order to guard against uncertain hardware specifications. Along one of the code's main execution branches (the so-called methods-of-lines integrator), the Fortran code was annotated with OpenMP directives in order to off-load much of compuatational work to GPUs. Along the second main branch (a spectral deferred corrections integrator), the AMReX CUDA-based approach was adopted. This latter approach required a complete rewrite of the associated Fortran code into C++. As shown in Fig. 19, both strategies performed equally well across a range of Summit nodes in a strong-scaling and weak-scaling performance analysis.

Although much work remains in the optimization of PeleC for Summit-like architectures, lessons learned from the PeleC work to date will be used in the initial port of PeleLM, scheduled for Quarters 2–4 of FY20. PeleLM is much more complicated than PeleC due to the much larger numerical time steps that the low Mach number formulation takes compared to that of the compressible formulation. First, the flow evolves on a manifold of constant thermodynamic pressure. The underlying algorithms enforce this during the flow evolution via the solution of a linear elliptic system. Next, the diffusion transport physics must be solved time-implicitly for numerical stability which involves an array of linear parabolic solves. Finally, the chemistry must be evolved time-implicitly.

Table 28: Summary of the performance of the interpolation and deposition kernels for a million particles employing the 4th-order interpolation and 43-stencil deposition on a 643 mesh for different architectures (ns/particle).

Machine name	Percival	Percival	Titan	Titan	Titan	Summit	Summit	Summit
Execution space	Serial	OMP	Serial	OMP	CUDA	Serial	OMP	CUDA
CPU cores/GPUs	1	64	1	16	1	1	21	1
Interpolation	933	17	345	41	22	130	6.3	1.3
Deposition	14,046	294	4,460	510	77	4,231	269	3.6

Initial ports of the Pele codes have centered on the mesh-based physics (advection, diffusion, reactions) integrated with time-explicit algorithms. Both codes are also being coupled to particle methods for multiphase physics processes, including liquid spray dynamics and radiation transport. Similar to Pele's dual-tracked strategy for mesh physics, there are two approaches to implementing spray coupling, and both are being developed in the context of PeleC first, and then will be migrated to PeleLM. The first approach uses a Kokkos-based library, Grit, for the particle data and interaction models and the second uses the AMReX-provided particle containers and iterators. PeleC-MP is the component of Pele for handling the particle-based multiphysics. This component leverages the particle support provided by AMReX and the effort to port particle physics to GPUs will culminate in a milestone scheduled for Quarter 4 of FY20. The excellent scalability of the AMReX particle containers is shown in the associated sections on AMReX. However, like the mesh data strategies, the AMReX approach requires a rewrite of the Pele multiphase interaction physics modules, from Fortran to C++. Alternatively, the Grit library works directly with the original Fortran source for the spray particle physics. It employs the Kokkos programming model for GPU acceleration and on-node shared memory parallelism. In Grit, the governing equations of the Lagrangian phase are solved explicitly through the Runge–Kutta method. The Eulerian simulations and MPI droplet exchange are conducted by PeleC through the AMReX particle container. The coupling between the dispersed and gaseous phases is through interpolation of variables from the Eulerian grid points of the gaseous phase to the particle positions, and the distribution of the source terms back to the grid points from the Lagrangian particles. The on-node performance of the interpolation kernel and the deposition kernel are benchmarked with respect to the number of droplets, mesh sizes and orders of the Lagrange polynomial on a single Volta GPU on Summit. A summary of the interpolation and deposition kernels benchmarks is shown in Table 28 with performance comparisons on Titan, the Intel manycore CPU Xeon 7230 on Percival at OLCF and on Summit. Based on the Kokkos programming model, Grit realized performance portability for different CPU and GPU architectures with a single implementation. The latest GPU accelerator, NVIDIA Volta, on the pre-exascale supercomputer, Summit, attains a 20-fold speedup compared with the speed of the NVIDIA Kepler GPU on Titan.

Additional features of the overall AMReX GPU strategy relevant to Pele developments include:

- Raw data (floating point, integer arrays and particles) is placed in managed memory. Application codes can opt out of the default and manage chosen objects separately by hand or with AMReX Memory Arenas. This decision simplifies the combined use of CUDA and directive-based offloading strategies.
- AMReX supports a C++ lambda-based launch system that offers inlining and portable GPU code.
- AMReX provides GPU-friendly implementations of common mesh, particle and particle-mesh operations, including operations such as Saxpy, reductions, PIC methods and particle neighbor list constructions.
- AMReX provides functions for common parallel communication operations for mesh and particle data, including ghost cell exchange and particle redistribution. These operations run on the GPUs without triggering host/device copies and have been optimized for performance on Summit.
- AMReX has no external software dependencies other than the standard software stack requirements such as working C++ compilers and MPI. However, AMReX does currently supply user-facing interfaces for the CVODE libraries supported by the SUNDIALS project, the external PETSc and Hypre algebraic multigrid solvers and HDF5. Aside from AMReX and the SUNDIALS library, the Pele codes have no additional dependencies.

Progress to Date

In preparation for the PeleLM port, a recent scaling study of the native geometric multigrid solvers in AMReX shows a roughly 4–5× speedup on a Summit node relative to a Cori Haswell node.

Next Steps

Following the successful strategy used to port PeleC, the Pele project will follow AMReX's lead to determine how to best access the API's and hardware accelerator(s) as they become available on Frontier. The AMReX team uses a local workstation with a Vega 10 AMD graphics card for active HIP-clang code development. Basic AMReX programs are successfully compiling and running with HIP-clang. HIP libraries, including hipRand and HIP's version of Thrust are going to be tested next. This development is currently in a separate branch, but will be merged into AMReX proper once fully tested. Code differences, bugs and feature requests have been identified and are being sent to AMD and ORNL to further improve AMReX and HIP for Frontier. We believe that the HIP-clang code path chosen for AMReX is well-suited for implementation within the Pele codes, and in FY20 we will begin to adapt PeleC to incorporate this approach and begin gathering performance data to assess its abilities.

AMReX has compiled and run with HIP-nvcc on Summit as an intermediate step towards HIP on AMD machines. AMReX will continue to use HIP-nvcc for a subset of bug and portability testing while HIP-clang is being developed. The PeleC porting work in FY20 will begin by following the AMReX lead here; we will get PeleC to build and run using HIP on Summit prior to working toward application on Frontier.

The Pele development team attended the Frontier Kickoff Workshop in September 2019. Pele was chosen as a Center for Accelerated Application Readiness (CAAR) code team to receive testing and expert assistance porting to AMD GPUs. That expertise will be utilized to port AMReX to HIP, as the underlying critical framework, and to ensure that the ports are compatible with the requirements of the Pele codes. CAAR and NESAP code teams are also provided access to an early test-bed system, which will be used for performance analysis with HIP.

AMReX established an AMD contact for HIP and ROCm questions at the workshop and has begun discussing bugs and feature requests needed for AMReX and its applications. The Pele development team will utilize this indirect connection in order to ensure that the needs of the Pele codes are taken into account.

Like the AMReX team, the Pele team anticipates DPC++ to be the preferred API to run on Intel systems and are awaiting formal releases of test platforms and standards before starting code development. The AMReX development team has attended available tutorials and seminars on DPC++, SYCL and other relevant Intel topics to keep up-to-date on the expected requirements for Aurora. The resulting knowledge transfer into the AMReX library and API access will be of direct benefit to the Pele development efforts.

The Pele application suite was chosen to participate in Argonne's Early Science Program (ESP). The provided expertise and hands-on Intel support will be used to help port AMReX itself to Aurora, much like the NESAP and CAAR programs have helped at NERSC and OLCF, respectively.

4.3 ExaSMR

Small Modular Reactors (SMRs) offer the prospect of 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 multi-decade history of experimental and operational data; for licensing, SMRs leverage that experience but are dependent 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 (CASL) DOE energy innovation hub has demonstrated the value of high-performance computing to the nuclear industry by deploying highly accurate Monte Carlo (MC) neutronics on DOE leadership-class platforms. The data sets generated by these first-of-a-kind simulations were used to validate the startup calculations of the new Westinghouse Electric Company AP1000 reactor. While this represented a significant advance, petascale computing and application limitations restrict the calculations to reactor startup 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 data sets of operational nuclear reactors. The MC neutronics method presents significant challenges related to 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
- Pin-resolved spatial fission power and reaction rate tallies

Details on the challenge problem specification are given in Table 29.

The objective of the simulation is calculating reactor startup conditions that demonstrate the initiation of natural circulation of the coolant flow 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 is targeting the Frontier architecture at ORNL, and OpenMC, which is targeting 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} particles per eigenvalue iteration
- Pin-resolved reaction rate with a single radial tally region and 20 axial levels
- 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
- Full core mesh 200×10^6 elements and 70×10^9 degrees-of-freedom

4.3.2 *ExaSMR: Figure of Merit*

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

$$\mathrm{FOM} = \frac{2}{\mathrm{FOM}_{\mathrm{MC}}^{-1} + \mathrm{FOM}_{\mathrm{CFD}}^{-1}}\,, \tag{7}$$

where each physics FOM is a ratio of a measured work rate to the baseline work rate, i.e.,

$$\mathrm{FOM}_{\mathrm{MC}} = \frac{W_{\mathrm{MC}}^{\mathrm{measured}}}{W_{\mathrm{MC}}^{\mathrm{baseline}}}\,, \tag{8}$$

$$\mathrm{FOM}_{\mathrm{CFD}} = \frac{W_{\mathrm{CFD}}^{\mathrm{measured}}}{W_{\mathrm{CFD}}^{\mathrm{baseline}}}\,. \tag{9}$$

Table 29: ExaSMR challenge problem details.

Functional requirement	Minimum criteria
Physical phenomena and associated models	Eigenvalue form of the linear Boltzmann transport equation with quasi-static nuclide depletion neutronics coupled to hybrid RANS/LES (or equivalent accuracy model), incompressible CFD with Boussinesq approximation or low-Mach incompressible CFD with non-zero thermal divergence.
Numerical approach, algorithms	Neutronics solver is Monte Carlo particle transport; CFD solver is spectral finite element on unstructured grids; physics are coupled using a quasi-static approximation.
Simulation details: problem size, complexity, geometry, etc.	The neutronics model has a minimum of 200k tally bins and 10B particle histories per eigenvalue iteration. The CFD model has a minimum of 200M elements and 70B degrees-of-freedom.
Demonstration calculation requirements	Run 30 eigenvalue cycles (10 inactive and 20 active) to estimate Monte Carlo particle tracking rate and 1,000 time steps toward steady-state convergence in the CFD solve. Only a single nonlinear (Picard) iteration is required. In order to facilitate comparison to the baseline measurement, the neutronics portion of this calculation requires six macroscopic reaction rates and a single radial region per pin.
Resource requirements to run demonstration calculation	2 hours at full system

The individual physics work rates are given by

$$\mathrm{W_{MC}} = \frac{\text{particles}}{\text{wall clock time}}\,, \tag{10}$$

$$\mathrm{W_{CFD}} = \frac{\text{degrees-of-freedom}}{\text{wall clock time}}\,. \tag{11}$$

The MC work rate is computed using only the active cycle particle tracking rate, which is more computationally intensive and typically occupies the majority of MC run time. If ensemble averaging is used to perform the CFD calculation, the number of degrees of freedom 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 to 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

- $\mathrm{FOM_{MC}}$ = 10.4M particles per second on Titan;
- $\mathrm{FOM_{CFD}}$ = 3.0B degrees-of-freedom per second on Titan.

4.3.3 ExaSMR: KPP Stretch Goal

The ExaSMR stretch goal is to improve the fidelity of both the MC and CFD models. For the MC problem, this will be to increase 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 as well as adding 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 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 degrees of freedom necessary to resolve the problem as well as placing more stringent requirements on the time steps needed to maintain numerical stability.

4.3.4 ExaSMR: Progress Towards Advanced Architectures

GPU Strategy

The use of Picard iteration for nonlinear coupling between physics applications in ExaSMR results in each code executing separately. This means that the MC neutronics and CFD solvers can adopt independent strategies for porting to new computing architectures. The Nek5000 CFD solver is being ported to GPU architectures in collaboration with the CEED codesign project. The GPU-enabled version of Nek5000, now known as NekRS, is utilizing optimized computational kernels from the libParanumal library, which is built on the OCCA performance portability library. OCCA defines a kernel language for writing accelerated code. Using just-in-time compilation, the kernels are compiled into architecture-specific code. Supported architectures include both NVIDIA and AMD GPUs. Support for Intel GPUs is expected to be available using an OpenMP back end. The libParanumal library provides optimized OCCA kernels for a variety of CFD-related tasks, including matrix-vector products, linear solvers, and algebraic multigrid preconditioning. The design of libParanumal is flexible enough that individual computational kernels can be separately optimized for a particular architecture if necessary.

Adapting an MC neutronics solver to GPUs is challenging because the amount of code within the primary parallel region is very large (more than 20,000 lines). This results in substantial effort to convert a given code to a new programming model. In order to reduce the risk associated with this porting process, ExaSMR has two different MC transport codes to target the two different exascale machines. The Shift code, developed at ORNL, will target the OLCF Frontier machine and OpenMC, developed at ANL, will target the ALCF Aurora machine. Shift currently uses the CUDA programming language to execute on NVIDIA hardware. AMD has developed the HIP language to provide a smooth transition for existing CUDA applications, allowing execution on both NVIDIA and AMD GPUs with minimal changes to existing code. HIP is therefore a natural selection for Shift and is currently the target programming model. OpenMC is currently planning to use OpenMP 5, which allows the offloading of work from the CPU for execution on the GPU. OpenMP 5 is one of the programming models favored by Intel for exploiting Intel GPUs.

Progress to Date

Initial work to accelerate the Nek5000 CFD code with GPUs used the OpenACC programming model. Although this OpenACC port did yield some performance gains on GPUs relative to CPUs, the performance ultimately fell below the expectations for the code. The amount of restructuring necessary to produce a performant GPU implementation led to the creation of NekRS, a new version of the Nek5000 code that has been substantially rewritten to support accelerators such as GPUs. This work has been performed in partnership with the CEED project. NekRS is heavily leveraging optimized computational kernels from the libParanumal library, which uses the OCCA performance portability model. Significant effort has been placed on optimizing a number of the libParanumal kernels used in NekRS. Figure 20 demonstrates one of the campaigns to improve performance of a particular kernel through a series of optimizations. The performance is compared to an empirical roofline model and demonstrates that a very high throughput can be achieved for sufficiently high polynomial order.

For problems of interest to ExaSMR, NekRS has resulted in a 4–5× speedup relative to the original OpenACC version of Nek5000. For a highly-detailed model of a single nuclear fuel assembly, NekRS achieved work rate of 20.7 degrees of freedom per second per time step on 660 nodes on Summit. Relative to the baseline work rate of 3.0 billion degrees of freedom per second per time step, this represents a demonstrated FOM value of almost 7. Assuming ideal weak scaling, this corresponds to an FOM value of 48 when extrapolated to the full Summit machine, indicating that NekRS has made significant progress towards the ExaSMR CFD performance objective. Because the underlying OCCA library is capable of generating code for AMD GPUs as well as NVIDIA GPUs, NekRS appears to be well positioned to execute effectively on Frontier.

During the initial portion of the ECP project, the Shift code was adapted to run on NVIDIA GPUs by rewriting its transport solver in the CUDA language. Following the initial port, the team explored a series of performance optimizations that led to a factor of 5–10 improvement in particle tracking rate in Shift.

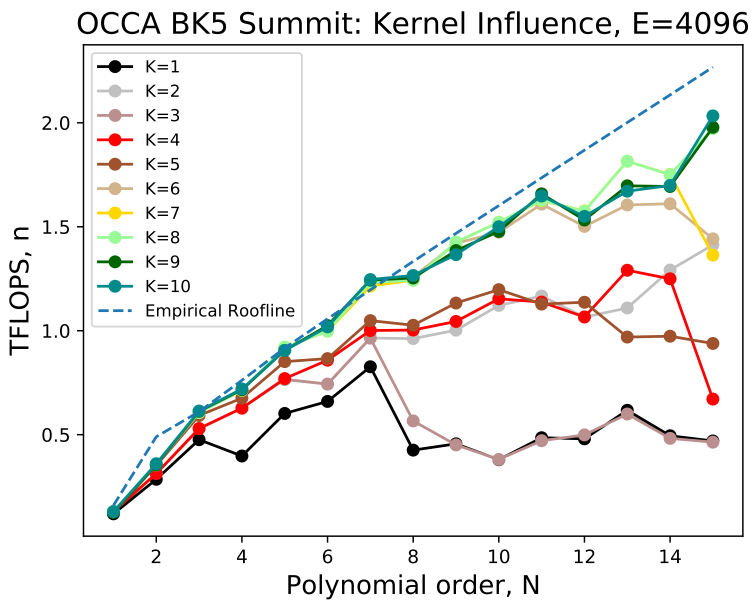


Figure 20: Performance tuning of NekRS OCCA kernels.

These modifications, summarized in Fig. 21, involved breaking large computational kernels into smaller work units to reduce thread divergence and increase device occupancy. Altogether, the Shift GPU port involved writing more than 20,000 lines of CUDA code as well as an additional 15,000 lines of testing code. Using the optimized CUDA code, Shift performed a simulation of a depleted SMR core on 4,096 Summit nodes, achieving a particle tracking rate of 242 million particles per second. Relative to the baseline rate of 10.4 million particles per second on Titan, this represents an MC FOM value of approximately 23. Extrapolating to the entire Summit machine of 4,608 nodes, this equates to a projected FOM of slightly more than 26. With the additional factor of more than 5 gain in theoretical performance moving from Summit to Frontier, Shift also seems very well positioned to meet the ExaSMR MC FOM target.

In order to prepare for porting to the Aurora architecture, several preliminary investigations have been performed with the OpenMC code. These efforts have included updating the XSBench mini-app and the creation of the RSBench mini-app to evaluate the performance of key computational kernels in OpenMC. After investigating a variety of potential programming models, it was decided that OpenMP offered the most favorable tradeoff of efficiency versus simplicity. The SYCL language was evaluated but determined to be too restrictive in the contructs exposed to the programmer to adequately address OpenMC needs. Similarly, the use of OpenCL was also considered but ultimately determined to be too verbose and complicated to be a viable option.

Next Steps

The NekRS code has made significant progress to date towards meeting FOM objectives on the Frontier machine. The next steps include performing performance assessments on both AMD and Intel GPUs once representative hardware is available. These performance assessments should allow the project to determine if there are any concerns related to performance on the architectures for either of the initial exascale machines. If performance issues are identified, it will be necessary to establish focused efforts to develop new optimized computational kernels or otherwise investigate remedies for performance bottlenecks. One additional remaining task for NekRS is to add the ability to perform conjugate heat transfer calculations, allowing the code to evaluate thermal diffusion within regions of a problem containing solid materials. This capability is needed to perform multiphysics coupling simulations within ExaSMR. It is a capability that exists in the original Nek5000 code but has not yet been implemented in NekRS. Conjugate heat transfer is not generally a performance bottleneck, so this effort is not viewed to introduce significant risk or invalidate existing performance assessments conducted with NekRS.

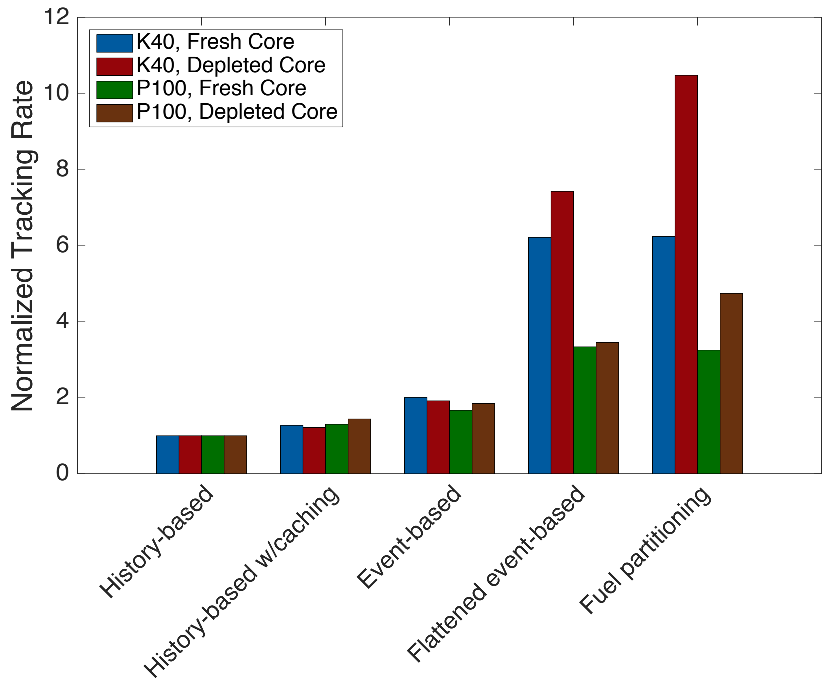


Figure 21: Performance gains through Shift algorithmic improvements on NVIDIA GPUs.

As noted in the previous sections, Shift is targeting the use of AMD GPUs by converting existing CUDA code to HIP, which is capable of executing on either AMD or NVIDIA GPUs. The next major task is to perform the porting of Shift's CUDA routines to HIP. AMD already has HIP libraries to replace many NVIDIA tools, including drop-in replacements for the NVIDIA Thrust and cuRAND libraries that are used within Shift. Once Shift has been ported to HIP, it will be necessary to evaluate the performance of the code on AMD hardware to determine if it is on track to meet FOM targets. It is expected that due to differences between the NVIDIA and AMD GPU execution models, there may be performance deficits that need to be addressed through additional algorithmic investigations.

Finally, in order to prepare the OpenMC code to run on Intel GPUs, a significant effort will be required to port it to OpenMP with an accelerator offloading model. OpenMC currently uses OpenMP for CPU-based multithreading, but has not yet started the process of porting to GPUs. This conversion, along with associated performance evaluation and algorithmic optimizations, is expected to require substantial effort over the next two years.

4.4 MFIX-Exa

Carbon capture and storage (CCS) technologies (e.g., oxy-fuel combustion, chemical looping combustion, and post-combustion capture systems) offer the most promising approaches for reducing CO_2 emissions from fossil fuel power plants. 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 scaleup of such reactors is known to be unreliable and the current approach requires building and testing physical systems at increasingly larger intermediate scales. The cost in both dollars and development time of having to build and extensively test systems at multiple intermediate scales is prohibitive. The development of high-fidelity computational tools capable of simulating such systems to enable the design and optimization of emerging CCS technologies is critically important to control costs and reduce the risk of designs failing to meet performance standards. The development of such tools and the simulations required is not possible without exascale computations. This work specifically targets scaleup of Chemical Looping Reactors (CLRs) using NETL's 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-Discrete Element Method (DEM) is an approach that allows for tracking of individual particles (DEM portion) within a continuum fluid phase (CFD portion). To date, the focus of existing MFIX CFD-DEM efforts has been on validation and development of physical models in the context of a relatively basic computational framework. MFIX-Exa will integrate expertise in high performance computing 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 containing 5B 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 representing the full-loop CLR geometry, 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 30.

4.4.2 MFIX-Exa: KPP Stretch Goal

The stretch goal is to run the challenge problem simulation until the gas concentrations at 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 major components of the CLR (air reactor, fuel reactor, and loop seal). The (cross-section averaged) CO_2 mass fraction at the exit of the fuel reactor will 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 disregarding the initial period, 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 5% difference between them, 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: Progress Towards Advanced Architectures

GPU Strategy

MFIX-Exa is an AMReX-based (§ 8.3) application and as such, has fully adopted AMReX's GPU strategy. To achieve the highest degree of portability, MFIX-Exa kernels were converted from FORTRAN to C++ utilizing AMReX mesh and particle iterators. This approach implicitly cycles through GPU streams on each iteration, keeping work in a given loop ordered but taking advantage of unused GPU resources when available. The raw mesh and particle data in MFiX-Exa is placed in managed memory. Individual kernels are launched on GPUs using AMReX's C++ lambda-based launch system that offers support inlining and seamless transition between CPU and GPU. Many of the more common operations in MFiX-Exa, such as global reductions, particle neighbor lists, and others take advantage of the native support in AMReX. Additionally, MFIX-Exa utilizes GPU aware AMReX-provided functions for common parallel communication operations for mesh and particle.

Because MFIX-Exa exclusively uses AMReX's internal GPU capabilities, MFIX-Exa contains no explicit CUDA, OpenACC or OpenMP GPU bound kernels. This should significantly reduce the work needed to migrate to AMD and Intel based accelerators once they are supported by AMReX.

Table 30: 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, the equations accounting 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 using a soft-sphere approach, and the contact forces are modeled 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 be used for the challenge problem as well. Model validation is outside the scope of the current project and will not be conducted.
Simulation details: problem size, complexity, geometry, etc.	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 $4.2\,\text{m} \times 1.1\,\text{m} \times 0.22\,\text{m}$ (height×width×depth). 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-quarters of the domain will not contain active computational cells and, therefore, will be removed to reduce memory usage. The domain can be fully covered by roughly 40k individual grids of 32^3 cells each; if only $\frac{1}{4}$ of the domain is relevant, one should need closer to 10k grids or a total of roughly 330M 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 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: $2FeO + \frac{1}{2}O_2 \rightarrow Fe_2O_3$ • Fuel Reactor: $4Fe_2O_3 + CH_4 \rightarrow 8FeO + CO_2 + 2H_2O$
Demonstration calculation requirements	Initial conditions: Minimizing the initial transient time (from guessed initial condition to 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 as move-gas in the standpipe and L-valve to facilitate solids movement. An operating temperature of 1,100 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 hours at full system.

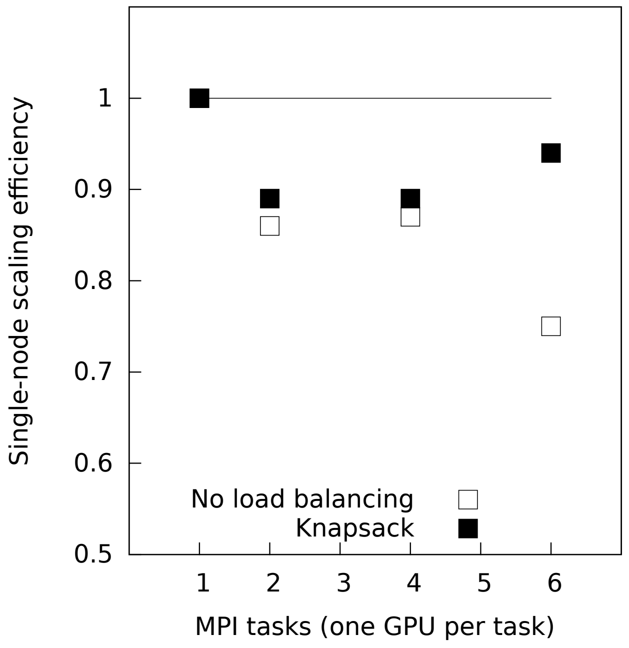


Figure 22: MFIX-Exa particle-advance scaling efficiency on a single Summit node with one GPU per MPI task. Closed symbols indicate runs with Knapsack load balancing.

Presently, MFIX-Exa has no external software dependencies other than AMReX and the typical software stack requirements such as working C++ compilers and MPI. However, this could change in the future as additional physical models are added.

Progress to Date

The MFIX-Exa fluid and particle solvers were migrated to GPU over the past fiscal year with the primary focus being on-node performance. The performance of MFIX-Exa on GPUs was assessed using a prototypical CLR geometry that is representative of MFIX-Exa's challenge problem. Here, an air reactor, fuel reactor, loop-seal, and L-value are initially seeded with high solids concentrations whereas the riser, crossover, and cyclone are initialized with a light solids loading. This disparity in local solids inventories across the device components gives rise to load imbalances in particle work while the full-loop geometry creates large void spaces whereby there is neither fluid nor particle work.

The total initial solids inventory is approximately 40 million particles, and the domain is decomposed into 36 grids ($32 \times 32 \times 32$) for a background $384 \times 32 \times 96$ mesh. The timing study focused on the average time per particle advance because at the time the timing study was conducted, the AMReX embedded boundary aware linear solvers were not fully ported to GPU.

A CPU-only run using 36 CPUs on a single Power9 Summit node with one MPI task per core averaged 1,094 seconds per particle advance. A $7\times$ speed up was observed when using one MPI task with one GPU and a $30\times$ speedup was observed when utilizing 6 MPI tasks with one GPU each.

A single-node parallel efficiency study was also conducted for the particle advance whereby the single-task-single-GPU run was compared to runs utilizing 2, 4, and 6 MPI tasks, with one GPU each. The first test did not use load balancing whereas the second test used MFIX-Exa's dual grid capability with the Knapsack load balancing strategy. As shown in Fig. 22, the tests without load balancing decrease in efficiency by 50 % as the number of MPI tasks is increased from 2 to 6. Conversely, a 94 % scaling efficiency is seen when the Knapsack load balancing is enabled.

Next Steps

MFIX-Exa plans to adopt changes suggested by the AMReX Co-Design Center for maximum portability and performance. As AMReX adds support for AMD accelerators, MFIX-Exa will leverage the expertise to incorporate the necessary changes to utilize those functionalities in preparation for Frontier.

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 [5] 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 the completion of 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 the coupling of two advanced and highly scalable gyrokinetic codes, XGC and GENE. The former is a Particle-in-Cell (PIC) code optimized for the treating the edge plasma, while the latter 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 whole device kinetic transport kernel for the WDM. A major thrust of the project is the coupling framework EFFIS 2.0 (End-to-end Framework for Fusion Integrated Simulation 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 is to be enhanced with communication-avoiding methods, task-based parallelism, in situ analysis with resources for load optimization workflows, and deep memory hierarchy-aware algorithms.

The resulting exascale application will be unique in its computational capabilities and will have potentially transformational impact in fusion science, e.g., 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 as well as 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 which aims to attain a tenfold energy gain. The physics objective is to predict one of the most important indicators for energy confinement in the H-mode: the plasma pressure "pedestal" height and shape. Realization of the H-mode with high edge plasma pressure and mild pedestal gradient is critical to the performance and success of ITER. Efficiency of the fusion burn is virtually determined by the height of the pressure pedestal at edge. The team's strategy will be to use WDMApp, which is focused on coupling the continuum code GENE in the core region and the PIC code XGC at the edge. Alternate code-couplings will also be considered for risk mitigation, i.e., the PIC code GEM in the core region.

The targeted minimum requirement for accuracy is 5% relative error. Due to 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 platform(s) 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 31 on the minimum criteria for the FOM run(s).

4.5.2 WDMApp: Figure of Merit

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

$$\text{FOM} = \frac{N_m N_t N_p}{\text{wall clock time}} \times 10^{-9}\,, \tag{12}$$

where N_m is the number of grid vertices, N_t is the number of simulation steps in specified wall clock time (e.g. 100 s), and N_p is the number of particles per grid vertex in the edge region. The baseline FOM is 1.76

Table 31: 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 one trillion in XGC
Demonstration calculation requirements	2,500 time steps
Resource requirements to run demonstration calculation	3 hours to run 2,500 time steps on 100% of exascale computer

on Titan based on a Probability Distribution Function (PDF) data exchange.

4.5.3 WDMApp: KPP Stretch Goal

The stretch goal for the project is to carry out core-edge coupled electrostatic turbulence simulations, modeling 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 infrequent exchange of PDF data. Further improvements are expected from asynchronous electron push in GPU and improvements in code-coupling efficiency and load-balancing.

4.5.4 WDMApp: Progress Towards Advanced Architectures

GPU Strategy

The goal of WDMApp is to predict ITER's edge pedestal height and shape by performing core-edge coupled simulations using high-fidelity gyrokinetic codes. To achieve the large-size ITER simulation aimed on exascale computers, an efficient weak-scaling is more relevant in the WDMApp project. Edge pedestal determines the steady-state fusion performance to a large degree in ITER. The edge gyrokinetic code XGC is used for the edge region and the two core gyrokinetic codes GENE and GEM are used for the core region. XGC and GEM are particle-in-cell codes and GENE is a continuum code. For risk mitigation, two different types of core gyrokinetic codes are utilized to couple to the XGC particle-in-cell code that is the only production gyrokinetic-code available for edge region. Due to the scale-inseparable interaction between the background plasma dynamics and the turbulence dynamics in complicated edge geometry, the edge code XGC takes up most of the computing time in the core-edge coupled simulation. On the other hand, the core codes assume a prescribed Maxwellian background plasma and calculate only the perturbed part of the particle distribution function, in contrary to calculating the total distribution function in XGC. Naturally, XGC occupies majority number of the HPC nodes. Hence, a good scalable GPU performance of XGC is a critical factor in the overall performance of WDMApp on exascale computers. Nonetheless, a good GPU performance of the core codes can be helpful in raising the FOM number. Thus, the project attempts to port and improve all three codes on GPU.

The core-edge coupled WDMApp has not been using GPUs yet. Running the coupled WDMApp code on GPU would require the core-edge coupling of the kinetic electron dynamics through a spatial interface. This work is presently on-going. Since the Poisson equation, which must be accompanied by global data communication, is solved at every ion time step in the WDM solver, only the subcycled kinetic electron dynamics are ported to GPUs in XGC. So far, the effort has been in the GPU porting of individual codes to prevent the GPU computing from becoming a hurdle when the core-edge kinetic electron coupling is established in the future.

XGC has two compute-heavy kernels: Kinetic electron push and nonlinear Fokker-Planck collisions. These two kernels take up over 95 percent of the compute time in production runs if GPUs are not utilized. The working strategy has been that these two heavy-weight kernels occupy GPUs while other light-weight kernels reside on CPUs, with GPUs and CPUs working asynchronously with an approximate work-load balancing. The electron push kernel is more offload-dependent than the nonlinear collision kernel. In order to mitigate this offload-dependence, XGC makes use of the Cabana particle library, which has been developed in the ECP-COPA project where XGC is part of, to allow the electron push kernel being interfaced to Kokkos. Kokkos utilizes each machine's native GPU-offload method as backend. For the nonlinear collision model, XGC uses OpenACC directives at the present time. For a better utilization of the Kokkos C++ library, all the Kokkos-calling routines in XGC have been converted from Fortran to C++.

GENE uses Fortran 2003/2008 constructs and exposed a number of compiler issues. OpenMP threading of MPI is not utilized. The GPU support entirely in Fortran has turned out to be not feasible at least in the short term due to lack of stable compiler support for OpenACC/OpenMP. Similar issues also prevented the use of CUDA Fortran. The approach chosen now for GPU off-loading is to use CUDA/C++ for the GPU kernels. We have separated out low-level computational kernels that form an abstracted "operations" class that can either call regular Fortran code or interfaces with GPU kernels implemented directly in CUDA C/C++. The idea behind this approach is similar to, say, using BLAS. The main code calls 'daxpy', and the underlying implementation may be a CPU-optimized BLAS or a GPU optimized BLAS. The switching between implementations is achieved using type-bound procedures in the operations class. Unfortunately, most of the computational kernels in GENE are not standard operations, so many custom interface and implementations needed to be created, though we leverage cuBLAS and cuFFT as well. Fortran 2003 ISO_C_BINDING is used for interoperability. CUDA managed memory is employed to support an incremental transition to CUDA kernels while other parts of the code still use Fortran CPU implementations. On the CUDA C++ side, a library named "gtensor" has been implemented to make writing CUDA kernels easier. It is modeled after the xtensor library `https://github.com/QuantStack/xtensor` but provides support for multi-dimensional arrays living in either host or device memory, similar to the CUDA thrust library or Kokkos. This makes it possible to naturally access Fortran multi-dimensional arrays on the CUDA side. In addition, gtensor supports lazily evaluated expressions using C++ template metaprogramming. This enables us to generate complex expressions that can then be evaluated in a single CUDA kernel all in one go.

GEM is a Fortran particle-in-cell code using OpenACC for GPU-offloading. OpenMP threading is not used currently. Kinetic electrons are not used in the GPU version of GEM, yet. Most of the effort has been spent in the communication minimization. Particle array, field array, equilibrium and profile array data are stored in GPU by default. Ion push and charge kernels are efficiently imported into GPU. Ion move and shift kernels are offloaded to GPUs but needs improvement: they occupy 60 % of computing time.

Progress to Date

A summary of the single-node, optimized CPU + GPU performance of the three WDMApp codes is compared to the one MPI rank per core CPU-alone performance in Table 32.

XGC

The baseline, single-node (CPU + GPU)/CPU acceleration factor is established to be 71× for the whole XGC code, using 1 MPI rank per core without any threading for the CPU-alone case and 1 MPI rank per GPU (6 MPI ranks per node) with 14 OpenMP threads per MPI (7 physical cores and 2 hyperthreads per core) for the CPU + GPU case. In order to avoid the possibility of memory overrun on single node, the problem size is weak-scaled down to 18.9M ions and 18.9M electrons on 3.74k grid vertices.

The whole production version XGC scales almost linearly all the way to the maximal number of Summit nodes (4,096 nodes tested out of 4,608 nodes), with asynchronous usage of all the CPUs and GPUs (see Fig. 23). Again, 6 MPI rank per node and 14 OpenMP threads per MPI rank (7 physical cores with 2 hyperthreads per core) have been used. Excellent scalability of the whole XGC code comes from the excellent weak scalability of the GPU-enabled kernels together with the minimal data-communication engineering.

Table 32: Single node acceleration of WDMApp component codes based on 1 MPI rank per core for CPU-alone performance.

Code	CPU method	CPU + GPU method	(CPU + GPU) / CPU
XGC	1 MPI/core, 0 threads, 1 node	1 MPI/GPU (6 ranks), 14 threads per MPI, 1 node	15
GENE	1 MPI/core, 0 threads, averaged over 42 nodes using 6 cores per node	6 CUDA-aware MPI ranks, 0 threads, 1 node	7.03
GEM	1 MPI/core, 0 threads, 1 node	1 MPI/core, 0 threads, 1 node	43

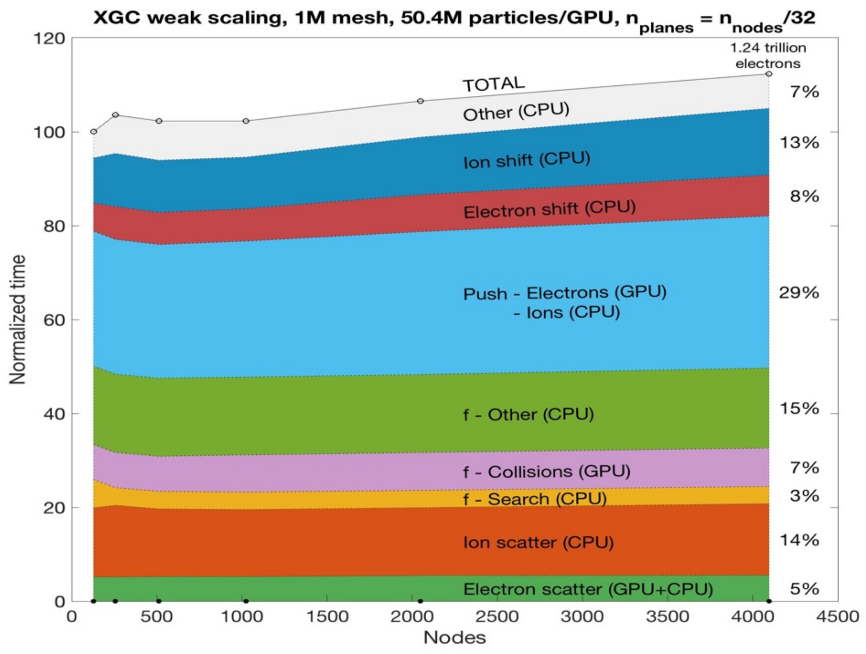


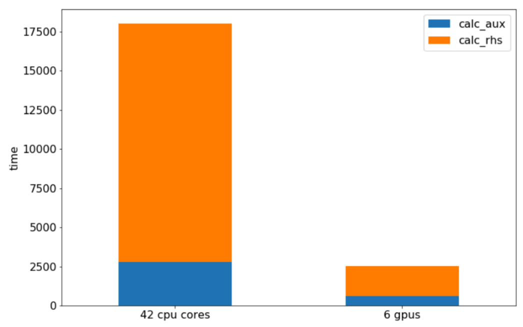
Figure 23: Weak-scaling performance of each XGC kernel for CPU + GPU computing on Summit.

Figure 23 shows breakdown of each kernel performance on Summit, weak-scaled on CPU and GPU together, starting from 128 nodes to 4,096 nodes. It can be seen that there is only about 12 % degradation in the measured compute time for the whole production code from 128 nodes to 4,096 nodes. Total number of weak-scaled particles (1.24 trillion ions and 1.24 trillion electrons) on the full-scale Summit corresponds to roughly the actual number of particles being used for high-fidelity ITER simulations at the present time.

GENE

Most of GENE runs to this point have been single-node runs, using the 6 GPUs per node. We use CUDA-aware MPI for communication. Due to the (relatively) high cost of host-device transfers and relatively low computational intensity of the code (it is mostly stencil computations, with some linear algebra matrix-vector products), it was necessary to fully port the timeloop to GPU. At that point, very little work is left for the CPUs, so we only run 6 MPI processes per node, which allows us to give the GPU kernels large pieces of work and reduces the amount of communication per ghost cells.

The following timing numbers are obtained from single-node Summit runs. We compare the performance of the code running on 42 CPU cores to running using the 6 GPUs (and 6 CPU cores to drive them). In



	42 cpu cores	6 gpus	speed-up
RK_step	77142.86	10966.33	7.03

	42 cpu cores	6 gpus	speed-up
calc_aux	2785.71	595.67	4.68
calc_rhs	15238.10	1933.33	7.88

Figure 24: Break-down of a single Runge-Kutta stage for the GPU GENE run.

order to have a direct comparison, we actually extrapolate the CPU performance on 42 nodes from runs that were using just 6 CPU cores, by dividing the timing numbers obtained by 7. This allows us to use exactly the same grid and same domain decomposition. This approach likely overestimates CPU performance a bit, due to the fact that we don't have the overhead of a domain decomposition into smaller pieces (and hence more ghost points, more communication) and that fewer CPU cores compete for shared resources like higher-level cache and memory bandwidth.

The overall timeloop using 6 GPUs is faster by a factor of 7.03x compared to using 42 CPU cores only.

Figure 24 breaks down a single Runge-Kutta stage, the basic building block for the timeloop into `calc_aux`, which computes some auxiliary quantities (e.g., gyro-averaged electrostatic potential), and `calc_rhs`, which computes and adds up the various terms of the right-hand side. Our initial focus had been on `calc_rhs`, so `calc_aux` has not seen much optimization yet. `calc_aux` involves a number of MPI communications; now that the computation has been accelerated very substantially the communication starts to show up as significant operations in the profiles. Break-down of individual terms that go into the right-hand side calculation shows that many kernels already achieve speed-ups of 15–30×, while some require more investigation and work. The calculation of the nonlinearity (`add_nl`) is the most expensive part of the right-hand size calculation, as it itself consists of a number of kernels, as well as FFTs to/from Fourier space.

GEM

Using one MPI rank per core without threading and OpenACC off-loading, GEM shows 43× acceleration of CPU + GPU performance compared to CPU alone on single node. The weak-performance degradation in the CPU + GPU computing from 1 node to 32 nodes is about 30 % (see Fig. 25). The cause for nonlinear increase in the time per step needs to be investigated, but MPI interference from other code runs could be non-negligible.

Next Steps

For the WDMApp coupled codes on Frontier and Aurora, a mixed offload strategy will be employed using Kokkos, HIP and OpenACC/OpenMP.

The most urgent task that WDMApp faces is to port the coupled XGC-GENE and XGC-GEM on Summit GPUs. This requires the coupling of kinetic electron dynamics between core and edge through spatial interface layer. After this is achieved, the coupling framework EFFIS2.0 will be used to enhance the performance of the coupled WDMApp codes up to the full Summit nodes. Optimization strategy of the code coupling on Aurora and Frontier will be established utilizing Summit within the EFFIS2.0 framework.

Individual codes will also be prepared to enable the coupled WDMApp code to run on Aurora and/or Frontier in the EFFIS2.0 framework. Because of the dominance of XGC in the performance of the WDMApp coupled codes, a particular attention will be given to its performance on Aurora and Frontier, in collaboration with the OLCF and ALCF facility liaisons and staff as well as vendors. XGC will be completely converted to C++, optimizing the Kokkos and OpenMP offloading capabilities.

For GENE and GEM, more time will be spent in optimizing individual non-performant kernels on Summit. The performance gains so far have come largely from porting operations straightforwardly, and from consolidating many small kernels into larger ones. Weak-scaling test to at least 1,024 Summit node will

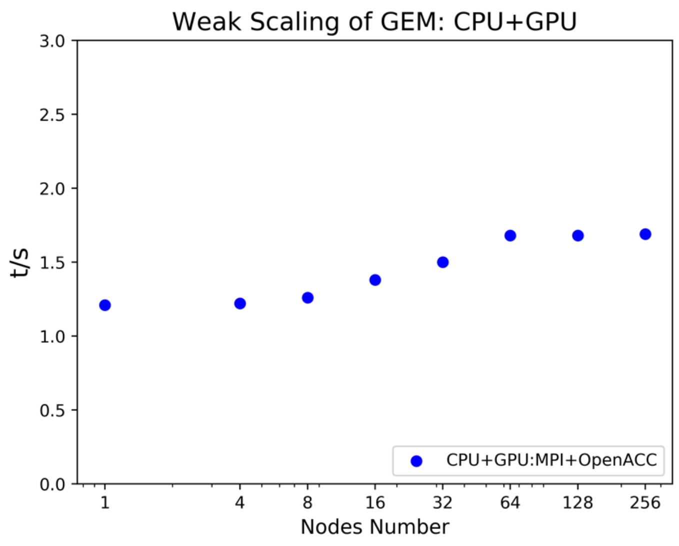


Figure 25: Weak scaling of GEM for CPU alone (one MPI rank per core) and CPU + GPU.

also be the next step.

Since GEM is utilizing OpenACC, we do not foresee serious GPU off-load issue on Aurora or Frontier, but we need to implement OpenMP threading as the next step. From the experience gleaned from XGC, multiple threading and hyperthreading could give additional performance enhancement.

The profile of GENE is rather flat, with about 30 kernels being where most of the time is being spent, so focusing on a small number of kernels will not make a big difference in overall performance. For Frontier, we will have to adapt to AMD's GPU programming model. Currently, GENE has a few kernels that are explicitly written in CUDA, so we will probably use the HIP compatibility layer there. Most of the GPU code is generated by gtensor, which is a performance-portable approach in some sense similar to Kokkos or Raja. Right now there are backends for generating NVIDIA CUDA kernels as well as regular C++ host code. Adding support for AMD ROCm/HC should be quite straightforward, and won't require any changes in the actual application code.

For the EFFIS2.0 framework, we are investigating the use of CudaMPI as one of the communication strategies/engines underneath the ADIOS-API. We are also investigating the use of in-line functions in ADIOS, which can allow one code to call another code, so that both of them can be on GPUs.

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: elementary physics, nuclear physics, material science, chemistry, and biology and even play a role in astrophysics and cosmology. A total of 30% 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 applications impacting lives, from sterilizing food or toxic waste to implanting ions in semiconductors, treating cancer, or 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 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 acceleration processes that develop over a large range of space and time scales. 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 injection, emittance transport, beam loading, tailoring of the plasma channel, and tolerance to nonideal effects (jitter, asymmetries, etc.) that are needed for the design of high-energy colliders, will necessitate a series of tens or hundreds of runs. This will require orders-of-magnitude speedup 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 (§ 8.3) 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, among others, 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 both the particles and the mesh.

The new software will enable, on exascale supercomputers, 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 not possible with present technologies, will bring huge savings in the research leading 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-TeV 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 the modeling of multi-TeV 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:

- Total energy gain of plasma accelerator of at least 100 GeV, using five accelerator stages or more.
- Maximum wall-clock time is the same as baseline.

Details on the challenge problem are given in Table 33.

4.6.2 WarpX: Figure of Merit

The figure of merit for WarpX is defined in the following

$$\text{FOM} = \frac{(\alpha N_c + \beta N_p) N_t B_A}{\text{wall clock time}}\,, \tag{13}$$

where

N_c:	number of grid cells,
N_p:	number of particles,
α, β:	relative cost of grid push and particle push,
N_t:	number of time steps.

The "Boost from Algorithms" (B_A) term includes additional boost coming from algorithm developments and is constrained $B_A \leq 5$.

The FOM without B_A gives the "raw" speed from running, on a newer and bigger machine, the same problem faster or at higher resolution (more grid cells, particles and time step) in the same time (just based

Table 33: 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 both laser-driven and charged-particle source terms.
Numerical approach, algorithms	PIC using finite-difference (FDTD) and/or pseudo-spectral (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, etc.	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 such that performance is accurately measured using a preloaded plasma column. Measure FOM for FDTD and FFT-based solvers.
Resource requirements to run demonstration calculation	2 hours on 100% of the machine.

on strong/weak scaling). Beyond this, speedup comes from (a) lower number of grid points and particles (using AMR) and (b) lower number of time steps, using a novel large time step (LTS) algorithm. Hence, B_A is the combination of two possible algorithmic boosts:

$$B_A = B_{\text{AMR}} B_{\text{LTS}} \, , \tag{14}$$

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 $\sim 2 \times 10^{10}$ on 6,625 compute nodes of Cori (FOM scaled to the full machine).

4.6.3 WarpX: KPP Stretch Goal

The goal of the project is to demonstrate the capability to perform the modeling of tens of consecutive multi-GeV stages. For a multi-TeV 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 ($B_A < 5$). The stretch goal is thus to demonstrate the capability to model as many consecutive multi-GeV stages as possible.

4.6.4 WarpX: Progress Towards Advanced Architectures

GPU Strategy

WarpX relies on the ECP AMReX framework in order to achieve performance portability on CPU and GPU, with a single-source code base. More specifically, AMReX provides a C++ function `ParallelFor` as well as similar reduction primitives, which are converted to GPU kernel launches when the code is compiled for GPU (and is converted to a simple `for` loop when the code is compiled for CPU). Note that this solution is similar to that used in Kokkos or Raja. WarpX has been extensively refactored, in order to use this performance-portable framework throughout the code.

In addition, AMReX provides containers for particles and mesh data. When the code is compiled for GPU, these containers are allocated in GPU managed memory. This means that host-device memory transfers are performed automatically whenever the data is used on the host or the device, and thus that the code does not need to contain explicit calls to e.g. `cudaMemcpy`.

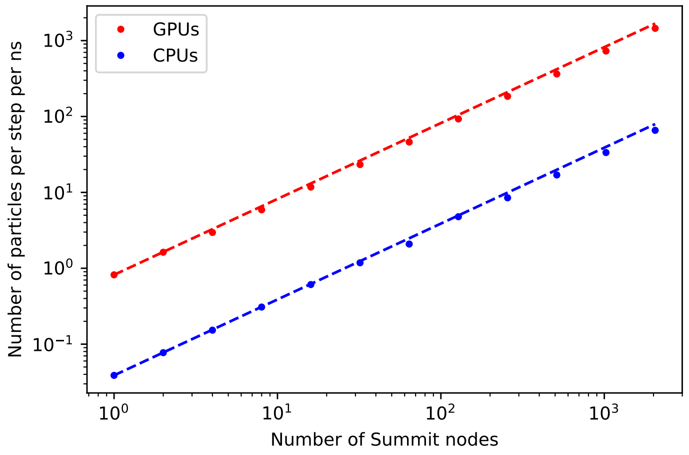


Figure 26: Results of a weak scaling study on a uniform plasma benchmark with WarpX, comparing GPU and CPU performance on Summit. The colored circles indicate simulation results, while the dotted line illustrates perfect weak scaling. Both cases used 8 particles per cell, Esirkepov current deposition, and 3rd-order interpolation. The GPU-accelerated runs used 6 GPUs per node, with 1 MPI task per GPU. The CPU-only runs used all 42 available POWER9 cores on a Summit node, using 6 MPI tasks and 7 OpenMP threads per node. Both runs exhibit excellent weak scaling. On 2,048 Summit nodes, the GPU-accelerated run was approximately 22 times faster that the CPU-only run.

During a typical WarpX GPU simulation, *all* the compute operations now run on the GPU. As a consequence, all the simulation data is automatically transferred to the device *on the first timestep* (via managed memory transfer), and then stays on the device for the rest of the simulation. As a consequence, host-device transfers are only needed when writing output data to disk.

Progress to Date

Prior to Summit, WarpX was running efficiently on CPUs and KNL architectures (NERSC Cori) via low-level Fortran routines (PICSAR) which were called from a high-level C++ code (WarpX code base). When transitioning to GPU, we initially experimented with OpenACC, in order to keep the same Fortran code base. However, we found this approach to be hindered by the limited availability and capabilities of compilers that support OpenACC in Fortran for GPU.

In the anticipation of Summit, we therefore transitioned to a C++-only strategy, based on the above-mentioned portability layer from AMReX. WarpX was extensively refactored, and all particle-mesh routines were converted to C++. The transition to a C++-only core enabled us to use a wider range of compilers for various architectural targets and single-source programming for both OpenMP, CUDA and upcoming HIP programming models improved code maintainability.

Benchmarks on Summit revealed excellent weak scaling, and a considerable speed-up from the use of GPUs, as shown on Fig. 26.

We also performed comparisons between Cori's KNL architecture and Summit's V100 architecture. In general, we noticed that the core compute routines (e.g. particle push, field update on the mesh, etc.) were typically much faster on V100, but that the routines associated with MPI communications (e.g. packing/unpacking of MPI buffers before/after MPI communications) did not benefit from the same speed-up. Upon further investigation, we noticed that these routines involved many small kernel launches, and that the kernel launch overhead (when running on V100) became a bottleneck in this case. These communication routines were therefore refactored, so as to have as few kernel launches as possible. For instance, in the case of guard cell exchanges, this was achieved by having a single kernel that loops over the 3D boundary of a subdomain and packs the guard cell data into different MPI buffers (as opposed to having one kernel launch

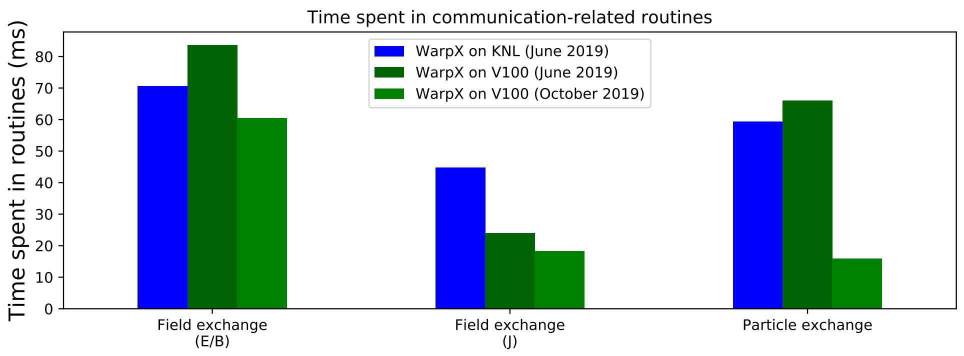


Figure 27: Time spent in communication-related routines (e.g. packing and unpacking of MPI buffers) in a benchmark involving a typical boosted-frame simulation of laser-wakefield acceleration with WarpX. The code was run on Cori (blue) and Summit (green), using 900 KNL nodes and 900 V100 GPUs on 150 Summit nodes respectively.

per subdomain face/edge). The resulting speed-up is shown in Fig. 27, which compares the time takenby these routines before this refactoring (June 2019) and after this refactoring (October 2019).

Next Steps

As mentioned above, WarpX relies on the AMReX framework for portability across different computing architectures. AMReX currently supports NVIDIA GPUs (through CUDA), and is working towards support for AMD GPUs through HIP. AMReX is also expected to support Intel GPUs via DPC++ or SYCL. On the WarpX side, supporting these architectures should involve only minimal changes, since the WarpX code base is isolated from these low-level programming models through the use of AMReX's `ParallelFor` function (and similar other high-level parallelization functions).

In addition, as raw compute power will increase on Aurora and Frontier, I/O is expected to become more and more of a bottleneck. The WarpX team is keenly aware of the potential limitations associated with this, and is exploring mitigation strategies, including fast, scalable I/O through ADIOS2, as well as in-situ data processing/reduction with Sensei, Ascent, or ADIOS2 sustainable staging transport.

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 34) 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.

Table 34: 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

5.1 ExaStar

This project develops a new code suite, Clash, which will be a component-based multi-physics AMR-based toolkit, accurately simulating coupled hydrodynamics, radiation transport, thermonuclear kinetics, and nuclear microphysics for stellar explosion simulations. Clash will reach exascale efficiency by building upon current multi- and many-core efficient local physics packages integrated into a task-based asynchronous execution framework based on current AMR technology. The fundamental goal in the development of Clash is to understand the production of the chemical elements in these explosions, particularly those heavier than iron. While astronomical observations reveal the production of the heaviest nuclei began early in galactic history, it is not known how and where these elements were formed. To address this topic via laboratory measurements, a series of Nuclear Science Long Range Plans have supported construction of radioactive ion beam facilities, culminating in the Facility for Rare Isotope Beams (FRIB). While FRIB is designed to acquire extensive data on the nuclei relevant for astrophysical nucleosynthesis, its end science 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 takes place. Through a better understanding of the sites where 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 three-dimensional simulation of the first 2 seconds of evolution after iron core bounce of a core-collapse supernova. The progenitor star model will be chosen at run time from the best available models. The most likely progenitor models are (a) the solar metallicity 12 solar$_{\text{mass}}$ progenitor of Sukhbold, Woosley, and Heger [6], chosen because it represents, in some sense, the "center" of the distribution of massive stars that produce core collapse supernovae (CCSNe), or (b) the binary merger model of Menon and Heger [7], 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 kilometer at the surface of the proto-neutron star (i.e., in the inner 100 km or so of the event). At least 20 energy groups will be used to resolve the spectra of neutrinos of all flavors (electron, mu, tau, and their anti-particles) from 0 to 300 MeV. An approximation to general relativistic gravity utilizing at least 12 moments in a multipole approach will be used, with the option to have a more realistic treatment if possible (e.g., conformally flat approximation). 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 Equation of State (EOS) that will provide pressures, entropies, and all other required thermodynamic values as required by, e.g., the hydrodynamics. The available set of coupled rates and EOS tables will include, at least, the SHF0 EOS of Steiner, Hempel, and Fischer [8]. Details on the challenge problem are listed in Table 35.

Table 35: 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, etc.	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 (electron, mu, tau, and their anti-particles) from 0 to 300 meV; an approximation to general relativistic gravity utilizing at least 12 moments in a multipole approach will be used, with the option to have a more realistic treatment if possible (e.g., conformally flat approximation)
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 hours

5.1.2 ExaStar: KPP Stretch Goal

The ExaStar stretch goal is a three-dimensional 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 spacetime solver and a GR magneto-hydrodynamics solver, which represent a significant advance in code capability. The initial conditions of the problem will consist of two neutron stars in a quasi-circular orbit with a separation sufficient to complete 2–3 orbits before mergers. The simulation domain will cover 3,000 km (roughly 300 neutron star radii) with a resolution using AMR of order 100 m. Instead of Newtonian gravity, dynamical spacetime solver using a high-order finite differencing scheme will be employed with corresponding Kreiss-Oliger dissipation terms to discretize spacetime variables in the BSSN formalism. In addition, a general relativistic ideal magneto-hydrodynamics solver will use the metric from the spacetime solver to advance the fluid dynamics while assuring 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 of order 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. The simulation will cover roughly the first 20 milliseconds 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: Progress Towards Advanced Architectures

GPU Strategy

The ExaStar codes are based on the AMReX (§ 8.3) library for block-structured adaptive mesh refinement. AMReX currently supports the use of CUDA, OpenACC and OpenMP. Internally, AMReX relies on CUDA for NVIDIA accelerators, and is in the process of developing core-level support for AMD accelerators based on HIP. AMReX provides the ability for users to construct loops over data using mesh and particle iterators. AMReX iterators implicitly cycle through GPU streams on each iteration, keeping work in a given loop

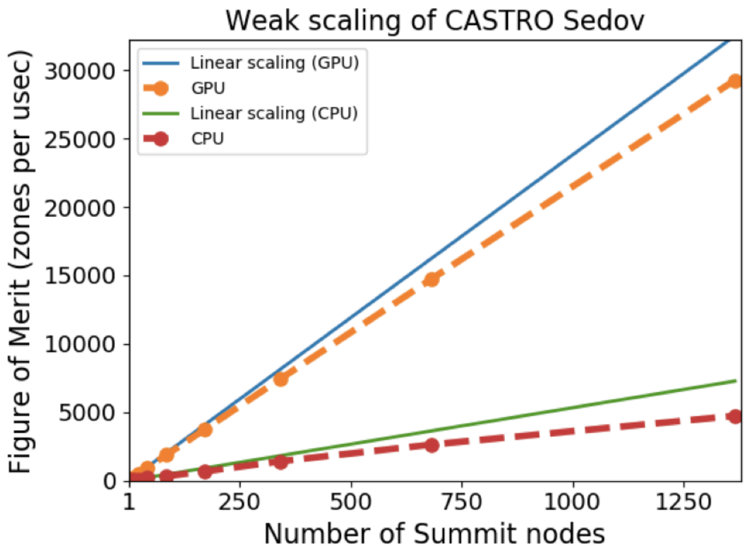


Figure 28: Weak scaling tests were performed for the Castro hydrodynamics solver for the Sedov problem. The GPU version is about an order of magnitude faster than the CPU version. Each configuration makes optimal use of all available node resources.

ordered but taking advantage of unused GPU resources when they are available. While Castro makes extensive use of the GPU-enabled capabilities of the AMReX iterators(using C++), FLASH (Fortran) relies primarily on support for alternative directive-based offloading strategies provided by AMReX.

FLASH GPU-enabled units have been primarily developed using OpenACC and OpenMP offload directives. We are presently transitioning all OpenACC kernels to OpenMP 4.5 to future-proof the code on Frontier and subsequent platforms.

Progress to Date

Castro's hydro solver has been ported to GPUs using a combination of the AMReX launch mechanism and custom kernel launches of CUDA Fortran routines. GPU performance is compared to CPU performance using all of the available node resources in a weak scaling benchmark in Fig. 28. Current work is concentrated on moving all kernel launches to the AMReX launch mechanism.

Efforts to adapt the FLASH hydrodynamics solvers to GPUs are being focused on a new magneto-hydrodynamics solver, Spark. That work is underway.

Thornado is developed as a collection of modules for neutrino radiation transport that can easily be incorporated into FLASH or Castro. For this reason, a primary focus of Thornado development is node-level performance on heterogeneous computing systems. GPU capability has been added via GPU directives and linear algebra libraries. In particular, the explicit and implicit updates in the neutrino transport solver have been moved to GPUs. Figure 29 demonstrates the performance of Thornado on one Summit node using GPU directives.

The equation of state for dense matter is an especially important feature of ExaStar challenge problem physics. WeakLib is a library and collection of microphysics for dense matter neutrino interactions that includes a tabulated nuclear equation of state. The relevant interpolation kernels were ported to the GPU with GPU directives. OpenACC or OpenMP offloading can be used by using the relevant preprocessor macro. The performance of important opacity interpolation modules was tested indirectly in the Thornado benchmark for Fig. 29.

The performance of Castro's reaction network was evaluated on SummitDev. As shown in Fig. 30 below, this approach works well for very small networks, but register pressure prevents networks larger than 15 species from running efficiently. This motivates further work to parallelize the reaction rate evaluation and linear algebra across threads in a thread block. This work is already underway in the XNet module used in

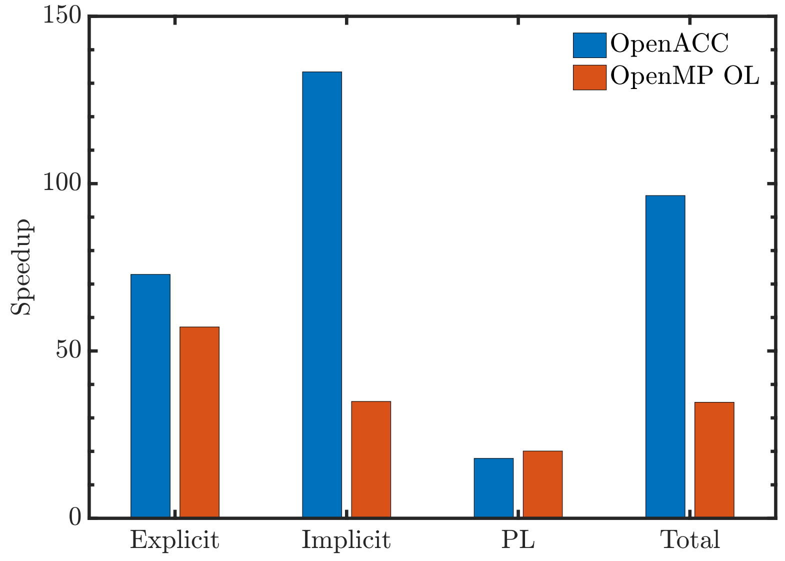


Figure 29: Performance of distinct Thornado components are measured for different programming models of OpenACC and OpenMP offload for a problem of a radiating sphere that mimics core-collapse supernova conditions. Speedup is that of one Volta GPU relative to a single Power9 CPU core.

FLASH. An earlier version of the XNet module has shown good speedups via GPU acceleration for large (i.e. 160 species) networks. Shown in Fig. 31 is the time to solution for a nuclear burning simulation with FLASH while employing an increasing number of CPU threads. In the GPU case, each CPU OpenMP thread is launching kernels on the GPU to perform the network solves.

Next Steps

Our GPU strategy moving forward is based on a three-step process: (1) finalize the GPU implementations in the individual physics modules, (2) interface these modules with FLASH/Castro, and (3) optimize and tune the individual GPU kernels in the context of executing the challenge problem. Thornado and WeakLib have both been fully interfaced with Castro; the FLASH interface is under active development. Both Thornado and WeakLib have already been extensively optimized for GPUs, but incorporation of asynchronous kernel launches may provide further benefit. The GPU implementation of the Spark module for magneto-hydrodynamics in FLASH is also under active development using a similar approach (i.e. via directives) as demonstrated with Thornado. A fully GPU-ported version of the XNet code for the reaction network is nearly complete and will replace the existing implementation in FLASH. This version will be a substantial improvement over the previous iteration which relied on GPUs only for the dense linear solve describing the coupled system of ODEs for the reaction network. Interfacing this new reaction network solver with FLASH will be straight-forward given the existing interface to the previous version of XNet.

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 as well as a number of breakthrough discoveries. Three key ingredients—dark energy, dark matter, and inflation—are signposts to further breakthroughs, as 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 as well as provide new classes of measurements, such as of neutrino masses. They may 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. They not only shed light on some of the deepest puzzles in all of physical science but also rank among the very largest and most scientifically rich simulations run on supercomputers today. Existing machines do not

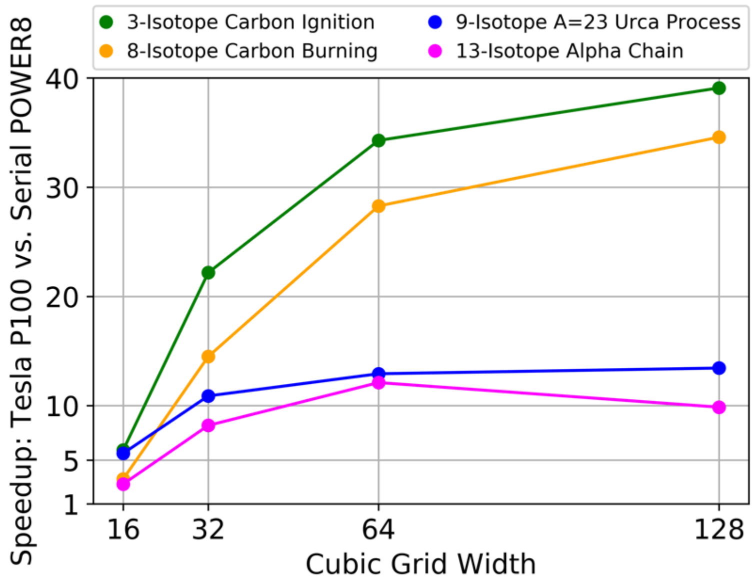


Figure 30: Castro reaction network calculation which uses a single thread on the GPU to time-integrate a single zone on the GPU.

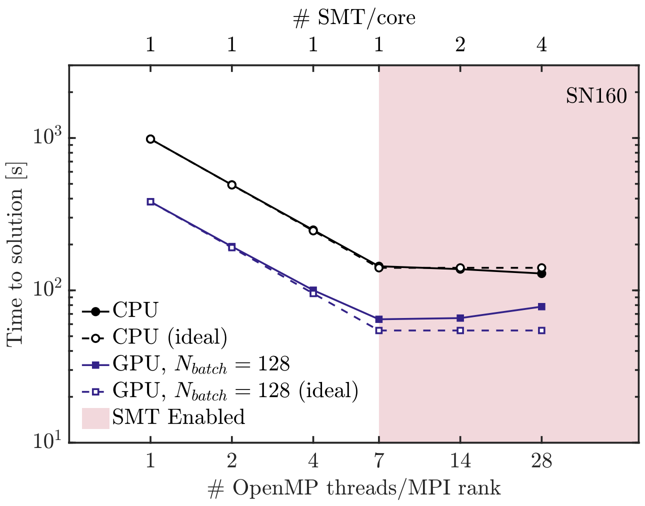


Figure 31: FLASH Burn Test on Summit. Single node using 6 MPI ranks (1 rank/GPU) with different levels of OpenMP CPU threading; 160-species network, explosive carbon burning.

have the performance and the memory needed to run the next-generation simulations that are required to meet the challenge posed by future surveys, whose timelines parallel that of the ECP. The ExaSky project extends the HACC and Nyx cosmological simulations codes so as to efficiently utilize 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 for verification of gravitational evolution, gas dynamics, and subgrid models in cosmological simulations at very high dynamic range.

In order to establish accuracy baselines, there are statistical and systematic error requirements on a large number of 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 a number of 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 a number of more significant problems such as particle shot noise and code evolution errors (including subgrid modeling biases). Strict accuracy requirements have already been set by the observational requirements for DOE-supported surveys such as the CMB-Stage 4 (CMB-S4), Dark Energy Spectroscopic Instrument (DESI) and the Large Synoptic Survey Telescope (LSST), which typically are sub-percent (statistical) over the range of well-observed scales. Systematic errors need to 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 $> 50\times$ FOM requirement. For a recent exploration of cosmological simulation errors in hydrodynamic simulations by ExaSky, see Emberson et al. [9], this paper uses the ExaSky-developed CRK-HACC code that implements the new CRK-SPH method [10] 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, 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 Gpc 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-size boxes will be run to cover the range of scales that need to be robustly predicted. The mass resolution of the simulations (in the smaller boxes) will go down to roughly a million solar masses for the baryon tracer particles and about five times this value for the dark matter particles. 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 consist of two very large cosmological simulations run with HACC that simultaneously address many science problems of interest. Setting up the science challenge problem in turn 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 astrophysics codes via a nested-box simulation approach, a medium-scale set for parameter exploration, and based on these results, implementing the final two large-scale challenge problem runs on the exascale platforms.

The challenge problem runs are of two different types. The first is a large-volume, high mass and force resolution gravity-only simulation (Table 36), and the second is a corresponding hydrodynamic simulation that includes detailed subgrid modeling (Table 37). 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 (internal to this set as well as 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 being *enforced*, as is mostly the case currently. The aim with this more physics-based approach is to achieve a more consistent approach in which

Table 36: ExaSky gravity-only challenge problem details.

Simulation target	Specifications
Initial Conditions	Multi-particle Gaussian random field initial conditions using a specified linear power spectrum at the initial redshift (based on ExaSky's 3-D SWFFT with up to 30k cube FFTs)
Boundary Conditions	Periodic (box size of 3 Gpc/h
Resolution	Force resolution, ∼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/heating, star/black hole/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, tSZ and kSZ statistics; light-cone outputs, galaxy summary statistics, strong/weak lensing for galaxy clusters, sky maps for optical and CMB observables; multi-probe cross-correlations

Table 37: ExaSky subgrid challenge problem details.

Simulation target	Specifications
Physical phenomena and associated models	Multi-scale cosmic structure formation—gravitational evolution, gas dynamics and subgrid models for astrophysical processes, including a number of 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, etc.	Multi-trillion multi-species particle simulation with fully representative subgrid modeling for the challenge problem; results must be available at low redshift (close to the current epoch)
Demonstration calculation requirements	1) Limited number of timesteps 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 hours of run time

the final results become independent of code parameters and are more easily interpreted. The new subgrid models not only need to be optimized for the exascale platforms, but they also have an effect on the temporal resolution of the simulations; once they are 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: Figure of Merit

In principle, the ExaSky FOM requires a discussion of several factors in order to arrive at a well-defined single number that can simultaneously capture a multi-dimensional 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) Addition of new science capabilities (physics complexity, which in the case of ExaSky, is a statement about subgrid models). Provided weak scaling targets are met, increases in problem size should provide a linear increase in the FOM; likewise, provided the strong scaling targets are met, problems at fixed size should run proportionally faster. Though ExaSky consists of two simulation codes, HACC and Nyx, since 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 since 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 either not available or it is not worth porting 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 timestep (higher time resolution) once the subgrid models are incorporated. The latter effect can be quite significant, leading to increases in the overall number of timesteps by one or two orders of magnitude, whereas the actual increase in time due to the additional work per timestep is only of order 10–20%. For this reason, the FOM remains controlled by the time spent in the main solvers, i.e., gravity + hydro, and therefore, the performance of these can be used to determine this value without having to consider how the new subgrid models would perform on the baseline platforms.

The current baselines for the FOM have been 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 have been carried out 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 wall-clock numbers from one set of simulation runs to another, provided 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).

The HACC FOM calculation is therefore based on the following steps (ADSE01-29):

1. Establish scaling of the code 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) and compute the per time step throughput (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 PFLOP (peak).
4. This methodology can be separately applied to the gravity-only part of the code, as well as to 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 using Summit as the base system, with appropriate scaling based on the known FOMs for Summit referred to the previous generation machines.

Example calculations of the FOM are given in ADSE01-29. Note that the methodology presented there presents an FOM that is based strictly on strong scaling: how much faster does an equivalent problem run on the system under test compared to the baseline? Such a measure does not account for the fact that the system under test may have significantly more RAM and therefore can attack larger problems than the baseline system. In the case of Titan and Summit for example, the RAM ratio is approximately (2.4PB/0.6PB) = 4, so Summit can be used to solve cosmological problems, which are typically memory bound, that would be impossible to do (in the same way) on Titan, but the FOM does not reflect this key advantage. The current FOM for HACC in gravity-only mode for Summit is 13.6 (reported in ADSE01-29). The current expectation is that for an exascale system this FOM will be greater than 60.

The initial hydro-solver FOMs have yet to be computed, since work on Summit optimization is continuing, but the baseline data is available in the ADSE01-29 report (later runs on Cori II have also yielded more information). The team expects to report the first hydro enhanced FOMs in the near future on the FOM dashboard.

5.2.3 *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 wall clock requirement for the demonstration calculation. The current estimate for the stretch goal is based on a $30{,}720^3$ particle problem (compared to 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 factor of 2 at the minimum.

In terms of additional capabilities, it is important for the ExaSky project that a number of data-intensive and Artificial Intelligence (AI)/Machine Learning (ML)-oriented analysis capabilities be available on the exascale systems. As the first table above 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 a number of capabilities of the exascale systems (file IO, AI/ML performance, etc.) aside from the purely computation-oriented requirements.

5.2.4 *ExaSky: Progress Towards Advanced Architectures*

GPU Strategy

The ExaSky challenge problem will be carried out with HACC, the Hardware/Hybrid Accelerated Cosmology Code, with gravity-only and with gravity plus hydrodynamics simulations. The challenge problem is a combination of these two types of simulations. The project's FOM is therefore associated with HACC's performance in both usage modes. The gravity-only version of HACC has been able to take advantage of GPUs ever since the initial development of the code. The first version for GPUs used OpenCL, a CUDA version was developed for Titan due to the better support available from NVIDIA. Given the diverse hardware in the exascale era, we recently revived the OpenCL version of HACC. This gives us a solid code base that should be able to run on either Aurora or Frontier. In addition, working closely with the vendors, we are exploring more vendor specific programming models, such as HIP for Frontier. Based on the OpenCL HACC version, it is straightforward to switch to a different programming model quickly if it turns out that the performance gains are considerable. The hydrodynamic version of HACC uses a new SPH method, CRKSPH, and has been run at scale on Intel KNL systems. A first GPU version has been run on Summit and shown to scale to the full machine. Currently we are optimizing this particular implementation.

HACC does not rely on any ST or external software in its critical path. However, we are planning to take advantage of several ST developments, in particular VeloC for a convenient and fast checkpoint/restart implementation that will allow us to store more frequent checkpoints, SZ for data compression in order to enable the storage of more detailed outputs from the simulations, and Cinema, for in-situ visualization.

Progress to Date

We have carried out three major science runs on Summit at the end of 2018. Each of the simulations was carried out on 4,096 nodes of Summit and used all available GPUs. The simulations evolved close to 2 trillion particles each, resulting in some of the largest cosmology simulations at the achieved resolution (see Figure 32). These simulations were carried out with the HACC gravity-only version. We used CosmoTools, HACC's analysis library, to carry out the first-level analysis completely on the fly. We used these simulations to identify possible improvements to enable even better performance in the future. We found a handful of bottlenecks in the in-situ analysis codes and were able to improve those considerably. The challenge problem will evolve $\sim 6.5\times$ more particles in the same simulation volume as these Summit runs and therefore will increase the resolution by this factor. The results from Summit are very promising for future prospects on exascale systems. We show the output from one rank from one of the Summit runs in Figure 33.

In addition to these three science runs, we have also carried out scaling runs and FOM runs with both the gravity-only and the hydro versions of HACC. Scaling results are shown in Fig. 33. We showed that both versions of the code scale up to the full machine. Our FOM includes time to solution as well as the particle loading, as both metrics are crucial for cosmological simulations. For the FOM we used baseline test problems set up on Theta (Intel KNL system at ALCF). The current projected FOM increase is measured to be 30.20 and the extrapolated FOM increase is 23.76. The FOM runs on Summit were carried out using 4,096 nodes and 6 GPUs per node, the Theta runs used 3,072 nodes of Theta. We also compared results from exactly the same problem size on Summit and Theta, using 128 nodes with 6 GPUs per node on Summit and 3,072 nodes on Theta. While the time to solution on Summit and Theta for the hydro version of HACC were similar, the gravity-only version which has been already maximally optimized for GPUs, was more than four times faster. We are currently working on further improving the performance of the hydro version of HACC on GPUs.

Next Steps

The highest priority is continued work to improve the performance of the HACC hydro kernels on Summit. The gravity-only version of the code is already very close to maximally performant. We have started to closely interact with both vendor teams, Intel and AMD, including dedicated hackathons to develop the optimal deployment strategy for HACC on both machines. We carefully follow the hardware and software developments for both systems and evaluate the ramifications of these developments on the projected performance of HACC periodically. We continuously compile and run on the latest software stack drops and measure and optimize performance on provided test hardware (which include using hardware simulators). A combination of these activities allows us to continuously project and refine the performance we obtain with our current software, and we plan/implement new required algorithms to meet any limitations we see (written in any programming language required). We continue to test different programming models to identify the best solution for HACC.

5.3 EQSIM

Large earthquakes present a significant risk around the world and are a major 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 (electric/gas distribution systems, renewable energy production facilities, nuclear power plants etc.). Beyond the DOE enterprise, addressing earthquake risk, both from the standpoint of life safety and damage/economic impact, is a major societal challenge for virtually every element of the built environment including transportation, health, data/commerce and all urban infrastructure. The tremendous developments occurring in high performance computing, 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 within grasp.

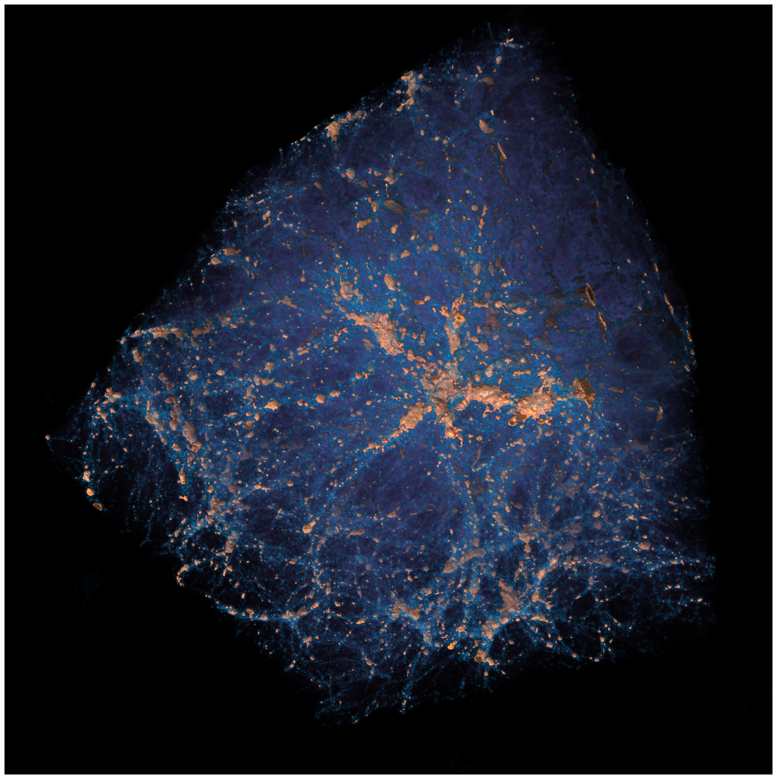


Figure 32: Visualization of the mass distribution from one rank for a large cosmology simulation carried out with HACC on 4,096 nodes of Summit. The whole simulated volume is 24,576 times as big. Light colors indicate high density regions.

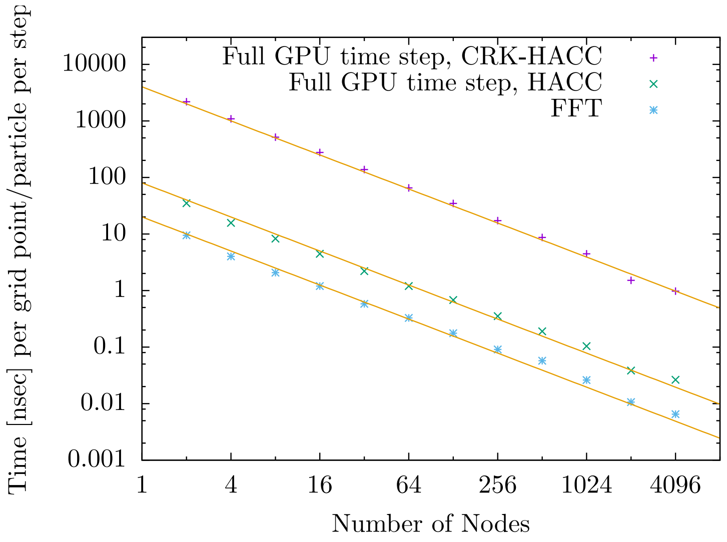


Figure 33: HACC weak scaling on Summit.

The EQSIM application development project is focused on creating an unprecedented computational toolset and workflow for earthquake hazard and risk assessment. Starting with a set of the existing codes, SW4 (a fourth order, 3-D seismic wave propagation model), NEVADA (a nonlinear, finite displacement program for building earthquake response), and ESSI (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 to infrastructure response (earthquake risk). The ultimate goal of the 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 utilize historical earthquake ground motions from many different locations to estimate future earthquake ground motions at a specific site of interest. Given the fact that 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 both 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 past decade, interest in utilizing 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. A fundamental objective of the EQSIM application development 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. A second 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, there are two fundamental challenges that must be addressed. First, 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 to allowing the necessary parametric variations necessary 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 structure at finer and finer scales, thus a companion schema for representing fine-scale geologic heterogeneities in massive computational models must be developed. For the purpose of evaluating regional-scale simulations and assessing progress on the application developments of this project, a representative large regional-scale model of the San Francisco Bay Area (SFBA) has been created. This model includes all necessary geophysics modeling features (3D geology, earth surface topography, material attenuation, nonreflecting boundaries, fault rupture models). For a 10 Hz simulation, the computational domain includes approximately two hundred and three billion grid points in the finite difference domain. The SFBA model provides the basis for testing and evaluating both advanced physics algorithms and computational implementations. Challenge problem details are given in Table 38.

5.3.2 EQSIM: Figure of Merit

The EQSIM project has established an application FOM that simply and clearly expresses both 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 wall clock 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 of the frequency resolved in the ground motion simulations requires essentially sixteen times the computational effort[2] and thus computational effort varies as frequency resolved to the fourth power. The

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

Table 38: 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 in order 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, etc.	Regional-scale simulations will typically encompass a large urban region surrounding the urban environments of interest and the regional earthquake faults (sources) of interest. For the EQSIM project a representative model for the San Francisco Bay Area (SFBA) has been 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 (performed annually in the project milestone plan to establish current application FOM) will revolve around the SFBA model with simulation of a representative $M = 7$ earthquake on the Hayward fault and simulation of a corresponding 90–120 seconds of earthquake motions. These runs measure the project's annual progress towards the exascale challenge problem.
Resource requirements to run demonstration calculation	Based on results to-date and the objectives outlined above, it is estimated that EQSIM would require ∼80–90 % with total integrated wall clock machine time usage on the order of 90–150 hours for a single earthquake scenario simulation. A single earthquake realization will need 3–5 hours.

initial application FOM was originally computed as

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

where $f_{\max}$ is the highest frequency (Hz) resolve in the regional ground motion simulation, the wall clock time (hours) is for one full rupture scenario simulation for a large earthquake (typically simulating on the order of 90 seconds 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 $V_{s\,\min} = 500\,\text{m/s}$ in the regional model.

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 as the model grid discretization is dependent on the minimum shear wave velocity in the model. In addition, regional simulations illustrate that in order to achieve realistic simulations of risk it would be necessary to reduce Vsmin 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

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

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 FOM has been tracked and updated since the start of the project, all values previously computed are consistent with the final FOM since they were computed using a model with $V_{s\,\min}$ of 500 m/s and thus $500/V_{s\,\min}$ was unity. More recently the team is moving towards lower $V_{s\,\min}$ values (e.g., $V_{s\,\min} = 250\,\text{m/s}$). As described below, a number of advancements have been achieved and the application FOM has progressed from 1.0 at the start of the development project to the recent highest achieved value of 66 on the Summit platform at OLCF.

5.3.3 *EQSIM: KPP Stretch Goal*

Ultimately, soft near-surface sediments can exhibit nonlinear softening behavior under strong earthquake ground motions. Representation of soil nonlinearity can be approximated in a number of 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 computationally challenging problem and would be a stretch goal for EQSIM depending on overall progress and performance on Exascale platforms.

5.3.4 *EQSIM: Progress Towards Advanced Architectures*

GPU Strategy

In FY19 the EQSIM project had the need and resources, as it transitioned from a seed to full-scale project, to take broader advantage of ECP ST developments. The engagement of the HDF5 team and the utilization of HDF5 to enhance parallel I/O was an important step towards preparation of the EQSIM workflow for execution on advanced platforms.

HDF5 has been used to implement a new geologic format, termed an *Sfile*, to store the material properties and topography/bathymetry of a region. The Sfile implementation offers significantly enhanced portability, improved parallel I/O performance, and reduced file size when compared to the previous raster file *Rfile* (a binary block structured format) format developed for SW4. The main disadvantage of the Rfile block storage paradigm is that material properties must be stored at all grid points below the highest topography in the model. This means that air properties must be stored for all points that are above the topography. The new Sfile format defines a number of curvilinear grids based on the topography, where grids near the surface of the earth have finer spatial resolution, eliminating the need to store grid values above the topography. As a result, the SFBA region Sfile is less than half the size of the corresponding Rfile (3.3 GB versus 7.1 GB),

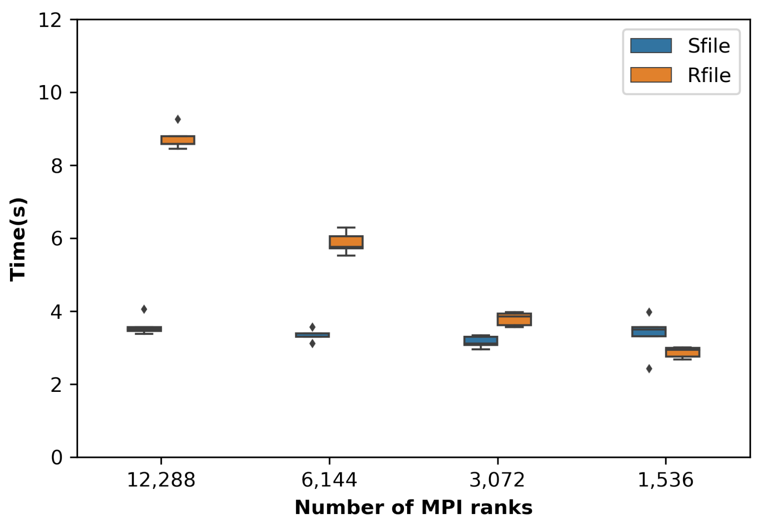


Figure 34: HDF5 enabled parallel I/O improvements for Hayward fault geologic structure read times (new geologic Sfile versus previous Rfile formats).

and can be accessed more quickly (a 2.5× read speedup was realized with SW4 using 12,288 MPI processes) when SW4 is running at large scale due to the efficiency of parallel HDF5 (Fig. 34).

The latest version of the EQSIM workflow also utilizes HDF5 to better manage fault-to-structure simulation results. SW4 ground motions can be saved in user-specified selected volumes using a standardized HDF5 container. This functionality is a valuable tool for coupling SW4 incident seismic waves to soil structure interaction (SSI) simulations through the Domain Reduction Method (DMR) using engineering finite element programs such as ESSI. Comparisons between motions computed with SW4 at the surface and from equivalent ESSI soil models (with SW4 motions injected at the boundaries of the finite element models) have been made and show exceptionally good agreement.

The ability to efficiently and appropriately integrate SW4 ground motions into engineering finite element models provides a unique ability to explore the interaction between complex incident seismic waveforms and soil-structure systems.

Also in FY19, the SW4 code was partially ported to the Sierra and Summit platforms through the use of RAJA C++ libraries. RAJA's C++ loop abstractions are designed for writing portable and performant grid-based simulation codes on high performance architectures. RAJA accomplishes this by separating the loop bodies from the actual methods used to execute the loop through the use of C++ lambdas. The loop body captured using one or more lambdas can be dispatched to run using a programming model and/or on a selected execution space (e.g CPU or GPU), through the use of execution policies that are selected at compile time. RAJA supports nested sequential and parallel loops as well as parallel loops with embedded reduction operations.

For GPU based platforms, SW4 uses Umpire managed memory to perform setup on a CPU and to automatically move the data to the GPU where it resides for the rest of the simulation. Some data is copied to the host for message passing and I/O. Loops that contain reduction operations must use special RAJA reduction variables and appropriate reduction policies must be used for correctness. The key computation loops in SW4 are perfectly nested triple and quadruple loops that require more complex execution policies for obtaining good performance. Some of the more complex loops had to be split along the coordinate directions to reduce register pressure and these kernels execute at up to 36 % of the peak floating-point performance of the GPU and up to 50 % of the maximum memory bandwidth. For nested loops, RAJA allows the nesting order to be changed using the execution policy and this is often required for getting performant code when switching architectures. RAJA also includes constructs for accessing shared memory on GPUs, coalescing loops. The RAJA port of SW4 has been executing on machines with Nvidia GPUs and Intel CPUs (Xeon and KNL). On future platforms, SW4 will require the new architecture to have a port of RAJA and Umpire and to support a unified address space. If these requirements are met, porting would simply involve the

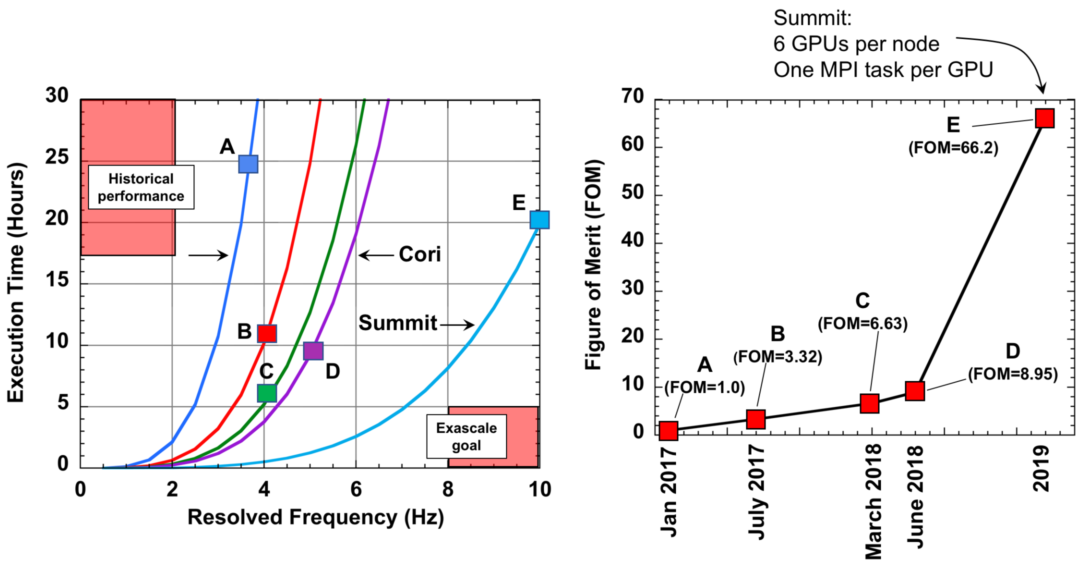


Figure 35: Performance metrics for a $M = 7$ Hayward fault earthquake simulation on both Cori and Summit (Note, these curves are for a minimum shear wave velocity of 500 m/s in the SFBA model).

generation and tuning of a new set of execution policies.

Progress to Date

The EQSIM project formally plans a regional-scale science/performance demonstration calculation in the 4th quarter of each fiscal year to evaluate the progress towards performance goals, and to demonstrate scientific and engineering insight into what are earthquake simulations of unprecedented resolution. As described below, the analysis of full, large-scale simulations of earthquake scenarios is also critical to informing the computational requirements for future realistic simulations.

In FY19 ground motion simulations with SW4 were executed for the first time on Summit. Relative to previous simulations using nearly all of Cori, a significant performance boost was realized as indicated in Fig. 35. For a SFBA model, a 10 Hz frequency resolution was successfully achieved on Summit, however this model only included near-surface soils with shear wave velocities down to 500 m/s. The evaluation of sensitivity studies with the large-scale simulation results has indicated that the regional model must capture near-surface soils, with shear wave velocities down to at least 250 m/s to fully represent the ground motion hazard [11]. Lowering the minimum shear wave velocity will mandate larger simulations going forward. To put the Summit simulation results in perspective, the SFBA simulation performed with EQSIM are compared to all previous regional-scale simulations of the SFBA in Fig. 36. It is clear that the EQSIM results, and the Summit simulation in particular, present a dramatic increase in earthquake simulation resolution.

Next Steps

Both HDF5 and RAJA will be essential elements of the EQSIM go-forward strategy for transition to new exascale platforms. As we look forward to the emergence of future platforms (Aurora, Frontier), the EQSIM team will work closely with the RAJA and HDF5 teams to identify necessary steps for successfully transitioning from Summit.

5.4 Subsurface

An urgent challenge is to understand and predict the reservoir-scale behavior as affected by the long-term integrity of the hundreds of thousands of deep wells that penetrate the subsurface for resource utilization. The

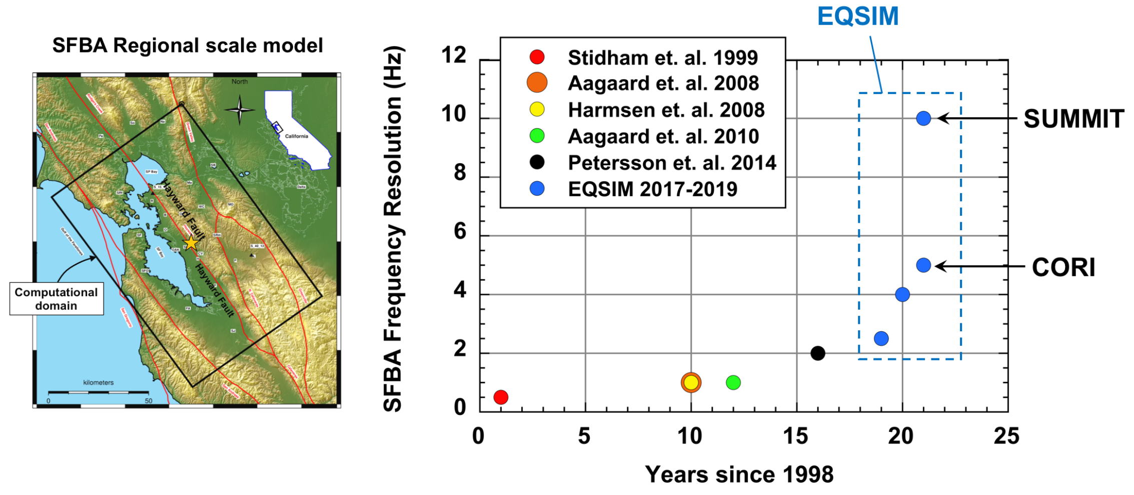


Figure 36: EQSIM SFBA simulation progress in context with all previous SFBA simulations.

performance of a wellbore hinges on the behavior of very thin interface features controlling the leakage of fluids along the well casing-cement boundary. Similarly, leakage of buoyant fluids (e.g., CO_2) through caprocks may be controlled by micron-scale asperities in fracture networks that are themselves subject to geomechanical and geochemical modification. At the reservoir or field scale ($\sim$1–10 km domain size), multiphase flow and reactions in fractured porous media are typically modeled using continuum models that make use of averaged quantities and bulk parameters that do not fully take into account 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 so as 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 multi-physics processes occurring 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 has been 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 degrees of freedom once the hydraulic, mechanical, and chemical variables are included. Based on prior experiments and modeling, the Challenge Problem is estimated to extend for 10 days of simulation in order 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 take place in the subsurface that involve the evolution of fractures, including both 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 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 have the potential to connect the reservoir to groundwater resources and the atmosphere (1). 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 storage of 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 need to be considered in order 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 because of 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 mechanical compression of the asperities (collapse of fracture pillars) or 2) by 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 deposition of minerals in the fracture (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 described above, 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., close to the wellbore) and timesteps 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 (such as permeability and reaction rate) that do not represent the true tortuosity of flow paths or reactive surface area of the material.

In contrast to the conventional treatment of wellbore failure, accurate prediction of fracture evolution depends on microscale resolution of fracture asperities (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 sub-critical fracture growth literature [12], the overall rate of fracture growth in these zones is controlled by coupled processes occurring at the fracture tip.

Microscale resolution is also needed to accurately predict fracture permeability, since real rough fractures are typically held open by asperities (pillars) of this scale. Chemical corrosion (dissolution) or mechanical corrosion (pressure solution) of these asperities occurs at the same micron scale. Chemical dissolution may actually have two opposing effects: 1) formation of channels or "wormholes" in the fracture plan that focus flow, thus accelerating the opening of the fracture, or 2) dissolution of fracture asperities, thus allowing the fracture to close under the ambient confining stress. The overall domain size of the fracture tip, $O(\text{cm})$, 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)$\times$1 cm (along an azimuth in the cement annulus)$\times$1 mm (in the radial direction) with 1 µm resolution. This domain size is assumed to be the minimum domain needed

Table 39: Subsurface challenge problem details.

Functional requirement	Minimum criteria
Modeled physics	Coupled flow, multi-component 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 (6 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$\times$ 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 $\Rightarrow$ 8640 global timesteps $\Rightarrow$ 4 weeks of machine time

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 takes into account the length scale for an REV. Order 1 mm in the radial direction is required to capture diffusive transport over long time scales.

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 (9,000 nodes) for 24 nm resolution of a 100 µm block of shale, the challenge problem is well into the regime of exascale resources as it is 50$\times$ current capability. Resources based on Summit Volta GPUs can also be estimated. Given that Summit has same high-bandwidth memory as Cori KNL (16 GB HBM), and 6 cards per GPU node, the requirement would be 80,000 Summit nodes. The problem specifications are summarized in Table 39.

5.4.2 Subsurface: KPP Stretch Goal

The Stretch Challenge focuses on the simulation of 100 m of a single wellbore with proposed cell resolutions down to 2 mm. The specifications for this problem are listed in Table 40. The wellbore scale component of this problem is a GEOSX simulation using continuum/Darcy scale assumptions with coupled physics consisting of solid mechanics, multicomponent multiphase flow, and fracture mechanics. While the wellbore scale (GEOSX) component of the Stretch Challenge Problem will be developed independently of the Base Challenge Problem, it is intended that the capabilities from the Base Challenge Problem are to be coupled with the wellbore scale through methods developed within the project (ADSE05-21).

5.4.3 Subsurface: Progress Towards Advanced Architectures

GPU Strategy

Table 40: 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	∼800M elements
Degrees of freedom	∼10B DOF
Domain decomposition and load balancing	2 ranks per node
Resource requirement to run stretch problem (based on current architecture)	1,600 Summit nodes
Simulation time for stretch goal	6 months of simulated time. Execution time dependent on the performance of the team's linear solver strategy.

ECP Subsurface is developing a new multiphysics capability that couples flow, transport and reactions using Chombo-Crunch with mechanics using GEOSX. There are two components to performance portability for this project: one for Chombo-based codes which includes Chombo-Crunch and the coupling framework, and one for GEOSX.

Chombo-Crunch will make use of a new abstraction layer called Proto. Proto is a C++ library 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 (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 major change from the previous approach is that all of 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` instances: (i) Application of stencils (apply). In Proto, stencils are first-class objects, separately defined and archived. (ii) Pointwise applications of user-defined functions applied to the values of the `BoxData` at each point in the rectangular patch (forall). Embedded boundary (EB) calculations are used to compute solutions on complex geometries. Because the nature of these algorithms is semi-structured, EB calculations require more complex data structures. The EBProto performance portability layer 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. This data lives on the device and requires complex indexing.
- Structures are built for fast stencil evaluation and pointwise function evaluation.
- A dictionary of stencil implementations is provided that can be easily accessed by the user. Most users will never have to see the graph or moment information.

The primary ST dependencies for Chombo-based codes for the port to GPUs are PETSc/hypre for AMG solvers and HDF5 for parallel IO.

The GEOSX portability strategy (GPU strategy) is centered on the use of LLNL's RAJA and CHAI libraries. In GEOSX we use RAJA achieve platform portable kernel launching as well as using some of the portability wrappers that RAJA provides to provide portable usage of shared memory, local arrays, reductions, and thread safe atomics. GEOSX utilizes CHAI to manage the motion of data between the different memory spaces. When coupled together, RAJA kernels that capture `CHAI::ManagedArray`'s will trigger motion of data between the memory spaces, when motion is required. In cases where the data has not been "touched" outside of the launch memory space (i.e. host or device), no data motion is required.

In addition GEOSX also has a strong dependence on linear solver packages. We currently have an interchangable interface between Trilinos, HYPRE, and PETSc that will allow GEOSX to select the best option for a given problem on a given platform (i.e. CPU, NVIDIA, AMD, etc.).

Progress to Date

We developed a performant implementation on Summit GPUs of Proto, including implementation and performance assessment of benchmark kernels:

1. We developed an initial reference implementation on Summit GPUs of EBProto, including implementation and performance assessment of benchmark kernels.
    (a) EBStencil
        i. Generalization of Stencil objects for EB Operators
        ii. Combination of a regular `Proto::Stencil`, and a sparse-matrix-vector operation near irregular cells.
    (b) EBDictionary
        i. Archived EBStencil objects to minimize graph walking
        ii. Recycles EBStencils across Chombo operators
2. Status
    (a) Complete prototype exists and runs in parallel
    (b) CPU and GPU capable
    (c) MPI + CUDA capable
3. We have deployed extensions of Chombo distributed-data API for representing heterogeneous memory (Chombo4).
4. We have an initial implementation of key components of EBChombo CFD code using EBProto, including interfaces to the PETSc/Hypre linear solvers.
    (a) EBProto-PETSc interface breadboard description is done
    (b) EBProto Advection (scalar) is done
    (c) EBProto Elliptic geometric multigrid solver is done

Chombo benchmark results are given in Tables 41 and 42. As shown in Table 42, performance of the initial implementation of Proto on Summit was not that great—in the 200–300 GFLOP range—for the 27-point Laplacian kernel benchmarks. We have since improved performance of that kernel to 2.5 TFLOP as reported in our annual review presentation. Arithmetic Intensity is still an issue. Complex algorithms in Chombo-Crunch are compositions of `Proto::Stencil` and `Proto::forall`. Many forall operations have low AI. Parallelism overheads are an issue for small patches kernel launch, or OpenMP fork-join. Realistic mixtures of Stencil and forall execute at $O(100)$ GFLOP. The plan for compositions of operators to get into TFLOP regime is to automate fusion. Performance is well enough in hand for Proto so that EBProto has been designed.

Complete design specification documents (math description, matching class design, doxygen reference manual) for Proto and EBProto and Chombo4, including specification and performance models for benchmark kernels have been written.

GEOSX currently runs low order finite element solid mechanics kernels on a single V100. We are currently porting our packing/unpacking functionality to execute on device using host pinned memory for running MPI

Table 41: Initial implementation on Summit of forall kernel benchmark, Riemann problem; measurements are in GFLOP.

nx	1 stream	2 streams	4 streams	8 streams
64	220	208	222	220
128	236	231	233	234
256	65	72	233	234

Table 42: Initial implementation on Summit of apply kernel benchmark (27 pt Laplacian); measurements are in GFLOP.

nx	1 stream	2 streams	4 streams	8 streams
64	232	231	231	231
128	243	242	243	243
256	268	276	271	265

across multiple GPU's. When comparing a XEON node with perfect thread scaling against an LLNL/Lassen node (4 × V100) the solid mechanics kernel shows a flat performance gain 80× for problems sizes from 1 million elements to 16 million elements. Given the known OpenMP thread scaling on the XEON node (∼60 %), and an estimated MPI scaling efficiency of ∼50 % on LLNL/Lassen, we are expecting to achieve a 70× performance gain in the node-to-node comparison for this kernel. These results are summarized in Table 43. In addition, the execution of the Small Strain Solid Mechanics element kernel achieves approximately 6 TFLOP on the single V100.

We have not yet integrated the GPU strategies for the linear solver packages into GEOSX. We plan on integration with the GPU strategies for Trilinos/PETSc/HYPRE in FY20.

Next Steps

The ECP Subsurface team has participated in the application readiness workshops and meetings for both Aurora and Frontier. The first was an Aurora programming workshop in September then an Intel-organized on-line workshop about the Aurora Exascale system in October. We also attended the Frontier Application Readiness Kick-Off Workshop in October. We have also been active in the Intel + LLNS: PathForward Monthly Tech Deep Dive since its inception.

In Chombo for any new architecture, the C++ abstraction layer, Proto, is designed such that the abstraction layer does most of the work to accommodate architectural changes. Implementations of Proto are customized for different platforms with no changes to applications code, leading to performance portability on current and next generation platforms. This approach reduces maintained code from O(100,000) lines of Chombo (now Chombo4, as the Proto binding) to 5,000 lines of Proto. The idea is to make use of the vendor toolchain and let the compiler do the work.

Based on the application readiness workshops programming model issues on Aurora are a concern. One thing we are trying to do is to get NERSC to use new Intel programming model for NERSC 9 so that we can

Table 43: Runtimes for Solid Mechanics Kernel with estimated node-to-node comparison for XEON vs V100.

Number of Elements	1M	2M	4M	8M	16M
XEON—single thread	838	1,676	3,352	6,705	13,410
XEON—perfect scaling	23.3	46.6	93.1	186.2	372.5
XEON—36 threads	31.4	63.2	130.3	274.0	609.3
Single Volta V100	1.18	2.28	4.46	8.87	17.59
4 × Volta—perfect scaling	0.29	0.57	1.12	2.22	4.40
Quartz Node/Lassen Node	79.1	81.8	83.5	84.0	84.7

have more time to develop and test. Concern is that Aurora relies on new compiler technology and Intel is the main provider on DPC++ trajectory. We plan to be application ready for any developmental version of Aurora when it becomes available. As for Frontier it is a Cray-run machine with which we have a good track record for performance portability and where sub-vendors deliver diagnostic tools. The AMD-GPU is an architecture that we are not familiar with. However, we are in a good position with the Proto abstraction layer to deal with the differences from VT100, for example. We plan to be application ready for any developmental version of Frontier when it becomes available.

The GEOSX plan for Aurora and Frontier has a hard dependency on RAJA and CHAI support for those systems. Currently RAJA is developing a backend for HIP support, which will be ready for Frontier. CHAI relies on allocators provided by UMPIRE, which currently has HIP malloc and pinned memory allocators implemented. We expect that the port to each new machine will involve the addition of some compile time tuning of the kernel launch parameters and register usage. The RAJA/CHAI/UMPIRE support for SYCL is still in the initial stages, but we expect that as details about Aurora (and the compiler support) become available we will have a clearer idea if any additional developments are required outside of RAJA/CHAI/UMPIRE.

5.5 E3SM-MMF

The goal of the E3SM-MMF project is to develop a cloud-resolving earth system model with throughput necessary for multi-decade, coupled high-resolution climate simulations. This next-generation model has the potential to substantially reduce major systematic errors in precipitation found in current models because of 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 the US and global economies, especially agriculture and energy production. Current earth system models possess limited ability to model the complex interactions between the large scale, mostly two-dimensional baroclinic atmospheric motions and the smaller scale three-dimensional convective motions found in clouds and individual storms. These motions and their interactions, to 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 (several simulated days). Running conventional climate models at this resolution, for 100-year simulations, requires a 5,000$\times$ 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 yet to be fully explored due to limited computing resources. This project will integrate a cloud-resolving convective parameterization (superparameterization) into the DOE E3SM Earth System model using the Multiscale Modeling Framework (MMF) and explore its full potential to scientifically and computationally advance climate simulation and prediction. The superparameterization will be designed to make full use of 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 model being developed 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 super-parameterization, 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 a calendar year.

Size requirements: The challenge problem size has several aspects: (1) Achieving cloud resolving resolution in the atmosphere super-parameterization, which is defined as at least 1km grid spacing in both horizontal and vertical directions. (2) Achieving 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). (3) Achieving an eddy resolving ocean/ice model, which is defined as a minimum 18 km resolution in equatorial regions, decreasing to 6 km

Table 44: 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, 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, etc.	Weather resolving atmosphere with a cloud-resolving super-parameterization 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, $\sim O(5{,}000)$ timesteps)
Resource requirements to run demonstration calculation	The full exascale machine for 12 hours and 20% of the machine for 20 days.

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 (4) model throughput necessary to perform the simulation campaign of the challenge problem in the course of one calendar year on the exascale Frontier system. The team's minimum requirement (10 member ensemble of 100 year simulations) requires the ability to run 1,000 simulated years in a calendar year, which can be achieved with a throughput rate of 5 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 with X equal to the SYPD performance of each ensemble member.

The E3SM-MMF will be evaluated using the E3SM water cycle metrics package (under development by the E3SM project) which will measure the ability of the model to simulate extreme storms and coastal inundation. Accuracy requirement will be to obtain accuracy similar or better than the high-resolution E3SM model while running 50× faster as measured by the FOM. Due to the large natural variability and chaotic nature of atmospheric and ocean dynamics, a rigorous assessment of these metrics requires the large ensembles of century length runs identified in the challenge problem, requiring 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 (described in Table 44) will use much fewer resources and is designed to show that the E3SM-MMF model can achieve the computational performance and stability necessary to complete the challenge problem. The performance will be established with a suite of short (5 day) strong scaling benchmark calculations and the stability of the model will be stablished with a single multi-year simulation.

5.5.2 *E3SM-MMF: Figure of Merit*

The E3SM-MMF's project FOM is the throughput of a cloud resolving Earth system model, measured in simulated-years-per-day (SYPD).

For the baseline, the team compares against the traditional E3SM model running at a global 3 km resolution. As it is currently not possible to run this resolution on Titan, benchmarks of the E3SM high resolution configuration (28 km) running on 20% of Titan are used, and then these results are scaled to the 100% of Titan and to 3km resolution. For the FOM speedup, the team will compare the baseline FOM to the performance of the E3SM-MMF model running with a cloud resolving convective parameterization on Summit, with GPU acceleration. This FOM speedup combines algorithmic speedup from the MMF approach, and GPU acceleration. The cloud resolving convective parameterization will be run at a resolution of at least

3km and potentially as fine as 1km.

Throughput was measured in simulated-years-per-day (SYPD) without I/O. I/O was excluded in order 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, and 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 throughput of the full Earth system model, 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, while the "G compset" isolates the performance of the ocean and ice 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 need to be constructed. This is not difficult, but it is non-trivial and time consuming. Based on the performance data collected in F and G compsets, the team can get a very good estimate of the performance of the coupled system via

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

This formula is based on current data which shows that coupling between components adds 20% to the overall cost, and 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).

Current progress towards the FOM is as follows:

- Baseline 2018/3: FOM = 0.005 (Titan with 17,576 nodes)
- Baseline revised, 2018/12: FOM = 0.011 (Titan, 18,700 nodes)
- E3SM-MMF, 2019/2: FOM = 0.095 (Titan, 2,700 nodes)
- E3SM-MMF, 2019/4: FOM = 0.34 (Summit, 1,024 nodes)

5.5.3 E3SM-MMF: KPP Stretch Goal

Develop an Earth system model with a fully cloud resolving 3 km atmosphere component and an eddy-resolving ocean and ice components, all while obtaining 1.0 simulated-years-per-day for a single ensemble member on Frontier or Aurora. For the stretch goal, they will use a E3SM configuration where the full atmosphere (not just the super-parameterization) is run at global 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 super-parameterization approach in the E3SM-MMF configuration.

5.5.4 E3SM-MMF: Progress Towards Advanced Architectures

GPU Strategy

The goal of the E3SM-MMF project is to develop a cloud-resolving earth system model with throughput necessary for multi-decade, coupled high-resolution climate simulations. We are using a MMF approach to cloud resolving modeling often referred to as super-parameterization. Our exascale challenge problem requires us to run a fully weather resolving atmosphere and cloud-resolving super-parameterization, an eddy resolving ocean and ice components, all while obtaining the necessary throughput to run 10–100 member ensembles of 100-year simulations. In this configuration of the model, the atmosphere component is by far the most computationally expensive, with most of the run time spent in the atmosphere's Cloud Resolving Model (CRM) code. The ocean component is the second most expensive. The remaining components are either relatively inexpensive, or dominated by communication/networking that would not benefit from GPU acceleration.

Our GPU strategy is thus to port the CRM and ocean components to the GPU. Our current approach is based completely on Fortran, MPI and OpenACC. It has consisted of extensive code refactoring to reduce GPU/CPU communication and to ensure that loops can be efficiently parallelized on the GPU with OpenACC directives. Our main focus is on on-node GPU performance. We continue to rely critically on MPI, but anticipate most performance gains on exascale systems will be through better node utilization. Longer term, for our Fortran components we plan on transitioning from OpenACC to OpenMP. We also need to update the microphysics in the CRM from single moment to two moment. For this work, we plan on adopting microphysics packages from the E3SM project that our being written in C++/Kokkos.

For the computational performance of the model, we thus rely heavily on Fortran, MPI and OpenACC, and on exascale systems expect to be also using OpenMP, C++ and Kokkos. In addition improved computation, we will also need significant upgrades to our I/O infrastructure in order to complete or exascale challenge problem. For this work, we rely on the PIO library, ADIOS and PNETCDF.

Progress to Date

On Summit we are currently running the model in time-slice configuration, which uses active atmosphere, land and sea ice components with prescribed sea surface temperatures and sea ice extent. We run the CRM on the GPU. In this configuration, the CRM represents 98 % of the cost of the atmosphere component, which in turn is close to 90 % of the cost of the full model.

The CRM code was ported to the GPU using OpenACC directives and CUDA Managed Memory with the PGI compiler. We undertook several major refactoring efforts to prepare this code for effective utilization of GPUs. We pushed the loop across different CRM instances down the callstack and into the CRM code as the fastest varying dimension for more exposed threading. We handle data explicitly, as "managed" data overheads were too large, and the PGI pool allocator performed poorly. All routines inside the main time stepping loop execute on the GPU without interruption, and we only send data to and from the GPU outside this loop

We show the weak and strong scaling performance of our application on Summit in Fig. 37. The weak scaling problem configuration uses a consistent 169 CRMs per Summit node with CRMs containing $64 \times 64 \times 58$ cells as we scale up. The strong scaling problem configuration uses 777,602 total CRMs, each containing $16 \times 16 \times 58$ cells. We have 100 % weak scaling parallel efficiency, and the strong scaling achieves 95 % and 65 % parallel efficiency on 1,150 and 4,600 nodes, respectively, compared to a baseline of 288 nodes. All timers are end-to-end for the entire model excluding file I/O. The benchmark uses the same problem configuration we plan to use in our 2019 and 2020 simulations on Summit (namely, 1-moment microphysics). The key metric for scaling and GPU utilization efficiency is the total number of cells per node. In the largest node count for strong scaling, we have 2.5M total cells per node on 4,600 nodes. Our upcoming Summit simulations will use 2.4M total cells per node on only 1,000 Summit nodes. Note that strong scaling degradation is not due to MPI overheads but rather due to load imbalance and GPU efficiency reduction due to smaller kernel runtimes. MPI overhead remains less than 1 % in these runs.

To estimate the GPU Performance, we used `nvprof` to directly measure the on-node performance in terms of percentage of peak double precision floating point operations per second. We have a total of 151 GPU kernels that cumulatively clocked 2.5 % peak flops on Summit for the OpenACC code, which is quite decent for a fluids application. This compares favorably to the 1.6 % peak result on Summit for HPCG, a benchmark that is a better representative of memory-bound application performance than Linpack.

Next Steps

To prepare our code to run on Aurora and Frontier, we have several types of work. For the Fortran components which are running well on Summit, we need to transition from OpenACC to OpenMP. We also have additional Fortran components for which our OpenACC GPU port is still in progress—in particular the ocean component and the RRTMGP radiation subcomponent in the atmosphere. This porting work needs to be finished and then also transitioned to OpenMP. In addition, we have algorithmic improvements needed in the MMF and the CRM, as well as improvements to the CRM to upgrade to two moment microphysics. For our algorithmic work, we are improving the atmospheric physics/dynamics coupling by moving the physics onto a lower resolution finite volume grid. This reduces some computational noise in the model and also improves the efficiency, reducing the cost of the model on both CPU and GPU systems. To upgrade the CRM to support two moment microphysics, we will be incorporating new code developed by the E3SM project. This code is written in C++ and supports GPUs via the Kokkos library. We also plan to further improve the GPU efficiency of the CRM's dynamical core through better timestepping algorithms and a WENO-type approach for dissipation. This subcomponent will be rewritten from scratch for optimal GPU performance.

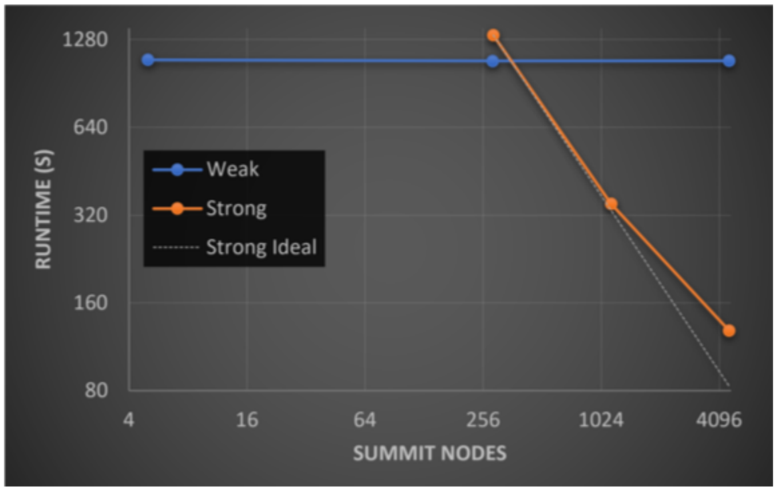


Figure 37: Strong and weak scaling of the E3SM-MMF on Summit for one model day. The weak scaling problem uses a consistent 169 CRMs per Summit node with CRMs containing $64 \times 64 \times 58$ cells. The strong scaling problem uses 777,602 total CRMs, each containing $16 \times 16 \times 58$ cells.

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 45) includes applications whose predictive capability is in part based on modern data analysis and machine learning techniques rather than strictly on approximate solutions to equations that state fundamental physical principles or reduced semiempirical models. These applications target the mission space of DOE and other relevant federal agencies such as NIH, 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 the development of the ECP applications capable of delivering demonstrable solutions to specific DAO challenge problem simulations 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 and simulation technologies 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 national power grids operate by maintaining balance between energy supply and demand. Energy is produced at generators and via renewables and other sources and is transmitted through a bulk power system at a frequency of 60 Hz. 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, both of which can permanently damage very large and expensive components. As a result, great care is taken to operate the grid with very high reliability within narrow operating voltage and frequency ranges.

Table 45: 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

Recovering from generation/load imbalance can be achieved by shedding load (deliberately allowing some load to go unserved creating a partial blackout) to preserve the functionality of the remainder of the power grid. But the behavior of the power grid can be influenced at many points within the system because of the increasing prevalence of cyber-enabled control and sensing, renewables (such as transient wind or solar power), plug-in storage devices (such as electric vehicles that can put power into the grid, or remove it), smart meters that can control load at a fine granularity (such as throttling home appliances or A/C 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, in which portions of the system are disconnected to remove the load they impose, which may miss more efficient strategies for dynamically achieving balance using a more complete spectrum of grid elements. A capability for discovering 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 response in a near-term timeframe (e.g., 30 minutes per NERC operating standards) to a variety of underfrequency hazard via physical and control threat scenarios using comprehensive modeling that includes generation, transmission, load, and cyber/control elements. This will make it possible to analyze a large number of sampled hazards to quickly discover strategies to reestablish balance with minimal impact to loads served. The ExaSGD team will compare the frequency recovery performance of a complex grid plus control system in the presence and absence of smart devices, renewables, and demand response technologies. This will involve at least two calculations of the distribution of severity of frequency response to grid hazards/effects relevant to the OE stakeholder community. Estimating these distributions involves the solution to a large number of optimal power flow calculations that consider different underfrequency scenarios. Each optimal power flow calculation requires the solution to a large-scale nonlinear optimization problem. Additionally, this challenge problem will consider the integrated execution of these optimization problems to warm-start subsequent power flow calculations across scenarios. Challenge problem details are shown in Table 46.

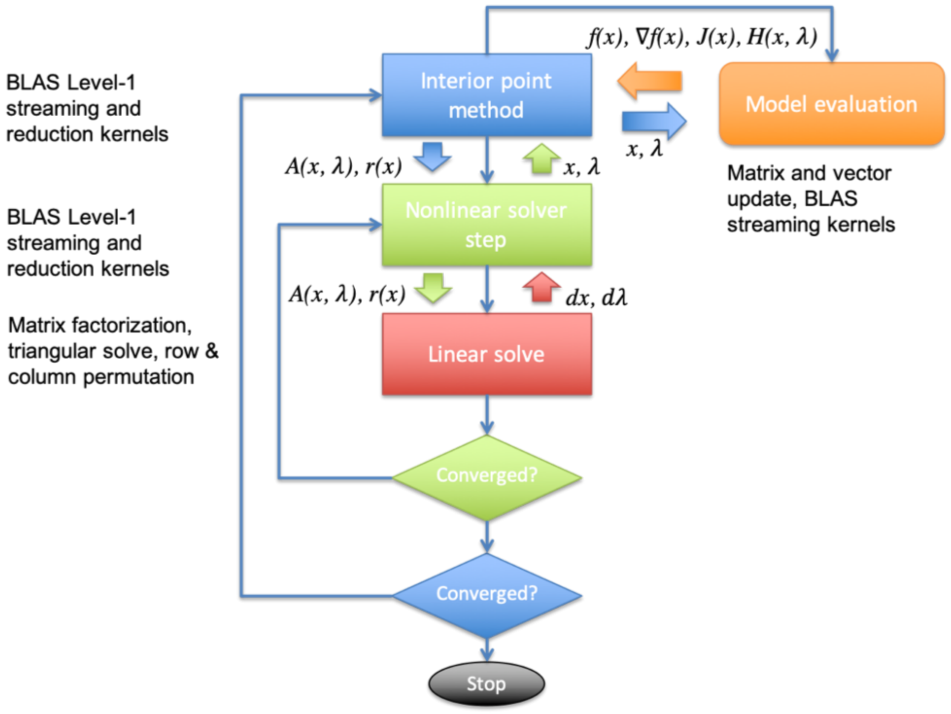
6.1.2 ExaSGD: KPP Stretch Goal

The KPP Stretch Goal is a transient security constrained AC optimal power flow problem with frequency recovery and restoration (TSCOPF-F). The objective is to determine control settings on generators, loads, and renewable energy resources, such that the system frequency is restored after the occurrence of the contingency fast enough to avoid loss of stability and prevent possible cascading outages. The content of the KPP Stretch Goal beyond the target criteria is:

- *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 large amount of physical and virtual storage further increases the complexity of the grid, not only in terms of more optimization parameters and scenarios to consider, but also in terms of qualitative changes in power grid behavior, such as possible transmission scale backflow.

Table 46: ExaSGD challenge problem details.

Functional requirement	Minimum criteria
Models	A nonlinear optimization model describing physical and control systems for grid operation including frequency-dependent fitness
Numerical Methods	Nonlinear optimization with equality and inequality constraints
Problem size and complexity	10^3–10^5 scenarios $10^5 \times 10^5$ system size 10^6 "N − 1" contingencies + 10^6 "N − ∗" contingencies 10^3–10^5 statistical samples Scenarios can be evaluated independently, but the team will apply loosely coupled scenario acceleration using partial solution strategies across scenarios
Demonstration calculation requirements	Application team will implement the whole application stack and demonstrate on models that capture a significant fraction of a model representative of the North American power grid. Initial calculations should require 10^3–10^5 scenarios and 10^2–10^3 statistical samples on a large portion of the full grid model (at least 10^4 components). Following success at this scale, the team will increase number of scenarios, samples, system size and contingencies targeting above ranges.
Resource requirements to run demonstration calculation	Production runs for challenge problem at highest scale will be ∼1 day on full exascale systems. • Aurora: 1 day on full system • Summit: 1 day on full system



$f(x), \nabla f(x), J(x), H(x, \lambda)$
BLAS Level-1 streaming and reduction kernels
Interior point method
Model evaluation
x, λ
Matrix and vector update, BLAS streaming kernels
$A(x, \lambda), r(x)$
x, λ
BLAS Level-1 streaming and reduction kernels
Nonlinear solver step
$A(x, \lambda), r(x)$
$dx, d\lambda$
Matrix factorization, triangular solve, row & column permutation
Linear solve
Converged?
Converged?
Stop

Figure 38: Schematic description of interior point method computational workflow.

- *Using transient rather than steady state constraints.* Transient analysis shows not only that a desired solution exists, but also that the control action can bring the system to that solution. TSCOPF-F will provide critical analysis capability needed to take advantage of new grid control technologies such as grid-forming (a.k.a. smart) inverters and dynamic shunt compensators. In TSCOPF-F, the objective function is obtained as the solution of a set of discrete differential-algebraic equations (DAEs). Such defined problem is far more computationally challenging because a system of DAEs needs to be solved every time the optimization solver requests the objective function update.

The stretch goal is designed to stress test the project's algorithms and libraries using a class of problems that have not been extensively studied even on sub-exascale benchmarks. If successful, the stretch goal will demonstrate the breadth of applications where ExaSGD technology can be deployed.

6.1.3 *ExaSGD: Progress Towards Advanced Architectures*

GPU Strategy

The ExaSGD project develops large scale optimal power flow simulations for power grids with multiple sources of uncertainty and multiple contingencies. To the best of our knowledge computations like this have not been performed on GPU accelerator platforms before and this project is pioneering methods for power grid analysis on GPUs.

The optimization algorithm used is based on interior point method [13]. The algorithm can be schematically described as in Fig. 38.

The interior point method implementation executes control logic and makes calls to BLAS Level-1 functions and the Newton solver. Our plan is to execute control logic on CPU and run compute intensive BLAS kernels on GPU. Host-device communication is minimal in this setting and consists only of sending error vector norms (single floating-point numbers) computed on GPU back to the control logic run on CPU. The Newton solver is structured in a similar way. It runs adaptive line-search algorithm control logic, which in turn calls certain BLAS Level-1 kernels and the linear solver. Similarly, the control logic will be run on CPU and the

BLAS Level-1 kernels will be executed on GPU. This approach is standard for abstract nonlinear algebra computations (see e.g. Ref. [14]). Different BLAS Level-1 kernels are available from multiple ECP projects and could be leveraged here [14, 15]. Nonstandard BLAS Level-1 kernels can be implemented with little developer effort using vendor-specific environments like CUDA or hardware abstraction layers such as Kokkos or RAJA.

The main computational challenges are in the linear solver implementation. The linear solution takes up more computational cost than all the other parts of the computation together. The linear problems typically arising in optimal power flow analysis are sparse symmetric indefinite. Furthermore, these problems are often ill-conditioned, so iterative linear solvers are largely ineffective there. The linear solver also has to provide matrix inertia to the optimization algorithm. Solvers that meet those requirements are mainly CPU based, such as MUMPS, Pardiso, or MA57. We assessed that development of GPU-based solvers in this area (both within and outside ECP) will not be at the pace that meets our project requirements.

To address this risk, we reformulated the optimization algorithm [16] so that instead of a large sparse linear problem, it leads to multiple dense linear problems of manageable size. In that way, we can leverage ECP sponsored libraries such as Slate [17] and MAGMA [18] who are far more mature in terms of code stability and performance than sparse direct linear solvers on GPU.

Another computational challenge is the model evaluation, which to the best of our knowledge has not been tried on GPUs before for power grid problems (or for any other problems defined by a complex network of algebraic and/or differential equations for that matter). The reason is the heterogeneous and irregular nature of power grid models that often leads to warp divergence and makes data coalescence challenging. Because of that, most of the prior work in the domain area has used highly optimized CPU code instead.

While power grid equations are highly irregular, they are assembled from component model equations for only a few different component model types (buses, transmission lines, generators, etc.). By evaluating the same component type on all threads within a warp, the warp divergence can be avoided. By grouping components that are coupled to each other within same thread block, data coalescence can be improved. We will use this approach to set up the problem so that power grid model equations are mapped to GPU threads to minimize warp divergence and improve data coalescence. We will prototype this mapping and model performance using mini-apps written in CUDA. We chose CUDA, because it gives us most opportunity to optimize the code and utilize hardware efficiently. Once satisfactory performance is achieved, we will investigate if similar performance could be attained when using hardware abstraction layers such as Kokkos and RAJA. Initially, we plan to have the model setup on CPU and model evaluations (called by an iterative solver) on GPU. The problem setup on GPU is more challenging, but at the same time less consequential as it adds one-time cost and is likely to be amortized when running complex iterative algorithms such as those in optimal power flow. The problem setup on GPU will be addressed at a later project stage.

Progress to Date

We created a mini-app in CUDA for model evaluation testing. The mini-app is emulating model evaluation in a mock-up solver environment. The solution vector is populated by random numbers instead of an optimization solver as shown in Fig. 38. We use synthetic grid models with the same size and similar graph properties as real power grid models. We tested the mini-app on a Power9/Volta architecture on Newell mini-cluster at PNNL and Ascent cluster at ORNL (both having same node architecture as Summit).

The initial implementation of the mini-app running on a single Volta device performed well and showed $>25\times$ speedup over the CPU implementation running on 16 threads. This addressed our main concern that model evaluation could create a performance bottleneck. The optimized mini-app showed flat scaling (Figure 2), indicating that a single model evaluation for problem sizes typical for optimal power flow analysis is not utilizing the entire Volta device. This was verified by profiling the code using NVIDIA Nsight tool.

The profiling also indicated that CUDA kernel launching latencies need to be taken into account when optimizing the performance. Kernels runs take between 5–20 µs (as timed by `nvprof`), while kernel launch latency on Power9 platform is typically between 5–7 µs (per private communication with NVIDIA engineers). A recommended approach to hide these latencies is to run multiple model evaluations on separate streams on each device. This approach is consistent with our optimal power flow analysis strategy, where we partition the large-scale problem into a large number of small to medium size problems (order of 10^4–10^5 unknowns) with low communication overhead. We are currently evaluating techniques for flooding GPU device with large number of small kernels to hide kernel launch latencies, including use of CUDA Graphs.

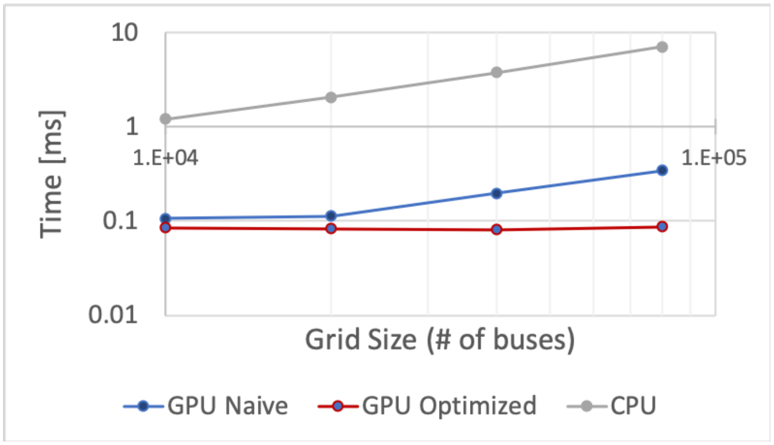


Figure 39: Preliminary performance results for model evaluation. Results are shown for CPU 16 threads, naive, and optimized single-device GPU implementation.

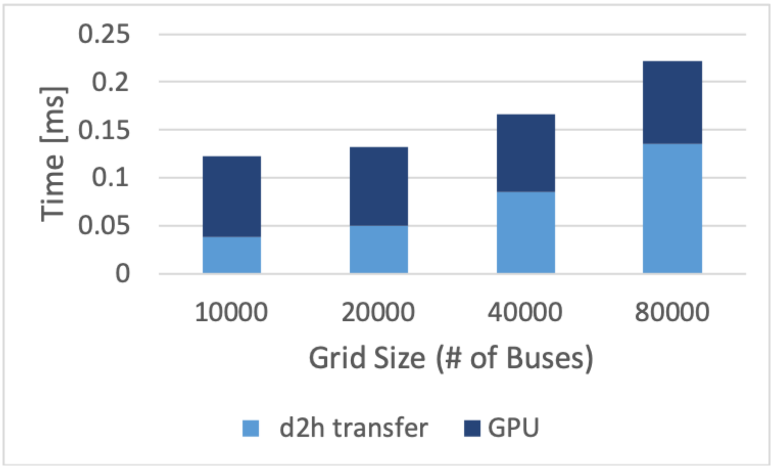


Figure 40: Host-device latencies compared to model evaluation runtimes.

We also measured host-device communication latencies relative to the model evaluation kernels runtimes on the device. For the smallest problem size we tested, moving data between the device and the host takes approximately the same amount of time as evaluating model residual function. As we go to larger problem sizes, host-device communication latencies become almost twice as long as kernel runtimes (see Fig. 40). This reinforced our initial assumption that for optimal performance, model data should remain on the device during the entire computation.

We tested performance of a sparse symmetric indefinite linear solver SSIDS from Spral library [19]. This solver capability matches requirement of our original approach. However, we found that SSIDS GPU performance is worse than its CPU performance for test cases arising from our optimal power flow analysis (Fig. 41). We performed tests for 4 different linear problems ranging in size from 13,000 to 300,000 and having number of nonzeros from 40,000 to 1,000,000. In each case CPU implementation running on 16 threads outperformed GPU implementation. Based on these testing results, in addition to literature survey, we concluded that GPU technology for sparse direct linear solvers may not be mature enough to support our project needs. That is why we decided to do algorithms-implementation co-design, where we reformulated the optimization algorithm, so that it recasts a large sparse linear problem in several manageable dense linear problems.

Next Steps

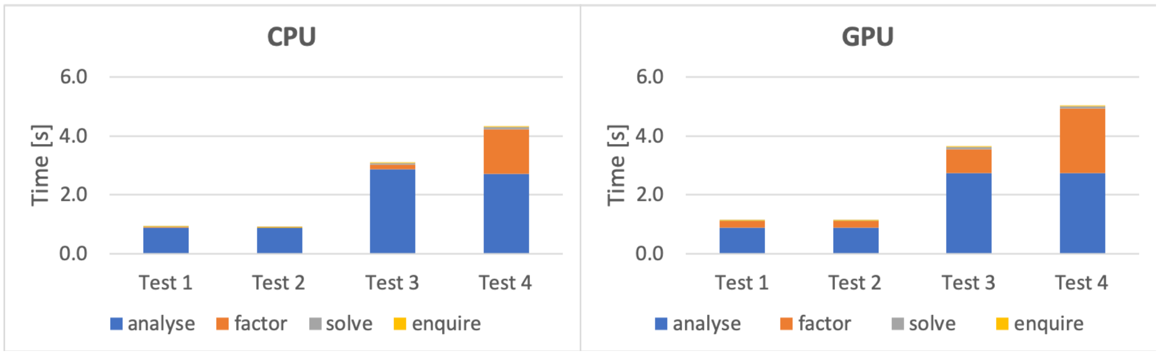


Figure 41: Sparse symmetric indefinite direct linear solver performance on CPU on 16 threads compared to performance on a single GPU device. Tests performed on Intel Xeon E5-2620 v4 and NVIDIA Tesla P100 device.

We will use CUDA mini-apps to prototype and optimize key computational kernels for our software stack on Summit platform. We chose CUDA because it is the most mature GPU technology to date and because it gives us best control over the GPU hardware. We will use hand-optimized CUDA code to set up performance targets for the portable version of our code.

We will rely on performance portability libraries such as Kokkos [20] or RAJA [21] to prepare our software stack to run on Aurora and Frontier architectures. We are still assessing which portability layer addresses our needs better. At this time, Kokkos seems to be a more mature and feature rich platform. UMPIRE/Chai/RAJA software provides a more modular environment and allows for gradual transition from vendor specific to portable code.

We will use our CUDA implementation to guide portable implementation of the software stack. We will leverage libraries developed at other ECP projects, in particular MAGMA and Slate, where applicable. In cases where portability abstraction creates a performance bottleneck, we will fall back to vendor specific solutions. Finally, we will engage Kokkos and RAJA developers when porting model evaluation kernels characteristic for power grid (and other complex network) models, which have not received sufficient attention in the past.

6.2 CANDLE

The 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 pre-clinical 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 is present in 30% of cancers. The third challenge (called the “treatment 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. While each of these three challenges are at different scales and have specific scientific teams collaborating on the data acquisition, data analysis, model formulation, and scientific runs of simulations, they also share several common threads. The ECP project *The Exascale Deep Learning and Simulation Enabled Precision Medicine for Cancer* focuses on the machine learning aspect of the three challenges and, in particular, builds on a single scalable deep neural network code called CANDLE (CANcer Distributed Learning Environment).

6.2.1 CANDLE: Science Challenge Problem Description

The CANDLE challenge problem is to solve large-scale machine learning problems for three cancer-related pilot applications: predicting drug interactions, predicting the state of molecular dynamics simulations, and predicting cancer phenotypes and treatment trajectories from patient documents. The CANDLE project

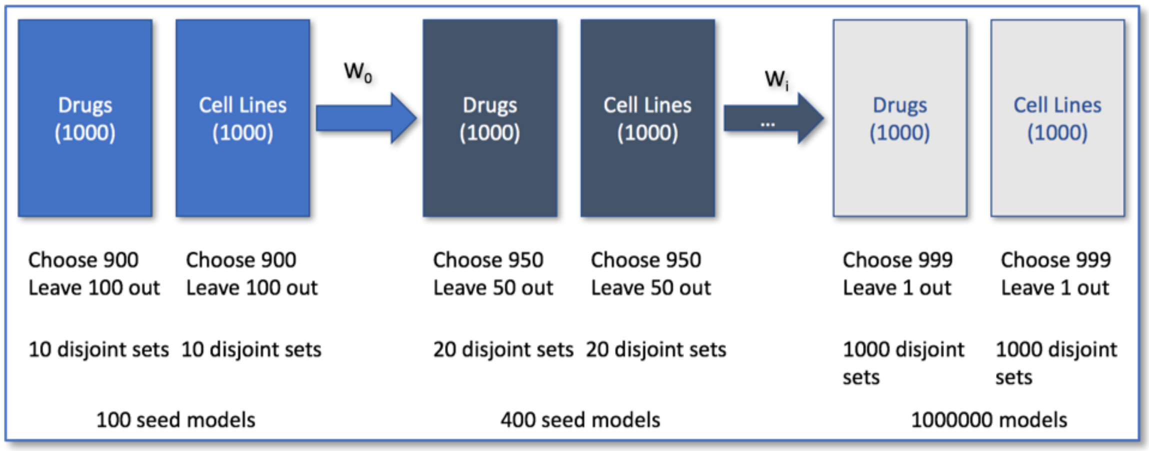


Figure 42: Illustration of a transfer learning design to train one million models for predicting the effect of a single drug on a single cancer cell line.

has three specific strategies to address these three challenges. For the drug response problem, unsupervised machine learning 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 RAS pathway problem, multi-scale MD runs are guided through a large-scale state-space search using unsupervised learning to determine the scope and scale of the next series of simulations based on the history of previous simulations. For the treatment strategy problem, semi-supervised machine learning 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. A 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). Take for example training a model that is specific to a certain drug and individual cancer. Starting with 1,000 different cancer cell lines and 1,000 different drugs, a leave-one-out strategy to create a high-resolution model for all drug by cancers requires approximately one million models. Yet, these models are similar enough that using a transfer learning strategy where 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 42 shows a general strategy of this weight sharing approach to training large numbers of models in which a transfer of weights delineates stages in the workflow.

In practice, speed up related to weight sharing can be discussed in the context of the challenge problem in terms of work actually done and naïve work done. Consider work actually done W_D being the number of jobs J times actual number of epochs E trained for all stages s,

$$W_D = \sum_{1}^{s} JE\,. \tag{17}$$

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

$$W_n = J_s E_c\,. \tag{18}$$

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

A number of parameters exist when considering accelerated model training via 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 or not to allow those weights to be updated in subsequent models. The requirements associated with the science challenge problem of training large number of high-resolution models are outlined in Table 47.

Table 47: 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 machine learning solutions will be computed for the three cancer pilots. • Pilot1—leave-one-out cross validation of roughly 1,000 drugs by 1000 cell lines. This involves roughly one 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 (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 were not in the training set of the model at iteration i. • Pilot2—state 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 1,000s of simulations, each of which are 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 10K curated text training sets. To accelerate model training, the team will use a transfer learning scheme with sharing of weights during training.
Demonstration calculation requirements	The computation performed at scale will be standard neural network computations, matric multiplies, 2D convolutions, pooling, etc. These will be specifically defined by the models chosen to demonstrate transfer learning. The computations performed at scale will require sharing of weights.
Resource requirements to run demonstration calculation	For each Pilot, it is estimated that each pilot problem will require up to 12 hours at full system.

6.2.2 *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 of the three pilot applications will focus on different deep neural network 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,

$$H = \frac{n}{\frac{1}{x_1} + \frac{1}{x_2} + \cdots + \frac{1}{x_n}} . \tag{19}$$

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 will need to choose a few numbers of models (ideally one per pilot project), fix the input data and fix the number of epochs associated with each model so that repeated measurements of the FOM can be compared to 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 where computational resources or training to convergence exceeds standard queue policies, and if it may be assumed that the time per epoch is constant, H_w remains a valid choice. The weighted harmonic mean of the rate of model training can be represented as

$$H_w = \frac{n}{\frac{e_{c1}}{e_1 x_1} + \frac{e_{c2}}{e_2 x_2} + \cdots + \frac{e_{cn}}{e_n x_n}} , \tag{20}$$

or

$$H_W = \frac{n}{\sum_{i=1}^{n} \frac{e_{ci}}{e_i x_i}} , \tag{21}$$

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 using training benchmarks for Pilot1 and Pilot3 models. Integration of benchmarks for Pilot2 models is part of the CANDLE stretch goal.

6.2.3 *CANDLE: KPP Stretch Goal*

CANDLE will consider two KPP stretch goals:

1. For the RAS pathway problem, Pilot2, multi-scale MD runs are guided through a large-scale state-space search 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 ever 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 1.6M new Cancer cases per year) for the next ten years. This patient profile will be combined with the GDC database to generate a national level tumor population estimate (including tumor type, genomic, transcript profile, etc.). The resulting "digital twin" database of 16M Cancer patients will then be processed using the drug response models to evaluate optimal drug

treatments strategies for each virtual patient. The team will use the “Uno-MT-UQ” drug response model to predict 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. In addition, the team will predict drug response for approximately 5,000 SoC drug combinations using our “Combo-UQ” model. The result would be the first ever national scale predictive oncology “benefit” map, outlining which Cancers, which regions and which population groups would benefit the most from a comprehensive implementation of precision medicine.

6.2.4 CANDLE: Progress Towards Advanced Architectures

GPU Strategy

The efficient use of the GPU is crucial to utilize exascale computing capabilities thoroughly. We continuously monitor the optimal use of computing resources from a single GPU level to the application level. We use profiling techniques to identify bottlenecks in our benchmark codes and refactor to achieve better GPU utilization. We periodically revisit the benchmark codes to spot room to improve.

Our GPU strategy for codes needed to complete our exascale challenge problems spans leveraging advances in hardware to improvements in the application codes. Each of these is discussed briefly.

- **Hardware**: As new hardware becomes available along with the necessary supporting libraries, we will examine the runtime and learning metrics associated with our benchmarks, and make necessary changes to leverage these advances.
- **Frameworks**: As new frameworks are released, we will analyze the runtime and learning performance of these changes. When appropriate, we make necessary changes to our code to benefit from performance improvements in new releases of the deep learning frameworks. We are able to achieve speedup with framework supported multi-GPU functionality. We profile multi-GPU scenarios to improve performance further, in addition to examining other mechanisms for data-parallelism and model-parallelism.
- **Libraries**: Improvement to the lower level libraries such as new releases of cuDNN and new optimization such as tensor cores are continuously evaluated.
- **Precision**: Mixed Precision or lower precision such as FP16, BFP16 should have a significant impact on runtime performance. Early tests show that learning metrics of the deep learning models are not significantly impacted by lower precision numbers. Multiple reports support the range of half-precision is sufficient for machine learning applications as long as the gradients are adequately scaled. The half-precision operation can speed up training significantly in Summit. The Nvidia V100 GPU offers 125 teraFLOPS with tensor cores, which is eight times larger than the single-precision FLOPS.
- **Weight Sharing**: Reducing the overall time to train thousands of models by sharing weights between networks has the potential to have a significant impact on the runtime performance of our benchmarks.
- **Model Properties**: Model Improvements based on things such as hyperparameters, network pruning, and learning schedulers are continuously pursued as part of our continuing pursuit of the best predictive model.
- **CANDLE Improvements**: Software improvements to CANDLE and the Benchmarks are resulting from ongoing profiling. At the application level, we exploit various resource-set configurations in Summit. CANDLE can combine 6 GPUs in a single resource-set and employ multi-GPU parallelization, or split into six resource-sets and run code in single GPU mode. In this case, the CANDLE workflow will be able to train six times more models at the same time. We will examine the best configuration to accomplish our challenge problem.

Progress to Date

We have been testing with a subset of challenge problems and are planning to use the multi-GPU mode in the next challenge problem run. We are still in the process of understanding the training characteristics and

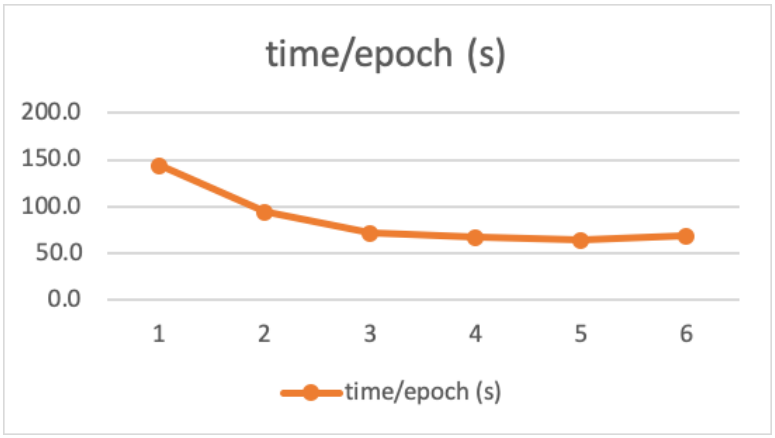


Figure 43: Training time on multi-GPUs.

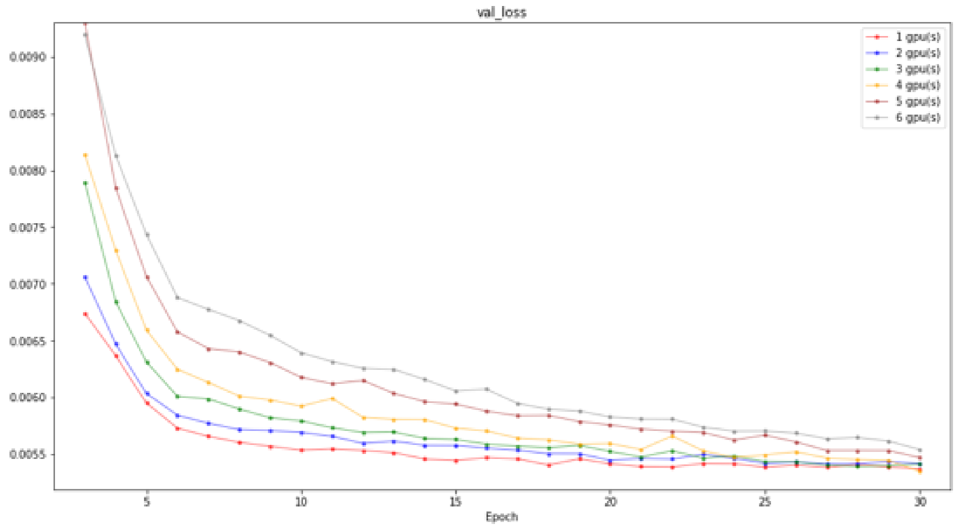


Figure 44: Validation loss changes during training.

relationships between runtime performance and model performance. We were able to gain 2× speedup with 5 GPUs (Fig. 43), but we observed some model accuracy degradation as we increase the number of GPUs. The models were not conversing as fast as the single-GPU mode (Fig. 44). We think degradation is negligible because, after 30 epochs (which is one-tenth of training requirements), the differences were becoming within 0.05 %.

We started evaluating half and mixed-precision training with our benchmarks. We observed 1.7× speedup on mixed-precision training (Fig. 45) as measured by the time to make one pass over the training data set. However, the model validation loss converged more slowly, thereby negating the runtime performance gains due to more passes over the data being required to achieve the same loss value. We are investigating this further and treat it as a preliminary result.

With one of our Pilot1 benchmarks (TC1), we used the automatic mixed-precision function provided by Keras. The feature is still in experimental mode. We will follow up with the framework updates. Besides, we tested half-precision training with the Pilot3 benchmark (Fig. 46). We compared training speed (time per epoch) between the single-precision and half-precision across one to six GPUs. We will continue to evaluate the mixed-precision training with other benchmarks with a careful review of speed gain training accuracy.

Next Steps

For Aurora, Intel GPU supports `bfloat16` with an 8× speedup. We aligned our efforts in the mixed-

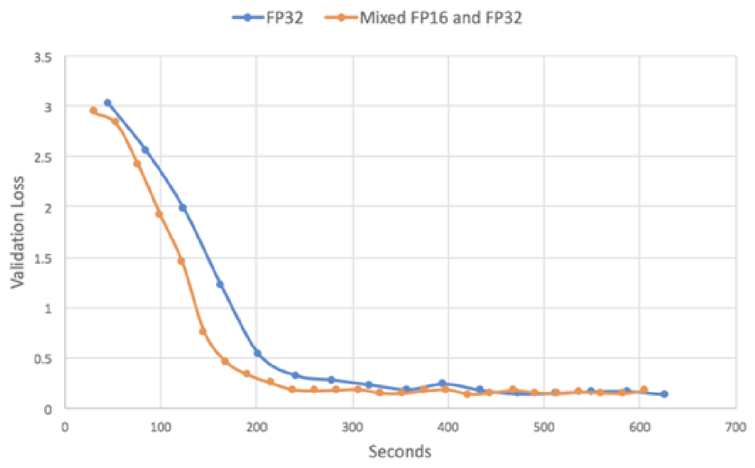


Figure 45: Validation loss change on time; benchmark TC1; single precision vs mixed.

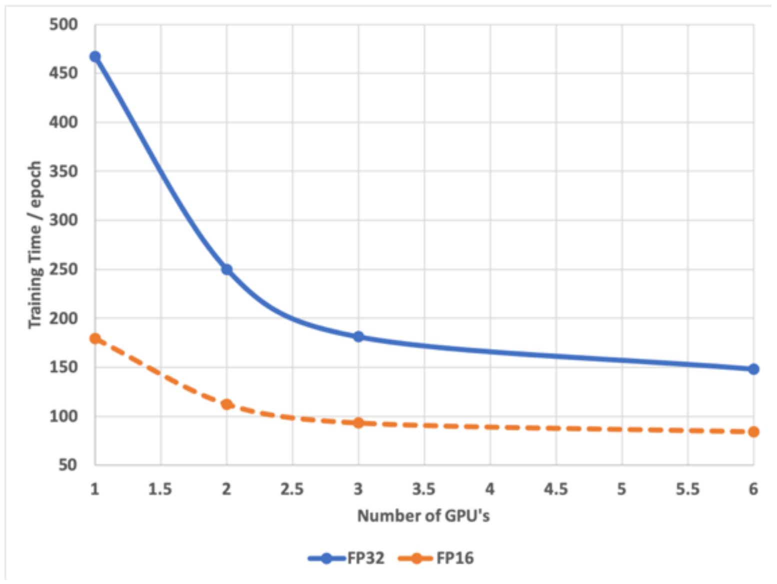


Figure 46: Training time difference between single-precision and half-precision over multi-GPUs; benchmark P3B4

precision training in this direction. We will evaluate Intel Deep Neural Network Library (DNNL) and collaborate by providing our examples and feedbacks.

For the Frontier machine, we will continue to collaborate with Cray and AMD engineers. The Frontier Centers of Excellence (CoE) machine for ECP projects, name Poplar, will be available by mid-Nov 2019. Once we can access the system, we will start evaluating the current version of software stacks in addition to the AMD deep learning libraries.

6.3 ExaBiome

Metagenomics—the application of high-throughput genome sequencing technologies to DNA extracted from microbiomes—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:(i) metagenome assembly, which takes raw sequence data and produces long genome sequences for each species; (ii) protein clustering, which finds families of closely related proteins; and (iii) signature-based approaches to enable scalable and efficient comparative metagenome analysis, which may show variability of an environmental community over time, for example.

The ExaBiome team has developed a scalable metagenome assembler, MetaHipMer, which scales well on thousands of compute nodes on today's petascale architectures and has already assembled large environmental data sets that had not been possible with previous tools. They continue to work on further scalability improvements across nodes and new node level optimizations to take advantage of fine-grained on-node parallelism and memory structures, including GPUs. MetaHipMer exhibits competitive quality with other assemblers, and the team continues to add innovations and parameters to control various aspects of how the data is analyzed, driven by the experience of science teams. MetaHipMer is designed for short read (Illumina) data, but a second assembler for long reads is also under development and shows even higher computational intensity, which may be a good fit for exascale systems. A second ExaBiome code, HipMCL, provides scalable protein clustering. HipMCL runs on thousands of nodes and has already been used to provide insight on the structure of protein families across hundreds of millions of proteins, a data set that was previously intractable. These codes and comparative analysis tools use some common computational patterns, including dynamic programming for string alignment (either DNA or proteins) with minimal edits, counting and analysis of fixed-length strings (k-mers), and a variety of graph and sparse matrix methods. The ExaBiome challenge problem focuses on metagenome assembly, but that capability will enable exascale feasibility for other bioinformatics problems in ExaBiome and more broadly.

6.3.1 ExaBiome: Science Challenge Problem Description

ExaBiome's challenge problem is to demonstrate a high-quality assembly or set of assemblies on at least 50 TB of environmental data (reads) that runs across a full exascale machine. The intent is to use a scientifically interesting environmental sample that may include multiple temporal or spatial samples and to be performed as a single assembly using complete sequence data. In contrast, current state-of-the-art assembly pipelines are forced to use subsampling when datasets get large, which limits the ability to assemble rare, low-coverage species, and with confusing duplications of genomes. Furthermore, assembling data across time and spatial scales together will not only enhance the assembly quality, but could reveal functions that otherwise would remain hidden. Addressing this challenge problem will demonstrate a first-in-class science capability using the power of exascale computing combined with novel graph algorithms. There are many potential beneficial science impacts, for example, enhancing 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 utilize an exascale machine. However, the challenge problem could also use synthetic data with environmental characteristics or an ensemble assembly of multiple independent environmental data sets. It may also use short reads, long reads, or a hybrid of the two. Challenge problem specifications for ExaBiome are listed in Table 48.

Table 48: ExaBiome challenge problem details.

Functional requirement	Minimum criteria
Models	Given a set of genome fragments 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, distributed hash tables.
Problem size and complexity	50 TB of environmental data (real or synthetic) using short reads, long reads, or a combination.
Demonstration calculation requirements	To demonstrate the challenge problem, the team will need to run a complete assembly, since all the stages need to 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 it is progressing at a reasonable rate. The team will also use intermediate problem sizes 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. The extrapolation starts with the fact that it required about 4.5 hours to assemble a 3 TB dataset on 1,000 Cori KNL nodes. The computational complexity scales with the size of the dataset (assuming good load balance), so a 50 TB dataset should take $\sim$75 hours on 1,000 Cori KNL nodes (ignoring for a moment the memory requirements). If 50% scaling efficiency from 1,000 nodes to the full Cori KNL system (9,000 nodes) is assumed, this should be around 17 hours, and assuming once again 50% scaling to a full exascale system (hence going from 30 PFLOP to 1 EFLOP) it could take about an hour.

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 environmental data. The ExaBiome project is developing exascale tools for 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 may also be combined with other data sets, including the NIH Sequence Read Archive (SRA) and the newly formed National Microbiome Data Collaboratory. The ExaBiome team is also building protein analysis tools that operate on large community databases, including exploration of deep learning techniques in addition to the PISA + HipMCL clustering pipeline. The comparative metagenomics work will also lead to new analysis techniques based on k-mer profiles, alignment against databases, or machine learning and may operate on assembled or unassembled data. Realization of this goal may also involve support for new long-read sequencing technologies or high-quality extension of those techniques, which may 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 n using environmental samples collected over space or time for DOE applications in energy and the environment.

6.3.3 ExaBiome: Progress Towards Advanced Architectures

GPU Strategy

The ExaBiome project has three major thrusts: genome assembly (MetaHipMer for short reads and diBELLA for long reads,), protein clustering (PISA plus HipMCL), and metagenome comparative analysis (metamer and others). The baseline challenge problem requires assembly using either MetaHipMer or diBELLA with the stretch goal involving the other tools. There are three computational kernels that dominate the local node computation including k-mer counting, alignment, and sparse matrix multiplication. There are other local computations, but these three dominate the computation across our application. k-mers are fixed length strings of either DNA or proteins, and alignment involves finding the minimum set of edits required to make two sequences (again either DNA or proteins) match. The dominant data structures across these applications are hash tables and (sometimes stored as a hash table) sparse matrices.

Because of the small set of computational kernels and existing 3rd-party libraries for all three problems, our GPU strategy involves use of optimized libraries for each algorithm that can be used on different alphabets (4-character DNA or 21-character proteins) and supports various heuristics for cutting down in the quadratic cost of pairwise alignment or the quartic running time for all-to-all-alignment. These heuristics are common in practice but tailored to a given problem, e.g., cutting off search for an alignment early if it is clear no high quality alignment is possible. We have explored existing 3rd party libraries, but are building our own implementations, currently optimized for NVIDIA hardware using CUDA, but adaptable to Intel and AMD GPUs with reasonable levels of effort. We have sent team members to meetings and training sessions to learn about the hardware and programming tools, and expect to use HIP on Frontier and SYCL on Aurora for our computational kernels. We will continue to track OpenMP, OpenCL, Legion, and Kokkos, but believe we can gain better performance and reasonable portability by having a library of kernels optimized for these systems.

MetaHipMer uses fine-grained one-sided communication which is mixed with computation rather to maximize overlap than having bulk-synchronous communication and computation phases. It is written in UPC and UPC++ and runs in top of GASNet. To address the need for low-overhead, modest-size communication events, we are exploring the use of accelerator-initiated communication that transfers data directly between accelerator memories on multiple nodes. We have worked closely with the PAGODA ECP team on UPC++ and GASnet and will continue to do so in identifying specific requirements for the use of accelerators in the ExaBiome applications.

Progress to Date

We have developed GPU implementations for two alignment algorithms (striped Smith-Waterman for short sequences and x-drop for longer ones), k-mer counting, and sparse matrix-matrix multiplication.

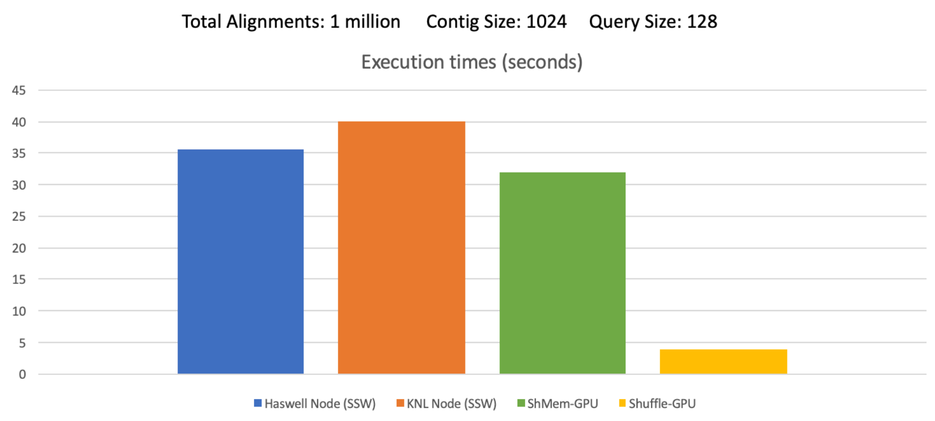


Figure 47: Single node SmithWaterman performance with our GPU code.

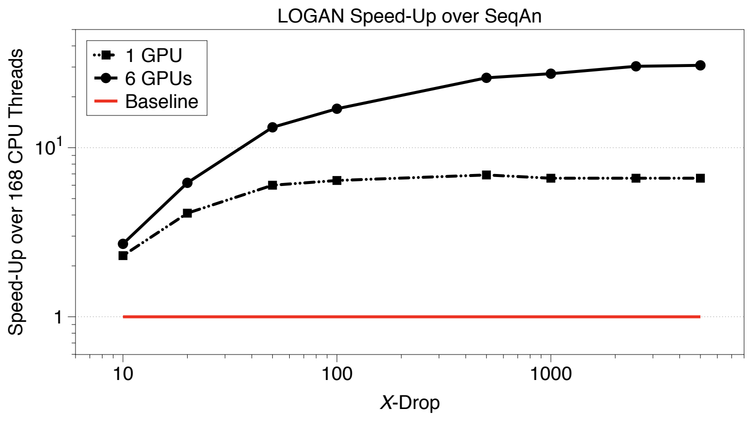
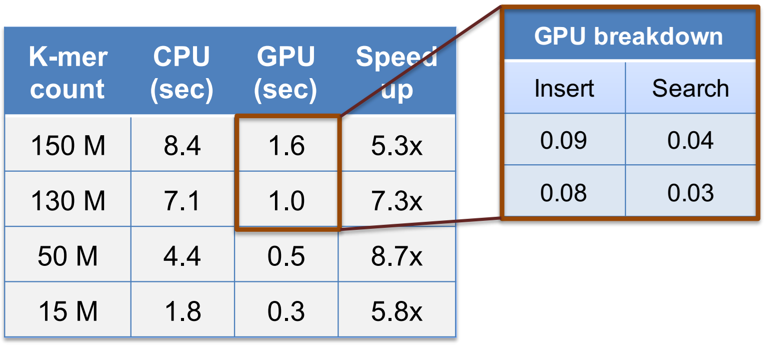


Figure 48: Single node x-drop speedup comparing our LOGAN GPU code to a popular multithreaded library, SeqAn.

Both of the alignment algorithms perform dynamic programming, filling in a nxm matrix for strings of length n and m. The use of Smith-Waterman in ExaBiome applications operates on relatively short strings, i.e., at least one string is a *read* of 100–200 characters, which is not sufficient to overcome GPU startup overheads. The longer reads supported for example in diBELLA can be over 10,000 characters, but the x-drop algorithm used in this case both limits the dynamic programming matrix to solutions that have limited numbers of mismatches (is dynamically banded) and also stops early when no good solution can be found. In both cases the running time is fairly short, so good GPU performance requires a "batched" approach that aligns a set of pairs in parallel on the GPU. (Note that this eliminates many of the existing implementations from consideration, because they get good GPU speedups on long strings but are not competitive in overall running time.)

Figure 47 shows the running time of a batched Smith-Waterman algorithm running on single nodes at NERSC, covering CPU (Intel Haswell), Manycore (Intel KNL) and GPU (NVIDIA V100) nodes. These benchamrks use one thread per core on the Haswell and KNL nodes, so maximizing parallelism there as well as the GPU node. The two GPU version are both from ExaBiome team in collaboration with the NERSC NESAP program, with the most recent version showing an 11× speedup relative to a single Haswell node.

Figure 48 shows the speedup of the x-drop alignment algorithm varying the cutoff value x. For larger values of x the running time is slower and the GPU speedup larger. Small values of x (under 100) can miss alignments that are important for overall assembly quality, although the relationship depends on other parameters in alignment. This comparison is done on Summit using the v100 GPUs and 168 threads on the Power9 CPU for the SeqAn code, a popular externally developed x-drop implementation.



K-mer count	CPU (sec)	GPU (sec)	Speed up
150 M	8.4	1.6	5.3x
130 M	7.1	1.0	7.3x
50 M	4.4	0.5	8.7x
15 M	1.8	0.3	5.8x

GPU breakdown	
Insert	Search
0.09	0.04
0.08	0.03

Figure 49: Single node k-mer counting GPU speedup relative to a 64-thread Haswell implementation.

Figure 3 shows the running time of k-mer counting on various numbers of k-mers (in millions). These single node results were taken on Cori and compare 64 threads on the Intel Haswell nodes to a single NVIDIA V100 GPU. The CPU code is based on the ExaBiome code from the single node diBELLA implementation (BELLA) and the GPU code is our own implementation.

The Markov Cluster (MCL) algorithm is one of the most popular algorithms for clustering biological data. HipMCL is a high-performance distributed memory implementation of the Markov Cluster algorithm that heavily relies on computations on sparse matrices. The MCL algorithm iteratively alternates between two successive steps of expansion and inflation until it converges. The expansion step performs random walks of higher lengths and it enables connecting to different regions in the graph. The inflation step aims to strengthen the intra-cluster connections and weaken the inter-cluster connections. One step of the random walk from all vertices can efficiently be implemented as sparse matrix squaring, a special case of sparse matrix-matrix multiplication (SpGEMM). Figure 50 shows the overall time for HipMCL running on 100 nodes of Summit with the time broken down into the major computation and communication phases. The optimized" code uses the GPU nodes with an without overlap, resulting in a 12.4× speedup relative to the MPI HipMCL code that uses maximum threading on the Power9 nodes. The algorithm uses different (pre-existing) GPU implementations for the matrix multiplication, since the best choice depends on the (sub)matrix characteristics.

Next Steps

There are several steps already underway in preparing our applications for exascale systems. The first is time breakdowns for the overall assembly process on large environmental genomes. Our target exascale problem is 50 Terabytes (TB), and we have completed a 3 TB assembly and are working on a larger one. However the dominant time breakdown varies with the input characteristics and has changed dramatically over the past several months due to scalability improvements and quality enhancements. This will give us a better idea of the total potential for GPU speedups from alignment and k-mer counting, for example, or whether communication performance will dominate. In addition, we plan to integrate the GPU-optimized alignment algorithms into the parallel many-to-many alignments that are performed in MetaHipMer, diBELLA, and PISA (the expensive matrix construction stage the precedes HipMCL). In addition, we are continuing to improve the GPU implementations, scale to multiple GPUs per node, and find new data layouts, parallelism strategies, or overlap opportunities. We are also tracking the sequencing technology that is evolving along with the exascale planning, and are placing additional resources into long read technology, both as a hybrid with short read data sets and standalone data.

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.

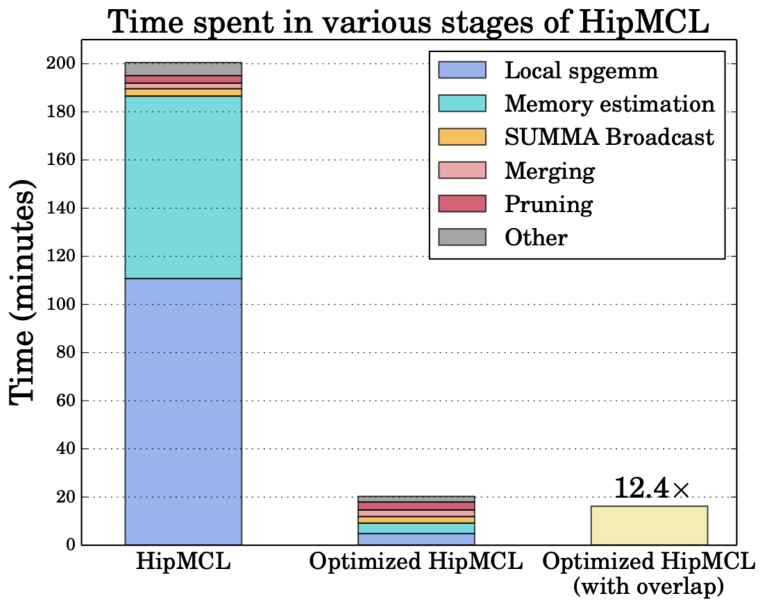


Figure 50: The running time HipMCL and the GPU-optimized HipMCL on a network of 35 million proteins and 17 billion connections on 100 nodes of Summit.

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.

Users of the LCLS require an integrated combination of data processing and scientific interpretation, where 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 (∼10 min) of data bursts, requiring commensurate bursts of exascale-class computational intensities.

The high repetition rate and ultra-high 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 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 using non-crystalline 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 may involve only a few atoms out of tens of thousands, and therefore the X-ray diffraction difference effects will only be on the order of 1-2% of 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 the creation of an automated analysis pipeline for serial femtosecond crystallography (SFX), also known as nanocrystallography. While 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.

The basic workflow is envisioned as a parameter-optimization inverse problem. Traditional crystallographic data analysis is used to determine approximate starting values for both the structure factors and geometric factors. The nanoBragg / CCTBX software is used to forward-simulate the diffraction images using a

Table 49: 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 calculation is on-going, ability to delegate data across multiple nodes for analysis, ability to exchange/average the parameter estimates across nodes, ability to offload the most computing intensive tasks (e.g. X-ray tracing modeling step with nanoBragg) to GPU accelerators.
Demonstration calculation requirements	At a minimum we'll run SFX against $O(10^7)$ images. If resources (e.g., memory) will be available, we'll perform the full demonstration calculation of 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 minutes to run the demonstration calculation for $O(10^9)$ images.

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 is 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 at the same time.

In order to show the scalability of the analysis pipeline, we plan to progressively increase the fraction of the machine used for reconstruction while keeping constant the number of diffraction images distributed across multiple nodes. 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).

6.4.2 ExaFEL: KPP Stretch Goal

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

Resource orchestration

This stretch goal will aim at scaling resource orchestration capabilities in order to allow:

- stream the science data from the beamline to the supercomputer at LCLS-II throughputs; and
- start analysis within seconds from the start of data collection.

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

Single Particle Imaging

This stretch goal will aim at scaling SPI reconstruction with the M-TIP algorithm (multi-tiered iterative phasing) in order 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 the scaling goal is due to the significant R&D challenge required to deserialize M-TIP and hence being able to scale. The main risk associated with being able to run against experimental data is that the ExaFEL team doesn't have control over which experiments will be performed at the LCLS during the lifetime of 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 non-identical 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, 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.

6.4.3 ExaFEL: Progress Towards Advanced Architectures

GPU Strategy

ExaFEL has a two prong GPU strategy: (1) optimize the most computing intensive ExaFEL kernels on accelerators and (2) optimize the execution of the ExaFEL applications by overlapping concurrent work on the CPU and GPU. Examples of the former effort during FY19 were porting/optimizing the experimental data analysis code for the Multi-Tiered Iterative Phasing for Fluctuation X-ray Scattering algorithm (M-TIP FXS) to the leadership class machine Summit and porting/optimizing the X-ray tracing package nanoBragg to the NESAP system (Cori rack with NVIDIA GPUs). An example of the overlapping effort in FY19 was the development of a proxy application for M-TIP FXS to exploit the advantages of the Legion task-based programming model. The next section details the progress to date in this two prong strategy.

Progress to Date

M-TIP FXS is an algorithm that takes as input the experimental data from light source experiments on samples such as viruses and provides structural information about ensembles of molecules. The acceleration effort for M-TIP FXS focused on developing code that executes on both the central processing (CPU) and acceleration (GPU) hardware on the Summit supercomputer cluster and on comparing the code performance before and after porting. Key takeaways are that the original code performance was dominated by Fourier and matrix-matrix operations. GPU acceleration via CUDA showed an 8-fold speedup. Note that 90 % of the code was ported to CUDA for this speedup.

On the nanocrystallography side, our eventual goal is to use exascale computing to improve the interpretation of protein diffraction by inverse modeling: to this end, we will modify the model of the crystal until the predicted pattern matches most closely with the observed data. NanoBragg is an algorithm that simulates an X-ray diffraction pattern given a complete description of a protein crystal. In the past year we participated in the NESAP program, aiming to develop a GPU-ready version of nanoBragg for Perlmutter. X-ray patterns are simulated by a single nanoBragg kernel, for which we have both CPU (OpenMP/C++) or GPU versions, held together by a Python framework. Acceleration is achieved (either OpenMP or GPU) by calculating image pixel intensities independently. Initially we found that a 2,000 seconds image simulation (Cori KNL, 16 OpenMP threads) was reduced to only 100 seconds using a single Nvidia device on the Cori-GPU testbed. Further analysis was performed in a NESAP hackathon addressing CUDA performance. We achieved an additional 7-fold GPU performance gain by refactoring our kernel calls to eliminate unneeded host-to-device and device-to-host memory transfer and to remove typecasting, and debug-mode instrumentation. Finally, the CUDA code was profiled, leading to an additional 1.6x gain by taking advantage of the fact that our use case involves sparse diffraction patterns, thus avoiding calculations for blank pixels.

On the programming model side, we have been investigating the potential of Legion to provide a flexible task-based execution model. Developers mark functions as tasks, and tasks are parallelized automatically by the system according to the dependencies that exist between tasks (based on what data is read and written by

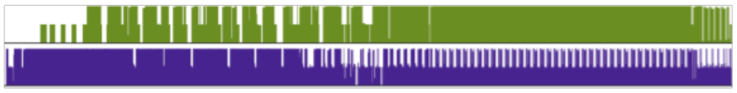


Figure 51: MiniFEL overlapping CPU/GPU utilization.

each task). Among other things, tasks make it very easy to express nested or hierarchical parallelism, as well as parallelism on heterogeneous processors such as accelerators, and make it easy to overlap communication and computation as well as work on the CPU and GPU. Thus we wanted to conduct an evaluation with an iterative code base, like M-TIP, that was critical to our future developments and that had the potential of showcasing Legion's advantages. To do this, we developed a proxy application called MiniFEL which implements a simplified subset of the functionality in the full M-TIP code base.

MiniFEL works by reading a set of diffraction images using the Psana analysis framework. It then uses the orientation of each image to merge it into a 3D diffraction volume. This 3D diffraction volume is then analyzed via two algorithms called HIO (hybrid input-output) and ER (error reduction) to recover the unknown phase information from the diffraction volume. This can then be used to reconstruct the 3D electron density of the molecule under consideration. The core HIO and ER loops involve the use of FFTs and inverse FFTs, in addition to some smaller kernels. These were implemented first in Python using NumPy, then ported to CUDA (using cuFFT) for execution on the GPU. The proxy application consists of about 400 lines of Python, with 230 lines of CUDA code, and additional 5,000 lines of Python code for detector corrections and other utility functionality. This small size was accomplished in part because the Python bindings for Legion are quite concise, and the core algorithms themselves were provided either by NumPy or by cuFFT. The Legion Python bindings were also expanded and made more complete as part of this work and a number of issues were addressed.

In order to evaluate the quantitative impact of Legion's ability to overlap CPU and GPU work, we performed an experiment on Summit with 10 concurrent reconstructions on 2 GPUs. By nature, the speedup that can be achieved by this method can be at most 2×, and depends on the ratio of work available to execute on the CPU and GPU. Our results serve as a proof of concept to demonstrate that this capability is available and works as intended. Due to the simplified nature of MiniFEL it was not necessary to use more GPUs or nodes.

Legion achieves a 1.8× speedup when overlapping work on both CPU and GPU, as compared to allowing the CPU to block and be idle while the GPU is working. In this test the preprocessing stages of MiniFEL (to align each image within the 3D volume) were executed on the CPU, while the reconstruction of the 3D electron density was performed on the GPU. Note that even in the reconstruction, a certain amount of work remained on the CPU, and Legion was able to successfully overlap this work by making use of task parallelism between the independent reconstructions.

Figure 51 shows a utilization plot of the CPU and GPU over time as the program is executing. This plot was generated automatically via Legion's profiling tools which record the utilization of each processor. As can be seen in the plot, the execution of the program is bimodal. In the first half the execution, data is still being actively collected, and the preprocessing of this data shifts the balance of work towards the CPU. In the second half of the execution, the collection and preprocessing of data is complete and the balance shifts towards the GPU. Legion automatically chooses the best scheduling based on the current balance of work between CPU and GPU.

Next Steps

Most of last year work was dedicated to single node acceleration of the ExaFEL codes. For next year, we plan to scale these codes on Summit and on the NESAP system. In addition, regarding the M-TIP effort, we'll investigate why porting the remaining 10 % of the code to CUDA did not produce a speedup. New risks and challenges include ensuring the accelerated M-TIP FXS code works on various GPUs that will be on the exascale computers. Finally, we'll start investigating the most computing intensive kernels in the SPI version of M-TIP.

Last year results gave us confidence that Legion is indeed able to overlap work on the CPU and GPU. Although this comparison focused on that specific aspect of overlap, Legion's task scheduling capability

Table 50: 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	MARBL	ATDM LLNL Application	KPP-2
2.2.5.03	SPARC and EMPIRE	ATDM SNL Applications	KPP-2

proved to be generally useful towards overlapping computation and communication, or computation and I/O, etc. We feel this capability will be especially important in the context of the upcoming exascale platforms as many of the hardware details and their performance are still unknown, and Legion's ability to flexibly schedule tasks provides valuable mitigation against potentially sources of performance degradation in more traditional, bulk synchronous programming models.

7. NATIONAL SECURITY APPLICATIONS

End State: Deliver comprehensive science-based computational weapons applications able to provide, through 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 include projects (Table 50) centered on the stewardship of the US nuclear stockpile and related physics and engineering modelling and scientific inquiries consistent with that mission space. The focus of the Advanced Technology Development and Mitigation (ATDM) element of the NNSA ASC Program is on the development of new nuclear weapons applications, one for each of the three NNSA Laboratories (LANL, LLNL, SNL), with each application having a specific challenge problem target, namely, a currently intractable 3D problem of interest. For LANL and LLNL, the demonstration application includes one or more weapons-relevant simulations. For SNL, the demonstration applications include weapons-relevant simulations of re-entry aerodynamics and electromagnetic plasma effects. All three applications must exhibit efficient use of both of the exascale architectures, although one may be more efficient than the others, as well as acceptable multi-node scaling on existing systems at NNSA laboratories. 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.

The outcomes and products of this activity will be integrated into the next generation of integrated and high-performance ASC codes on advanced (next decade) architectures. The LANL approach is to concurrently develop a flexible framework, code infrastructure, and physics components, based on the outcome of an initial feasibility and scoping study. LLNL is developing algorithms to minimize data motion relative to computation, which are then incorporated into prototype codes built on a lightweight software layer. The SNL approach is to build agile components on top of a comprehensive toolkit, which includes a data model, an abstraction layer, and high-quality solvers. 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.

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 and time scales, and drive requirements for exascale computing. LANL is developing a next-generation multi-physics code for national security applications that focuses on 3D multi-physics, mesoscale insight for extreme condition materials, and high-energy density physics simulations.

7.1.1 *Ristra: Science Challenge Problem Description*

The ultimate goal of LANL's Ristra Next-Generation Code Project is to create a set of codes that must:

1. Solve multi-physics problems using computational methods with characteristics of those required for national security problems;
2. Do so in an efficient way on emerging high-performance computing (HPC) architectures leading to exascale;
3. Provide a flexible, extensible, productive programming environment featuring 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 testbed for the evaluation of novel programming models and data management technologies.

Computer science technologies that allow 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 multi-physics simulation, in addition to an exploration of programming models for emerging architectures.

Key to the architecture of Ristra's applications is FleCSI (Flexible Computer Science Infrastructure), an abstraction layer that provides the desired separation of concerns between computational physics and computer science. FleCSI provides an abstract data model supporting compile- and run-time configurability for implementing a variety of discretizations (mesh and mesh-free) and physics fields and operators over them, together with an abstract execution model that can target a variety of underlying parallel programming runtimes from well-established options (MPI) to ambitious new programming systems such as Legion, a data-centric model with out-of-order task execution. Evaluation of Legion's potential is an important goal for Ristra.

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

- *High Energy Density Physics for Inertial Confinement Fusion.* Ristra's Symphony code is an unstructured multi-material radiation hydrodynamics application that features a multi-scale algorithm for the radiation solve: a fully-coupled low-order radiation hydrodynamics system is updated by a high-order radiation solver which has the potential to be executed asynchronously (work in progress).
- *Multi-Scale Hydrodynamics of Materials in Extreme Conditions.* Ristra's FUEL code is an unstructured multi-material Arbitrary Lagrangian-Eulerian (ALE) hydrodynamics code that can be coupled to complex material models in order to take account of mesoscale physics, such as grain structure, in the dynamic response of materials. Mesoscale modeling is computationally intensive, and the multi-scale approach has potential for effective use of exascale-class systems, as well as providing a promising target for data-driven machine-learning (ML) techniques.

7.2 MARBL

LLNL is developing next-generation multi-physics simulation capabilities for national security applications and has adopted a modular approach to code development. A foundational component of this approach is the Axom computer science (CS) toolkit which provides infrastructure for the development of modular, multi-physics 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 Magneto Hydrodynamics (MHD), EOS and material strength studies as part of the NNSA's SSP.

MARBL is 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. A 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/accuracy per unknown compared to standard low-order finite volume schemes and because they have computational characteristics which play to the strengths of current and emerging HPC architectures. Specifically, they have higher FLOP/byte ratios meaning that more floating-point operations are performed for each piece of data retrieved from memory. This leads to improved strong parallel scalability on GPU platforms and increased computational efficiency.

A key goal for MARBL is enhanced end-user productivity including improved workflow for problem setup and meshing, simulation robustness, support for UQ 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 multi-physics simulations on advanced architectures and less manual user intervention.

MARBL is part of a larger effort, the Multi-Physics on Advanced Platforms Project (MAPP). MAPP also includes the separate development of national security application codes following the same basic development philosophies outlined above; modular infrastructure leveraging Axom developments, modular physics capabilities and multiple options for every major physics capability.

7.2.1 MARBL: Science Challenge Problem Description

Along with the other NNSA labs, the team is targeting a FY20 demonstration of a 3D multi-physics problem of interest to the stockpile stewardship program at LLNL on one-quarter of the Sierra machine as well as a performance portability demonstration by running a similar problem on the SNL Astra (ARM based) platform. The calculation will be multi-physics in nature, high-resolution and will utilize multiple new algorithms and novel capabilities developed as part of this work.

Success of MAPP will ultimately be determined by the degree of adoption of its simulation tools by the LLNL user community. To this end, emphasis at this relatively early stage of development is being 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 multi-material radiation-magneto-hydrodynamics, thermonuclear burn for ICF fusion calculations, general equations of state, material opacities and electrical conductivities, simulation diagnostics and queries, in situ analytics/rendering, and parallel computational and file IO performance at a massive scale. In addition, performance of the new codes on advanced architectures like the GPU based Sierra system at LLNL is critical. Portability of the software stack and long-term maintainability are critical as well, 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 multi-physics simulation at scale in the NNSA and has already produced first-of-a-kind simulation results using such methods.

7.3 SPARC and EMPIRE

The development of exascale computing presents an opportunity for Sandia to develop new capability to impact the labs' broad national security mission space. Sandia is pursuing development of two new applications under ATDM: SPARC, a hypersonic reentry simulation capability, and EMPIRE, an electromagnetic plasma physics simulation activity. These applications are using simultaneously developed next-generation components, exploiting Sandia's vision to enable applications to build on foundational capabilities developed and deployed by other teams that provides leverage and potential for reuse and increased impact. The ATDM project incorporates formal tools for planning, development, testing, version control, code reviews, and deployment, for both applications.

SPARC for Virtual Flight Testing

The SPARC (Sandia Parallel Aerodynamics and Reentry Code) application represents a revolutionary hypersonic reentry simulation capability that captures the random vibration and thermal environments created by re-entry of a vehicle into the earth's atmosphere. SPARC incorporates the innovative approaches of ATDM projects on several fronts, including effective harnessing of heterogeneous compute nodes using

Kokkos, exascale-ready parallel scalability through asynchronous multi-tasking, uncertainty quantification through Sacado integration, implementation of state-of-the-art reentry physics and multiscale models, use of advanced verification and validation methods, and enabling of improved workflows for users.

EMPIRE for Electromagnetic Plasma Physics

The EMPIRE application is an advanced electromagnetic and plasma physics capability that will support analysis of system-generated electromagnetic pulse (SGEMP) and source-region electromagnetic pulse (SREMP) phenomena. Validated computational simulation tools are critical because certain 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 EM fields over a wide range of conditions. EMPIRE incorporates the innovative approaches of ATDM projects on several fronts, including effective harnessing of heterogeneous compute nodes using Kokkos, exascale-ready parallel scalability through asynchronous multi-tasking, uncertainty quantification through Sacado integration, implementation of state-of-the-art plasma physics, use of advanced verification and validation methods, and enabling of improved workflows for users.

7.3.1 SPARC and EMPIRE: Science Challenge Problem Description

Each of Sandia's applications will perform a science challenge problem for the FY20 NNSA milestone that will run on at least a quarter of the Sierra machine, at least half of Trinity, and the entirety of Astra. Running at this scale will show the progress made in developing simulation capabilities that can operate effectively across the user workflow, from mesh generation, to problem throughput, file I/O, and visualization.

SPARC's Science Challenge Problem

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 simulation of the flowfield around the vehicle (including the aft end and its wake) 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 fluctuations predictions by the turbulence models will be used to analyze the vibrational response of the vehicle and its internal components.

EMPIRE's Science Challenge Problem

The science challenge problem for EMPIRE is to perform large-scale kinetic plasma simulations of an experiment fielded at the National Ignition Facility (NIF). In this problem, an X-ray source interacts with metallic surfaces to generate a plasma by the photoelectric effect, that then interacts with the geometry to generate currents on components of interest. This simulation will demonstrate the capability of EMPIRE and builds upon the progress made in FY19 on 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 on a mission-relevant problem to achieve a resolution fidelity beyond what is possible with the current plasma simulation capability.

8. CO-DESIGN

End State: Develop cross-cutting, motif-based CD software technologies and integrate them into applications, providing them the potential to fully utilize 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 CD of exascale software and hardware technologies through proxy application software.

The co-design activity includes six co-design centers focused on specific computational motifs. These target cross-cutting algorithmic methods that capture the most common patterns of computation and communication

Table 51: 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	Co-design Center for Online Data Analysis and Reduction at the exascale	KPP-3
2.2.6.04	CoPA	Co-design Center for Particle Applications	KPP-3
2.2.6.05	AMReX	Block-Structured AMR Co-design Center	KPP-3
2.2.6.06	CEED	Center for Efficient Exascale Discretizations	KPP-3
2.2.6.07	ExaGraph	GraphEx Co-design Center	KPP-3
2.2.6.08	ExaLearn	Co-design Center for Exascale Machine Learning Technologies	KPP-3

(known as motifs) in the ECP applications. The current list of motifs includes structured and unstructured grids (with adaptive mesh refinement), dense and sparse linear algebra, spectral methods, particle methods, Monte Carlo 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 may emerge over the lifetime of the ECP (including machine learning methods, the focus of a recently added co-design center), will be considered within the co-design activity, with the exception of dense and sparse linear algebra, which is supported within the ST focus area.

Each of the six funded co-design centers (Table 51) 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 co-design centers. The integration activity will assess application predictive maturity (e.g., as guided by the SNL Predictive Capability Maturity Model) and integration of exascale technology through regular, independent application assessments.

The goal of the co-design 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 co-design centers, a Proxy Applications project is managed within the co-design activity; its mission is to improve the quality of ECP proxy applications (“apps”) and maximize the benefit received from their use, including by maintaining and distributing the ECP Proxy App Suite. Proxy apps are tools to explore algorithms, data structures/layouts, optimizations, etc., and the associated tradeoffs 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 conducts unbiased evaluations of the capability, performance and scaling, and performance portability of ECP application codes.

8.1 CODAR

A 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 only at 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 major implications for the design of various elements of exascale systems.

CODAR, a co-design center focused on **online data analysis and reduction** at the exascale addresses this issue. Working closely with the ECP applications, CODAR is undertaking a focused process that targets both common data analysis and reduction methods (e.g., anomaly detection and 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, and applications in order to understand and guide tradeoffs in the development of applications and software frameworks, given constraints relating to application development 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 tradeoffs related to the treatment of scientific results created by scientific models, (2) produce high-performance implementations of data analysis and reduction methods, (3) enable easy and efficient integration of those methods with applications, and (4) contribute to the co-design 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 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 co-design tradeoff investigations.

8.1.1 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 co-design experiments, and (2) the development of high-performance implementations of online data analysis and reduction methods that are inspired by unique data access requirement and unfulfilled application needs. Descriptions of the components developed by CODAR are provided in the Infrastructure and Online Methods subsections, while a more detailed description of how the software is used with applications is provided in the Co-design Engagements and Integration Points section.

Infrastructure: Cheetah, Savanna, Chimbuko

Cheetah enables co-design experiments for improving performance and functionality of online analytics and reduction for exascale science. Cheetah is the interface for defining the co-design experiments. Specifically, the Cheetah component works to define a set of conventions and re-usable scripts for conducting parameter sweep experiments on different science application scenarios that are necessary for co-design studies.

Savanna is the runtime that can launch and manage and the individual co-design 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 enaction and enforcement of those workflows and their policies. The transitions between different scheduler systems, parameters set on command line versus 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 data sets. Because of the vast volume of data being produced by todays scientific simulations and experiments, lossy data compression allowing user-controlled loss of accuracy during the compression is a relevant solution for significantly reducing the data size. However, lossy compressor developers and users are missing a tool to explore the features of scientific data sets 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 needs to detect and track blobs and streamers in 5D gyrokinetic tokamak simulations, such as XGC.

MGARD is a technique for multi-grid adaptive reduction of data. Special attention is given to the case of tensor product grids, where the team's approach permits the use of non-uniformly spaced grids in each direction, which can prove problematic for many types of data reduction methods. An important feature of CODAR's approach is the provision of guaranteed, computable bounds on the loss incurred by the reduction of the data and the preservation of quantities of interest and statistics.

8.1.2 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, all running at the same time on the same or different nodes. Performance challenges can occur within any component and/or as a result of communication among components. Decisions that need to be made related to performance include the placement of components across the cores within the nodes (heterogenous node usage) or running the components in separate nodes (homogeneous node usage), mapping components to utilize CPU, GPU, and other resources, the types of memory to use within the nodes and across nodes for the different components (e.g., NVRAM), and the selection of 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 allow 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 (TAU and Chimbuko) that provides performance feedback.

2. **Application components**: The performance of individual application components is viewed as being 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 toolkit, the work to optimization 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 one to perform, collate, and analyze experiments to identify performance bottlenecks and either address them for internally developed software or relay them to application development and software technology 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 does lead to the application idling while waiting for I/O to complete. At the same time, the addition of the online data analysis and reduction methods needs to fit within resource restrictions (e.g., cores and impact on application

Table 52: CODAR KPP-3 goals and metrics

Passing value	Stretch value	Tentative present value
3	6	1

performance) set by the application development teams. If the team can simultaneously satisfy all these resource restrictions, it will be able to provide the right information at the right time and place to accelerate scientific discovery.

8.1.3 CODAR: Co-design Engagements and Integration Points

CODAR's engagement with application development and ST teams consists of three activities: infrastructure integration and co-design studies for online data analysis and reduction; the development of application-inspired data analysis and reduction methods; and investigations of 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.

Application-inspired Methods

In addition to the infrastructure integration and co-design 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., multi-grid or hierarchical access).

The application-inspired methods pipeline includes methods that are being developed into a production capability including the Feature Tracking Kit (inspired by WDMApp), the MGARD multi-grid data reduction method (inspired by combustion) and the performance anomaly detection methods (inspired by NWChemEx) and included in Chimbuko. In addition, activities in the exploration phase include developing improved statistical metrics for compression quality analysis (inspired by ExaSky and WDMApp and to be include in Z-checker) and numerical optimization-based compression (inspired by EXAALT (§ 3.4)). Two activities were stopped after prototyping: the functional data analysis topic (inspired by WDMApp) and the hierarchical data analysis topic (inspired by ExaSky (§ 5.2)). Further development of the hierarchical data analysis methods for computing halo centers may be undertaken by the ExaSky project and implemented in their 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 [22] that was 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` is 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 need to run multiple components at the same time, in such a way that termination of one component does not cause a computation to fail. This exercise thus constitutes a valuable co-design exercise for exascale system software.

8.1.4 CODAR: Progress Towards Advanced Architectures

GPU Strategy

Our GPU portability strategy depends on the type of software as described below.

Table 53: CODAR code base

Package name	LOC	Target exascale challenge problems	Computational motifs
Cheetah	∼2k	Launch and manage the execution of a set of co-design experiments on supercomputer platforms, as specified by co-design 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	10k	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 multi-component application (e.g., 1+ application modules, analysis modules, reduction models) on a supercomputer platform	Performance portability
Z-checker	40k	Data reduction error analysis (already used to assess compression errors for ExaSky, ExaFEL, EXAALT, GAMESS; to also be used for NWChemEx, QMCPACK).	Data reduction

The Cheetah/Savanna software focuses on command-and-control scheduling and placement for online, in situ, and code coupling scenarios. A key concern is a dependency for Savanna on the details of the scheduler interface for pinning and managing placement of processes onto CPU and GPU cores, as that is a primary testing point for a number of our client technologies. The abstractions in Savanna will allow us to add the necessary support for the schedulers to perform these placement studies.

The Chimbuko software analyzes performance trace data in real time for exascale applications and workflows and uses the TAU software to supply the performance data that is analyzed. At present TAU only collects performance data on the CPUs and so the Chimbuko services are only running on the CPU's. We are actively working with the TAU team to enable data collection on the GPUs, at which point we would move the Chimbuko analysis services to the GPUs to analyze the data where it is produced to avoid heavy data transfer costs.

FTK is a data analysis method for tracking features in the simulation data as they evolve over time. We are currently prototyping on-node GPU parallelism with Kokkos for feature tracking in 4D spacetime regular grid data. The kernel function that scans individual mesh elements is executed by Kokkos for performance portability.

MGARD is a compression methods being developed for data that resides on structured and unstructured grids. It was observed that about 75 % of the time spent in compression was in the backend lossless compressor used in MGARD. We will incorporate GPU-enabled lossless compressors including http://cudpp.github.io and http://bzip2-cuda.github.io, which use the CUDA programming model. We are also implementing the rest of MGARD to use CUDA in all of the time-consuming kernels.

Z-checker tests the quality of lossy compressors on simulation data. This tool integrates compressors that have been ported to GPUs using the programming model of their preference. The Z-checker calculation of statistical lossy compression quality metrics will to be ported to GPUs using Kokkos.

Progress to Date

The Cheetah/Savanna orchestration software has been ported to Summit and we have successfully orchestrated and performed demonstrations of the technologies on that system using the WDM and Gray-Scott miniapps. The Gray-Scott miniapp in conjunction with various compression and data reduction technologies is a key component, and we recently ported this miniapp to Kokkos and have an OpenACC version that is almost complete. This miniapp allows us to demonstrate the Cheetah/Savanna capabilities on using both CPUs and GPUs on Summit.

We have run Chimbuko on Summit using 1,000 nodes utilizing 20 MPI threads per node for evaluating a NWChem workflow, where TAU is used to collect the performance information that Chimbuko analyzes. When TAU support for capturing performance information on GPUs matures, we be able to run further studies using the GPUs on Summit.

We have been prototyping FTK support for on-node parallelism on a Linux sandbox with an NVIDIA GPU. The distributed CPU-only feature tracking has been benchmarked on Theta up to 1,024 processes with 40 % parallel efficiency. We anticipate similar scaling on GPU machines after the on-node parallelism is fully implemented and tested on a single node.

Our current progress on GPU optimization for MGARD has been done only on Summit. We worked with the fusion dataset from the WDMApp and showed that we can reduce the data by 8× when running MGARD on one Summit GPU. We have begun to investigate the asynchronous data movement from the code's main memory to MGARD routines, but since the WDMApp utilizes GPU, we were asked to use a small set of separate nodes. We use ADIOS to move the data in memory from the nodes running the WDMApp to the data reduction nodes, and in our tests it requires just one Summit node to reduce the data.

For Z-checker, only the compressors can utilize the GPUs so far. We plan to use Summit when port the statistical metrics integrated in Z-checker to use the GPUs. We are still in the phase of identifying application quality analysis metrics. Once identified, we will integrate in Z-checker the GPU version of the analysis code, if available.

Next Steps

For Cheetah/Savanna, we are targeting approaches to continue to understand the future scheduler and interconnect changes on the new platforms as we generalize the Cheetah/Savanna task placement interface to support different hardware through a common API. We also are continuing to work with the component technologies important to our codesign studies in order to understand how they will port to the new hardware so we can address the most relevant codesign workflow topics for them.

For Chimbuko, we are working with TAU to enable performance data collection on CPUs and GPUs. Once established we will explore where best to place the performance data analysis for both Summit and for the coming Exascale systems based on the data movement pathways and costs.

Our plan for FTK is to further implement and test FTK on a single GPU node and then benchmark FTK on Summit in order to prepare for running on Aurora and Frontier.

The next steps for MGARD are to integrate GPU lossless compression routine to MGARD and investigate Kokkos as the performance-portable programming model for the "rest" of MGARD.

We will use early prototype and/or simulators to port Z-checker statistical metrics for Aurora and Frontier accelerators using the portable Kokkos programming model. If performance does not acheive expectations, we will first attempt to optimize the Kokkos code before resorting to porting the critical kernels to the native accelerator programming model used on Aurora and Frontier.

8.2 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 (including ExaMiniMD, ExaSP2, CabanaMD, and CabanaPIC) and libraries. Libraries are modular instantiations that multiple applications can utilize 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 production codes (existing or newly developed), with both productivity and performance benefits.

8.2.1 *CoPA: Algorithms and Software Objectives*

PROGRESS/BML

The QMD capabilities are included in a computational framework aiming to foster developments in computational chemistry and electronic structure packages. This framework consists of two libraries (PROGRESS and BML) where 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 package (BML) 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) were found to be 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 while 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 (techniques) developed within PROGRESS. The current focus is on low-level BML library implementations on accelerated architectures (e.g., GPUs).

Cabana

The CoPA particle toolkit (dubbed "Cabana") is a collection of software packages which will allow scientific software developers targeting exascale machines to develop scalable and portable particle-based algorithms and applications. This toolkit provides an open-source implementation of a variety of basic particle-based algorithms and data structures applicable to a wide range of application types including (but not limited to) PIC and its derivatives, MD, and SPH codes. Cabana is written in C++, and usable by application codes written in C++, C, and Fortran.

Cabana provides native particle data structures, parallel programming APIs, and algorithm implementations using those data structures. These algorithms will span the space of particle operations necessary to support each relevant application type. This includes intra-node (local and threaded) operations on particles and inter-node (communication between nodes) operations to form a hybrid parallel capability. The initial set of algorithms sort particles 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 toolkit will be interoperable (e.g., easily permit linking and allow for simultaneous use) with other ECP scientific computing libraries which the user may leverage for other needed services not provided by, or not in the scope of, the Cabana toolkit. Use of Cabana in concert 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 inter-node (MPI) or intra-node (primarily Kokkos) level. The parent LAMMPS MD code, used in the EXAALT (EXascale Atomistics for Accuracy, Length, and Time) ECP project, has a similar modular structure as well as similar 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). While

ExaMiniMD was the initial MD proxy app, the newer CabanaMD provides full integration with the Cabana particle toolkit. Both provide useful platforms for development, as well as a compact expression of use cases which can be used for vendor engagement and experimentation. Code developed in ExaMiniMD has been ported both 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).

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. While several methods, including particle-particle particle-mesh (PPPM, or P3M) and particle-mesh Ewald (PME), utilize FFTs, others, including the fast multipole method (FMM), do not. As an initial step towards a general long-range solver library that enables the user to readily switch between and evaluate different algorithms for their particular use case(s), 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 $10{,}000^3$ 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.

8.2.2 CoPA: Performance Objectives

PROGRESS/BML

The challenge is to develop libraries that are portable across the range of available multicore, many-core 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 (including MAGMA, OpenMP offload, cuSPARSE/cuBLAS, and SLATE). This is exacerbated by the varying maturity of compliers and the underlying hardware, e.g., for OpenMP offload. 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

A 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 a user 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 may more easily vectorize on modern architectures and to make these developments accessible to users of the library. As a result, a user can compose their application with the Cabana library and compile it with cross-platform support, creating a single implementation which 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 re-implemented with a SHMEM option to test performance against conventional MPI. 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) are also being tested. While this work is driven by LAMMPS, it mostly involves internal Kokkos code, and will thus benefit other Kokkos-based applications within ECP.

SWFFT/fftMPI

Both 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 two-dimensional subgrid ("pencil" decomposition) is scalable but requires 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) versus 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 of 1d FFTs for GPUs and multi-core CPUs.

8.2.3 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 have been integrated into LATTE, and the team works closely with EXAALT members to co-design new capabilities and modifications as needed. As discussed above, libraries and proxies have been 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 which 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 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 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 co-design centers, potential integration of the Cabana and AMReX libraries for block-structured adaptive mesh refinement with particles has been discussed. While a direct coupling has not been achieved, the MFIX-Exa (§ 4.4) project (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 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 in their first quantitative performance assessment by comparing their computation and memory behavior to 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

Table 54: CoPA KPP-3 goals and metrics

Passing value	Stretch value	Tentative present value
3	7	4

into the CoPA co-design 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 co-design engagement. In this way, an apples-to-apples comparison can be made before and after code refactoring or adoption.

Software Technologies

The MAGMA library has been 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 added into BML 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 their experience. Performance of BML was assessed 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 offload 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 which Cabana uses to represent particle data and to implement algorithms and operations on particle data. Cabana developers have 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. In addition, 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 (Canning and Dongarra) have also occurred; they will use fftMPI as a baseline in their benchmarking and new algorithm development.

Vendors

The team has 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 NERSC GPU hackathon. Using the Cabana library. Team members have participated in the NVIDIA Summit on Summit/Sierra (SoSS) meetings and bi-weekly conference calls per GPU issues on Summit and Sierra. The team held videoconferences and in-person 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. While the team anticipates the Kokkos library to be transferable, the choice of AMD for the Frontier GPU will require a comparable deep engagement with AMD comparable to SoSS with NVIDIA.

8.2.4 CoPA: Progress Towards Advanced Architectures

GPU Strategy

CoPA includes multiple GPU strategies summarized as follows. CoPA depends on underlying portable libraries where possible, optimized for pre-exascale and exascale hardware. The LAMMPS/SNAP Potential Implementation and the Cabana particle toolkit are dependent on Kokkos to supply the GPU interface.

The PROGRESS/BML Libraries for QMD codes relies on a couple different strategies. The BML library relies on the MAGMA library for dense matrix operations on GPU, and eventually SLATE. *OpenMP offload* is the choice implementation on GPUs for sparse matrix operations, such as ELLPACK. There are still considerable differences in performance across compilers for OpenMP offload. We trust that there will be

Table 55: CoPA code base

Package name	LOC	Target exascale challenge problems	Computational motifs
PROGRESS/BML	70,900	EXAALT (LATTE)	Sparse/dense linear algebra
Cabana	10,000	ExaAM (ExaMPM), WDMAPP (XGC)	Particles, Long-Range Solvers
ExaMiniMD	6,200	EXAALT (LAMMPS)	Particles
SWFFT	3,600	ExaSky (HACC)	FFT
fftMPI	6,400	WarpX	FFT

increased performance as we reach exascale. The pre-exascale and exascale GPU BLAS and SPARSE libraries such as CUBLAS and CUSPARSE from NVIDIA will be used as needed. The PROGRESS library relies on BML to supply the GPU interface.

Progress to Date

The LAMMPS/SNAP Potential Implementation required complete restructuring of the SNAP algorithm and data structures to run efficiently on GPUs. The OLCF NERSC GPU Hackathon served as an opportunity to improve performance. SNAP GPU Performance over time is shown in Fig. 52.

The efficient implementation of the SNAP potential for GPU acceleration required re-engineering the SNAP kernel from the ground up, adopting compact memory representation and improving memory access patterns, breaking up of the force kernel into sub-kernels and pushing atom/neighbor parallelism into the sub-kernels.

This resulted in a significant increase of roughly 5× in performance over the baseline implementation of the SNAP benchmark running on NVIDIA V100 GPUs. Deploying the new kernel on Summit resulted in a simulation throughput of 175,100 atoms-steps/wall-clock second per node, exploiting all 6 GPUs and 42 CPU cores on a node. Extrapolated to the full machine, this predicts an increase of 90× in the EXAALT FOM over baseline on the ALCF/Mira system, putting EXAALT on track to meeting, and even exceeding KPP targets on the eventual exascale systems.

Future plans for GPU acceleration include Cuda kernel fusion and hierarchical threading strategies for improved performance on small systems (per GPU), use of mixed-precision data for improved performance on large systems, faster neighbor-list building algorithms, improved FFT methods which leverage vendor GPU libraries, and strategies to reduce data transfer overheads (between the CPU and GPU).

The PROGRESS/BML + MAGMA Libraries have enabled EXAALT/Latte scaling on Summit for metallic systems. The density matrix build for Silver nano-particles of increasing size was run on a single Summit node using diagonalization through MAGMA. Figure 53 shows performance for CPU only versus CPU + GPU (a Power9 CPU using 21 threads and 1 V100 GPU). BML was compiled with MAGMA to handle on node dense matrix operations. We see similar performance on CPU and CPU + GPU for small system sizes. Natrices of small sizes (<3,000) do more compute on the CPU for diagonalization in MAGMA. As the matrix size/system size increases the CPU + GPU is the clear winner with good speedup.

Currently, small system GPU utilization and performance is lacking. We will be exploring optimizations to improve small system performance for EXAALT/Latte by considering code refactoring, moving more work to the GPU, and exploring the use of Multi-Process Service (MPS).

The Cabana Particle Toolkit relies on Kokkos for its GPU interface. Performance for the classical MD mini-app, CabanaMD, is shown in Fig. 54 using the Leonard-Jones force potential. The impact of data layout and communication decisions on performance across different hardware is shown by routine. Another particle-in-cell mini-app, CabanaPIC (loosely based on VPIC), is GPU enabled, uses the existing Cabana communication plan and halo functionality.

WDMapp/XGC has adopted Cabana/Kokkos for handling data management and kernel execution for easy portability across architectures. This provides an easy, flexible framework for porting more kernels to GPU. Comparison and scaling across Summit is shown in the left plot of Fig. 55. A performance comparison

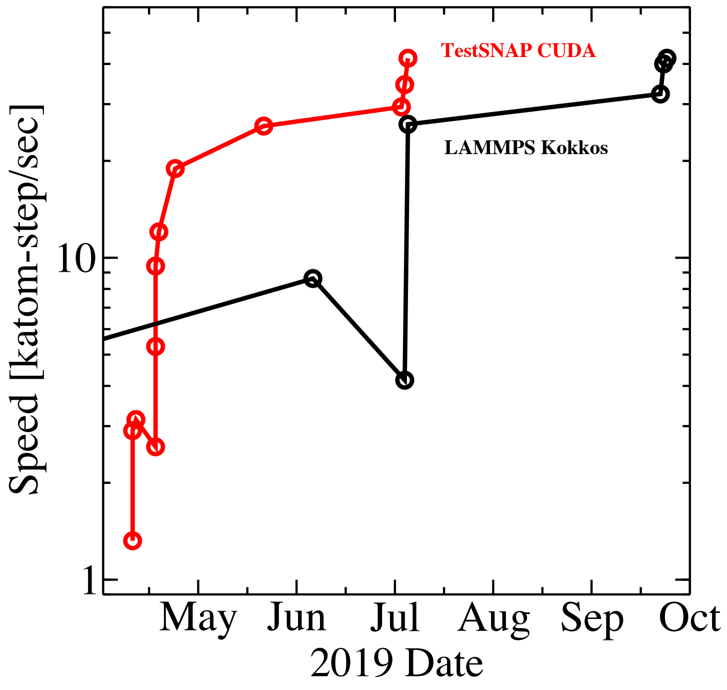
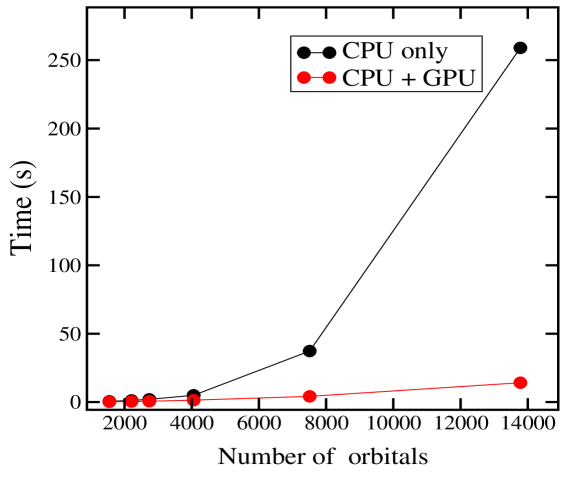
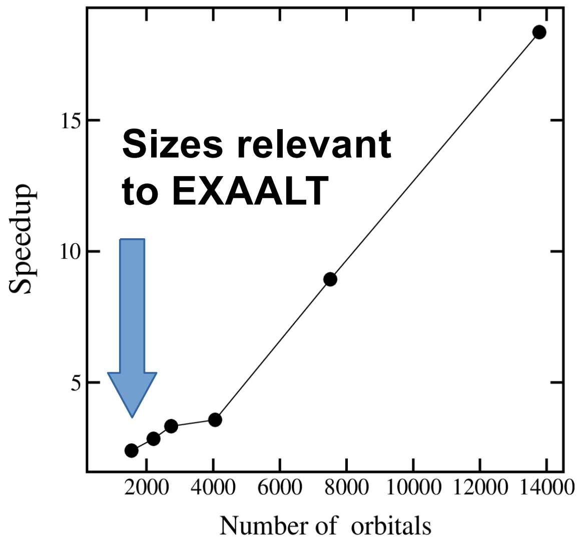


Figure 52: SNAP Performance improvement on a single Summit node May-October 2019.

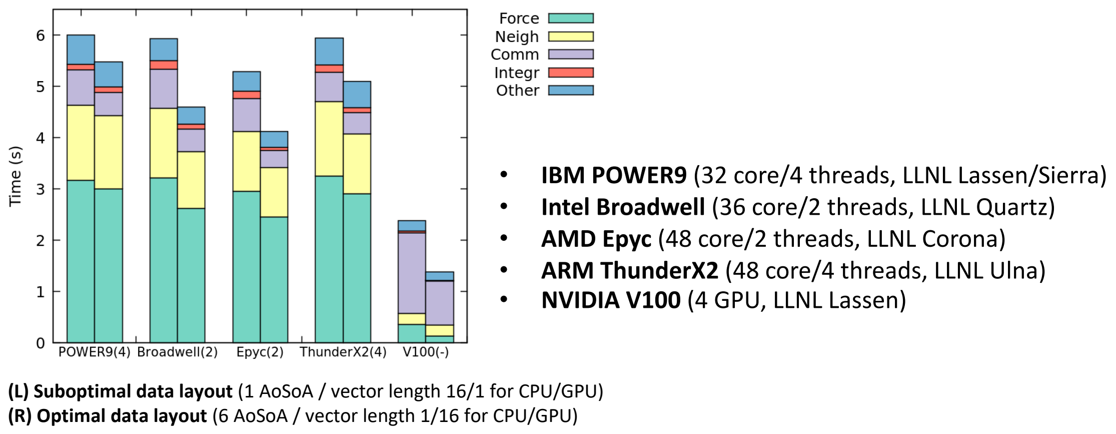


(a)



(b)

Figure 53: Scaling (a) and speedup (b) of PROGRESS/BML on a single Summit node.



- **IBM POWER9** (32 core/4 threads, LLNL Lassen/Sierra)
- **Intel Broadwell** (36 core/2 threads, LLNL Quartz)
- **AMD Epyc** (48 core/2 threads, LLNL Corona)
- **ARM ThunderX2** (48 core/4 threads, LLNL Ulna)
- **NVIDIA V100** (4 GPU, LLNL Lassen)

(L) Suboptimal data layout (1 AoSoA / vector length 16/1 for CPU/GPU)
(R) Optimal data layout (6 AoSoA / vector length 1/16 for CPU/GPU)

Figure 54: Compared performance of CabanaMD using different data layouts on different architectures.

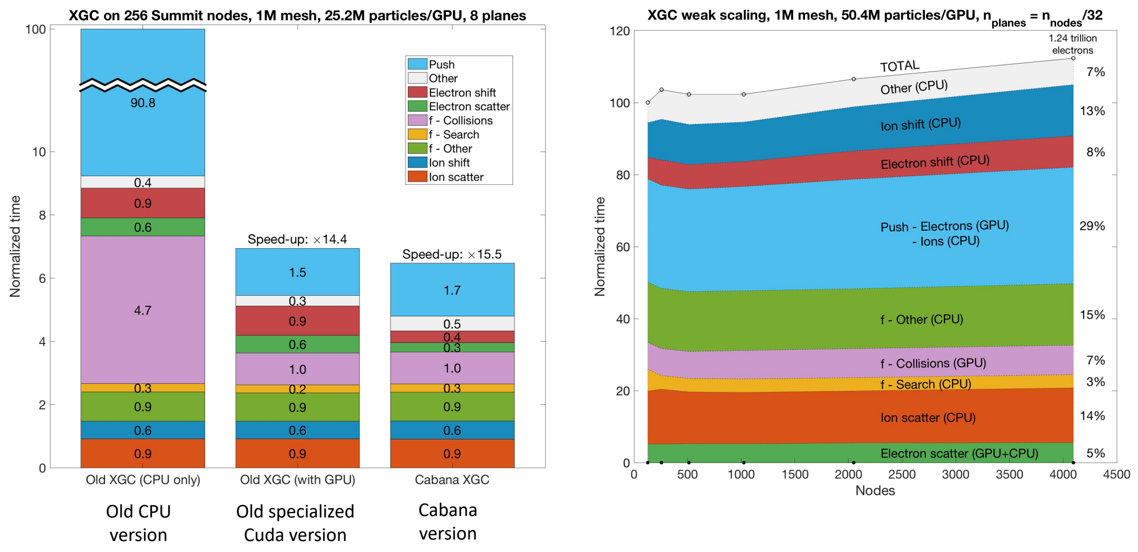


Figure 55: XGC-Cabana Summit performance comparison and scaling.

of the old XGC CPU version, the old specialized XGC Cuda version, and the new XGC Cabana version shows improved performance. Weak scaling is shown in the right plot of Fig. 55.

CopA/Cabana enabled XGC big science studies on full-scale Summit. The improved Cabana version performs better than the original GPU version XGC to full-size Summit for a production case tokamak plasma and geometry.

Next Steps

Development and running on Summit will initially prepare us for moving to early Aurora and Frontier. We will explore moving more computation to the GPUs. We will take advantage of the Aurora and Frontier GPU libraries/capabilities that will be made available on Summit using the NVIDIA GPUs.

A number of CoPA members attended Aurora and Frontier Workshops in-person and on-line. We will identify all the necessary mappings for our libraries and mini-apps. At this stage we will focus on converting our libraries and mini-apps so they can run on the current software stacks available. We may have to wait for Kokkos, MAGMA, and SLATE implementations. At this time compiling and correctness are important. Performance will come later. We will take the opportunity to explore the performance analysis tools in preparation for optimizing for exascale hardware.

The PROGRESS/BML libraries will continue their current strategy of using MAGMA for dense matrix operations, and *OpenMP offload* for sparse (ELLPACK, CSR) matrix operations on Aurora and Frontier. As needed, we will take advantage of the MKL (Aurora) and blas and sparse libraries (AMD) as available. Cabana will continue to employ its current implementation strategy of using the MPI + Kokkos programming model for performance portability. Two Kokkos back-end implementations are currently planned to be available for both Frontier and Aurora. Given the similar accelerator-based design of these two platforms as compared to Summit and Sierra we expect this programming model to continue to be successful.

8.3 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 a large number of applications relevant to DOE. Six 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), 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 utilize exascale architectures effectively.

AMR reduces the computational cost and memory footprint compared to a uniform mesh while preserving the local descriptions of different physical processes in complex multi-physics 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 subcycling 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 a 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 is 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 a number of 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.

8.3.1 *AMReX: Algorithms and Software Objectives*

The goal of the AMReX co-design 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 referred to as a level. Only the coarsest level is required to cover the domain, though finer levels can cover it as well.
- The complete mesh hierarchy on which field variables are defined is the union of all the levels. Proper nesting is enforced, i.e. 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 multi-physics 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 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 utilize 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 embedded boundaries (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 use of managed memory on hybrid CPU/GPU systems.

This basic framework includes native geometric multigrid solvers, with support for solving systems arising from embedded boundary discretizations, as well as interfaces to external solvers such as hypre and PETSc. The team provides an interface to SUNDIALS for integration of stiff ODEs. 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 (Structure-of-Arrays (SoA) versus Array-of-Structures (AoS)), particle communications and support for particle algorithms in an AMR context. For embedded boundary representations of complex geometry, AMReX provides support for the necessary 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 multi-level 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 in particular to support traditional subcycling-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. A 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. While 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 allows application developers sufficient access to the underlying data structures to allow them to implement and optimize new discretizations as well.

8.3.2 AMReX: Performance Objectives

The applications AMReX supports represent a wide range of multi-physics applications with different performance characteristics. Consequently, AMReX needs to provide a rich set of tools to allow 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++.

On-node parallelism

The basic AMReX paradigm is distribution of one or more patches of data to each node with an owner-computes rule to allocate tasks between nodes. For code executing on CPUs, AMReX supports logical tiling for cache re-use and OpenMP threading. Tile size can be adjusted at run time to improve cache performance; tile size can also vary between operations. AMReX includes both a standard synchronous strategy for scheduling tiles as well as an asynchronous scheduling methodology. The asynchronous scheduling uses runtime support from Perilla. AMReX also provides extensive support for kernel launching on GPU accelerators (using C++ lambda functions) and for the effective use of managed memory, that allows users to control where their data is stored. While much of the internal AMReX functionality currently uses CUDA for maximum performance on current machines, AMReX supports the use of CUDA, 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 co-exist 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. Particle tiling differs from grid tiling, however, in that the particle tiling determines the memory layout of the particles whereas tiling of mesh data does not change the layout

Fork-Join Parallelism

In many multi-physics applications, within each time step there are different physical processes that can be advanced independently. AMReX includes support for higher-level asynchronous task description and execution based on a fork-join approach. To enable fork-join task parallelism, a simple programming interface is provided for developers to express their coarse-grained tasks and the data those tasks require. The runtime mechanisms for parallel task forking, handling data migration, and task output redirection is also provided.

Linear solvers

A key feature of a number of applications typically solved using AMR is that they require 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 may be constrained by the synchronization points/barriers imposed by linear solvers within a time step.

The team has 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 (merging boxes in the AMR hierarchy to enable additional coarsening as part of the multigrid algorithm) and consolidation (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 provides MPI communication routines that operate directly on GPU memory buffers, so that unnecessary device-to-host and host-to-device copies are not triggered on machines, such as OLCF's Summit, that support GPUDirect. For particles, this requires operations such as sorting, searching, and stream compaction that work efficiently on the device. The current approach to this problem uses Thrust, a parallel algorithms library maintained by NVIDIA and distributed as part of the CUDA Toolkit. As the AMReX-based applications move more of their functionality onto GPUs, the team will work in conjunction with them to assess which components of the algorithm are most effectively offloaded to the GPU and which components stay on the CPU without negatively impacting overall performance. This raises the possibility of executing on the CPU and GPU in parallel and or executing different algorithm components on different GPUs. This will potentially require some redefinition of the underlying algorithms as well as software infrastructure to support the implementation. The team's fork-join approach provides a suitable framework that could be generalized to handle these types of algorithms.

Performance characterization

Predicting the performance of complex multi-physics AMR applications a priori is challenging at best. The team has developed a sophisticated set of profiling tools for performance characterization of AMReX-based applications. The toolkit includes measurement of both computation and communication. The software can be used to obtain anything from a broad overview of performance to detailed measurements localized to a specific portion of the algorithm. In addition to a brief summary report, the tools create a database during execution that can be queried using a graphical interface to extract detailed information. This overall performance measurement system is augmented with performance modeling methodology that enables the user to construct a model of execution including both compute and communication to understand performance behavior and how it depends on machine characteristics. This type of performance model enables the user to evaluate the potential impact of algorithm changes on performance prior to full implementation.

I/O; In situ analysis / visualization

AMReX provides native high-performance I/O, as well as a prototype for reading and writing in HDF5 format, for checkpoint/restart or further post-processing. The AMReX native format is supported by VisIt, ParaView and yt. AMReX is also working with ALPINE and SENSEI teams to integrate an interface for applications to move data to appropriate analysis and visualization packages in order to perform analysis and visualization in transit or in situ.

8.3.3 AMReX: Co-design Engagements and Integration Points

ECP Applications

Six ECP application projects in the areas of accelerator design (WarpX), astrophysics (ExaStar), combustion (Pele), cosmology (ExaSky), multiphase flow (MFIX-Exa) and additive manufacturing (ExaAM) include codes based on AMReX. All codes make use of the basic mesh data structures and iterators along with additional capabilities as discussed below.

- WarpX is a multilevel electromagnetic PIC code for simulation of 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 of these use AMReX. Particles can be used as tracer particles in Castro and FLASH5, and in a Monte Carlo 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, the linear solvers are used for pressure solves in a projection formulation and for the implicit treatment of viscous terms, and CVODE will be used to evolve the chemical reactions within the reactor.
- The compressible combustion code, PeleC, and the low Mach number combustion modeling code, PeleLM, are both based on AMReX. Both will use the EB methodology to represent the problem geometry, and possibly CVODE to evolve the chemical mechanism. 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 both as tracer particles and to represent sprays.
- One of the codes in the ExaAM project, TruchasPBF, is based on AMReX. TruchasPBF targets part-scale process and melt pool physics.

In addition, the AMReX co-design center has regular communication with the CoPA co-design center regarding best practices for particle data layout and operations. There has been no direct use of shared code, but discussions are on-going.

ECP Software Technologies

Table 56: AMReX KPP-3 goals and metrics

Passing value	Stretch value	Tentative present value
3	7	6

Table 57: 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 AMReX co-design 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. The AMReX co-design center had a shared fate milestone with the SUNDIALS project (2.3.3.05) for the SUNDIALS team to develop and release a vectorized version of CVODE, and for the AMReX team to provide an interface for its use in AMReX. This milestone has been successfully completed and the vectorized version is in use in the Nyx code. Plans are underway for development of a new version of CVODE that will work effectively on GPUs. In addition, the AMReX co-design center has regular interactions with:

- the HDF5 project (2.3.4.08). A prototype routine to write AMReX plotfiles in HDF5 format (as an alternative to the native I/O format) is being re-written for application ease of use; both ExaSky and MFiX-Exa plan to take advantage of this functionality.
- the ALPINE project (2.3.4.12) to determine the best ways for ALPINE and SENSEI to support the AMReX-based application projects. Preliminary interfaces to both exist in the AMReX git repo and the ExaSky and MFiX-Exa projects are currently exploring this functionality.
- Finally, the AMReX team worked with the xSDK team (2.3.3.01) to ensure xSDK compatibility and interoperability; AMReX was part of the 2018 xSDK release and will be part of the 2019 release as well.

ECP Vendor Interaction

The AMReX vendor liaison continues to interact with a number of vendors, reading PathForward milestones and participating in PathForward reviews. The liaison regularly provides feedback geared towards 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 in order to ensure that AMReX software development is in line with trends in architecture and system software.

8.3.4 AMReX: Progress Towards Advanced Architectures

GPU Strategy

The AMReX GPU strategy is to make it as easy as possible for AMReX-based applications to write code that is both performant and portable. To this end, it currently supports application code that uses CUDA, OpenACC or OpenMP. Internally, AMReX relies on CUDA for NVIDIA accelerators, and is in the process of developing core-level support for AMD accelerators based on HIP. A similar migration strategy will be implemented when appropriate for supporting Intel accelerators. AMReX is currently expecting to support Intel through DPC++, but is waiting for Intel to release test hardware and standards before making a final decision.

Originally the AMReX core framework included Fortran routines; almost all of these have been replaced by C++ versions, which makes adapting to future architectural changes more straightforward by not having to wait for Fortran compiler support and interoperability. The remaining Fortran routines will be completely replaced by mid-FY20. It is important to note that the Fortran interfaces will remain and AMReX-based applications will continue to be able to use Fortran without penalty.

AMReX provides the ability for users to construct loops over data using mesh and particle iterators. AMReX iterators implicitly cycle through GPU streams on each iteration, keeping work in a given loop ordered but taking advantage of unused GPU resources when they are available. AMReX iterators have an implicit device synchronize built into their destructors, as in most cases, the results of a loop will be used almost immediately. This can be turned off with a flag when desired, but provides a clean, readable and understandable workflow for most applications, especially when initially porting to GPUs.

Additionally, features of the overall GPU strategy include:

- Raw data (floating point, integer arrays and particles) is placed in managed memory. Application codes can opt out of the default and manage chosen objects separately by hand or with AMReX Memory Arenas.
- AMReX supports a C++ lambda-based launch system that offers inlining and portable GPU code.
- AMReX provides GPU-friendly implementations of common mesh, particle and particle-mesh operations, including operations such as `saxpy`, reductions, PIC methods and particle neighbor list constructions.
- AMReX provides functions for common parallel communication operations for mesh and particle data, including ghost cell exchange and particle redistribution. These operations run on the GPUs without triggering host/device copies and have been optimized for performance on Summit.

AMReX has no external software dependencies other than the standard software stack requirements such as working C++ compilers and MPI. However, AMReX does currently supply user-facing interfaces for the CVODE libraries supported by the SUNDIALS project, the external PETSc and hypre algebraic multigrid solvers and HDF5.

Progress to Date

The performance of AMReX on GPUs is best measured by the performance of AMReX-based applications. We report here some recent performance results for three of the AMReX-based codes and refer the reader to the subsections on those codes (see ExaStar § 5.1, MFiX-Exa § 4.4, and WarpX § 4.6 for further details on the results below).

WarpX, an AMReX-based electromagnetic PIC code for modeling particle accelerators, reports a 7× speed-up on its extrapolated KPP FOM on Summit relative to the Cori KNL nodes (this run used half of Summit). Additionally, on a uniform plasma benchmark, WarpX reports a $\sim 20\times$ speedup on Summit comparing 6 GPUs per node to 42 POWER9 cores (using 6 MPI and 7 OpenMP threads per node). These runs also exhibited nearly perfect weak scaling up to 2,048 nodes as shown in Fig. 56. In WarpX, the entire PIC loop runs on the GPUs and the data is only communicated back to the CPU for the purpose of I/O.

Castro, an AMReX-based code for compressible astrophysics that is part of the ExaStar project, reports a 10x speedup for the explicit hydrodynamics on Summit (6 GPUs/node) relative to multicore using MPI + OpenMP (6 MPI ranks × 7 OpenMP threads/rank). In addition it reports a 40× speedup of the time integration of the reaction network.

MFiX-Exa, an AMReX-based code for modeling gas/solid multiphase flow with the goal of simulating a full chemical looping reactor, reports a two-orders-of-magnitude speedup for the particle work on a Summit node with 6 GPUs relative to a Cori node with 36 cores.

In addition, a recent scaling study of the native geometric multigrid solvers in AMReX shows a roughly 4–5× speedup on a Summit node relative to a Cori Haswell node.

Next Steps

For Frontier-targeted code development, the AMReX team uses a local workstation with a Vega 10 AMD graphics card for active HIP-clang code development. Basic AMReX programs are successfully compiling and running with HIP-clang. HIP libraries, including hipRand and HIP's version of thrust are going to be tested next. This development is currently in a separate branch, but will be merged into AMReX proper once fully tested. Code differences, bugs and feature requests have been identified and are being sent to AMD and ORNL to further improve AMReX and HIP for Frontier.

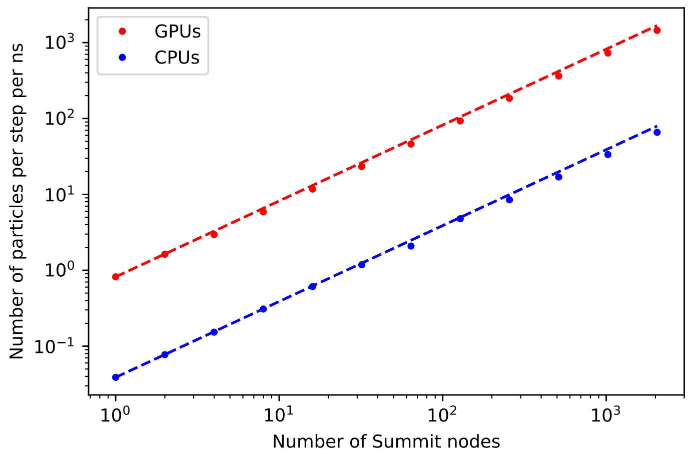


Figure 56: Results of a weak scaling study on a uniform plasma benchmark with WarpX, comparing GPU and CPU performance on Summit. Both cases used 8 particles per cell, Esirkepov current deposition, and 3rd-order interpolation. The GPU-accelerated runs used 6 GPUs per node, with 1 MPI task per GPU. The CPU-only runs used all 42 available POWER9 cores on a Summit node, using 6 MPI tasks and 7 OpenMP threads per node. Both runs exhibit excellent weak scaling. On 2,048 Summit nodes, the GPU-accelerated run was approximately 22 times faster that the CPU-only run.

AMReX has compiled and run with HIP-nvcc on Summit as an intermediate step towards HIP on AMD machines. AMReX will continue to use HIP-nvcc for a subset of bug and portability testing while HIP-clang is being developed.

The AMReX development team as well as numerous application teams attended the Frontier kickoff workshop in September. ExaStar, ExaSky and Pele were chosen as CAAR code teams to receive testing and expert assistance porting to AMD GPUs. That expertise will be utilized to port AMReX to HIP, as the underlying critical framework. CAAR and NESAP code teams are also provided access to an early test-bed system, which will be used for performance analysis with HIP.

AMReX established an AMD contact for HIP and ROCm questions at the workshop and has begun discussing bugs and feature requests needed for AMReX and its applications.

For Aurora preparation, the AMReX team anticipate DPC++ to be the preferred API to run on Intel systems and are awaiting formal releases of test platforms and standards before starting code development. The AMReX development team has attended available tutorials and seminars on DPC++, SYCL and other relevant Intel topics to keep up-to-date on the expected requirements for Aurora.

The Pele application suite was chosen to participate in Argonne's Early Science Program. The provided expertise and hands-on Intel support will be used to help port AMReX itself to Aurora, much like the NESAP and CAAR programs have helped at NERSC and OLCF, respectively.

8.4 CEED

Efficient exploitation of exascale architectures requires a rethink of the numerical algorithms used in large-scale applications of strategic interest to the DOE. These architectures favor algorithms that expose ultra-fine-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, where practical efficiency is measured by the accuracy achieved per unit computational time. One of the few viable approaches to achieve high-performance in this case is to use matrix-free high-order finite element methods, since these methods can both 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 (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 to traditional low-order approaches.

The CEED co-design center is a focused team effort to develop the next-generation discretization software and algorithms that will enable a wide range of finite element applications to run efficiently on future hardware. High-order methods are the logical choice for this, from a mathematical (higher-quality simulations) perspective, as well as from HPC (better performance) and risk mitigation perspectives (range of orders provides flexibility in the uncertain exascale hardware and software environments). Their efficiency extends to problems with unstructured non-conforming 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 that will 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 miniapps that combine applications-relevant physics with key high-order kernels that capitalize on matrix-free forms for efficient performance. These miniapps 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.

Up-to-date information about the project is available on the website, `http://ceed.exascaleproject.org`, including recent publications and news items. The software catalog, including the libCEED low-level API library, the CEED benchmarks, the Laghos and Nekbone miniapps, the libParanumal set of GPU kernels, and the high-order Field and Mesh Specification, FMS, as well as mirrors of the major CEED packages are freely available on GitHub at `https://github.com/CEED`.

8.4.1 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 integra-differential equations. It has been exhaustively studied, both theoretically and in practice, in the past 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 physics fields and potentially the geometry of mesh elements.

While they live on unstructured grids and result in globally sparse matrices, locally on each element high-order finite elements have a dense Cartesian structure, thus offering a bridge between the unstructured/structured and sparse/dense worlds and 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, while 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 having 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 anticipated 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.

The majority 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 only $O(p^2)$ references for the operators, while the number of flops scales as $O(np) = O(Ep^4)$. (Here, n is the total number of gridpoints, E is the number of elements, and p is the approximation order.) Further operations, involving 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.

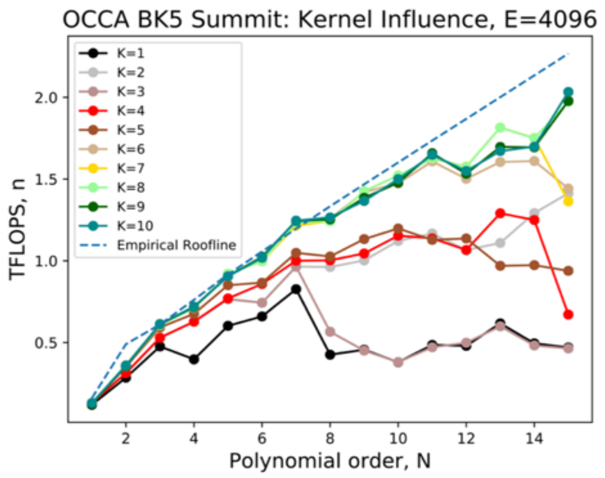
CEED is focused on developing optimal implementations of finite element operator evaluators and solvers across a variety of applications. One of the ways that these developments will be manifest is through a lightweight library, libCEED, and related mini-apps and examples. libCEED is designed to allow 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 frontend apps and backend kernels which supports efficient operator description (not a global matrix). This description is based on a purely algebraic finite element operator decomposition and thus applicable to many applications. The frontend application can use any compatible backend, and any (performant) high-order code should be able to plug in at that front end (now or in the future). The backends can be added independent of frontends and are divorced from finite element information, allowing computer scientists to optimize the evaluation without domain knowledge. These backends can be selected for best performance, optimized for order, device (CPU, GPU, etc.) and can use just-in-time compilation to deliver performance. Each new backend is automatically usable in all compatible frontend applications, giving a broad impact of the optimization efforts.

8.4.2 CEED: Performance Objectives

Petascale finite element codes currently execute with up to a million CPU cores 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 to have enough work per core to offset communication overhead and to have linear system solvers that are scalable both 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 having bounded iteration counts at these scales, where even the coarse-grid problem, which has $O(1)$ Degrees of Freedom (DOF) per core, consists of millions of DOF. For high-order methods, there are two prevailing strategies for preconditioning: 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 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 AMG while 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 will be a major 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 will be to realize high performance per node, where 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



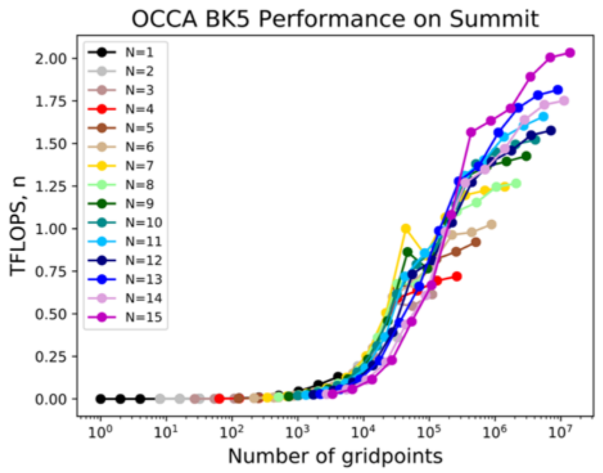


Figure 57: Significant variance in the bake-off results.

operators on the reference element—which are readily expressed in the form of BLAS3 `dgemms`.

For CPUs, optimizing the dgemms for small matrices of order $p = 6$–16 (typically) is the standard starting point for performance tuning. However, to further push development and to 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 degrees for freedom and quadrature points. These benchmarks have been used to compare Nek, 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 assessing overall performance of a given code-problem coupling on a particular platform.

For GPUs, the CEED development is taking advantage in particular 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 will perform as well as hand-tuned CUDA. MAGMA is directed towards fast tensor contractions and batched dgemms that are appropriate for the size of matrices encountered in high-order finite element codes. Currently, all the miniapps have GPU versions and the team is in the process of bringing these technologies to first wave applications.

High-order alone is not sufficient 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 has 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 TFLOP on a V100 GPU on Summit as shown in Fig. 57. 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, realization of fast implementations on GPUs and other accelerators requires a deeper understanding of the architecture than most application scientists would desire. An objective of CEED is to make these fast implementations accessible through a convenient interface, libCEED, which will support 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 will allow these implementations to be leveraged on future 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 $p = 1, \dots, 16$ 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.

8.4.3 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 miniapps and software technologies.

Application engagement comes in two primary forms. Initially, CEED will 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 will include all developments directed at Nek5000 and MFEM, including extensions to GPUs and Aurora. Secondly, CEED will reach 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 will 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 will be reaching out to these applications to assess the potential of libCEED-enabled performance gains.

Benchmarks: Bake-Off Problems (BPs) and Bake-Off Kernels (BKs)

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

Table 58: CEED KPP-3 goals and metrics

Passing value	Stretch value	Tentative present value
3	6	2

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 matvec. Thus, high-order methods require a new "format" that still represents a linear (or more generally non-linear) 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 (CPUs, GPUs, etc.). 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 FMS (Field and Mesh Specification) 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 testbed for multi-level 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 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.

8.4.4 *CEED: Progress Towards Advanced Architectures*

GPU Strategy

Table 59: CEED code base

Package name	LOC	Target exascale challenge problems	Computational motifs
Nek5000	100k Fortran/C	ExaSMR, ExaWind, E3SM	High-order methods, matrix-free computation, scalable solvers
MFEM	164k C++	MARBL, ExaAM, GEOS, SNLApp	High-order finite elements, unstructured AMR, computation, matrix-free computation, scalable solvers
libCEED	22k C	CEED, SNLApp, E3SM, GEOS	Low-level API library for efficient high-order operator evaluation
libParanumal	112k C++/OKL	CEED	Prototype GPU accelerated algorithms for high-order finite element methods
MAGMA	740k C/C++	CEED	Numerical linear algebra batched linear algebra, tensor contractions, dense and sparse matrix computations
OCCA	44k C++	CEED	Portable many-core programming platform
PUMI	82k C++	CEED	Parallel, unstructured, mesh infrastructure

The CEED team has been developing GPU-performant algorithms in all of its software components including: the *low-level* OCCA, MAGMA and libCEED libraries; the Nekbone, Laghos and libParanumal *miniapps*; and the *high-level* MFEM and Nek codes.

At the low-level, OCCA is providing performance portable way of describing high-order finite element kernels that can target all of the Summit, Aurora and Frontier architectures. An important feature of OCCA is its support for JIT compilation, which allows for example kernel fusion with user-provided input, which is critical for achieving top level GPU performance. OCCA is being used in several CEED products, including libCEED, libParanumal, MFEM and Nek. Another low-level GPU library is MAGMA, which focuses on batched dense contraction applied to high-order finite element kernels and is currently integrated in libCEED. In this approach the dense contractions are expressed through Batched BLAS APIs, e.g., batched matrix-matrix multiplications (gemms). The Batched BLAS APIs extend the BLAS standard and are co-designed with vendors, targeting libCEED performance portability across architectures through the use of BLAS standards. The best performance in libCEED is currently obtained for the 'cuda-gen' backend, which is implemented using NVRTC.

At the high-level, the Nek team is developing NekRS, a C++ version of Nek5000 utilizing OCCA and the libParanumal elliptic solver. With version 4.0, MFEM introduced general support for hardware accelerators such as GPUs, and for low-level libraries/programming models such as CUDA, OCCA, libCEED, RAJA, and OpenMP.

Progress to Date

MFEM-4.0 introduced GPU support in the library based on new backends and kernels working seamlessly with a new lightweight memory spaces manager as illustrated in Fig. 58. Several of the MFEM example codes (ex1, ex1p, ex6, and ex6p) and the Laghos miniapp can now take advantage of this GPU acceleration. One main feature of the MFEM performance portability approach is the ability to select the backends at runtime: they can be mixed to take full advantage of heterogeneous architectures. Algorithms well suited for multi-core CPUs can still be used, while kernels more suited for GPUs can be launched on these architectures. Most of the kernels are based on a single source, while still offering good performance and being able to use efficiently hardware. This follows MFEM's design philosophy and is made possible by integrating the support

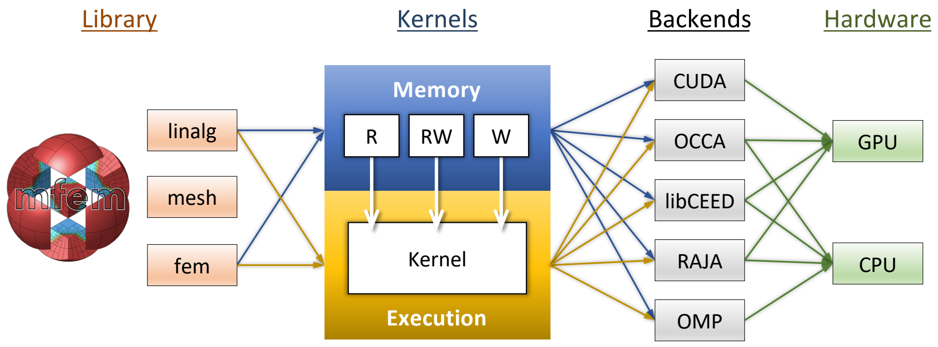


Figure 58: Diagram of MFEM's modular design for accelerator support, combining flexible memory management with runtime-selectable backends for executing key finite element and linear algebra kernels.

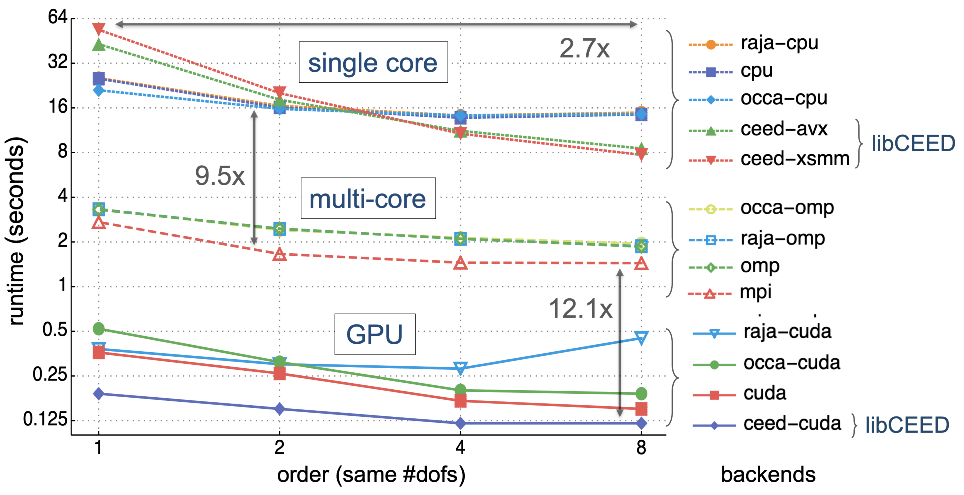


Figure 59: Initial results with MFEM-4.0: Example 1, 200 CG-PA iterations, 2D, 1.3M DOF, GV100, sm_70, CUDA 10.1, Intel(R) Xeon(R) Gold 6130@2.1 GHz.

directly at the library's finite element components level. Many linear algebra and finite element operations can now benefit fully from the new GPU acceleration.

Figure 59 presents initial MFEM performance results measured on a Linux desktop with a Quadro GV100 GPU, sm_70, CUDA 10.1, and Intel(R) Xeon(R) Gold 6130 CPU @ 2.10 GHz. Note that this configuration is very similar to a compute nodes of LLNL's Sierra machine. Single-core, multi-core CPU, and single-GPU performance for different discretization orders is shown, keeping the total number of DOF constant at 1.3 millions in 2D. Results from all backends supported in MFEM 4.0, as well as recent results based on the libCEED library (integrated with MFEM) are included. The plot clearly shows that GPU acceleration offers a significant gain in performance relative to multi-core CPU. Comparing the single-core CPU backends, we see that libCEED (ceed-avx and ceed-libxsmm) brings better performance for orders above two. For GPUs, libCEED (eed-cuda) presents an improvement over all other GPU backends.

NekRS is a new C++ variant of Nek5000 based on OCCA and libParanumal coming out of Warburton's group at Virginia Tech. libParanumal is a test platform for exploring advanced algorithms for PDEs. NekRS development started in January, 2019.

Figures 60 shows performance results for the current version of NekRS performed on Summit for a 17×17 rod-bundle flow simulation. The mesh uses 277,000 elements of order $N = 7$ ($n = 95$M gridpoints total). The Reynolds number is 5,000 based on hydraulic diameter. Periodic boundary conditions are used in the axial flow direction and the initial conditions comprise an array of meandering vortices.

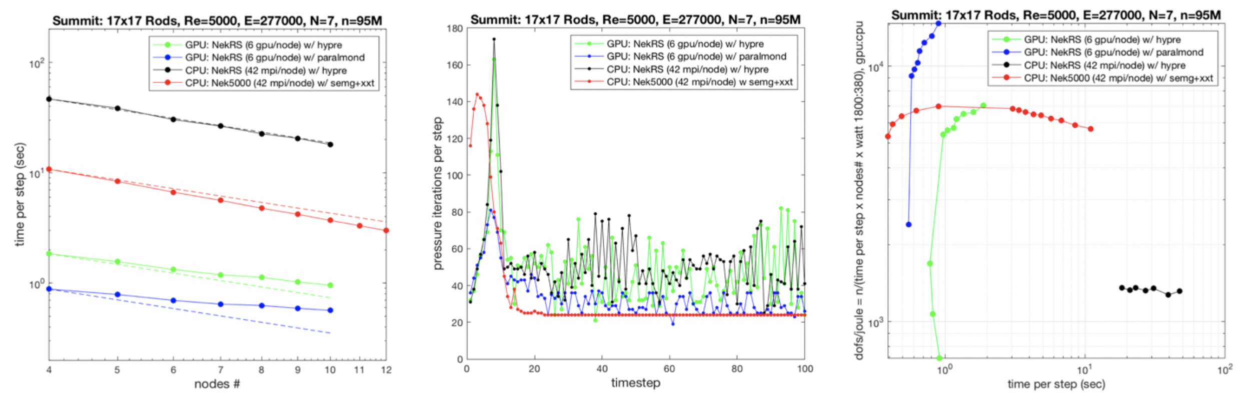


Figure 60: NekRS and Nek5000 performance of GPUs versus CPUs on Summit for turbulent flow simulations with $Re = 5{,}000$ for a 17×17 rod-bundle geometry using total number of grid points $n = 95{,}011{,}000$. Based on timings from Step 11 to 60, time-per-step with ideal scalings shown as dashed lines (left), pressure iterations per step (center), and DOF-per-J with respect to time-per-step (right) are shown.

Figure 60, left, shows strong scaling results on a few nodes of Summit using NekRS with six V100 GPUs per node or NekRS/Nek5000 with 42 CPUs per node. For the CPU version, NekRS uses Hypre as a coarse grid solver. In this case, NekRS is about 4× slower than Nek5000 because the pressure solver is not yet as optimized as the highly-tuned solver in Nek5000. For the GPU, the NekRS results improve substantially when the coarse grid solver is based on the AMG solver ParAlmond.

Figure 60, center, shows the pressure iteration counts for each of the four cases. Nek5000 uses Schwarz-smoothed p-multigrid; NekRS uses Chebyshev smoothing. When ParAlmond is used for the coarse-grid solve the NekRS iteration counts improve by a factor of two and are on par with those of Nek5000. The Chebyshev smoother requires more work per iteration than the Schwarz-based smoother.

With ongoing effort on the pressure solve we anticipate a 2× reduction in NekRS solution times, which will put it on par with the strong-scaled solution times of Nek5000 with more than 2× energy savings that are already observed for NekRS on Summit's V100s (Fig. 60, right).

Next Steps

The CEED team is actively engaged with Intel in preparation for the Intel-based Aurora architecture. Activities in this area include: performance projection for Nekbone on the new Aurora architecture, estimated to achieve over 50× FOM speedup over Sequoia's baseline; attending the Aurora workshop at ANL (9/17-9/19, 2019) by 5 CEED members; initial porting of NekRS on an Aurora development system, running with OpenCL (via OCCA) on a single Intel Gen9 GPU; and further kernels optimization of OCCA, libCEED, libParanumal, and Nek5000 planned and/or in development.

CEED researchers are also actively engaged with porting to and evaluation of AMD GPUs in preparation for the AMD-based Frontier machine. Activities in this area include: support for HIP in OCCA and MFEM-4.0; initial MFEM performance runs on the LLNL Corona cluster, which has Radeon Instinct MI25 GPUs (weak double precision); initial results with libParanumal on Radeon VII (good double precision) which on certain benchmarks have achieved 75 % of the NVIDIA V100 peak performance at 10 % of the cost; and collaboration with AMD on fixing slow linking times with the HIP compiler which was reported to hamper development at the CEED annual meeting, and was addressed by AMD engineers in a couple of weeks.

Table 60 demonstrates NekRS baseline of performance measured on a single GPU on V100 and Intel Gen9. Simulations are performed for turbulent flows with a Reynolds number of 8,000 using triangle-shaped pipe geometry with 9,234 elements of order $N = 7$ ($n = 3$M gridpoints total). Wall boundary in spanwise and periodic boundary in stream directions are considered with turbulent initial condition. Timings are measured from 100 timestep runs.

Table 60: NekRS baseline of performance measured on a single GPU, Intel Gen9 (Aurora development system) vs. NVIDIA V100 using API backends of CUDA and OpenCL. Simulations are performed for turbulent flows with $Re = 8{,}000$ for a triangle-shaped pipe geometry using total number of grid points $n = 3{,}167{,}262$.

systems	API backend	accumulated time 100 steps (s)	time per step (s)	ratio (Gen0/V100)
Intel Gen9 (Iris@JLSE/ANL)	OpenCL	1.784 98e3	17.84	1.00
NVIDIA V100 (Nurburg@ANL)	OpenCL	3.885 53e1	0.388	45.93
NVIDIA V100 (Nurburg@ANL)	CUDA	3.755 09e1	0.375	47.53
NVIDIA V100 (Summit@OLCF)	CUDA	3.836 53e1	0.386	46.53

8.5 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 and GMT.

8.5.1 ExaGraph: Algorithms and Software Objectives

Through engagements with AD projects, work in the first year was 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), ExaWind (§ 4.1), ExaSGD (§ 6.1) and SuperLU/STRUMPACK. Key contributions from the first year's work were in the development of perfect matching with heuristics for maximizing the sum of the weights of the matched edges (highlighted in an ECP news article dated 08-27-2018). Contributions were also made to the efforts of ExaBiome through scalable implementations of Markov clustering. However, in order to increase further engagement with ECP Application Projects, the team reached out to all the projects with a brief questionnaire and received good feedback. Based on the results of this survey, the team aims to include at least two new applications to closely integrate with and meet their needs by developing specific algorithmic kernels that can be seamlessly integrated with the applications, and to find wider usage for the tools that have already been developed in the project.

The team has started to explore porting and optimization of key graph algorithms (weighted approximate matching and graph clustering) targeting important accelerator architectures. While design efforts related to the hardware design are fairly committed at this point of time, there is ample scope to improve the support of runtime and middleware systems in order to enable efficient execution of combinatorial and graph algorithm on the exascale systems. ExaGraph intends to develop at least two prototype implementations for benchmarking efforts through the Proxy Applications Project and publish technical articles on the different aspects of performance such as impact on energy consumption, communication models and memory footprints.

As far as software engagements are concerned, the team has 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. The team plans to:

Table 61: ExaGraph KPP-3 goals and metrics

Passing value	Stretch value	Tentative present value
2	4	2

- develop and deliver a nested-dissection based sparse matrix ordering capability that is not just open-source but has a permissive BSD-style license;
- develop a new parallel symbolic sparse matrix factorization algorithm using GraphBLAS primitives that can also run on GPU accelerated clusters; and
- develop an implementation of half-approximate matching using UPC++ to provide insight for the UPC++ team on addressing the needs of irregular applications.

8.5.2 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 multi-GPU implementations for graph clustering.

8.5.3 ExaGraph: Co-design Engagements and Integration Points

- SuperLU/STRUMPACK: In addition to existing accomplishments on developing a heavy-weight perfect matching (HWPM) algorithm in collaboration, 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.
- ExaBiome: Through a close working relationship with ExaBiome (§ 6.3), the ExaGraph team has already 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). Work is underway to deliver a distributed algorithm for generating protein similarity graphs, which will be the input for HipMCL. Future plans also include a joint milestone on end-to-end GPU acceleration of this protein clustering pipeline.
- 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.
- 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.
- Software Technologies: ExaGraph is working on developing a PGAS based implementation of half-approximate matching using UPC++.
- Benchmarking/Vendors: ExaGraph is working with the AMD's Path Forward team to port and optimize the proxy application on graph clustering targeting accelerator (GPU) platforms.

Table 62: ExaGraph code base

Package name	LOC	Target exascale challenge problems	Computational motifs
CombBLAS	∼50,000	Protein similarity construction, protein clustering, factorization-based sparse solvers	All graph algorithms that can be implemented as matrix algebra
Vite	∼7,000	Model reduction in power grid simulations; computational biology	Graph clustering
Matchbox	∼34,000	Task assignment, algebraic multigrids, sparse direct solvers	Graph matching
Zoltan2		Mesh and matrix partitioning	Graph partitioning
KokkosKernels		FEM assembly, AMG setup	Graph coloring

8.5.4 ExaGraph: Progress Towards Advanced Architectures

GPU Strategy

The following strategies are being adopted to exploit heterogeneous (GPU) resources to accelerate graph algorithms:

- CUDA based development for NVIDIA GPUs: We will develop code using MPI + OpenMP + CUDA for systems equipped with NVIDIA GPUs;
- OpenMP Offloading: We will develop code using MPI + OpenMP offloading for systems equipped with Intel and AMD GPUs.
- Kokkos: Kokkos Kernls and Trilinos implementations of graph algorithms use productivity tools such as Kokkos to enable portability across different systems equipped with GPUs from different vendors.
- Relying on problem-specific high-level libraries that run on GPU, including GraphBLAST and CUDASW++
- SYCL: As a stretch goal, we will consider developing simpler algorithms in SYCL targeting multiple systems equipped with Intel, AMD and NVIDIA GPUs.

The ST and external software dependencies are:

- Critical dependecies: MPI, OpenMP, OpenMP offloading, CUDA.
- SYCL, Kokkos, GraphBLAST, Gunrock, Ligra.

Progress to Date

The following graph algorithms are currently under development:

- Graph partitioning based on spectral partitioning. We are developing the first multi-GPU graph partitioner. Our algorithm is a variation of spectral partitioning and we use the Trilinos and Kokkos libraries to write performance portable code across CPU, GPU, and future exascale architectures. We have some results on Summit shown in Tables 63 and 64. Our use case is the application has a certain number of MPI ranks, and each rank corresponds to either a CPU core or a GPU. We show our spectral method is faster on GPU than CPU. It is about 12× faster (on GPU) than ParMetis (which only runs on CPU).
- Graph clustering: Our algorithms based on modularity optimization are currently under active development. We are currently fixing minor bugs on the Volta GPUs.

Table 63: Spectral partitioning runtime with MPI-only and MPI + CUDA on 24 processes per GPU.

graph	MPI-only (s)	MPI + CUDA (s)	speedup
web-NotreDame	0.53	0.64	0.83
language	1.29	0.92	1.40
coPapersCiteseer	1.20	0.72	1.67
rajat30	1.44	2.39	0.60
Stanford_Berkeley	1.03	0.97	1.06
ASIC_680ks	1.25	0.67	1.87
ASIC_680k	0.68	1.55	0.44
eu-2005	1.78	0.97	1.84
hollywood-2009	8.97	1.77	5.07
FullChip	9.92	12.38	0.80
com-Orkut	40.83	4.17	9.79
wikipedia-20070206	48.41	5.18	9.35
cit-Patents	23.61	3.79	6.23
com-LiveJournal	20.60	3.02	6.82
circuit5M	15.89	10.27	1.55
wb-edu	17.60	1.56	11.28
uk-2005	133.62	12.62	10.59
it-2004	149.36	12.47	11.98
twitter7	1 087.02	61.16	17.77
com-Friendster	1 408.27	84.93	16.58
geomean—all graphs			3.27
geomean—all graphs with >1M vertices			6.83

- Influence maximization: Our algorithms for influence maximization (submodular optimization) currently scale on Summit. Our initial results demonstrate strong scaling with up to 64 nodes using all the Power9 CPU cores and all the six GPUs on the node. A summary of tabulated results and strong scaling results are shown in Table 65 and Fig. 61.
- PISA: Parallel protein sequence aligner code currently spends 80–90 % of its execution in pairwise sequence alignment. In collaboration with Exagraph, we have been developing various GPU optimized versions of pairwise alignment (Striped Smith Waterman, X-drop, etc.). We will integrate them to PISA during the next two years. This is already a major milestone for Exagraph.
- Sparse Solvers: We are currently working on a GraphBLAS compatible symbolic factorization step, in collaboration with the factorization-based solvers project (SuperLU/STRUMPACK).

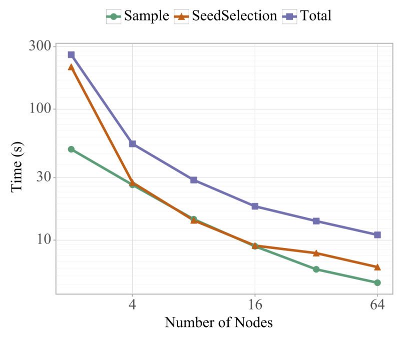
Next Steps

The current next step plans in ExaGraph are:

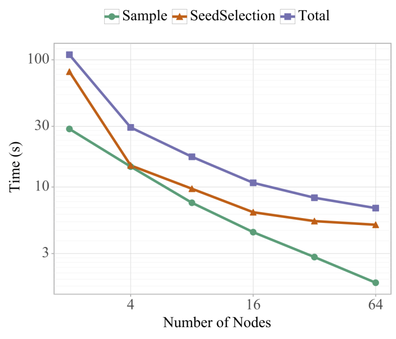
- To enable the GraphBLAS-based implementations to use GPU implementations. Currently the closest tool that suits this purpose is GraphBLAST (`https://github.com/gunrock/graphblast`).
- Work with AMD on porting the CUDA-based code to use AMD GPUs.
- Use OpenMP offloading to enable execution on GPUs for the graph clustering problem, in collaboration with the SOLLVE project.

Table 64: Runtime (s) comparison of spectral partitioning (SpecPart) against ParMetis (24 processors/GPUs).

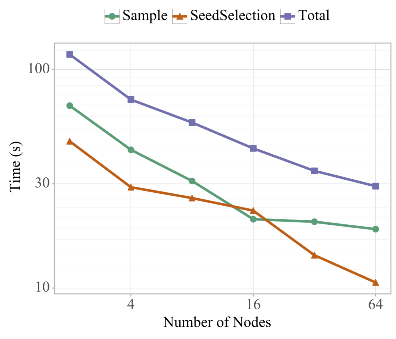
graph	ParMetis	SpecPart	speedup
web-NotreDame	0.46	0.64	0.72
language	0.65	0.92	0.71
coPapersCiteseer	0.60	0.72	0.83
rajat30	274.31	2.39	114.77
Stanford_Berkeley	7.01	0.97	7.23
ASIC_680ks	0.44	0.67	0.66
ASIC_680k	84.91	1.55	54.78
eu-2005	36.21	0.97	37.33
hollywood-2009	37.84	1.77	21.38
FullChip	6 951.30	12.38	561.49
com-Orkut	116.54	4.17	27.95
wikipedia-20070206	170.00	5.18	32.82
cit-Patents	8.65	3.79	2.28
com-LiveJournal	28.91	3.02	9.57
circuit5M	5 967.20	10.27	581.03
wb-edu	10.54	1.56	6.76
uk-2005	>7 200	12.62	-
it-2004	>7 200	12.47	-
twitter7	>7 200	61.16	-
com-Friendster	>7 200	84.93	-
geomean			12.68



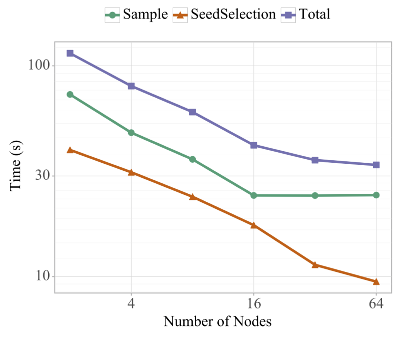
(a) web-Google



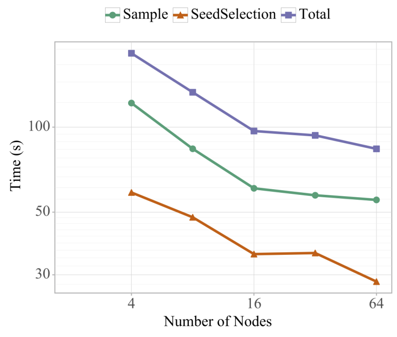
(b) web-BerkStan



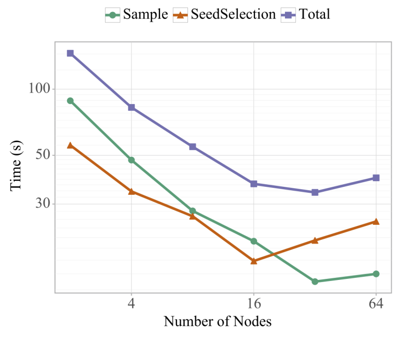
(c) wiki-topcats



(d) soc-pokec-relationships



(e) soc-LiveJournal1



(f) com-orkut.ungraph

Figure 61: Summit IC model, parameters: $\epsilon = 0.13$, $k = 100$.

Table 65: Comparative evaluation of CuRipples relative to previous implementations of IMM—both serial (IMM_{seq}) and parallel ($\text{IMM}_{\text{opt/mt/edison}}$). Abbreviations used: number of cores (C), GPUs (G), and nodes (N).

System	Time (s)	Speedup	Scale
com-Orkut ($\epsilon = 0.5$, $k = 100$)			
IMM_{seq}	28 024.56	1.00	1C
IMM_{opt}	9 027.50	3.10	1C
IMM_{mt}	1 319.21	21.24	20C (1N)
$\text{CuRipples}_{\text{dgx-1v}}$	35.47	790.09	80C+8G (1N)
$\text{CuRipples}_{\text{newell}}$	43.72	641.00	128C+4G (1N)
com-Orkut ($\epsilon = 0.13$, $k = 200$)			
$\text{IMM}_{\text{edison}}$	294.51	95.16	3 072C (64N)
$\text{IMM}_{\text{edison}}$	47.77	586.61	49 152C (1 024N)
$\text{CuRipples}_{\text{summit}}$	36.30	772.03	2 688C+384G (64N)
soc-LiveJournal1 ($\epsilon = 0.5$, $k = 100$)			
IMM_{seq}	16 434.81	1.00	1C
IMM_{opt}	3 954.59	4.16	1C
IMM_{mt}	1 026.21	16.02	20C (1N)
$\text{CuRipples}_{\text{dgx-1v}}$	70.23	234.01	80C+8G (1N)
$\text{CuRipples}_{\text{newell}}$	65.26	251.84	128C+4G (1N)
soc-LiveJournal1 ($\epsilon = 0.13$, $k = 200$)			
$\text{IMM}_{\text{edison}}$	190.94	86.07	3 072C (64N)
$\text{IMM}_{\text{edison}}$	55.12	298.16	49 152C (1 024N)
$\text{CuRipples}_{\text{summit}}$	106.43	154.42	2 688C+384G (64N)

8.6 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 will provide 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 will undertake a focused co-design process that targets learning methods common across these applications. This includes deep neural networks of various types (RNNs, CNNs, GANs, etc.), kernel and tensor methods, decision trees, ensemble methods, graphical models and reinforcement learning methods. ExaLearn will engage 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, given constraints relating to application development costs, application fidelity, performance portability, scalability, and power efficiency.

ExaLearn will identify the fundamental ML challenges associated with ECP and concentrate 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 is 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. Building on extensive collaborations with application teams over the past years, the team will attach dedicated application "catalysts" to specific ECP application teams to ensure continuous, high-bandwidth information and technology exchange between ExaLearn and applications as well as with experimental user facilities. In addition, a continuous outreach program will engage the wider data science and ECP software library communities. Finally, the ExaLearn co-design center will serve as a focal point for exascale learning technology interactions with vendors.

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 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.

8.6.1 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 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 deliver a series of software releases that combine multiple open-source packages that span the ML/DL 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

Table 66: ExaLearn KPP-3 goals and metrics

Passing value	Stretch value	Tentative present value
2	3	1

address particular scalability impediments. Examples of 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.

8.6.2 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 applications of interest across the DOE. The performance objectives will be tailored to each application within the four application pillars: surrogates, control, inverse, and design. Within each of these applications areas the goal will be 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 will be 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 day, 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.

8.6.3 ExaLearn: Co-design Engagements and Integration Points

Initial engagements with AD are primarily with three projects: ExaSky (§ 5.2), ExaAM (§ 3.5), and CANDLE (§ 6.2). Surrogate model development for cosmology is the focus with ExaSky, and the development of reinforcement models with ExaAM. With ST, the two primary engagements are with the SOLLVE project (to advance OpenMP integration) and the Legion project (to explore a potential threading model for large scale deep learning software).

ExaLearn also has crosscutting teams responsible for interactions with PathForward hardware vendors and ST projects. The PathForward cross-cutting effort facilitates communications and collaborations between ECP and PathForward projects in the ML domain. This will be accomplished in three primary ways:

- Direct vendor engagement. Discussing the ExaLearn needs with the vendors, attending the vendor "deep-dive meetings," and understanding the new vendor technologies. Two caveats: this requires significant travel and need to be under the vendor NDAs (the latter may be easier for some labs than others).
- ECP application and experimental facility engagement. Work with other ExaLearn application and cross-cutting areas to develop proxy apps to accurately reflect the system utilization (computation, communication, etc.) of the key ML kernels and how they are applied to applications. The ability of proxy applications to improve the design of scalable software solutions for deep learning methods is high priority.
- ECP ML proxy applications. Work with ECP proxy application team to define ExaLearn proxy applications and problem sets as part of the ECP proxy application suite. This includes working across ExaLearn to ensure the PathForward vendors have ML/DL proxy applications and data sets relevant to ECP's ML/DL needs.

In a similar manner, the ST effort relies on the other ExaLearn crosscutting teams to reach out to the larger ECP ST Community. Since the ExaLearn Software Infrastructure and Performance/Scaling crosscut teams also benefit from understanding and engaging with the PathForward vendors, this crosscut supports integration and communication among these teams.

8.6.4 ExaLearn: Progress Towards Advanced Architectures

GPU Strategy

ExaLearn's strategy is to provide exascale ML software for use by ECP Applications Projects, other ECP Co-design Centers, DOE Experimental Facilities and DOE Leadership Class Computing Facilities. Scalable deep learning software toolkits are being built upon the LBANN software stack, which includes: LBANN a deep learning toolkit, the Hydrogen GPU-accelerated distributed linear algebra library, and Aluminum a GPU-centric accelerated communication library. This framework heavily leverages custom CUDA kernels, cuDNN, and cuBLAS for node-local computation. Using this software stack, all of the distributed deep learning runs directly on the GPUs with GPU-centric communication operations, leaving the host CPUs for data marshalling, staging, and preprocessing.

To enable scalable training of regressive and generative models, we developed new methods of extending generalizable parallel distributed convolutions to work on 3D data cubes. This work was performed in the Livermore Big Artificial Neural Network (LBANN) toolkit and built on prior work that implemented spatially distributed two-dimensional convolutions. Additional work has gone into creating the parallel data ingestion pipeline to support data movement and distribution of these large samples, which are approximately 1 GB in size per sample. To accelerate the per-step data exchange during stochastic gradient descent (SGD), we extended LBANN's in-memory distributed data store to handle the cosmology data cube format. CosmoFlow is a deep learning tool that allows determination of the initial condition (IC) parameters of a universe based on the simulated 3D distribution of mass in the universe. This work is a continuation of the earlier effort extended to ingest input composed of four redshifts distribution and regress four IC parameters. The CosmoFlow code was ported to Keras, as well as shared with Cerebras and LBANN groups for their benchmarking. The multi-GPU version of CosmoFlow executed on Summit allowed us to search the hyperparameter space.

ExaLearn's softwared dependencies are:

- LBANN: distributed deep learning toolkit akin to Pytorch and TensorFlow
- Hydrogen: GPU-accelerated distributed linear algebra with explicit host and device memory allocation
- Aluminum: GPU-centric asynchronous communication library that enables MPI operations to be scheduled along with GPU kernels on streams CUDA, cuDNN, cuBLAS, NCCL.

Progress to Date

Our surrogate model is based on the Generative Adversarial Networks (ExaGAN, formerly CosmoGAN) framework, and we are experimenting with Variational Autoencoders (VAE). Both models will eventually need hyperparameter tuning at exascale to extend to larger maps and build conditional generative models. For complex conditional models on large maps (currently 5,122 and intent to go to 10,243), we expect to use intra-node model parallelism (up to a few GPUs) and inter-node data parallelism (tens to hundreds of nodes) for faster experimentation during algorithm development and final model training. When such models' hyperparameters are being tuned, they will need thousands of nodes. After training, when applied to generation, this GAN/VAE approach would be coupled with large-scale cosmology applications running on exascale machines (e.g., ExaSky, § 5.2, apps HACC and Nyx).

As part of the ExaLearn 1.0 release, we have published the 3D spatially distributed convolution optimizations within the LBANN framework. This builds on the existing DistConv extension and includes early phase optimizations for parallelizing I/O for large samples. Table 67 summarizes three models, each of which corresponds to the training dataset of 1,283, 2,563, and 5,123 volumes. For each of the models, we have applied several extensions to the original baseline model. First, we add a batch normalization layer after every convolutional layer. Ravanbakhsh et al. [23] reported that batch normalization was critical in training a similar model. However, in the original CosmoFlow model, it was dropped due to the computational cost of batch normalization, especially in a distributed training setting. We present training results in both configurations and observe that while batch normalization increases memory requirements, it improves final prediction accuracy. Second, in order to simplify comparison of the three models, we insert additional pooling layers in the 2,563 and 5,123 models (the pool6 layer in both models and the pool7 layer in the 5,123 model). Finally, we experimentally identified several minor parametric changes that improve prediction accuracy or

Table 67: CosmoFlow network architecture, where W_i is the input width of the spatial dimensions. A stride 1 convolution and stride 2 pooling is used unless explicitly mentioned. All layers use a padding width of 1.

Layer		**Output width**		
Name	Weights	$W_i = 128$	$W_i = 256$	$W_i = 512$
conv1	16×3^3	128^3	256^3	512^3
pool1	16×3^3	64^3	128^3	256^3
conv2	32×3^3	64^3	128^3	256^3
pool2	32×3^3	32^3	64^3	128^3
conv3	64×3^3	32^3	64^3	128^3
pool3	64×3^3	16^3	32^3	64^3
conv4	128×3^3 (stride of 2)	8^3	16^3	32^3
pool4	128×3^3	4^3	8^3	16^3
conv5	256×3^3	4^3	8^3	16^3
pool5	256×3^3	2^3	4^3	8^3
conv6	256×3^3	2^3	4^3	8^3
pool6	256×3^3	N/A	2^3	4^3
conv7	256×3^3	2^3	2^3	4^3
pool7	256×3^3	N/A	N/A	2^3
fc1	$2\,048 \times 2\,048$	2 048	2 048	2 048
fc2	$2\,048 \times 256$	256	256	256
fc3	256×4	4	4	4
# conv. ops. [GFLOP/sample]		55.55	443.8	3 550
Forward [GFLOP/sample]		18.52	147.9	1 183
Memory [GB/sample]		0.824	6.59	52.7
# parameters ($\times 10^6$)		9.44	9.44	9.44

simplify the implementation of distributed convolution, including removal of biases and use of padding in convolutional layers.

Simulation ML approaches have been attempted at full-machine scale on Summit and Cori with the ExaLearn CosmoFlow software running stably at 2,400 nodes on Summit resulting in 111× speedup with 128 nodes as shown in Fig. 62. Exploiting 2-way spatial + 2-way sample parallelism and reducing memory pressure scalable 3D distributed convolution is a game changer. Every code in ECP using AMReX could take advantage of this including Combustion-Pele (§ 4.2 and ExaStar (§ 5.1).

Next Steps

The goal of the Performance crosscut area is to identify opportunities to scale up traditional ML and DL training methods on leadership-class DOE HPC systems. For this first milestone, we noted the Surrogate application pillar had identified an ML training task and associate data set that was amenable to scaling up the training environment. To address the scalability challenges, we leveraged the open-source DL training framework being developed by LLNL under multiple projects, including CANDLE. The LBANN framework is a distributed memory DL toolkit intended for researching training and inference at large scales. LBANN already has demonstrated the capabilities to train on Sierra and Summit. Prior work in this framework developed support for accelerating training of convolutional neural networks (CNNs) via a hybrid spatial/sample distribution known as the DistConv extension of LBANN.

Our ML work use case leverages OpenAI Gym, PyTorch, TensorFlow, PetSC, LAMMPS, CUDA, profilers, and used ParaView a little as workflow building blocks. For scalable deep learning, we are building on

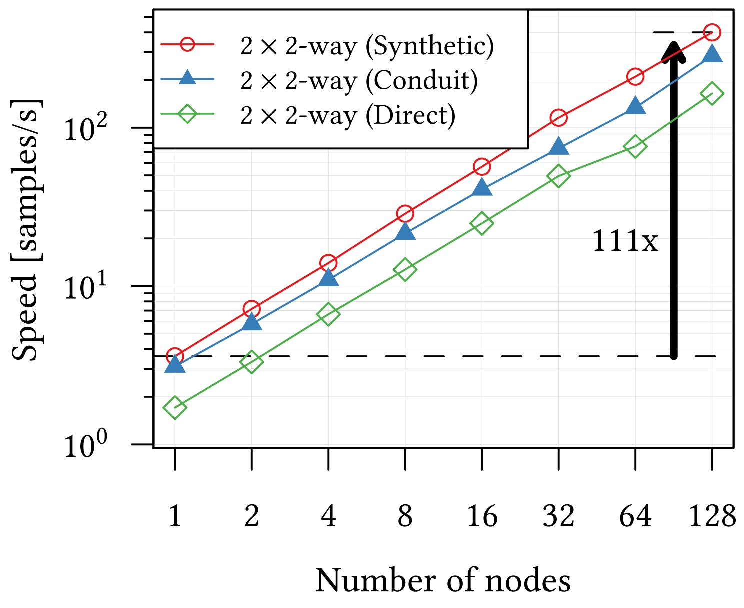


Figure 62: Weak-scaling performance when training CosmoFlow using 2-way spatial and 2-way sample parallelism.

LBANN and CANDLE, both partially developed under ECP projects. For LBANN, we have a toolkit that provides easy management of large-scale parallelism while adopting a frontend that looks like an industry tool (PyTorch). Furthermore, we are adopting support for standard model exchange formats such as ONNX. Raw capabilities built into LBANN are easily generalized to other applications. We have developed a flexible data ingestion frontend that can natively execute Python code snippets from standard industry training pipelines. Demonstrated unique capability to train models that provide quantitatively better science at unprecedented scale on Sierra. Ongoing benchmarking for the I/O is taking place on Summit, and software is getting ported to Perlmutter EA systems. Within the surrogate application pillar, the task of training deep neural networks, such as CosmoFlow and CosmoGAN, is extremely GPU friendly. The LBANN scalable deep learning tool has trained CosmoFlow on 512 GPUs (128 nodes). LBANN also has trained another CycleGAN network using 16,384 GPUs on all of Sierra.

ExaLearn is training a large 3D convolutional neural network (CNN) for cosmological analysis of dark matter distribution from simulations. As the training data scales, the anticipated bottleneck becomes the I/O necessary to deal with the large numbers of substantial training data. Our strategy is to characterize and provide solutions for the I/O bottleneck. The first step involved benchmarking the I/O of LBANN, the system used by the Surrogates team for the CNNs. The initial benchmarks were performed on Cori. Next, these code and training data were moved to Summit at the OLCF, where benchmarking was performed. The I/O bottleneck on Summit was significantly less than on Cori. For next steps in LBANN one performance optimization that we are working on is the use of the float16 (FP16) data type within LBANN and use of the TensorCores. Training to-date in LBANN has been done at float32 (FP32) due to earlier software design decisions within the LBANN toolkit. The Hydrogen GPU accelerated distributed linear layer is already engineered for FP16 and the layer-wise support for FP16 in LBANN should be complete by the end of November 2019. Once this support is in place, we will tune the custom kernels in LBANN for using the TensorCores. Beyond these next steps, we are refactoring the GPU support within LBANN, Hydrogen, and Aluminum to be GPU platform agnostic, initially bringing in support for AMD GPUs using HIP. At LLNL we have a new AMD-based testbed called Corona and we have an initial port of the LBANN software stack onto the AMD toolchain. We are still working on tuning the performance using the HIP/ROCm software stack. Additionally, we will look at fusing key layer patterns to enable fused GPU kernels. Scalable deep learning provides 10× improvement in prediction accuracy with large samples for the development of a cosmological surrogate model. Simulation ML approaches have been carried out at full-machine scale on Summit with the

ExaLearn CosmoFlow software running stably at 2,400 nodes.

ACKNOWLEDGEMENTS

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] 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.*, 137:15781–15794, 2015.

[2] AMB2018-01 Description. https://www.nist.gov/ambench/amb2018-01-description, 2018.

[3] Kayahan Saritas, Wenmei Ming, Mao-Hua Du, and Fernando 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*, 10(1):67–74, 2019.

[4] Hyeondeok Shin, Ye Luo, Panchapakesan Ganesh, Janakiraman Balachandran, Jaron T. Krogel, Paul R. C. Kent, Anouar Benali, and Olle Heinonen. Electronic properties of doped and defective NiO: A quantum Monte Carlo study. *Phys. Rev. Materials*, 1:073603, Dec 2017.

[5] ITER. https://www.iter.org, 2019.

[6] Tuguldur Sukhbold, S. E. Woosley, and Alexander Heger. A high-resolution study of presupernova core structure. *The Astrophysical Journal*, 860(2):93, Jun 2018.

[7] Athira Menon and Alexander 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*, Apr.

[8] A. W. Steiner, M. Hempel, and T. Fischer. Core-collapse Supernova Equations of State Based on Neutron Star Observations. *The Astrophysical Journal*, 774:17, September 2013.

[9] J. D. Emberson, Nicholas Frontiere, Salman Habib, Katrin Heitmann, Patricia Larsen, Hal Finkel, and Adrian Pope. The borg cube simulation: Cosmological hydrodynamics with crk-sph. *The Astrophysical Journal*, 877(2):85, May 2019.

[10] Nicholas Frontiere, Cody D. Raskin, and J. Michael Owen. Crksph – a conservative reproducing kernel smoothed particle hydrodynamics scheme. *Journal of Computational Physics*, 332:160–209, Mar 2017.

[11] A. J. Rodgers, A. Pitarka, and D. McCallen. The effect of fault geometry and minimum shear wavespeed on 3D ground motion simulations for an Mw 6.5 Hayward Fault Scenario earthquake, San Francisco Bay Area, northern California. *Bulletin of the Seismological Society of America*, 109(4):1265–1281, 2019.

[12] S.M. Wiederhorn. Subcritical crack growth. In RJ BROOK, editor, *Concise Encyclopedia of Advanced Ceramic Materials*, pages 461 – 466. Pergamon, Oxford, 1991.

[13] Andreas Wächter and Lorenz T Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106(1):25–57, 2006.

[14] Alan C Hindmarsh, Radu Serban, and Aaron Collier. example programs for ida v5.0.0. Technical report, 0. Technical Report UCRL-SM-208113, LLNL.

[15] Michael A Heroux and James M Willenbring. A new overview of the trilinos project. *Scientific Programming*, 20(2):83–88, 2012.

[16] Cosmin 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*, 133:337–348, 2019.

[17] Jakub Kurzak, Panruo Wu, Mark Gates, and Ichitaro Yamazaki. Designing SLATE: Software for linear algebra targeting exascale. 2017.

[18] Jack Dongarra, Mark Gates, Azzam Haidar, Jakub Kurzak, Piotr Luszczek, Stanimire Tomov, and Ichitaro Yamazaki. Accelerating numerical dense linear algebra calculations with GPUs. In *Numerical computations with GPUs*, pages 3–28. Springer, 2014.

[19] Jonathan D Hogg, Evgueni E Ovtchinnikov, and Jennifer A Scott. A sparse symmetric indefinite direct solver for gpu architectures. *ACM Trans. Math. Softw.*, 42(1):1–1, 2016.

[20] H Carter Edwards, Christian R Trott, and Daniel Sunderland. Kokkos: Enabling manycore performance portability through polymorphic memory access patterns. *Journal of Parallel and Distributed Computing*, 74(12):3202–3216, 2014.

[21] Richard D Hornung and Jeffrey A Keasler. The RAJA portability layer: overview and status. Technical report, Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States), 2014.

[22] Scientific data reductions benchmarks. https://sdrbench.github.io, 2019.

[23] Siamak Ravanbakhsh, Junier Oliva, Sebastien Fromenteau, Layne C. Price, Shirley Ho, Jeff Schneider, and Barnabás Póczos. Estimating cosmological parameters from the dark matter distribution. In *ICML*, 2016.

A. APPLICATION CODE SUMMARY

Table 68 gives a brief summary of the codes used by AD Application projects, the primary languages and their strategy for utilizing the GPUs.

Table 68: 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	Fortran	CUDA, OpenACC
Combustion-PELE	PeleLM	Fortran	CUDA, OpenACC
WarpX	WarpX + PICSAR	C++	AMReX abstractions
ExaSMR	Nek5000	Fortran	OpenACC
ExaSMR	NekRS	Fortran	libParanumal (OCCA)
ExaSMR	OpenMC	C++	OpenMP, OpenCL or SYCL
ExaSMR	Shift	C++	CUDA
WDMApp	GENE	Fortran	OpenMP
WDMApp	GEM	Fortran	OpenACC
WDMApp	XGC	Fortran	OpenMP, OpenACC
MFIX-Exa	MFIX-Exa	C++	AMReX abstractions
ExaWind	Nalu-Wind	C++	Kokkos
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