EXASCALE COMPUTING PROJECT

ECP Software Technology Summaries

This is a compilation of summaries of software technology products that were funded under the Exascale Computing Project. The research, development, and deployment activities of these code teams produced an integrated, readily deployable, portable, production-ready, and high-performance software stack that encompasses program development tools, scalable and resilient numerical algorithms, storage solutions, data analytics, and visualization capabilities to enable and accelerate the development of mission-critical applications for exascale supercomputers.

July 2024

This research was supported by the Exascale Computing Project (17-SC-20-SC), a joint project of the U.S. Department of Energy's Office of Science and National Nuclear Security Administration, responsible for delivering a capable exascale ecosystem, including software, applications, and hardware technology, to support the nation's exascale computing imperative.

U.S. DEPARTMENT OF ENERGY | Office of Science

NNSA National Nuclear Security Administration

EXASCALE COMPUTING PROJECT

CONTENTS

ADIOS Provides an Adaptable I/O Framework for Industrial, Academic, and Exascale Applications

The Motivation

The Adaptable I/O System (ADIOS) in the Exascale Computing Project (ECP)'s Data and Visualization portfolio, addresses data management and in situ analysis concerns for industrial, academic, and exascale applications. The ADIOS team successfully optimized I/O on exascale architectures according to modern software practices, which means the software is scalable, maintainable, sustainable, and extensible while ensuring performance.

The developers describe ADIOS as a framework that puts computation in the proper place at the proper time in a data-rich environment. The framework was designed to enable scientists and engineers to describe their data and how they would like to use it without being distracted by details of the underlying storage technology and file formats.

The Solution

To provide a very simple way to realize extreme-performance I/O, the project focused on transforming the existing ADIOS 1.x software (which has been used successfully on petascale resources), into a community framework that can efficiently utilize exascale hardware. This effort required refactoring to create a generalized software ecosystem (rather than a point solution) that can be customized and shared among a variety of users, exascale applications, and hardware technologies. Software modularity, flexibility, and extensibility were all achieved by using C++ and extending, tuning, and hardening the core services such as I/O and staging to support applications, architectures, and technologies up to and including exascale systems. The result is a sustainable, maintainable, and more approachable all-scale framework that can be used by a wide community of users and developers.

The Impact

The ECP team adopted a project approach that focused on doing deep dives of software needs with scientists, end users, and developers to ensure that the software development process led to specific, verifiable results that positively impact their customers. This approach meant working directly with select US Department of Energy (DOE) and other ECP application developers to understand their requirements for scalable and high-performance I/O, staging, or code coupling solutions to avoid any I/O bottlenecks in their production runs. Notable projects include the following:

- GEM: a kinetic code used to model the central core plasma in a tokamak reactor
- X-point Gyrokinetic Code (XGC): a code used to model the edge of the plasma in a tokamak that includes some of the more complex features of the magnetic field
- WarpX: an exascale application for plasma accelerators that enables the exploration of outstanding questions in the physics of the transport and the acceleration of particle beams in long chains of plasma channels
- Particle-In-Cell Scalable Application Resource (PICSAR): a high-performance repository intended to help scientists with porting their particle-in-cell codes to the next generation of exascale computers
- Energy Exascale Earth System Model (E3SM): a cloud-resolving earth system model with the throughput necessary for multidecade, coupled high-resolution climate simulations

Sustainability

The challenge of supporting hardware portability and runtime performance tuning was addressed by paying special attention to the architecture of the ADIOS code base and adopting state-of-the-art software engineering methodologies to make it easier for the DOE community to use, maintain, and extend the ADIOS framework. The new code base is

governed with state-of-the-art software development practices, including a GitHub workflow of pull requests with reviews, continuous integration that enforces well-tested changes to the code only, and nightly builds to efficiently catch errors on various combinations of architectures and software stacks.

For More Information and References

- ADIOS can be installed by using Extreme-Scale Scientific Software Stack binaries or containers or by using custom source code builds via Spack: https://e4s.io/.
- ADIOS is open source and available at https://github.com/ornladios/ADIOS2/.
- Scott Gibson's podcast episode "ADIOS: Providing A Framework for Scientific Data on Exascale Systems" provides more information about the ADIOS project.
- The ADIOS page on the ECP website provides an overview.

Dyninst – Dynamic Instrumentation to Efficiently Obtain Performance Profiles of Unmodified Executables

The Motivation

Dyninst provides a standalone set of tool kits with platform-agnostic interfaces designed to provide the high-performance, portable, and feature-rich analysis of binary files across a wide range of heterogeneous CPU and GPU hardware architectures. Tools that leverage Dyninst can analyze binaries on any supported platform without the need for modification or recompilation.

The Exascale Computing Project (ECP) focused on adding support for the analysis of GPU binaries to enhance of this binary analysis toolkit.

The Solution

The project team created platform-independent representations for GPU binaries from AMD, Intel, and NVIDIA to generate tooling that can reduce analysis time for even the largest applications. In particular, ECP performance analysis tools can now use Dyninst to instrument binaries and attribute CPU and GPU performance measurements back to detailed source code contexts, thereby enabling accurate and actionable development efforts for authors of scientific software.

The project team worked with community stakeholders, including standards committees, vendors, and open-source developers, to improve hardware and software support for building performance analysis tools. This resulted in the implementation of a first in the world set of binary analysis techniques for accelerator hardware from multiple vendors.

The Impact

The Dyninst toolkit is foundational to all of the ECP integration demonstrations, which is why the ECP invested significant funding and software development time.

The project team delivered a portable tool kit for building low-overhead, full-featured tools across a wide variety of post-exascale hardware via E4S, vendor-provided software stacks, and as an open-source project used by dozens of teams across the world. This also makes Dyninst available to the global HPC, cloud, and industry communities.

Dyninst has been integrated with the following ECP clients:

- HPCToolkit is designed to measure and analyze the performance of applications, libraries, and frameworks within and across the compute nodes of GPU-accelerated platforms. Developers use it to identify bottlenecks and inefficiencies that keep codes from achieving exascale performance.
- TAU (Tuning and Analysis Utilities) is an open-source performance evaluation tool kit. TAU is developing production quality, standard conforming OpenMP tool support for the latest OpenMP 5.2 specification and beyond.

Sustainability

Because the tool kit interfaces do not require users to have a working understanding of the underlying hardware, a tool that leverages Dyninst can analyze binaries on any supported platform without the need for modification or recompilation. This ensures continued and expanded use of the Dyninst toolkit on supported GPU accelerators by those

who wish to gain an understanding of application performance on their systems — even when they don't have access to source code. This is particularly useful to commercial users and those who use vendor libraries as it is unlikely that they will have access to source code.

As a demonstration of the success of the ECP effort, Dyninst successfully builds and runs on early access exascale systems. HPCToolkit, for example, uses Dyninst to analyze AMD CPU binaries to attribute performance measurements to source code contexts, including functions, loops, and lines. Such cross-platform binary analysis capability is essential to understanding and optimizing application performance, which ensures continued use and need for this toolset on new hardware platforms.

For more information and references:

- Dyninst can be installed though E4S binaries, containers, or via custom source code builds via SPACK: https://e4s.io.
- See https://www.paradyn.org/ for more information and a list of recent papers.

E4S and SDK: Enabling Software Interoperability

The Exascale Computing Project (ECP) has delivered a very large number of impactful software technologies. But many scientific applications don't rely on only one set of software capabilities, and it is important for these software products to behave as an ecosystem instead of in isolation. The new challenge in high performance computing (HPC) is to ensure different software products can effectively interoperate to maintain a sustainable future.

Achieving this interoperability requires software to be tested together and be built in common ways. ECP offers two solutions: the Extreme-scale Science Software Stack (E4S) and the software development kits (SDKs).

E4S encompasses the full suite of ECP's software technology products, and provides HPC users with improved access to these products with preinstalled versions, the Spack build cache—Spack being ECP's software deployment tool that handles software dependencies—and Spack recipes that can be customized to specific needs. Since the majority of users will likely only require a subset of ECP software, E4S offers the option to pick what they need and turn off anything else. E4S also coordinates with HPC facilities to help deploy the software on various machines.

SDKs, meanwhile, offer specific categories of individual software products with common use cases. These groups of software are specifically tailored to be able to work together. For example, the programming models and runtimes SDK focuses on enabling robust, low-cost exascale applications, while the development tools SDK addresses common problem patterns at scale. Other SDKs include the math libraries SDK, the data and visualization SDK, and the scientific workflows SDK. Each SDK relies on its developer community to identify their own specific needs, and the SDK teams find and test ways to use related software together and deliver them through E4S.

Both E4S and SDKs provide important solutions to government agencies—including the National Oceanic and Atmospheric Administration and NASA—as well as to industry users. By increasing the interoperability between ECP's software technologies and coordinating their delivery, E4S and SDKs have helped increase the availability, quality, and long-term sustainability of HPC software.

ExaHDF5: Exascale Enhancements for the Popular HDF5 Scientific Data Library

The Motivation

The Hierarchical Data Format version 5 (HDF5) is the most popular high-level I/O library for scientific applications to write and read data files. The HDF Group released the first version of HDF5 in 1998. Over the past 20 years, numerous applications in scientific domains and in finance, space technologies, and many other business and engineering fields have used HDF5. It is the most used library for performing parallel I/O on existing high-performance computing (HPC) systems at US Department of Energy (DOE) supercomputing facilities.

Modern hardware has added more levels of storage and memory. Therefore, the ExaHDF5 team of the Exascale Computing Project (ECP) aimed to enhance the HDF5 library so that it can more effectively use the hardware capabilities of these new platforms. The ExaHDF5 team also added new capabilities to the HDF5 data management interface to support other formats and storage systems.

The Solution

To address changes in modern supercomputer storage systems and account for increases in the volume and complexity of application data, the ExaHDF5 team added new features to leverage commonly available capabilities in the latest exascale architectures.

The ExaHDF5 team has worked with numerous ECP application teams to integrate asynchronous I/O, subfiling in the Cabana framework, and caching. The Virtual Object Layer (VOL) and interoperability features with PnetCDF (Parallel Network Common Data Form) and ADIOS (Adaptable I/O Systems) data open up the rich HDF5 data management interface to science data stored in other file formats. The team created the dynamically pluggable Virtual File Drivers (VFD) feature, which allows the HDF5 file format to be extended to new sources and destinations of I/O, including GPU memory and cloud storage.

The Impact

The ExaHDF5 team has released these new features in HDF5 for broad deployment on HPC systems. The technologies developed to use the massively parallel storage hierarchies being built into pre-exascale and current exascale systems enhance performance and scalability. The enhanced HDF5 software has also been refactored to achieve efficient parallel I/O on these exascale (and other) systems. Owing to the popularity of HDF5, the greater versatility and performance will positively impact many DOE science and industrial applications.

The benefits span many applications, workloads, and users. NASA, for example, gives HDF5 software the highest technology readiness level (TRL 9), which is a category that contains actual systems that have been flight proven through successful mission operations. In the ECP ecosystem, numerous applications have depended on HDF5 for performing I/O.

Sustainability

The critical nature and broad user community ensure that ExaHDF5 will support application needs far into the future. HDF5 and other subproducts released by the ExaHDF5 team are now readily available via the Extreme-Scale Scientific Software Stack (E4S), making them available to any user wishing to use HDF5 on an HPC platform or cloud provider.

Additional Information and References

- ExaHDF5 installation through E4S binaries, containers, or custom source code builds via Spack: https://e4s.io
- HDF5 GitHub repository: https://github.com/HDFGroup/hdf5/
- The HDF Group: https://www.hdfgroup.org/
- Asynchronous I/O VOL connector: https://github.com/hpc-io/vol-async/
- Cache VOL connector: https://github.com/hpc-io/vol-cache/
- H5bench (HDF5 benchmark suite): https://github.com/hpc-io/h5bench/

ExaWorks Finds Common Ground Inside the Workflows Community to Create Robust, Sustainable, Accelerator-capable HPC Workflows

The Motivation

The consensus in the HPC community is that the most practical path to continued performance growth in the near term will be architectural specialization in the form of various kinds of accelerators (which can speed some applications by up to 100×). New software implementations along with new mathematical models and algorithmic approaches are necessary to advance the science that can be done with these specialized architectures, but applications are just part of the HPC ecosystem, which is embedded in advanced scientific workflows. The complexity and performance demands of these heterogenous HPC workflows highlight the need for ExaWorks, an Exascale Computing Project (ECP)–funded effort that provides access to *portable* hardened and tested workflow components through a software development kit (SDK).

The need for the ExaWorks SDK is motivated by the observation that even with the historical prevalence of workflows in HPC, no single tool has demonstrated that it can meet the needs of all HPC projects. As explained in "ExaWorks: Workflows for Exascale," these legacy solutions were created through bespoke processes that unfortunately resulted in stove-piped software solutions. An explosion in the number of per-project independent solutions is making development and support of workflows increasingly unwieldy, expensive, and unsustainable. Furthermore, the complexity of these workflows is expected to increase because of the need to run on multiple heterogeneous computing platforms and exploit all levels of parallelism across multiple disparate distributed computing environments.

The Solution

Thinking beyond current and legacy HPC workflows, the ExaWorks project created a general component-based infrastructure designed to support a diverse range of workflows as a first-order requirement. The beauty of this approach lies in the generality of workflows that can be created with ExaWorks for all major computing environments, including DOE supercomputers, cloud environments, and academic/industry HPC clusters. The ExaWorks SDK enables the creation of HPC, AI, and industry workflows in all these environments. This generality is realized by providing developers access to hardened and tested components they can use for the development and maintenance of an interoperable ecosystem of workflow systems and components.

The Impact

The value of the ExaWorks SDK has been proven many times including by finalists and winning projects in the prestigious Gordon Bell competition. The ExaWorks components can be quickly deployed via the freely and conveniently deployed E4S (https://e4s.io) in the cloud or on-premises as containers and binary distributions. Custom deployments can be performed from source builds using the SPACK package manager.

Sustainability

To lay a sustainable groundwork for transforming workflows research and development, the ExaWorks project is partnering with other initiatives to bring the international workflows community together as discussed in "A Community Roadmap for Scientific Workflows Research and Development".

To date, the ExaWorks project has successfully brought together stakeholder communities at a series of summits that represented HPC facilities, application developers, and numerous workflow systems. The ExaWorks team created the Workflows Community Initiative (WCI) to bring people together to find a common ground inside the community. The WCI has quickly grown into a thriving community with many members (170 as of July 2023), numerous workflow

systems cataloged (29 as of July 2023), an active jobs board, and a regular workflow newsletter. The project has also received seed money based on the success of the ECP-funded work to create the Center for Sustaining Workflows and Applications Services (SWAS) after ECP funding was closed.

For more information and references:

- To install the ExaWorks SDK via binaries, containers, or source code builds via SPACK: https://e4s.io
- "A Community Roadmap for Scientific Workflows Research and Development"
- "ExaWorks Provides Access to Community Sustained, Hardened, and Tested Components To Create Gordon Bell Prize–Winning Workflows"
- The Workflows Community Initiative (WCI)
- The Center for Sustaining Workflows and Applications Services (SWAS)

Flang – An LLVM-based Open Source Fortran Frontend

The Motivation

Today there are several commercially-supported Fortran compilers, typically available from one vendor and often for a limited set of platforms. None of these are open source. While the GNU gfortran compiler is open source and available on a wide variety of platforms, the source base is not modern LLVM-style C++ and the GPL license is not compatible with LLVM — a powerful set of compiler and toolset technologies. This places limits on scientific collaborations, and thus has an impact on broader community participation and adoption.

Removing these limits provided the motivation for the Exascale Computing Project (ECP) Flang project, which created a source base with the maturity, features, and performance of proprietary solutions along with the cross-platform capability of GNU compilers, and which is licensed and coded in a style that will be embraced by the LLVM community. Additional challenges come from supporting all Fortran language features, language extensions (e.g., OpenMP, OpenACC), and scalability required for effective use of exascale-class systems.

The Solution

The Flang project provides an open-source Fortran standard compiler front-end for the LLVM Compiler Infrastructure (see http://llvm.org). Leveraging LLVM, Flang will provide a cross-platform Fortran solution available to ECP and the broader international LLVM community. The goals of the project include extending support to GPU accelerators, thus targeting GPU-accelerated Exascale systems, and supporting LLVM-based software and tools of interest to a large, deployed base of Fortran applications.

LLVM's growing popularity and wide adoption make it an integral part of the modern software ecosystem. Flang provides a foundation for Fortran that will complement and interoperate with the Clang C/C++ compiler and other tools within the LLVM infrastructure. The project provides a modern, open-source Fortran implementation that is stable, has an active footprint within the LLVM community, and will meet the specific needs of ECP as well as the broader scientific computing community.

Beyond parsing and semantics, the project has focused on creating a Fortran-centric intermediate representation (Fortran IR, or FIR) that leverages recent activities within Google. (See "MLIR: accelerating AI with open-source infrastructure" to understand how MLIR addresses the complexity of modern, accelerated hardware technology).

The Impact

With the adoption of Flang into the LLVM community, the project successfully introduced, developed and delivered a solid, alternative Fortran compiler for DOE's platforms. Flang also established and has been growing a strong community around it.

It is critical that the Flang project continues to act as a good shepherd within the LLVM community. This engagement is in the best interest of ECP as well as the long-term success of Fortran and enables leveraging the significant momentum and strengths of the LLVM and Fortran codebases, both of which are in wide use and are generally accepted.

Sustainability

Flang was formally accepted as an official component of LLVM in 2019 and merged portions of its initial code base into the main LLVM repository in April 2020. Work continues today with a growing set of contributors to the code base. This,

plus the need for a modern, platform agnostic Fortran compiler that does not limit potential collaborations ensures a long lifespan.

For more information and references:

- Flang can be installed though E4S binaries, containers, or via custom source code builds via SPACK: https://e4s.io.
- https://github.com/llvm/llvm-project/
- https://blog.google/technology/ai/mlir-accelerating-ai-open-source-infrastructure/

Flux – A Next-Generation Workload Management Framework

The Motivation

Flux is a next-generation workload management framework for HPC. Flux maximizes scientific throughput by scheduling the scientific workloads as requested by HPC users.

The workload manager is responsible for efficiently delivering compute cycles of HPC systems to multiple users while considering their diverse resource types—e.g., compute racks and nodes, central and graphics processing units (CPUs and GPUs), multi-tiered disk storage.

Two technical trends are making even the best-in-class products significantly ineffective on exascale computing systems. The first trend is the evolution of workloads for HPC. With the convergence of conventional HPC with new simulation, data analysis, machine learning (ML), and artificial intelligence (AI) approaches, researchers are ushering in new scientific discoveries. But this evolution also produces computing workflows—often comprising many distinct tasks interacting with one another—that are far more complex than traditional products can sufficiently manage. Second, hardware vendors have steadily introduced new resource types and constraints into HPC systems. Multi-tiered disk storage, CPUs and GPUs, power efficiency advancements, and other hardware components have gained traction in an era in which no single configuration reigns. Many HPC architectures push the frontiers of compute power with hybrid (or heterogeneous) combinations of processors. The workload management software must manage and consider extremely heterogeneous computing resources and their relationships for scheduling in order to realize a system's full potential.

The need to schedule work for modern workflows motivated the Exascale Computing Project (ECP) Flux project to create a highly scalable scheduling solution that supports high-performance communication and coordination across hundreds of thousands of jobs, which could not be accomplished with traditional HPC schedulers. Nearly all existing scheduling systems were designed when the workflows were much simpler. These problems have led users to develop their own ad hoc custom scheduling and resource management software or use tools that perform only workflow management or only scheduling.

The Solution

Flux manages a massive number of processors, memory, GPUs, and other computing system resources—a key requirement for exascale computing and beyond – using its highly scalable fully scheduling and graph-based resource modeling approach.

Thus, Flux provides first-class support for job coordination and workload management, which avoided the legacy issue of groups individually developing and maintaining ad hoc management software. With Flux, a job script with multiple tasks submitted on a heterogeneous HPC system remains simple, requiring only a few more lines within the script.

The Impact

Flux was the basis for a workflow of ECP ExaAM (ExaConstit) that improved throughput by 4×. The Flux-based ML drug design workflow was part of an SC20 COVID-19 Gordon Bell finalist submission.

See https://flux-framework.org/ for more examples of the impact of this flexible resource management framework.

Sustainability

Flux manages a massive number of processors, memory, GPUs, and other computing system resources. The ubiquity of these heterogenous architectures running at scale ensures a continues user base and active user community. Flux is also part of the ExaWorks SDK, which brings together four seed workflow technologies, specifically Flux, Parsl, RADICAL, and Swift/T.

For more information and references:

- Flux can be installed though E4S binaries, containers, or via custom source code builds via SPACK: https://e4s.io.
- http://flux-framework.org
- "Flux Framework: A flexible framework for resource management customized for your HPC site."
- "Flux: Overcoming scheduling challenges for exascale workflows"
- "The Flux Supercomputing Workload Manager: Improving on Innovation and Planning for the Future"

HPCToolkit: A General, Vendor-Agnostic, Platform-Agnostic, Performance-Monitoring Toolkit

The Motivation

Today's fastest supercomputers and new exascale systems all employ GPU-accelerated compute nodes. Almost all the computational power of a GPU-accelerated supercomputer comes from the GPUs rather than CPUs. Efficiently running on these GPU-accelerated systems is challenging because they have complex memory hierarchies that include multiple memory technologies with different bandwidth and latency characteristics. Adding to this complexity in GPU-accelerated systems are the nonuniform connections between the different memory spaces and computational elements (e.g., CPU and GPU devices).

Application developers can employ abstractions to hide some of the complexity of these parallel systems. However, performance tools that use each of the features in these heterogeneous systems provide an easier path to increased application performance. The purpose of these tools and specifically the Exascale Computing Project (ECP) HPCToolkit is to provide feedback so developers can improve application performance and efficiency. This objective requires that the performance tool appropriately measure many different hardware devices and provide analysis capabilities so developers can assess how well the individual hardware features are being used. To close the feedback loop, the tool kit must use the measurements to create actionable feedback about the application software and libraries to guide developers as they work to improve the performance, efficiency, and scalability of their applications.

The Solution

The ECP's HPCToolkit team needed to deliver a production-ready, vendor-agnostic tool kit that can measure application performance on several exascale platforms. The team focused on adding new capabilities to measure and analyze interactions between the application software and key hardware subsystems in extreme-scale platforms, including the GPUs and their complex memory hierarchies in GPU-accelerated compute nodes.

This effort required that the HPCToolkit team enhance their software to incorporate emerging hardware and software interfaces for monitoring code performance on both CPUs and GPUs, thereby extending the capabilities of the HPCToolkit software to better measure and analyze computation, data movement, communication, and I/O as an application executes. The additional specificity provides application developers with more information to pinpoint scalability bottlenecks, quantify resource consumption, and assess inefficiencies.

The team also worked to improve performance attribution inside codes that are already optimized to support large collections of complex node-level programming models. This work included providing information for the vendor-specific programming models used on US Department of Energy (DOE) exascale systems. The team also added support to use GPU binaries in the Dyninst binary analysis tool kit, which other ECP tools also use.

To meet application developer needs, the project team has been working with various ECP teams to ensure that they can leverage HPCToolkit's capabilities to measure, analyze, attribute, and diagnose performance issues on ECP test beds and forthcoming exascale systems.

The Impact

The HPCToolkit enables application teams to assess performance on GPU-accelerated systems via a robust and vendor-agnostic tool that can support production-level, exascale ECP applications. The success of these ECP efforts will benefit the general scientific community and help to increase their application performance on other systems.

Sustainability

To provide a sustainable foundation for performance measurement and analysis, the project team worked with community stakeholders, including standards committees, vendors, and open-source developers. This work involved improving hardware and software support for measurement and attribution of application performance on extreme-scale parallel systems.

To develop a sustainable, platform-agnostic, performance-monitoring tool kit, the project team engaged with various DOE hardware vendors to improve support for performance measurement in next-generation GPUs.

The team also worked with a variety of software teams to design and integrate new capabilities into operating systems, runtime systems, communication libraries, and application frameworks. These efforts ensure that HPCToolkit can accurately measure and attribute code performance on extreme-scale and other parallel systems.

All these activities and the general availability of the software—including via the Extreme-Scale Scientific Software Stack (E4S)—ensure a robust and vibrant user base and user community.

Additional Information and References

- HPCToolkit installation through E4S binaries, containers, or custom source code builds via Spack: https://e4s.io
- http://hpctoolkit.org/
- https://www.paradyn.org/
- https://www.github.com/dyninst/dyninst/
- Dyninst paper: Xiaozhu Meng et al., “Parallel Binary Code Analysis,” *PPoPP '21: Proceedings of the 26th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming* (2021): 76–89, https://doi.org/10.1145/3437801.3441604.

HeFFTe – Highly Efficient Multidimensional FFTs with Excellent Large Node Count Scalability

The Motivation

HeFFTe (highly efficient FFTs for Exascale, pronounced “hefty”) enables multinode and GPU-based multidimensional fast Fourier transform (FFT) capabilities in single- and double-precision.
The Exascale Computing Project (ECP) heFFTe project developed a sustainable high-performance multidimensional Fast Fourier Transforms (FFTs) for Exascale platforms. Multidimensional FFTs can be implemented as a sequence of low-dimensional FFT operations in which the overall scalability is excellent (linear) when running in large node count instances.

The need for scalable multidimensional FFTs motivated the development and main objectives of the heFFTe project to: (1) Collect existing FFT capabilities from ECP application teams; (2) Assess gaps, extend, and make available various FFT capabilities as a sustainable math library; (3) Explore opportunities to build multidimensional FFTs while leveraging on-node concurrency from batched FFT formulations; and (4) Focus on capabilities for exascale platforms.

The Solution

HeFFTe leverages established but ad hoc software tools that have traditionally been part of application codes, but not extracted as independent, supported libraries. This required mitigating challenges in the following areas:

- Communications and GPU Optimizations: FFTs are communication bound. A central focus in heFFTe is on algorithmic design to minimize communication and to provide efficient GPU implementations. Other strategies include the use of mixed-precision calculations and data compression for reduced communications (including lossy, e.g., using ZFP compression) to reduce communications overhead.
- Evolving Design: The heFFTe library was designed to support the fftMPI and SWFFT functionalities, which are already integrated in ECP applications. This compatibility means that heFFTe directly benefits these applications and provides integrated solutions.
- More functionalities and application-specific optimizations: These were added through heFFTe backends to support various ECP applications.
- Autotuning Performance portability: These were addressed through use of standards (like 1D FFTs from vendors), portable linear algebra using MAGMA, and parameterized versions that will be autotuned across architectures as required.

The Impact

Considered to be one of the top 10 algorithms of the 20th century, the FFT is widely used by the scientific and high-performance computing (HPC) communities. Over the years, this demand has motivated the use in many applications including molecular dynamics, spectrum estimation, fast convolution and correlation, signal modulation and many wireless multimedia applications. For example, the distributed 3D FFT is one of the most important routines used in molecular dynamics (MD) computations. Its performance can affect MD scalability.

Sustainability

The importance of the FFT to scientific computing, broad utilization and need for the heFFTe multinode and GPU-based multidimensional fast Fourier transform (FFT) capabilities ensures broad, and thus continued use by many in the HPC, cloud, commercial, and academic computing communities.

For more information and references:

- The heFFTe library can be installed though E4S binaries, containers, or via custom source code builds via SPACK: https://e4s.io.
- See https://icl.utk.edu/fft/ for more information and a list of recent papers.
- The ECP highlight, "heFFTE – A Widely Applicable, CPU/GPU, Scalable Multidimensional FFT that Can Even Support Exascale Supercomputers"

Kokkos: Hardware-Independent C++ Applications

The average full-time developer writes about 20,000 lines of useable code per year. Most applications require about 500,000 lines, on top of relying on other libraries that can total millions—like mathematical libraries responsible for certain types of calculation, or visualization libraries that create scientific images. This presents a problem in high performance computing (HPC), where a new system is deployed every few years, and rewriting applications for each new system is an expensive solution.

The Exascale Computing Project (ECP)'s Kokkos project provides a unique set of semantics that allows developers to write their code independently of the HPC platform being used. Implemented in C++, a common and powerful programming language, the Kokkos model does not depend on the implicit differences between systems. With a common interface, Kokkos enables developers to write their code just once and port it from platform to platform, saving significant amounts of time and money.

At least 1,200 developers from over 150 institutions use Kokkos for a total of about 300 different HPC projects. Within ECP, over 50% of the projects that use C++ rely on Kokkos for portability and longevity.

Part of the success of Kokkos is due to how easily it interfaces with other products, such as the math libraries and other tools developed by ECP. These tie-ins provide users with the ability to write larger applications with a much smaller time investment.

Kokkos has a strong history of engagement with the HPC community, providing tutorials and training materials to its wide range of users. Its legacy will continue beyond ECP, as there are currently no other solutions for C++ portability that operate at its scale.

LLVM: An Essential Collection of Modular and Reusable Compiler and Toolchain Technologies

The Motivation

LLVM, winner of the 2012 ACM Software System Award, has become an integral part of the software-development ecosystem for optimizing compilers, dynamic-language execution engines, source code analysis and transformation tools, debuggers and linkers, and a host of programming-language and toolchain-related components. Although LLVM is well suited to the compilation of code from C++ and other languages on CPU hardware, and it supports compilation for some models of GPU hardware, it lacks the high-level optimizations necessary to enable performance-portable programming across future architectures. As a result, LLVM has relied on architecture-specific autotuning to achieve high performance. Unfortunately, tuning every workload in this way for multiple target architectures is impossible.

Therefore, the Exascale Computing Project (ECP) LLVM effort focused on enabling parallelism constructs to be understood and optimized within parallel programs. The effort also involved providing high-level loop transformations to leverage complex memory hierarchies and parallel-execution capabilities.

The Solution

The solution focused on enhancing the LLVM analytic capabilities, loop-optimization infrastructure and autotuner, and intermediate representation (IR). Many other optimizations were also performed.

To enhance the LLVM analytic capability, the team focused on improving both interprocedural analysis and representation of parallelism constructs. Information can now be communicated across boundaries otherwise imposed by parallelism constructs, thereby enabling the LLVM toolchain to transform the parallelism constructs themselves.

The team also worked to enhance the LLVM loop-optimization infrastructure. The objective was to allow the direction of a sequence of loop transformations to loop nests, thereby exposing these features to users via Clang pragmas (in addition to making the features available at an API level to tools such as autotuners), enabling those transformations to execute as specified, and otherwise enhancing the loop-optimization infrastructure.

Additionally, the team examined fundamental IR-level enhancements (as part of the Kitsune development) and runtime call aware optimizations that address the classical lowering of parallelism into runtime calls. The latter mechanism is being implemented upstream as an OpenMP optimization pass, whereas the Kitsune work is currently more exploratory. To address autotuning and the need for code specialization, the team developed a just-in-time compilation technology that integrates naturally with the C++ language and enables embedding of domain-specific languages into C/C++ programs.

The Impact

The LLVM toolchain has become an essential part of the open-source community. Over the years, it has become a bedrock project consisting of several subprojects, many of which are being used in production by a wide variety of commercial and open-source projects and being widely used in academic research.

All performance enhancements to the open-source LLVM project benefit numerous applications written in a variety of languages. Along with the ECP Flang Fortran front-end project, LLVM also supports C, C++, and Objective-C. Many other projects use components of LLVM for various tasks. Through these external projects, LLVM can be used to compile Ruby, Python, Haskell, Rust, D, PHP, Pure, Lua, Julia, and several other languages.

Sustainability

LLVM provides an essential, core technology for nearly every aspect of scientific computing. It has grown a broad and active community of people interested in building these great low-level tools.

Additional Information and References

- LLVM installation through Extreme-Scale Scientific Software Stack binaries, containers, or custom source code builds via Spack: https://e4s.io
- https://llvm.org/
- https://foundation.llvm.org/
- "Kitsune: Efficient, General-purpose Dynamic Software Updating for C"

MFEM – High-Performance Mathematical Algorithms and Finite Element Discretizations that Exploit Complex Computer Architectures and GPUs

The Motivation

The MFEM library effort focused on providing high-performance mathematical algorithms and finite element discretizations for next-generation high-order Exascale Computing Project (ECP) and NNSA Advanced Technology Development and Mitigation (ATDM) applications. MFEM is used in many projects, including BLAST, Cardioid, Palace, VisIt, RF-SciDAC, FASTMath, xSDK, and CEED in the ECP.

To enable many applications to take advantage of unstructured mesh adaptivity, the MFEM team developed adaptive mesh refinement (AMR) algorithms at the library level, targeting both conforming local refinement on simplex meshes and non-conforming refinement for quad/hex meshes. As part of the efforts in the ECP co-design Center for Efficient Exascale Discretizations (CEED), the MFEM team also developed mathematical algorithms and software implementations for finite element methods that exploit increasing on-node concurrency targeting multiple complex architectures (e.g. GPUs).

The Solution

A main component of these efforts was the development of ATDM-specific physics enhancements in the finite element algorithms in MFEM and the MFEM-based BLAST Arbitrary Lagrangian-Eulerian (ALE) code, in order to provide efficient discretization components for LLNL's ATDM efforts, including the MARBL application.

A second main task in the MFEM project was the development of unique unstructured adaptive mesh refinement (AMR) algorithms in MFEM, that focus on generality, parallel scalability, and ease of integration in unstructured mesh applications. The new AMR capabilities can benefit a variety of ECP apps that use unstructured meshes, as well as many other applications in industry and the SciDAC program.

Another aspect of the work was the preparation of the MFEM finite element library and related codes for exascale platforms by using mathematical algorithms and software implementations that exploit increasing on-node concurrency targeting multiple complex architectures (e.g., GPUs).

The Impact

This work includes the libCEED low-level API library, the Laghos miniapp, and several other efforts available through CEED. To reach its many customers and partners in NNSA, DOE SC, academia, and industry, the MFEM team delivers regular releases on GitHub that include detailed documentation and many example codes. Code quality is ensured by smoke tests with GitHub actions on Linux, Mac, Windows, and nightly regression testing at LLNL.

The open-source MFEM finite element library realizes 10000 downloads/year from 100+ countries. The MFEM community includes more than 450 members. The software is available via Spack and OpenHPC. The application outreach and the integration in the ECP ecosystem is further facilitated by MFEM's participation in ECP's xSDK and E4S projects.

Sustainability

The pervasive need for these numerical methods plus the addition of GPU acceleration ensures a large and active user base and community.

In addition to being open-source, the MFEM project hosts an annual workshop and FEM@LLNL seminar series. The project website also hosts Gallery, Publications, Videos and News pages for additional outreach.

For more information and references:

- MFEM can be installed though E4S binaries, containers, or via custom source code builds via SPACK: https://e4s.io.
- https://mfem.org/

MPICH: Efficient Data Transfer on Exascale HPC Systems

Supercomputers consist of thousands to tens of thousands of computing nodes, each with separate central processing units (CPUs) and graphical processing units (GPUs—accelerators for improving computational efficiency) that have their own memory. Data often needs to move between CPUs and GPUs, and between CPUs or GPUs across the entire supercomputer. Efficiently supporting this data transfer is a major challenge in high performance computing (HPC) and scaling this process up for new machines is especially difficult, as new systems are increasingly more complex.

On the other hand, the scientists using these machines often don't need to know how the data transfer process works. The Message Passing Interface (MPI) is a standard way to achieve the necessary portability. It consists of a set of standards describing hundreds of communication functions that, when implemented, allows scientists and developers to write their application once, and move it to a different HPC system with good performance.

While MPI provides the set of standards, the Exascale Computing Project (ECP)'s MPICH library implements these standards. One of the most popular implementations of MPI, MPICH turns the standards into a library that programmers on an HPC machine can use. MPICH addresses many challenges in scaling up data transfer, including performance and memory usage, to ensure the MPI standard is efficiently implemented on new machines.

Almost all ECP projects rely on MPI for interprocess communication, and all three Department of Energy exascale computing systems—Frontier at Oak Ridge, Aurora at Argonne, and El Capitan at Lawrence Livermore—use an MPICH-based MPI implementation. The vendors of these systems, Intel and HPE/Cray, provide an optimized version of MPICH for their customers.

One of the goals of ECP was building exascale supercomputers and enabling applications to run efficiently on these new systems. MPICH directly helps this goal.

PAPI: A Universal Interface for Hardware and Software Metrics

Performance metrics that monitor statistics like processor usage are available in all computers, from smartphones and laptops to supercomputers. These metrics are important for identifying and addressing bottlenecks to making software more efficient. This task gets more and more difficult as computers become more and more complex, making it particularly tricky for high performance computing.

Each type of hardware component—computer processor, memory, or interconnect, for example—has its own vendor-provided interface for monitoring performance and power statistics, as well as configuring hardware parameters, such as power caps. The Performance Application Programming Interface (PAPI), developed by the Exascale Computing Project (ECP), provides a unified consolidation of these statistics in a general interface that is portable across platforms. With PAPI, users can easily access these statistics rather than having to rely on multiple vendor-specific products. PAPI, like other performance monitoring software, runs in the background of ongoing applications, monitoring various hardware and software events to gauge their impact on software performance.

A few key points, however, set PAPI apart from other performance monitoring software. PAPI allows users to register and monitor new software-defined events, in contrast to most vendor tools, which typically only monitor hardware. Furthermore, software-defined events introduced into a library can be incorporated into any application that uses the library, allowing developers to monitor how well each library works within a certain application. These features enable software and application teams, both within and external to ECP, to better understand and improve the performance and power usage of their own software layers, often serving as a foundational tool while providing uniformity across systems.

PAPI has become a ubiquitous tool—often seen as a requirement in high performance computing and is pre-installed on most systems. In addition to providing support for the latest hardware and software layers, ECP also improved PAPI's sustainability by enabling integration into Spack and the Extreme-scale Scientific Software Stack (E4S) and ensuring software robustness through continuous integration and deployment. With the ongoing integration of new monitoring capabilities for advanced hardware and software technologies, PAPI is well-positioned to meet the emerging needs of the high-performance computing community, continuing to make an impact well beyond the ECP era.

PEEKS and Ginkgo – Performance Portable Numerical Capabilities for Scientific Computing

The Motivation

Numerical software has an enormous impact on scientific computing because it acts as the gateway middleware that enables many applications to run on state-of-the-art hardware.

The Exascale Computing Project (ECP) Production-ready, Exascale-Enabled Krylov Solvers (PEEKS) effort focused on communication-minimizing Krylov solvers, parallel incomplete factorization routines, and parallel preconditioning techniques, as these building blocks form the numerical core of many complex application codes.

Those looking to solve sparse linear systems will be interested in the separate ECP Ginkgo project, which brought GPU acceleration to this modern high-performance linear algebra library for manycore systems.

The Solution

To provide performance portability, the PEEKS project relied heavily on the Kokkos and Kokkos Kernels libraries as they provide kernels that are performance portable across a variety of platforms, including CPU and GPU (NVIDIA, AMD, Intel). To increase portability, the project worked to reduce reliance on the NVIDIA UVM (Unified Virtual Memory), which is not widely supported.

The Ginkgo library design was guided by combining ecosystem extensibility with heavy, architecture specific kernel optimization using the platform-native languages CUDA (NVIDIA GPUs), HIP (AMD GPUs), DPC++ (Intel GPUs), and OpenMP (Intel/AMD/Arm multicore).

The Impact

GPU-acceleration in these production-quality scalable packages means that applications can exploit the scalability and performance of newer, exascale-capable hardware. More specifically:

- **PEEKS**: Applications that rely on the following four solver packages on ECP platforms will benefit from the PEEKs effort: distributed linear algebra (Tpetra), Krylov solvers (Belos), algebraic preconditioners and smoothers (Ifpack2), and direct solver interfaces (Amesos2).
- **Ginkgo**: The Ginkgo library contains functionality for solving (sparse) linear systems via iterative and direct solvers, preconditioners, algebraic multigrid (AMG), mixed-precision functionality, and batched routines. Weak and strong scalability of the functionality up to thousands of GPUs has been demonstrated.

Sustainability

PEEKS provides exascale-enabled capabilities in a robust, production-quality software package, thus ensuring continued use by applications that rely on Tpetra, Belos, ifpack2, and Amesos2.

Ginkgo provides native support for NVIDIA GPUs, AMD GPUs, and Intel GPUs to ensure successful delivery of scalable Krylov solvers in robust, production-quality software that can be relied on by ECP applications. Ginkgo is part of the Extreme-scale Scientific Software Development Kit (xSDK), which is part of E4S. This ensures both accessibility and exposure to a large user base.

For more information and references:

- PEEKS and Ginkgo can be installed though E4S binaries, containers, or via custom source code builds via SPACK: https://e4s.io.
- The PEEKS homepage: https://icl.utk.edu/peeks/
- Ginkgo project website: https://ginkgo-project.github.io/
- Ginkgo open-source Git repository: https://github.com/ginkgo-project/ginkgo/
- Ginkgo performance database: https://ginkgo-project.github.io/gpe/

Petsc/TAO – A Long-Term Investment in Essential Software Infrastructure for The Scientific Community

The Motivation

The Portable, Extensible Toolkit for Scientific Computation (Petsc) reflects a long-term investment in software infrastructure for the scientific community. The toolkit provides scalable solvers for nonlinear time-dependent differential and algebraic equations that includes numerical optimization. Petsc is also referred to as Petsc/TAO because it also contains TAO, the Toolkit for Advanced Optimization, software library. (Note: the capitalization of PETSc recently changed to Petsc.)

The Solution

The overall strategy for accelerator support in Petsc/TAO is based on a separation of concerns – a powerful computer science design principle that allows Petsc users who program in C/C++, Fortran, or Python the ability to employ their preferred GPU programming model, such as Kokkos, RAJA, SYCL, HIP, CUDA, or OpenCL. Support for GPU devices required innovative solutions as discussed in the paper "Toward Performance-Portable PETSc for GPU-based Exascale Systems".

The ECP Petsc/TAO project added or augmented a plethora of capabilities in the toolkit. Work included:

- Addressing the memory bandwidth limitations of traditional sparse-matrix-based techniques for linear, nonlinear, and ODE solvers, as well as optimization algorithms.
- Revising synchronization to enable better scaling.
- Adding support for various accelerators and their associated programming models.
- Developing and implementing a coordination layer for ensemble workloads that enables load balancing and the ability to grow and shrink a collection of running simulations.

The Impact

Within the ECP, the Petsc/TAO team has been working with applications such as Chombo-Crunch, which addresses carbon sequestration, and the whole device model application (WDMApp) for fusion reactors. The US Department of Energy identified WDM as a priority for "assessments of reactor performance in order to minimize risk and qualify operating scenarios for next-step burning plasma experiments".

Sustainability

The production-quality Petsc/TAO toolkit illustrates a time-proven use case in connecting users and developers, who together have added capabilities and adapted the software to the unforeseen and radical system architectures that have been installed in datacenters over a period of decades. To remain fresh, the project provides biannual releases.

Petsc/TAO provides an exemplary use case in how to incorporate good software practices that worthy of study as the Petsc/TAO effort has successfully navigated the HPC landscape to earn both funding and extensive community support over many generations of supercomputers. All three are essential and need to be taken as an object lesson for any aspiring software effort (e.g., good software practices, funding, and community support).

For more information and references:

- Petsc/TAO can be installed though E4S binaries, containers, or via custom source code builds via SPACK: https://e4s.io
- PETSC/TAO: How to Create, Maintain, and Modernize a Numerical Toolkit Throughout Decades of Supercomputer Innovations
- Toward Performance-Portable PETSc for GPU-based Exascale Systems
- http://www.mcs.anl.gov/petsc

PnetCDF – A Widely Used Parallel I/O Library for Application Developers and Library Authors

The Motivation

PnetCDF (Parallel netCDF) is a high-performance parallel I/O library for writing multidimensional, typed data in the NetCDF portable format. PnetCDF is widely used in the weather and climate communities, among other fields, for its high performance and standard format. Both computing facilities and vendors recognize PnetCDF's role in computational science and provide it as a prebuild module. The ECP included PnetCDF in the E4S software collection due to its importance.

Libraries such as HDF, ROMIO, and PnetCDF have a significant history with over a decade of production use, yet significant changes were needed to address the scale, heterogeneity, and latency requirements of upcoming applications. This motivated the targeting of specific use cases in pre-exascale systems, such as E3SM that require output at scale using the netCDF-4/PIO/PnetCDF preferred code path. The PnetCDF team continues to address concerns, to maintain portability and performance, and may develop new capabilities as needs arise.

The Solution

The ECP PnetCDF development has taken place over the past 5 years with a level of effort sufficient to demonstrate performance and scalability as an integrated software component on new HPC platforms. The project has focused on ensuring broad use, along with assisting in performance debugging on new platforms and augmented existing implementations to support new storage models (e.g., burst buffers).

PnetCDF can be directly called by applications and via the ROMIO MPI-IO implementation, or indirectly as an integrated part of popular libraries such in HDF5 and netCDF-4. Under the ECP, PnetCDF has developed several new capabilities:

- Significant performance enhancements to both PnetCDF itself and the ROMIO MPI-IO implementation.
- New "burst buffer" feature to use novel storage hierarchies.
- Interoperability with HDF5 through a VOL component allowing HDF5 applications to read NetCDF formatted datasets.

Capability Development and Approach

The PnetCDF project followed two integration approaches. First, as a portable and widely used library, PnetCDF is routinely installed by facilities as part of the default software environment. PnetCDF's facility and vendor partners recognize its utility to multiple applications. With PnetCDF's focus on portability and minimal external dependencies, operators can typically install it with only a small amount of time and effort. Second, PnetCDF has a long track record of responsiveness and community involvement.

Consistent with this approach, PnetCDF has integrated with the following clients:

- Characterizing PnetCDF in ECP applications via Darshan. Darshan is a lightweight I/O characterization tool that captures concise views of HPC application I/O behavior.
- Integration into the Frontier exascale supercomputer software stack.
- Integration into the Sunspot software stack (as proxy for the Aurora exascale supercomputer).
- Integration into E3SM, which is a fully coupled, state-of-the-science Earth system model capable of global high-definition configuration. The E3SM code has long relied on PnetCDF.

The Impact

The impact can be recognized via these integration efforts:

- **Darshan**: The Darshan integration project enabled a much richer data capture capability via an enhanced PnetCDF module implementation. In particular, the DataLib team implemented fine-grained metrics for data I/O using the PnetCDF format and associated library. This provided important performance insight to users of PnetCDF, which is a frequently used data format in the ECP ecosystem, particularly for atmospheric modeling.
- **Frontier**: Facilities and their users value PnetCDF highly enough to include it in the software environment on the Frontier platform. This integration deploys PnetCDF on Frontier, making it available to ECP clients and to clients outside and beyond the ECP.
  PnetCDF also provides a test suite for verifying previously installed versions of itself. The team documented correct behavior with the installed PnetCDF library but also identified (and worked around) an issue in the Lustre file system on Frontier. The PnetCDF validation suite packages the 105 (as of this writing) C and Fortran tests to create and read NetCDF datasets under various conditions. The suite targets these tests at existing installations of PnetCDF. The tests check whether PnetCDF functions run and compare data with known-good values.
- **Sunspot**: Similar to the benefits for Frontier using proxy hardware for the Aurora exascale supercomputer.
- **E3SM**: E3SM developers regularly run climate experiments on cutting-edge platforms. They are now running on Frontier using PnetCDF as the default module.

Sustainability

The integration efforts demonstrate that PnetCDF will continue to be used successfully by application and library teams.

- **Darshan**: the new PnetCDF capabilities have been integrated into the Darshan repository as of version 3.4.1, and they can be included in a Darshan build using standard Spack parameters. Since PnetCDF is a heavy-weight library, it is appropriate to have this capability integrated as an optional extension, which preserves the option for a leaner and simplified Darshan build.
- **Frontier**: Extensive performance evaluation has been reported on Frontier supercomputer at scale, comparing performance of HDF VOL against PnetCDF and the traditional HDF IO system. The PnetCDF Darshan capabilities are accessed via an lmod module. Multiple versions are available, which demonstrates continued update and deployment activities.
- **Sunspot**: As with Frontier, the PnetCDF Darshan capabilities are accessed via an lmod module. The software passed the PnetCDF correctness testing, indicating it represents a viable capability.
- **E3SM**: By making the PnetCDF integration a default I/O on Frontier, the team has the potential to get adoption from diverse user communities on Frontier. In addition, the performance data being automatically ingested into PACE enables the possibility of continuous performance improvement in a way transparent to the users.

For more information and references:

- PnetCDF can be installed though E4S binaries, containers, or via custom source code builds via SPACK: https://e4s.io.
- http://cucis.ece.northwestern.edu/projects/PnetCDF .
- Project GitHub repository: https://github.com/Parallel-NetCDF/PnetCDF/
- Project website: https://parallel-netcdf.github.io/
- See the ECP DataLib project for more data related ECP projects. An overview can be found in "Datalib Innovates with Some Of The Most Successful Storage and I/O Software In The DOE Complex".

PowerStack – A Vendor-Neutral Solution to Access the Power and Performance Capabilities of Low-Level Hardware

The Motivation

An HPC system's power, performance, and scientific throughput are affected by configurations and requirements at different levels of the system, such as job schedulers, runtime systems, and the tuning of the applications themselves. Each level presents different challenges and opportunities for optimization.

The Exascale Computing Project (ECP) PowerStack effort was motivated by the needs of supercomputer users who are not familiar with low-level architecture details across different vendors, and some power and performance dials require the user to have elevated privileges. As a result, accessing power and performance optimization features that are complex and vendor specific can be chaotic, unwieldy, and error prone from the users' perspective.

The Solution

PowerStack developed hierarchical interfaces for power management at three specific levels: batch job schedulers, job-level runtime systems, and node-level managers. Each level in PowerStack provides options for adaptive and dynamic power management depending on the requirements of the supercomputing site under consideration.

PowerStack relies on Variorum—an extensible, open-source, vendor-neutral library for exposing power and performance capabilities of low-level hardware dials across diverse architectures in a user-friendly manner.

The Impact

Variorum provides vendor-neutral APIs such that the user can query or control hardware dials without needing to know the underlying vendor's implementation (e.g., model-specific registers or low-level sensor and GPU interfaces). These APIs permit integration with higher-level system software such as schedulers and runtime systems. The HPC PowerStack Initiative uses these APIs to better manage power, energy, and performance of diverse HPC architectures through various metrics.

These APIs enable application developers to gain a better understanding of power, energy, and performance through various metrics. Additionally, the APIs enable system software to control hardware dials to optimize for a particular goal when integrated with a software stack comprising Flux, PowerAPI, Caliper, Kokkos, LDMS, and others.

Sustainability

The HPC PowerStack community meets regularly, details of which can be found at their website. This ensures user engagement.HPC PowerStack also closely collaborates with the following community efforts:

- PowerAPI
- EEHPC WG

To support general sustainability, PowerStack (Variorum) has been integrated into E4S and has a Spack package. Numerous Variorum integrations assist in growing both adoption and a user community.

- Variorum integrations: Flux Power Manager: https://github.com/flux-framework/flux-power-mgr/
- PowerAPI and Variorum integration: https://github.com/LLNL/variorum-integrations/tree/powerAPI/pwrAPI/

- Caliper service: https://github.com/LLNL/Caliper/tree/master/src/services/variorum/
- Kokkos Variorum plug-in: https://github.com/kokkos/kokkos-tools/tree/develop/profiling/variorum-connector/
- GEOPM Integration of Variorum: https://github.com/amarathe84/geopm/
- LDMS Variorum sampler: https://github.com/ovis-hpc/ovis/tree/OVIS-4/ldms/src/contrib/sampler/variorum_sampler/

For more information and references:

- PowerStack can be installed though E4S binaries, containers, or via custom source code builds via SPACK: https://e4s.io.
- https://variorum.readthedocs.io/en/latest/PowerStack.html
- Variorum repository: https://github.com/LLNL/variorum/
- msr-safe: https://github.com/LLNL/msr-safe/
- Variorum Spack mirrors: https://github.com/LLNL/variorum-spack-mirrors/
- The ECP podcast, "Efficiently Using Power and Optimizing Performance of Scientific Applications at Exascale"

RAJA: Implementing C++ Portability for High Performance Computing

Exascale computing systems—high performance computers that can run at least 10^{18} calculations per second, like Aurora at Argonne National Laboratory, Frontier at Oak Ridge National Laboratory, and the upcoming El Capitan at Lawrence Livermore National Laboratory—require specialized software. In other words, if a programmer writes code for a standard HPC system, it does not easily transfer up to systems at the exascale level. This ends up presenting complex challenges and a very large time investment for developers, who may need to rewrite their code.

RAJA, a software technology developed by the Exascale Computing Project (ECP), helps programmers with necessary code transformations. RAJA allows developers to write their code in C++—a versatile and commonly used programming language—to recompile their code to run on different high performance computing systems without major changes to the source code. This provides the developers with flexibility: They can use a language familiar to them and only consider the backend details of their system to the extent that is necessary for advanced performance tuning. For developers, it's simply a matter of substituting a C++ language type in their code with another, indicating which system it will be running on.

Used in at least eight different ECP projects and many other Department of Energy laboratory applications, RAJA provides a significant return on investment when preparing codes for new computing platforms. In some cases, RAJA has enabled a more than 20-fold reduction in application preparations for Lawrence Livermore's upcoming El Capitan supercomputer.

RAJA will continue to be developed and widely accessible beyond the closeout of ECP and will continue its impact on computing platforms within the Department of Energy. Its use for educational purposes in university coursework teaching parallel programming will also help train the next generation of computing experts.

SLATE – A GPU-Enabled Replacement for the Venerable ScaLAPACK Library

The Motivation

SLATE (Software for Linear Algebra Targeting Exascale) provides fundamental dense linear algebra capabilities including parallel BLAS (basic linear algebra subprograms), norms, linear system, least squares, singular value, and eigenvalue solvers. These dense linear algebra routines are a foundational capability used in many science and engineering applications and are used as building blocks in other math libraries. As such, SLATE seeks to provide a complete linear algebra library offering many capabilities that diverse clients can build upon.

Recognizing the inherent challenges of designing a software package from the ground up, the SLATE project started with (1) a careful analysis of the existing and emerging implementation technologies,(2) followed by an initial design that (3) has solidified. The team will continue to reevaluate and refactor the software as needed.

This motivated the design, implementation, and deployment of software that:

- Aggressively performs GPU offloading: Virtually all the processing power in modern systems is provided by GPUs. Achieving efficiency requires aggressive offload to GPU accelerators and optimization of communication bottlenecks. Where applicable, highly optimized vendor implementations of GPU operations are used. Where necessary, custom GPU kernels are developed. Communications operations are overlapped with computation to increase performance whenever possible. Attention has been paid to GPU memory usage as SLATE's target applications generate large dense matrices to solve, with dimensions exceeding 100,000, often exceeding the memory of a single node.
- Preserves flexibility: The team recognized that standardized solutions for GPU acceleration are still evolving and are in a state of flux. SLATE uses modern C++ features and recent extensions to the OpenMP standard, which may not be fully supported by compilers and their runtime environments. The team also optimized existing matrix-multiplication routines to automatically use mixed-precision iterative refinement algorithms when appropriate.

The Solution

The ultimate objective of SLATE is to replace the venerable ScaLAPACK (Scalable Linear Algebra PACKage) library, which has become the industry standard for dense linear algebra operations in distributed memory environments but is past its end of life and can't be readily retrofitted to support GPUs. Primarily, SLATE aims to extract the full performance potential and maximum scalability from modern multi-node HPC machines with many cores and multiple GPUs per node. This is accomplished in a portable manner by relying on standards such as MPI and OpenMP.

SLATE also seeks to deliver dense linear algebra capabilities beyond the capabilities of ScaLAPACK, including new features such as communication-avoiding and randomized algorithms, as well as the potential to support variable size tiles and block low-rank compressed tiles.

To be as widely available as possible, SLATE provides several interfaces, including a native C++ interface, C and Fortran interfaces, and a ScaLAPACK compatible wrapper. All the libraries have Spack, CMake, and makefile builds and are in the E4S and xSDK distributions to ease integration with applications and facilities.

The Impact

SLATE improved performance of several major routines, including Cholesky, QR, eigenvalue, and singular value decompositions. More components of Cholesky and QR are now on the GPU. For the eigenvalue problem, the team parallelized a major component and accelerated it using GPUs, resulting in significant improvements. The team also added computation of eigenvectors to the eigenvalue solver and are working on the divide-and-conquer algorithm, which exhibits better performance than the QR iteration algorithm for computing eigenvectors.

SLATE has been integrated with the following clients:

- STRUMPACK is a library for dense and sparse linear algebra using hierarchical rank-structured matrix approximations.
- WarpX enables computational explorations of key physics questions in the transport and acceleration of particle beams in long chains of plasma channels. PICSAR is a library of modular physics routines for PIC codes.
- NWChemEx is a widely used open-source computational chemistry package that includes both quantum chemical and molecular dynamics functionality.
- BLAS is a collection of low-level matrix and vector arithmetic operations. LAPACK is a library of dense and banded matrix linear algebra routines such as solving linear systems.

Sustainability

Sustainability is achieved through a flexible design and community engagement. The SLATE team interacts on a regular basis with the OpenMP community, represented in ECP by the SOLLVE project, and with the MPI community, represented in ECP by the OMPI-X project and the Exascale MPI project. The SLATE team also engages the vendor community through contacts at HPE Cray, IBM, Intel, NVIDIA, AMD, and ARM.

A well-established community of ScaLAPACK users effectively guarantees an extensive user community. The SLATE team also packaged the BLAS++ and LAPACK++ portability layer as stand-alone libraries for other applications to leverage for portability and modern C++ semantics.

For more information and references:

- SLATE can be installed though E4S binaries, containers, or via custom source code builds via SPACK: https://e4s.io.
- http://icl.utk.edu/slate
- SLATE: Design of a Modern Distributed and Accelerated Linear Algebra Library
- Portable and Efficient Dense Linear Algebra in the Beginning of the Exascale Era

SOLLVE: Compiling Code for Exascale Computers

All modern computing machines parallelize tasks—split and distribute them among multiple processing elements simultaneously—to run more quickly. However, most legacy code is written in older programming languages that do not intrinsically support parallel expression, such as C and older versions of C++ and Fortran. Furthermore, most scientific source code is written in a way that does not automatically permit safe parallel execution. Real-world scientific applications, such as biological models and solar physics, require code augmentation to run in parallel, especially on GPU-based supercomputer systems.

OpenMP is a community standard and programming system that helps enhance codes to enable parallelization, allowing them to run at scale on large systems. However, the challenge remains to compile this code to run on supercomputers.

The Exascale Computing Project (ECP) software project SOLLVE helps address this issue. SOLLVE promotes advances in the OpenMP standard and creates compiler implementations for OpenMP to help enable these large, historic scientific programs to run on supercomputers, providing solutions to exascale challenges. As OpenMP syntax modifies C/C++/Fortran code and SOLLVE-developed tools compile the enhanced OpenMP code—in other words, translates it into zeros and ones that a computer can understand—the result brings frequently used legacy code into the future of computing.

SOLLVE is a community-driven project. The team works with application developers to help determine their needs in order for their code to run better, faster, and more efficiently on exascale systems. Before an OpenMP specification is formed, any desired new features must be voted on, ratified, and approved by the OpenMP community, of which SOLLVE is an integral part.

For OpenMC—an ECP Monte Carlo neutron and photon transport code that aims to model the entire core of a nuclear reactor—SOLLVE helped deliver key kernel performance on multiple GPU platforms without user code changes via OpenMP GPU support capabilities. This is just one example of the community-driven implementation improvements SOLLVE has enabled. SOLLVE tools are available to any interested application teams. SOLLVE tools have been well-received by the scientific community, and SOLLVE's customer focus makes its tools crucial for industry vendors, who leverage SOLLVE's investments within their own tools. SOLLVE's primary goal is to help drive OpenMP design and adoption to optimize its functionality for exascale systems. SOLLVE will continue to provide critical capabilities beyond ECP.

Spack – The Essential Tool Used to Build the ECP Software Stack on All Target Platforms

The Motivation

Spack is a package manager for building and installing software on systems ranging from Linux and macOS laptops to exascale supercomputers. It was created to make HPC software complexity manageable, and allow users to easily build and install optimized, reproducible, and reusable HPC software. Spack has a broad appeal as it isn't tied to a particular language; users can build a software stack in Python or R, link to libraries written in C, C++, or Fortran, and easily swap compilers or target specific microarchitectures.

The Exascale Computing Project (ECP) adopted Spack as the standard tool to build the ECP software stack on all target platforms. (See E4S for more information about the Extreme Scale Software Stack and its use in scientific computing, industry and the cloud.) This decision was motivated by the observation that the space of possible ways to build software is combinatorial in size, and software reuse is hindered by the complexity of integrating a large number of packages and by issues such as binary compatibility. Building modern software to run correctly and with the highest possible performance on a system is a difficult task. It simply is no longer practical for users to manually download the source code for the packages they wish to install — including all the associated software that must be built and installed to get the software to run — and then iteratively build, debug, and manually tune all this software (potentially many packages) to achieve high performance.

Spack bridges this divide through automation and portable package recipes; specialized packages can be built per-site if needed, or lowest-common denominator packages can be built for those cases that do not need highly optimized performance. Packages are relocatable and can be used outside their original build environment. They can also be installed directly, without building, from the Spack binary cache.

These reasons motivated the ECP to adopt this essential technology to provide a packaging solution that can deploy on bare metal, in a container, or be rebuilt for a new machine on demand. Moreover, Spack provides environments that enable a number of software packages to be deployed together either on an HPC system, in the cloud, or in a container.

The Solution

Spack automated much of the package build and deploy process. This means that collections of packages can be built consistently and compatibly from source and for multiple configurations including different package versions and combinations of versions; different compilers and operating systems; different low-level support libraries for the CUDA, HIP, and SYCL GPU layers; MPI libraries such as Open MPI and MPICH; and low-level mathematical libraries such as BLAS, LAPACK, and solvers.

The Spack solution has three main components.

1. The core tool encodes a complex model of software that understands not just fixed, scripted configurations of packages but a constraint model for ensuring their compatibility when combined.
2. Spack's so-called concretizer allows developers to provide an abstract specification of the package they want to build and then fills in the rest of the build parameters for that package to ensure a consistent build compatible with application binary interfaces.
3. Spack is also a library of over 7,400 open-source package build recipes with over 1,200 contributors from around the world. These contributors push 400–600 contributions per month to GitHub. Spack is also a cloud service for CI. As commits come in from contributors on GitHub, a cloud-based CI system ensures that recipes continue to work in all the required configurations.

The Impact

Spack has broad adoption in the open-source community as an elegant solution toward solving many of the challenges presented by building software with many dependencies. Within the ECP, Spack plays a central role in the software development and delivery processes by supporting turnkey builds of the ECP Software Technology (ST) software stack for the purposes of continuous integration testing, installation, and seamless multiproduct builds. Spack's cloud CI system runs over 115,000 builds per week—both in Amazon Web Services instances and on the U. The Oregon Frank cluster comprises a variety of the ECP-relevant GPU nodes that can be used for testing.

The Spack development, maintenance, and testing effort has been ongoing throughout the life of the ECP. All of the L4 ECP projects have both an entry in the E4S test suite as well as 'smoke test' functionality in their Spack packages. These testing modalities are automatically executed by the E4S team when assessing releases. The Spack smoke tests are also executed by the Spack CI pipeline when evaluating pull requests for a package as well as part of preparation for a Spack release.

The E4S Spack Build Cache now includes binaries for ppc64le as well as x86_64 and includes over 53,000 total binaries. The E4S containers now support custom images for ECP applications, such as WDMApp, ExaWind, and Pantheon. A regular cadence of E4S releases will continue, with updated ST products, broader facility adoption, and inclusion in vendor offerings.

Sustainability

Spack continues to gain penetration across the ECP, and is the de facto delivery method for ECP ST products building from source. The ECP provided Spack packaging assistance for ST users and DOE facilities and has developed new capabilities for Spack that enable automated deployments of software at facilities, in containerized environments, and as part of continuous integration.

The E4S project continues, but funding is needed to continue this essential service.

For more information and references:

- SPACK can be installed though E4S binaries, containers, or via custom source code builds via SPACK: https://e4s.io.
- https://github.com/spack/spack

STRUMPACK/SuperLU: Portable Solvers for Large Linear Systems

Large systems of linear equations are key kernels in a variety of scientific projects. These are analogous to the systems of equations solved in basic algebra—two or three equations describing known relationships between various factors, with the same number of unknown variables that need to be resolved—but scaled up to millions, making them impossible to solve by hand and difficult to solve computationally.

Because scientific discoveries depend so heavily on these equations, many scientific computing applications require a method for solving them. Instead of reinventing the wheel for each specific use case, the Exascale Computing Project (ECP) invested in developing a set of products that can be reused and benefited from by many: the STRUctured Matrix PACKage (STRUMPACK) and Supernodal LU package (SuperLU). Together, STRUMPACK/SuperLU provide fast and efficient direct solving and preconditioning capabilities for large-scale, complicated sets of linear equations. Their methodologies enable vast memory savings compared to other similar software.

STRUMPACK/SuperLU act as building blocks within a larger, interconnected set of mathematical tools. As large computational problems are broken down into smaller pieces, STRUMPACK/SuperLU take care of a specialized portion. Other mathematical libraries then rely on these results to perform their own tasks.

STRUMPACK/SuperLU are open source and portable, and can be used with a variety of different central processing unit (CPU) and graphical processing unit (GPU—a type of accelerator that helps with computational efficiency) types, as well as on different operating systems. Their capabilities have been utilized by the fusion community, engineers, and more, overcoming limitations of other solvers. For example, SuperLU provided a key capability for fusion energy reactor design in plasma simulations, and STRUMPACK was used in MFEM, an ECP-developed library for solving partial differential equations—such as the very challenging three-dimensional Maxwell equations. The speed at which SuperLU and STRUMPACK solve these difficult problems can be further enhanced by up to a factor of three to ten, respectively, when GPUs are added. These critical functionalities and are expected to continue to be used well past ECP.

Sundials and hypre: Accelerated Mathematical Libraries for Science and Engineering

The Motivation

Scientific and engineering computing often requires the application of sophisticated mathematical techniques to solve systems of equations that reflect or model something of interest in the real world. Part of the magic of mathematics is that the same or similar mathematical techniques can work with systems of equations for wildly disparate physical phenomena. Well-designed libraries make these techniques (many of which are non-trivial to implement) available for use by scientists and engineers in many fields. These libraries have a significant impact on application portability and performance as the algorithms and software implementation act as the gateway middleware that dictates what hardware can be used and how efficiently an application can use its capabilities.

The ECP SUNDIALS and hypre mathematical libraries are two well-designed, robust libraries that have been implemented, and verified using modern software techniques. They give scientists and engineers the numerical methods they need to solve problems efficiently on a plethora of state-of-the-art hardware on systems ranging from laptops to exascale supercomputers.

The Solution

Scientists often define equations in terms of the rate of change of something with respect to time, where time is considered the independent variable. Such problems are expressed as ordinary differential equations (ODEs) where the term *ordinary* is used to indicate functions of one independent variable and the derivatives of those functions. The SUNDIALS project focuses on delivering adaptive time-stepping methods for dynamical systems which permit bigger time steps when timescales are longer and smaller time steps when timescales are shorter. The ability to adapt the timestep can both increase accuracy and deliver a faster time to solution.

This contrasts with another broad class of equations referred to as partial differential equations (PDEs) which are used to describe change with respect to more than one independent variable. Scientists use the high-performance preconditioners and solvers in the hypre library to solve large, sparse linear systems of equations on massively parallel computers. This versatile library with a well-designed API can be used to solve many types of linear systems. Common examples include finite-element and finite-difference methods that are applied to scalar PDEs (i.e., one unknown per grid point) or systems of PDEs that contain multiple variables per grid point.

The Impact

The SUNDIALS and hypre libraries give users the ability to exploit the performance and power efficiency of accelerators to run experiments on laptops, large and ensemble production runs on tera- and peta-scale systems, and when needed, at scale on the largest exascale systems. This efficiency can reduce runtime and gives users in industry and the general HPC community the ability to run on the most cost- and power-efficient hardware platform. The broad hardware support gives organizations the ability to pick and choose the best platform that suits their needs and budget, be it on-premises or in the cloud.

Sustainability

Through community efforts and good software design, the SUNDIALS and hypre mathematical libraries continue their proven track record in supporting application developers through generations of increasingly more performant and

scalable hardware platforms — usually without requiring application code modifications. With ECP support, these libraries have been modernized and are now accelerator and exascale capable.

For more information and references:

- Install via binaries, containers, or source code builds via SPACK: https://e4s.io
- Sundials and hypre: exascale-capable libraries for adaptive time-stepping and scalable solvers
- See also: ECP Mathematical Libraries

SZ – An Award-Winning Lossy Compressor for Data Reduction with User Settable Error Bounds

The Motivation

Increased data size due to higher performance computing systems and updated scientific instruments has motivated the development of error-bounded lossy compressors for scientific datasets. GPU acceleration has commoditized high-performance hardware, and the advent of exascale supercomputers means that computational power can now exceed I/O performance by orders of magnitude. This motivated the development of innovative approaches to data as even commonly used checkpoint/restart operations, which act as a safeguard against hardware failure in distributed systems, are challenging to implement cost effectively at scale.

Lossy compression can be an acceptable solution for many scientific applications when:

- the compression only removes information that does not impact scientific discovery,
- the compression and decompression operations are fast enough to avoid raising a performance issue, and,
- the lossy compressor needs to be effective at providing much higher data reduction ratios than lossless compression.

The Solution

The Exascale Computing Project (ECP) VeloC-SZ project focused on the SZ lossy compressor because it offered an excellent compression ratio and very low distortion and runtime. This motivated further work to improve this software-based compressor to better support ECP scientific datasets. These improvements were performed while ensuring that user-set error controls were respected, including during the refactoring process to turn the SZ codebase into production-quality software package that could be integrated into leading I/O libraries.

The ECP contributions to SZ include (1) optimized SZ compression ratios (improved by up to a factor of 6 compared with the initial algorithm) with accuracy based on end-user needs; (2) improved compression speed (up to 3 orders of magnitude) when using the GPU.

implementation of SZ, which supports multiple supercomputers with different architectures (e.g., Aurora, Frontier, Summit); (3) refactoring of SZ in C++ to support a composable compression framework and all data types used in ECP applications (version 3.0 of SC received an R&D 100 award in 2021); (4) integration of SZ into the HDF5 and ADIOS I/O libraries; and (5) improved SZ code robustness and testability to make it production ready.

The Impact

The ECP SZ project multifaceted approach successfully targeted integration into science applications and I/O libraries such as HDF5 and ADIOS. This made SZ available to a broad range of potential clients both inside and outside of the ECP community. The 2021 R&D 100 award reflects recognition by the global scientific community on its usefulness.

Sustainability

Integration of the production quality code into popular I/O libraries such as HDF5 and ADIOS makes the SZ compressor available to many users and ensures a continued and long term userbase. Direct client integrations of SZ include ECP applications HACC (cosmology), Nyx (cosmology), LAMMPS (molecular dynamics), and LCLS crystallography (x-ray light

source) reflect high-visibility application use cases. Both library and direct application integration successes demonstrate the long-term viability of the SZ compressor.

For more information and references:

- SZ can be installed though E4S binaries, containers, or via custom source code builds via SPACK: https://e4s.io.
- https://szcompressor.org/
- "Two MCS technologies win R&D 100 awards for 2021"

Trilinos – An Essential Set of Solver Capabilities for Scientific Computing on CPU and GPU Systems

The Motivation

Trilinos is a collection of open-source reusable software packages particularly for linear, nonlinear, optimization, and uncertainty quantification solvers. It is meant to provide building blocks for developing large-scale scientific applications. The project contains a large and growing collection of solver capabilities that can utilize next-generation platforms, in particular scalable multicore, manycore, accelerator and heterogeneous systems.

For portability reasons, the Exascale Computing Project (ECP) Trilinos project worked to suppress the use of Unified Virtual Memory (UVM). UVM is convenient as it provides a single memory address space accessible from any processor in a heterogenous system, be it a CPU or GPU. Suppressing the use of UVM in the Trilinos source code proved to be critical for portability as UVM is not available on all GPUs. Thus, any reliance on UVM precluded access to many academic, cloud and exascale platforms by applications that have incorporated Trilinos building blocks. This motivated and explains the extensive ECP effort to remove the UVM dependency from the Trilinos codebase. Note that Trilinos will still enable UVM when it is available.

The Solution

Eliminating the reliance on UVM was necessary for portability reasons, but it also provided two additional benefits: (1) it simplified debugging, which can be quite a difficult issue with UVM and (2) requiring explicit programmer memory management helped deliver more predictable performance. The tradeoff is additional programmer effort as managing memory efficiently in a heterogenous CPU and GPU computing environment can be complicated, delicate work.

The ECP primary development focused on the following four packages:

- Tpetra: distributed-memory matrix and graph operations
- Belos: distributed-memory iterative Krylov solvers
- Ifpack2: algebraic preconditioners (e.g., relaxation, one-level domain decomposition, incomplete factorization)
- Amesos2: direct solver interfaces (native Trilinos solvers such as KLU2 and Tachos and external solvers such as SuperLU, STRUMPACK, and Pardiso-MKL)

The Impact

The impact has been through the primary strategy of direct integration with science applications.
Along with removing the dependency on UVM, the team added a HIP backend to target AMD GPUs. This new capability allows the use of Trilinos solvers on machines with AMD GPUs as well as NVIDIA GPUs. The new HIP-enabled Trilinos solver components have been integrated into the public Trilinos Git repository and passed its tests.

Sustainability

The Trilinos building blocks are a core dependency for many scientific applications. Expanded support for new architectures ensures sustained, continued use by the large Trilinos user community. The refactored UVM-free code means that important components for linear solvers and ECP applications can run on GPU-accelerated hardware from multiple vendors, ensuring cross-platform portability and viability. Industry and cloud users along with the global HPC community benefit from this portability, which will help expand the global userbase.

For more information and references:

- Trilinos can be installed though E4S binaries, containers, or via custom source code builds via SPACK: https://e4s.io.
- https://trilinos.github.io/
- See also https://trilinos.github.io/ForTrilinos/ for portable object-oriented Fortran interfaces to Trilinos C++ packages.
- "Helping Large-Scale Multiphysics Engineering and Scientific Applications Achieve Their Goals"

Umpire: A Portable Library for Memory Resource Management

The Motivation

Umpire is a portable library for memory resource management. It provides a unified, high-level API in C++, C, and Fortran for resource discovery, memory provisioning, allocation, transformation, and introspection.

The Exascale Computing Project (ECP) developed Umpire to target the porting issues faced by legacy codes. Other projects that address this porting issue include the ECP's RAJA and Copy Hiding Application Interface (CHAI) projects. Where other performance portability frameworks may require a larger up-front investment in data structures and code restructuring, Umpire is noninvasive and allows codes to separately adopt strategies for loop parallelism, data layout tuning, and memory management. Legacy applications need not adopt all three at once; they can gradually integrate each framework at their own pace and with a minimal set of code modifications.

The Solution

Umpire leverages the abstraction mechanisms available in modern C++ (C++11 and higher) compilers, such as lambdas, policy templates, and constructor/destructor patterns (e.g., resource acquisition is initialization) for resource management. The objective is to provide performance portability at the library level without special support from compilers.

Targeting this level of the software stack gives US Department of Energy developers the flexibility to leverage standard parallel programming models, such as CUDA and OpenMP, without strictly depending on robust compiler support for these APIs. If the necessary features are unavailable in compilers, then library authors need not wait for these programming models to be fully implemented. These libraries allow applications to work correctly and to perform well even if some functionality from OpenMP, CUDA, the threading model, and other models is missing.

This capability is possible because a flexible allocation process supports the Umpire user interface. The interface is the same regardless of which resource is housing the memory. Key operations are supported. For example, memory can be managed and viewed. An abstract operations interface is provided for modifying and moving data between Umpire and allocators, and custom algorithms can be applied with code-specific strategies.

The Impact

Accessing the parallelism of the hardware platform is essential for achieving high performance on today's hardware platforms.

Sustainability

Umpire is open source. It also provides user guides and tutorials.

Additional Information and References

- Umpire installation through Extreme-Scale Scientific Software Stack binaries, containers, or custom source code builds via Spack: https://e4s.io
- Umpire GitHub repository: https://github.com/LLNL/Umpire

UPC++ and GASNet-EX Deliver Increased Performance Through Reduced Communication Costs

The Motivation

UPC++ is a C++ library supporting Partitioned Global Address Space (PGAS) programming. The PGAS programming model provides an alternative to the popular Message Passing Interface (MPI) programming model. PGAS offers significant advantages in performance as well as programmer productivity. These advantages translate into a programming model and communications library that can scale efficiently to potentially millions of processors, while still delivering high-performance on smaller platforms.

UPC++ leverages the underlying GASNet-EX communication library to deliver efficient, low-overhead Remote Memory Access (RMA) and Remote Procedure Call (RPC) on HPC systems. Thus, the ECP UPC++ software can make use of modern system capabilities such as Remote Direct Memory Access (RDMA) offload capabilities and native on-chip communication between distinct address spaces.

The Solution

The ECP project focused on three guiding principles:

1. All communication is asynchronous, allowing overlap of computation and communication, and encouraging programmers to avoid global synchronization.
2. All communication is syntactically explicit, encouraging programmers to consider the costs of communication.
3. UPC++ encourages the use of scalable data-structures, avoiding non-scalable library features.

The ECP GASNet-EX effort updated the 20-year-old GASNet-1 PGAS codebase and communication system. The ECP solution included an implementation overhaul along with this major redesign of the GASNet-1 software interfaces.

UPC++ provides the high-level productivity abstractions appropriate for PGAS programming such as: remote memory access (RMA), remote procedure call (RPC), support for accelerators (e.g., GPUs), and mechanisms for aggressive asynchrony to hide communication costs.

The Impact

UPC++ has proved it is able to provide efficient, low-overhead RMA and RPC on HPC systems. This includes accelerated transfers to and from GPU memory. Demonstrations show that the UPC++ library delivers robust performance scalability, even on large modern supercomputers. Near-linear weak scaling on a distributed hash table is one example.

Numerous application use cases demonstrate that UPC++ can deliver high performance and portability on systems ranging from laptops to exascale supercomputers. Examples include: MetaHipMer2 metagenome assembler, SIMCoV viral propagation simulation, NWChemEx TAMM, and graph computation kernels from ExaGraph. See the Berkeley Lab upcxx wiki for a list of notable applications/kernels/frameworks using UPC++.

Sustainability

UPC++ enables the programmer to compose the powerful productivity features of the C++ standard template library seamlessly with RPCs to help the programmer create understandable and maintainable code. This includes the use of C++ lambda expressions which have many programming advantages.

Ease of use and proven application performance means that UPC++ and the GASNet-EX communications library will maintain a large application portfolio and user base in the future.

For more information and references:

- UPC++ and GASNet-EX can be installed though E4S binaries, containers, or via custom source code builds via SPACK: https://e4s.io.
- "Pagoda updates PGAS programming with scalable data structures and aggressively asynchronous communication"
- https://bitbucket.org/berkeleylab/upcxx/wiki/Home
- https://docs.nersc.gov/development/programming-models/upcxx/
- https://gasnet.lbl.gov/dist-ex/README

VeloC-SZ – A Lossy Compressor for Structured and Unstructured Scientific Datasets

The Motivation

The VeloC-SZ project is extending and improving the SZ error-bounded lossy compressor for structured and unstructured scientific datasets. SZ offers an excellent compression ratio and very low distortion and compression time. This additional work was motivated by the need to improve the SZ lossy compressor for Exascale Computing Project (ECP) scientific datasets, thereby ensuring that user-set error controls are respected, integrating SZ into the leading I/O libraries, and delivering production-quality software.

The Solution

Specifically, ECP contributions to SZ include (1) optimized SZ compression ratios (improved by up to a factor of 6 compared with the initial algorithm) and accuracy based on end-user needs; (2) improved compression speed (up to 3 orders of magnitude) when using the GPU implementation of SZ, which supports multiple supercomputers with different architectures (e.g., Aurora, Frontier, Summit); (3) refactoring of SZ in C++ to support a composable compression framework and all data types used in ECP applications (the resultant SZ3.0 received an R&D 100 award in 2021); (4) integration of SZ into HDF5 and ADIOS; and (5) improved SZ code robustness and testability to make it production ready.

The Impact

SZ uses a multifaceted approach to integration, targeting integration into science applications and integration into I/O libraries such as HDF5 and ADIOS to make itself available to a broader range of potential clients after the ECP. Direct client integrations of SZ include ECP applications HACC (cosmology), Nyx (cosmology), LAMMPS (molecular dynamics), and LCLS crystallography (x-ray light source). The integration demonstrations span both AD clients and I/O library use cases.

See SZ products and their utilization on the SZ homepage. SZ received an R&D award in 2021.

Sustainability

SZ is also available through the DAV software development kit and through E4S, enabling users to use SZ on an HPC platform.

For more information and references:

- VeloC-SZ can be installed though E4S binaries, containers, or via custom source code builds via SPACK: https://e4s.io.
- SZ lossy compression website, which provides numerous links to SZ products and their utilization.

VTK-m: Enabling Parallelism for Scientific Visualizations

Scientific visualizations are a primary mechanism for scientists to understand and share their results. However, most tools developed prior to the exascale era of computing do not properly scale up to high performance computing (HPC) systems. As HPC systems grow to use more accelerators, applications require a larger degree of parallelism in order to take full advantage of these systems. Current visualization applications cannot achieve this necessary level of parallelism.

The Exascale Computing Project (ECP) software VTK-m fills in this critical gap. VTK-m takes care of all back-end operations that need to occur to enable scientific data to be rendered into a visual form. The application iteratively builds upon pre-existing visualization tools to take advantage of processing technologies on HPC systems, allowing visualization applications to run more computations per compute node. Users also have the option to integrate VTK-m into a rendering program of their choice or use its own rendering capabilities.

Currently used worldwide both as a standalone product and with other Department of Energy products—such as ParaView, VisIt, and Ascent—VTK-m is extremely versatile. It can be used with different processor architectures, embedded in other visualization solutions, and applied directly to scientific simulations. For example, the ECP project Whole Device Model Application (WDMApp) used VTK-m to design Poincare maps—plots of magnetic fields around a tokamak, which require thousands of iterations—for fusion simulations. This cut down their visualization times from hours to minutes and allowed scientists to obtain more granular views of the magnetic fields over time across the entire simulation.

With its critical functionality and demonstrated usability with other tools, VTK-m software is poised to continue seeing utilization beyond ECP. The team hopes its user base will continue to grow as scientists use VTK-m as a valuable platform for future research.