| title | authors | keywords | abstract |
|---|---|---|---|
| Automated Acceptance Testing in HPC with buildtest | Shahzeb Siddiqui, Erik Palmer, Justin Cook and Wyatt Spear | HPC Testing<br>DOE-Facilities<br>Spack | Supercomputers are complex systems where software stacks are deeply integrated with hardware. During their service life, these systems undergo numerous software updates and hardware changes which can result in an unstable system. After each set of changes is made, testing is necessary to detect lingering issues and regain confidence in the system. Buildtest provides a thorough and consistent testing framework designed to optimally address this issue in high performance computing (HPC) environments. Support teams can leverage buildtest to painlessly write tests and automate the acceptance testing process. In this way, buildtest's comprehensive suite of tools simplifies stack maintenance and promotes user trust in the system.<br><br>The buildtest framework has several key features that support teams to create and run routine testing of large software stacks. In buildtest, tests are written declaratively in YAML, are known as buildspecs and validated using JSON schemas. Buildtest integrates with common HPC elements, such as batch schedulers, environment modules, the Spack package manager and the Cray Programming Environment. It can be run on login or compute nodes. Buildtest has several methods for discovering buildspecs based on filename, directory path, tag names and filter buildspecs during the build phase. Test results and corresponding metadata are stored in a JSON that can be queried for further analysis. Lastly, test reports can also be uploaded to CDash for viewing of test results in a web browser.<br><br>This tutorial will provide a functional understanding of buildtest with a mix of short lectures and hands-on activities. The lectures will present the essential concepts followed by hands-on activities to apply those concepts into practice. Through hands-on activities, attendees will learn basic usage of the buildtest command line interface, how to write their own tests, integrate tests from the Spack package manager, and run buildtest in an HPC environment. We will present the buildtest setup at NERSC, and how we leverage Gitlab CI/CD to automate execution of tests. In addition, we will share real test examples that are run on Perlmutter to help attendees understand types of test that one could typically run on their HPC system.<br><br>This tutorial will be run on NERSC systems, and we will provide attendees with training accounts. The attendee is required to bring a laptop in order to participate in the tutorial. One must have a terminal client in-order to SSH into the NERSC system. The attendee is expected to join the buildtest slack channel which will be used as the primary communication channel for the tutorial. |
| FFTX: Next-Generation Open-Source Software for Fast Fourier Transforms | Brian Van Straalen, Phil Colella, Franz Franchetti and Patrick Broderick | Fourier Transform<br>FFT<br>FFTX<br>FFTW | The FFTX project has two goals:<br>(1) The development of performance portable, open-source FFT software system for modern heterogeneous architectures (i.e. GPUs) to provide a baseline capability analogous to FFTW for CPU systems. (2) The support applications-specific optimizations corresponding to integrating more of the algorithms into the analysis / code generation process. Our approach is based on code generation using Spiral, an analysis and code generation tool chain for FFTs and tensor algebra algorithms; an FFTX user API implemented in standard C++; and a factored design that allows FFTX / Spiral to be more easily ported across multiple platforms.<br><br>In this tutorial we will describe the current progress of FFTX towards achieving goals (1) and (2). We will give an overview of the FFTX approach to obtaining performance portability; the current FFTX APIs for 1D and 3D FFTs on GPU systems, and how to use FFTX; and a discussion of the co-design process by which we undertaking to achieve our second goal. |
| Power Management on Exascale Platforms with Variorum | Tapasya Patki, Stephanie Brink, Aniruddha Marathe and Barry Rountree | power management<br>low-level knobs on AMD CPUs and GPUs<br>LDMS power plugin<br>Flux power<br>Caliper power | As we push the limits of supercomputing toward exascale, resources such as power and energy are becoming expensive and limited. The Argo project is building portable, open-source system software to improve performance and scalability with a focus on resource management, memory management, and power management. In this tutorial we focus on the power management component of Argo, aimed at developing an<br>end-to-end power management approach that includes node-level, job-level and cluster-level policy integration.<br><br>The focus of this tutorial will include be to provide an overview of power management knobs on pre-exascale and upcoming exascale systems. We will first discuss the challenges in power management and the supporting hardware features that can be leveraged by the HPC software stack. Then, we will introduce Variorum, a user-space, open-source vendor-neutral library for power management and control that supports Intel (CPUs and GPUs), AMD (CPUs and GPUs), IBM, Nvidia, and ARM platforms through a common API. The tutorial will cover details about supporting power management on upcoming El Capitan and Aurora systems. Finally, the tutorial will discuss how Variorum can be used by application developers for telemetry as well as control, including details of integrations with LDMS, Caliper, Kokkos and Flux. The tutorial will provide experience to developers and software tool developers as to how Variorum can be leveraged for optimization with adaptive system software and scientific workflows through demonstrations as well as hands-on experience. |
| Autotuning ECP codes using the GPTune package | Yang Liu, Younghyun Cho, Hengrui Luo and Xiaoye Li | autotuning<br>Bayesian optimization<br>transfer learning<br>crowd tuning<br>database | GPTune is a Bayesian-optimization-based performance autotuner that has been applied to many HPC codes including ECP math libraries: ScaLAPACK, PLASMA, SLATE, SuperLU_DIST, STRUMPACK, Hypre, MFEM, SUNDIALS and heFFTe, and application codes for fusion simulation, accelerator modeling and quantum chemistry calculation.<br><br>Compared to existing general-purpose autotuners, GPTune supports the following advanced features: (1) dynamic process management for running applications with varying core counts, and reverse communication-style interface to allow flexible MPI environments, (2) incorporation of coarse performance models and hardware performance counters to improve the surrogate model, (3) multi-objective tuning of computation, memory, accuracy and/or communication, (4) multi-fidelity tuning to better utilize the limited resource budget, (5) effective handling of non-smooth objective functions, (6) an advanced hybrid surrogate model for handling mixed-type tuning parameters, (7) transfer learning across multiple tasks and machines, (8) a crowd database with web interfaces and programmable APIs for performance data sharing and sensitivity/importance analysis.<br><br>In this tutorial, we will provide a brief overview of GPTune with hands-on exercises using a variety of representative examples illustrating the above-mentioned features of GPTune. |

| | | | |
|---|---|---|---|
| ExaWorks: Developing Robust and Scalable Next Generation Workflows Applications and Systems | Daniel Laney, Kyle Chard, Shantenu Jha, Justin Wozniak and Rafael Ferreira da Silva | Workflow<br>Applications<br>Machine Learning<br>Exascale<br>Automation | Workflows applications are critical to scientific discovery. Technology trends and the convergence of traditional High Performance Computing (HPC) with new simulation,<br>analysis, and data science approaches provides unprecedented opportunities.<br>Traditional approaches to workflow applications and systems development have<br>scalability and robustness limits. ExaWorks provides a robust SDK with well-defined and scalable component interfaces which can be leveraged by new and existing workflow applications and systems. ExaWorks will address sustainability and facilitate software convergence in the workflows community.<br><br>This tutorial will present the ExaWorks SDK, and its constituent components: Flux,<br>Parsl, RADICAL-Cybertools (RCT), and Swift/T. These components are widely used,<br>and available tools for developing workflow applications. This tutorial will outline today's most common workflow motifs on HPC platforms, illustrate science examples of these motifs, and discuss solutions using ExaWorks. 33% of our tutorial is dedicated to presentations from experts, and 67% to hands-on sessions. Attendees will gain practical knowledge to develop best workflow practices to manage large-scale campaigns on the largest supercomputers. At the end of the tutorial, they will be able to apply these tools and techniques to their advanced workflows with minimal programming effort. |
| CMake Tutorial | Zack Galbreath, Betsy McPhail and Markus Ferrell | cmake<br>build systems<br>testing | An interactive, half-day tutorial on writing and extending CMake build systems. Topics include:<br><br>* introduction to CMake: what problems it is meant to solve?<br>* how to get help with your CMake build systems<br>* how to use CMake<br>* variables and cache control<br>* building libraries<br>* "Modern CMake": specifying usage requirements on your CMake targets<br>* techniques to achieve faster build times<br>* installing artifacts from your project<br>* testing with CTest and CDash<br>* controlling the behavior of find modules<br>* using generator expressions<br>* importing and exporting CMake targets from your project<br>* MPI<br>* CUDA<br>* Fortran |
| Accelerate Your Application I/O with UnifyFS | Michael Brim, Cameron Stanavige, Kathryn Mohror, Sarp Oral and Ross Miller | I/O<br>application performance<br>file system | UnifyFS is a user-level file system that highly-specialized for fast shared file access on HPC systems with distributed node-local storage, e.g., burst buffers. In this tutorial, we will present users with an introductory overview of the lightweight UnifyFS file system that can be used to improve the I/O performance of HPC applications. We will begin at a high level describing how UnifyFS works with node-local storage and how users can incorporate it into their jobs. Following this, we will dive into more detail on what kinds of I/O UnifyFS currently supports and what we expect it to support in the future. We will also discuss the interoperation of UnifyFS with HDF5 and MPI-IO. |
| SuperLU and STRUMPACK: GPU accelerated sparse factorization solvers | Pieter Ghysels, Sherry Li and Yang Liu | solvers<br>preconditioners<br>sparse<br>dense<br>linear algebra<br>rank-structured<br>hierarchical matrix<br>compression | Matrix factorizations and the accompanying solution algorithms (e.g., triangular solution associated with the LU factorization) are often the most robust algorithmic choices for linear systems from multi-physics and multi-scale simulations. They are indispensable tools for building various algebraic equation solvers. They can be used as direct solvers, as coarse-grid solvers in multigrid, or as preconditioners for iterative solvers. On the software and implementation side, it is imperative to exploit multiple levels of parallelism presented by the heterogeneous node architectures through well orchestrated use of MPI, OpenMP and GPU programming frameworks such as CUDA/HIP/OneAPI and external libraries like MAGMA and SLATE. In this tutorial, we illustrate how, during the lifetime of the ECP project, we ported two sparse solver libraries, SuperLU_Dist and STRUMPACK, to GPU architectures from NVIDIA, AMD and Intel. We show that with careful implementation these solvers can achieve a significant fraction of the peak performance of the accelerators used in the current and planned leadership compute facilities.<br><br>Through hands-on exercises, the participants will learn how to use each solver most effectively for their target problems and parallel machines or specific accelerator. We also illustrate the use of SuperLU_Dist and STRUMPACK through different interfaces from other math libraries such as PETSc, Trilinos and MFEM.<br><br>As we are approaching the exascale computing era, demand for algorithm innovation is increasingly high. It is imperative to develop optimal-complexity scalable algorithms both in flop count and more importantly in data movement, such as, in the form of communication-avoiding formulations, and low-rank and butterfly compressions. We give a brief overview of the recent algorithmic innovations in SuperLU and STRUMPACK. |

| | | | |
|---|---|---|---|
| Steering Intelligent Workflows with ECP Toolkits | Logan Ward, Stephen Hudson, John-Luke Navarro, Shantenu Jha and Justin Wozniak | workflows<br>artificial intelligence<br>optimization<br>exaworks<br>candle<br>exalearn<br>petsc | Techniques and software for “dynamic” workflows - where the workload performed by a supercomputer is automated by algorithms that learn from emerging data - are growing in complexity and promise. Recent work from numerous ECP teams have demonstrated how coupling steering algorithms to conventional workflow systems can lead to improved scientific outcomes like faster protein folding, molecular discovery, hyperparameter optimization, and accelerator tuning. This tutorial will overview the algorithms and software behind these advances with a focus on the lessons learned in scaling these approaches and integrating them with scientific applications.<br><br>Our tutorial will spend 30 minutes each on tools developed by four different ECP subprojects, listed below. In each, we discuss their unique approaches to steering ensembles, what challenges they have encountered on the road to exascale, and the types of applications they support best. In total, we will introduce common patterns of “intelligent” workflows and showcase how they will shape science at the exascale.<br><br>Colmena (ExaLearn): Colmena is a Python library for expressing complex steering strategies as cooperative “agents.” Agents define event-driven strategies, such as “submit a machine learning training task once 10 simulations complete,” that control which work is being executed on one or more computing resources via ExaWorks workflow engines. Our tutorial will use examples of molecular design and water cluster optimization to explain how to design effective steering methods and scale workflows across thousands of nodes.<br><br>libEnsemble (PETSc): libEnsemble is a Python toolkit for coordinating asynchronous and dynamic ensembles of calculations. Users select or supply “generator” and<br>“simulation” functions that respectively produce candidate parameters and perform computations on those parameters. We will introduce libEnsemble’s generator/simulation interface alongside other critical capabilities like handling intermediate data and adaptively reallocating compute resources. We'll also discuss use cases that have driven these features and design decisions.<br><br>RASCAL (RADICAL Adaptive and Steered Campaigns to Accelerate Learning) Toolkit (ExaWorks): The RADICAL-Cybertools based RASCAL toolkit comprises several frameworks to support adaptive and steered computational campaigns. We will discuss (i) the methods and science use cases supported, (ii) outline the different frameworks developed to support them, and (iii) discuss the common and underpinning middleware that enables efficient execution at exascale.<br><br>Supervisor (Integration of ECP AD CANDLE and ECP ST ExaWorks):<br>The Cancer Distributed Learning Environment (CANDLE) is focused on scalable deep learning approaches to cancer |
| Julia programming for Exascale | Johannes Blaschke, William Godoy, Valentin Churavy, Pedro Valero-Lara, Philip Fackler, Hyun Kang, Youngsung Kim, Sarat Sreepathi, Michel Schanen and Jeffrey Vetter | Julia<br>Performance Portability<br>Workflows<br>LLVM<br>GPU/Accelerator Programming | Julia is an open-source programming language targeting scientific computing and data science with many abstractions built into the language and ecosystem. It has been gaining popularity since its first stable release in 2018 with the promise of being an approachable language that compiles to efficient native code via LLVM, while providing rich capabilities for data analysis and visualization.<br><br>In this tutorial, we will first present a gentle introduction to the language’s unique combination of features, such as: i) basic project structure and package management, ii) rich numerical and array syntax for kernel programming on heterogeneous systems, iii) high-level functionality for composable and interactive data analysis workflows allowing it to be easily integrated into existing high-performance computing (HPC) workflows, iv) powerful macro syntax for code instrumentation, v) lightweight interoperability with C, C++, Fortran, and Python, vi) HPC packages of interest including community support, and viii) creating a package environment, tests and reproducible project. The second part of this tutorial will introduce the audience to hands-on examples on how to write portable high-performance code targeting heterogeneous NVIDIA, AMD and Intel systems as well as Julia’s multithreading capabilities. Overall, the tutorial targets those interested in achieving performance portability and interactive data analysis via a unified language for their co-design process, and/or for participants interested in expanding their portfolio of programming languages. |
| Software practices for better science: testing, reproducibility, and documentation | David Bernholdt, Patricia Grubel, Mark Miller, David Rogers and Gregory Watson | Software productivity<br>Software quality<br>Reproducibility<br>Software testing<br>Lab notebooks for computational research | As many ECP projects begin their transition from major development towards production science, this tutorial will offer key strategies to help projects improve their science. The tutorial will focus on testing strategies (design and selection in different contexts), reproducibility concerns and the creation of “lab notebook”-style documentation. These practices will provide you and your team detailed information about what to do and why. We’ll offer practical strategies, based on experience in a broad range of projects, that can help improve the effectiveness in going from software to science. |

| | | | |
|---|---|---|---|
| OpenMP 5+ tutorial | Colleen Bertoni, Sunita Chandrasekaran, Johannes Doerfert, Abid Malik, Dossay Oryspayev, Swaroop Pophale and Tom Scogland | OpenMP offload<br>accelerator programming<br>pragma-based | While most HPC developers have OpenMP experience, many are not familiar with the latest OpenMP features and how to use them in their codes. Modern OpenMP can be used in creative ways to map parallelism to current and emerging parallel architectures. Yet it is much broader and more complex than OpenMP was even just three years ago, and a significant amount of training is needed if programmers are to be able to exploit it fully.<br><br>We propose a tutorial that describes recent and new features of OpenMP, with a focus on those features that have proven important for ECP. We will cover emerging programming styles and best practices for early adopters so that they can efficiently program accelerators, orchestrate work across CPUs, GPUs and the network, and take advantage of the memory hierarchy.<br><br>The OpenMP 5.0 specification was released at SC'18 with exciting new functionality. OpenMP 5.1 was released at SC'20. OpenMP 5.2, a revision for clarifications in the Specification, was released at SC '21. In SC '22 TR11 will be released. Thus, it is important not only for developers to be aware of the current standard, and what is available today, but also what is coming next and what will be available in the exascale time frame.<br><br>Finally, we also want to use this opportunity to discuss with the application teams' areas where they can influence the design choices and the open challenges that we haven't solved yet but that we will address in the future.<br><br>The tutorial is expected to cover the following topics: Overview of what is available today in OpenMP 5.2, An overview of the accelerator programming model, Examples on how to map data structures, How to exploit parallelism in the target regions, Fortran and C/C++ examples and specific features, Demo and hands-on sessions. |
| Dense & Sparse Linear Algebra and FFT Libraries: SLATE, Ginkgo, MAGMA, heFFTe | Mark Gates, Stan Tomov, Sebastien Cayrols, Daniel Bielich, Ahmad Abdelfattah, Pratik Nayak and Natalie Beams | linear algebra<br>FFT<br>GPU computing<br>distributed computing | This tutorial focuses on the SLATE, Ginkgo, and MAGMA linear algebra libraries and the heFFTe multidimensional FFT library. These libraries are designed as portable ways to harness today's supercomputers with multicore CPUs and multiple GPU accelerators. The tutorial covers practical aspects in setting up matrices and calling the libraries in your application. No prior knowledge is required. The format will be as four segments, so attendees can attend just the applicable segments, or all four.<br><br>SLATE is a modern C++ library developed as part of ECP to replace ScaLAPACK for solving dense linear algebra problems on distributed systems. It supports multicore CPU nodes or hybrid CPU-GPU nodes, using MPI for communication and OpenMP tasks for node-level scheduling. It covers parallel BLAS, solving linear systems (LU, Cholesky, QR, symmetric indefinite), least squares, symmetric eigenvalue and SVD solvers. SLATE includes a ScaLAPACK compatible interface to aide transitioning existing applications, as well as C and Fortran interfaces.<br><br>Ginkgo is a high performance numerical linear algebra library focusing on solution of linear systems and sparse matrix operations, providing highly tuned kernels for cross-platform performance on multi-threaded CPUs as well as NVIDIA, AMD, and Intel GPUs. It implements many state of the art Krylov solvers and preconditioners. It also has interfaces to many well known application libraries such as MFEM, deal.ii, SUNDIALS, XGC, etc.<br><br>MAGMA is a C library for accelerating dense and sparse linear algebra on a node using multiple GPUs. It covers a large part of LAPACK's functionality: LU, Cholesky, QR, symmetric indefinite, eigenvalue and singular value solvers. It has a batch component for solving many small problems in parallel, and tensor contractions for high-order finite element methods. The sparse component accelerates many iterative algorithms such as CG, GMRES, and LOBPCG. MAGMA also includes a Fortran interface.<br><br>The Highly Efficient FFTs for Exascale (heFFTe) library provides multidimensional Fast Fourier Transforms (FFTs) for Exascale platforms. FFTs are in the software stack for almost all ECP applications. The tutorial will cover heFFTe's APIs, approach, performance, and use in applications. heFFTe leverages existing FFT capabilities, including 1-D FFTs, FFTMPI, and SWFFT. |
| Updates to ADIOS2: Storage and in situ I/O | Norbert Podhorszki and Scott Klasky | I/O<br>Data<br>in situ<br>code coupling | The ADIOS I/O framework provides data models, portable APIs, storage abstractions, in situ infrastructure, and self-describing data containers. These capabilities enable reproducible science, allow for more effective data management throughout the data life cycle, and facilitate parallel I/O with high performance and scalability. In this tutorial, participants will learn about the ADIOS concept and APIs and we will show a pipeline of a simulation, analysis and visualization, using both storage I/O and in situ data staging. Participants will also learn how to use state-of-the art compression techniques with ADIOS. We will introduce the new BP5 file format and show new functionalities around handling many steps in a single dataset efficiently.<br><br>Finally, we will discuss how to use ADIOS on the LCFs for the best storage performance, in situ data analysis and code coupling. This is a short, 2-hour long tutorial that aims to teach the basic concepts, demonstrate the capabilities and provide pointers to interested parties to incorporate ADIOS into their science applications. |

| | | | |
|---|---|---|---|
| E4S: Extreme-scale Scientific Software Stack | Sameer Shende | E4S<br>Spack<br>SDK<br>Containers<br>HPC<br>AI/ML<br>AWS | The DOE Exascale Computing Project (ECP) Software Technology focus area is developing an HPC software ecosystem that will enable the efficient and performant execution of exascale applications. Through the Extreme-scale Scientific Software Stack (E4S) [https://e4s.io], it is developing a comprehensive and coherent software stack that will enable application developers to productively write highly parallel applications that can portably target diverse exascale architectures. E4S provides both source builds through the Spack platform and a set of containers that feature a broad collection of HPC software packages. E4S exists to accelerate the development, deployment, and use of HPC software, lowering the barriers for HPC users. It provides Spack build cache for package binaries, container images, build manifests, and turn-key, from-source builds of popular HPC software packages developed as Software Development Kits (SDKs). This effort includes a broad range of areas including programming models and runtimes (MPICH, Kokkos, RAJA, OpenMPI), development tools (TAU, HPCToolkit, PAPI), math libraries (PETSc, Trilinos), data and visualization tools (Adios, HDF5, Paraview), workflows (flex), and compilers (LLVM), all available through the Spack package manager. E4S includes bare-metal installation of packages on HPC systems at ORNL, ALCF, and NERSC, containerized deployment using Docker, Singularity, and Shifter, as well as cloud based images on AWS. The tutorial will describe how to use E4S and contribute to it. |
| Autotuning ECP Applications at Scale with ECP PROTEAS-TUNE/ytopt and PETSc/TAO libEnsemble | Xingfu Wu, Prasanna Balaprakash, Jeffrey Larson, John R. Tramm, Brice Videau, Michael Kruse, Paul Hovland and Mary Hall | Autotuning<br>Machine learning<br>ytopt<br>libEnsemble<br>OpenMC | ytopt is a Python machine-learning-based search software package developed within the ECP PROTEAS-TUNE project. The ytopt software adopts an asynchronous search framework that consists of sampling a small number of input parameter configurations and progressively fitting a surrogate model over the input-output space until exhausting the user-defined maximum number of evaluations. The framework is designed to operate in the single manager multiple workers parallelization scheme, where the manager node fits the surrogate model and generates promising input configurations and worker nodes perform the computationally expensive evaluations and return the outputs to the manager node. libEnsemble (ensemble management for exascale platforms) is a Python toolkit for coordinating workflows of asynchronous and dynamic ensembles of calculations across massively parallel resources developed within the ECP PETSc/TAO project. libEnsemble helps users take advantage of massively parallel resources to solve design, decision, and inference problems and expands the class of problems that can benefit from increased parallelism. In this tutorial, we will present our methodology and framework to integrate ytopt with the libEnsemble to take advantage of massively parallel resources to accelerate the autotuning process. Specifically, we will focus on autotuning ECP OpenMC, an open source Monte Carlo particle transport code. There are many application parameters with large ranges needed to be tuned for OpenMC, for instance, for a large problem size, just one parameter the number of particles in-flight is in the range of 100k to 8 million. Its default setting is 1 million, but for different architectures with distinct memory sizes, setting the proper maximum number of particles in-flight helps achieve the best performance (particles/s). Exhaustively evaluating all parameter combinations becomes very time-consuming. Therefore, autotuning for automatic exploration of the parameter space is desirable. We will apply the ytopt autotuning framework with libEnsemble to the ECP application MPI/OpenMP offload version of OpenMC to tune its performance based on a user-defined metric such as its Figure of Merit (particles/s), application execution time, or energy efficiency EDP (Energy Delay Product) on OLCF Frontier TDS Crusher and show good autotuning results. If ALCF Aurora TDS Sunspot is available before the tutorial, we will present its autotuning results as well. |
| DAOS Tutorial | Kevin Harms, Johann Lombardi and Mohamad Chaarawi | DAOS<br>Storage<br>I/O | DAOS (Distributed Asynchronous Object Store) is the dedicated high performance platform storage for Argonne's Aurora Exascale supercomputer. Aurora will feature both traditional Lustre based Parallel File System (PFS) and the DAOS storage system. In order for users to obtain the highest performance for I/O on Aurora, the DAOS storage system will be necessary to use. DAOS provides users an extensive set of capabilities and features to tune I/O performance. The tutorial will cover the various concepts needed for users to effectively use DAOS on Aurora. The session will provide an overview of the DAOS hardware for Aurora and Sunspot.<br><br>From the user perspective, DAOS is composed of various components such as pools, containers, objects and keys. The structure of DAOS will be explored from a user facing perspective with an emphasis on how DAOS is deployed on Aurora. The Aurora site-specific configuration and behaviors will be discussed outlining the best practices for performance under different workloads. The session will have a demonstration that starts with an initial empty pool allocations and walks through adding users, initializing containers and then performing I/O to DAOS from applications running on Sunspot. |

| | | | |
|---|---|---|---|
| HPCToolkit: Emerging Performance Tools for Exascale Computing | John Mellor-Crummey and Wil Phan | ECP ST<br>performance tools<br>application performance | To address the challenge of performance analysis on the US DOE's forthcoming exascale supercomputers, Rice University has been extending its HPCToolkit performance tools to support measurement and analysis of GPU-accelerated applications. To help developers understand the performance of accelerated applications as a whole, HPCToolkit's measurement and analysis tools associate profiles and traces with calling contexts that span both CPUs and GPUs.<br><br>To help developers understand the performance of complex GPU code generated from high-level programming models, HPCToolkit constructs sophisticated approximations of call path profiles within GPU kernels. To support fine-grained analysis and tuning, HPCToolkit uses PC sampling and instrumentation to measure and attribute GPU performance metrics to source lines, loops, and inlined code. To supplement fine-grained measurements, HPCToolkit can measure GPU kernel executions using hardware performance counters. To provide a view of how an execution evolves over time, HPCToolkit can collect, analyze, and visualize call path traces within and across nodes.<br><br>Over the last year, the project team has<br><br>- integrated support for measurement and analysis of AMD, Intel, and NVIDIA GPUs into spack releases of HPCToolkit,<br><br>- added support for monitoring executions on Intel GPUs that use Intel's emerging Level Zero runtime,<br><br>- added support for instrumentation-based measurement of kernel execution on Intel GPUs using binary instrumentation with Intel's GTPin,<br><br>- reduced the size of HPCToolkit's measurement data and analysis results by using a sparse-tensor representation for performance metrics,<br><br>- improved the scalability of HPCToolkit's post-mortem analysis by using "streaming aggregation", which employs a combination of shared-memory and distributed memory parallelism for speed and scalability, and<br><br>- improved HPCToolkit's graphical user interface for analyzing measurements of GPU-accelerated programs.<br><br>This tutorial will present:<br><br>- an overview of HPCToolkit's measurement and analysis capabilities for GPU-accelerated applications, |
| AMD Software Tools for Exascale Computing | Timour Paltashev, Nicholas Malaya and Robert Robey | HPC tools<br>GPU tools<br>GPU Profilers<br>GPU Debuggers<br>AMD software tools | Moderator: Timour Paltashev, AMD<br><br>This tutorial demonstrates software tools that support AMD CPU and GPU hardware. The tools are currently all available on Frontier.<br><br>omniTools: Jonathan Madsen and Xiaomin Lu, AMD<br><br>omnitrace provides profiling and tracing for AMD GPU applications. Using binary instrumentation, sampling, and user-defined hooks, it produces high-level summaries and interactive trace visualization.<br>omniperf is a performance profiler for AMD GPUs using all approved hardware counters. It includes speed-of-light, memory chart, and roofline analysis with CLI and GUI interfaces.<br><br>PAPI: Giuseppe Congiu, Innovative Computing Laboratory, University of Tennessee, Knoxville<br><br>PAPI provides an interface to performance counters of most hardware components. We demonstrate the measurement of compute and power on AMD GPUs. We then show PAPI's use of the new ROCprofiler "intercept mode" for fine-grained kernel performance measurements.<br><br>LIKWID: Karlo Kraljic, HPE Pointnext Services, Hewlett-Packard GmbH<br><br>The LIKWID command line utilities have been enhanced for AMD GPUs. We'll show LIKWID wrapper on AMD GPUs generating performance metrics from an unmodified application. Then shown is how to modify an application with the "Marker API" for finer-grained analysis.<br><br>TAU: Sameer Shende, University of Oregon<br><br>The TAU Performance System(R) now supports AMD CPUs and GPUs. We show TAU's instrumentation for ROCprofiler, ROCtracer, Kokkos, OpenMP and support for LLVM based compilers. We'll also demonstrate TAU's 3D profile browser, paraprof.<br><br>HPCToolkit/DynInst: John Mellor-Crummey, Rice University and Barton Miller, University of Wisconsin |

| | | | |
|---|---|---|---|
| Performance Evaluation using the TAU Performance System | Sameer Shende and Kevin Huck | TAU<br>Level Zero<br>oneAPI<br>CUDA<br>CUPTI<br>OpenACC<br>OpenMP<br>OMPT<br>MPI<br>E4S<br>AWS<br>Containers | To meet the needs of computational scientists to evaluate and improve the performance of their parallel, scientific applications, we present the TAU Performance System and describe how it supports the GPU platforms from NVIDIA, AMD, and Intel. This talk presents TAU as a versatile tool for performance profiling and tracing that supports both direct instrumentation and sampling. With event-based sampling in TAU, it is possible to evaluate the performance of un-instrumented applications. TAU can collect data at the fine granularity of statements as well as routines. It supports direct instrumentation of MPI, pthread, OpenMP (including target offload directives), OpenACC, DPC++ runtime (Level Zero), ROCm, and CUDA and can be used with sampling. The talk will describe the elements of TAU and show a demonstration of TAU's usage on the Extreme-scale Scientific Software Stack (E4S) AWS image. |
| Whole Device Modeling: First-principles Core-Edge Coupling Workflow Tutorial with WDMApp | Eric Suchyta and Scott Klasky | whole device modeling<br>plasma physics<br>workflows<br>applications | WDMapp formulates plasma whole device modeling with a modular approach, by coupling multiple applications, where each has been optimized for a specific regime of the physics. The project has employed three first principles gyrokinetic simulation codes, XGC for the edge calculations, and GENE or GEM for the core computations, as well as developed the appropriate technology needed to transform between the different representations and meshes adopted by the respective application. In this tutorial, we demonstrate how to concurrently run spatially coupled core and edge applications, configured for kinetic electrons and electromagnetic perturbations, where data is exchanged between the two codes in an overlap region common to both simulations. This is the problem that constitutes the basis for the project's KPP-FOM metrics. For the tutorial, we also overview the software and workflow management frameworks on which WDMapp depends. This includes ADIOS configuration of in-memory data movement between application, as well as workflow composition and execution with EFFIS. The workloads will be composed for and executed on the latest ECP hardware (i.e., Crusher or Frontier contingent upon access and availability). Furthermore we incorporated this technology to loosely couple performance modeling, using TAU, data reduction, using MGARD, and Visualization using VTM-M, to expedite the WDMApp knowledge discovery. Many of the tools and technologies discussed in this tutorial utilize ECP software technologies |
| Performance Tuning with the Roofline Model on GPUs and CPUs | Samuel Williams, Jaehyuk Kwack, Neil Mehta and Nicholson Koukpaizan | Roofline<br>Performance Analysis<br>GPU<br>CPU | The HPC community is on a never ending quest for better performance and scalability. Performance models and tools are an integral component in the optimization process as they quantify performance relative to machine capabilities, track progress towards optimality, and identify performance bottlenecks. The Roofline performance model offers an insightful and intuitive method for extracting the key computational characteristics of HPC applications and comparing them against the performance bounds of CPUs and GPUs. Its capability to abstract the complexity of memory hierarchies and identify the most profitable optimization techniques has made Roofline--based analysis increasingly popular in the HPC community. This 180-minute tutorial is centered around four components. First, we will introduce the Roofline model and discuss how changes in data locality and arithmetic intensity visually manifest in the context of the Roofline model. Next, we will introduce and demonstrate the use of Roofline analysis in NVIDIA Nsight Compute and discuss several real-world use cases of Roofline in the NERSC NESAP effort. After a short break, we will return and introduce and demonstrate the use of AMD's Roofline tool for analysis of AMD GPU-accelerated applications running at OLCF. Finally, we will conclude by introducing and demonstrating Intel Advisor's capability for Roofline analysis of applications running on Intel GPUs and CPUs as well as Roofline use cases at ALCF.<br><br>* Introduction to the Roofline Model (45 mins) Samuel Williams<br>* Roofline Analysis using NVIDIA Nsight Compute (45 mins) Neil Mehta<br>* Roofline Analysis using AMD Roofline Tool on AMD GPUs (45 mins) Nicholson Koukpaizan<br>* Roofline Analysis using Intel Advisor on Intel GPUs and CPUs (45 mins) JaeHyuk Kwack<br>* Closing Remarks / Q&A (5 mins) |
| RAJA Portability Suite Tutorial | David Beckingsale, Kristi Belcher, Mike Davis and Richard Hornung | memory management<br>programming models<br>portability<br>RAJA<br>Umpire<br>CHAI<br>c++ | With the rapid change of computing architectures, and variety of programming models, the ability to develop performance portable applications has become of great importance. This is particularly true in large production codes where developing and maintaining hardware specific versions is untenable.<br><br>To simplify the development of performance portable code, we present the RAJA Suite, a collection of C++ libraries that allow developers to write single-source applications that can target multiple hardware and programming model back-ends. RAJA decouples loop bodies and execution via programming model-specific implementations using standard C++11 features. This approach enables developers to tune loop patterns, rather than individual loops, and enables applications to be tailored at compile time to specific compute architectures. Umpire provides a programming-model agnostic way to manage memory, in addition to more complex memory allocation strategies that can improve application performance. CHAI is an array abstraction that can automate data motion between memory spaces in a heterogeneous system. When coupled with RAJA and Umpire, this benefit is provided without additional programmer intervention. In this tutorial, we present a walkthrough of the features of each of the three RAJA suite libraries, and show how they can be combined in application inspired examples which will allow attendees to understand how these products may be used in their own applications. |

| | | | |
|---|---|---|---|
| A hands-on introduction to the OCCA portability framework | Tim Warburton, Noel Chalmers, Kris Rowe and Saumil Patel | portability<br>performance<br>GPU<br>CUDA<br>HIP<br>SYCL | OCCA is an open source, portable, and vendor neutral framework for parallel programming on heterogeneous platforms. The OCCA API provides unified models for heterogeneous programming concepts—such as a device, memory, or kernel—while the OCCA Kernel Language (OKL) enables the creation of portable device kernels using a directive-based extension to the C-language.<br><br>Mission critical computational science and engineering applications from the public and private sectors rely on OCCA. Notable users include the U.S. Department of Energy and Shell.<br><br>Key themes of the OCCA framework are portability, performance, transparency, and extensibility. Portability is achieved through multiple backends (CUDA, HIP, SYCL, OpenCL, OpenMP, Metal), interoperability with backend API and kernels, and language support for C, C++, and Fortran. JIT compilation and caching of kernels enables the definition of constants during runtime, facilitating performance. Transparent translation of kernels makes it easy for developers to understand how their code is mapped to each platform. The ability to extend the OCCA API and to implement new kernel attributes means the framework can easily be customized for specific use cases.<br><br>In this tutorial, you will learn the fundamentals of the OCCA framework: hardware discovery and selection; memory management; kernel definition, JIT compilation, and execution; and device synchronization. These concepts will be presented through a series of progressive, hands-on examples relevant to computational science and engineering applications. Guidelines for achieving performance will be discussed, along with other best practices for OCCA application development. A vignette on extending the OCCA framework will examine real-world examples---drawing from ECP libraries and applications, such as libParanumal and NekRS.<br><br>The tutorial will close by highlighting ways to become involved with and contribute to the OCCA framework, which is a truly open-source and community driven project. |
| The Template Task Graphs Programming Paradigm | Thomas Herault, Joseph Schuchart and George Bosilca | Task-Based Runtime System<br>Distributed Computing<br>Accelerated Computing<br>Unbalanced and Irregularly Shaped Applications | The PaRSEC team will highlight the Template Task Graph (TTG) programming paradigm, the concepts, benefits and requirement of the programming approach as well as the practical aspects necessary to start using TTG on various platforms to write portable task-based applications. The team will provide direct support during the tutorial as well as through GitHub and a mailing list after the tutorial.<br><br>Template Task Graphs have been developed to enable a straightforward expression of task parallelism for algorithms working on irregular and unbalanced data sets. The TTG Application Programming Interface employs C++ templates to build an abstract representation of the task graph, and schedule it on distributed resources. It offers a scalable and efficient API to port complex applications on top of task-based runtime systems to gain access to asynchronous progress, computation/communication overlap, and efficient use of all computing resources available on the target system. In this tutorial, we will introduce TTG and its main concepts and features through a variety of applications, ranging from well-known regular to irregular and data dependent examples.<br><br>The tutorial, which features many hands-on examples, presents how to install TTG on the ECP platforms and other environments, how to integrate TTG with your application using CMake, how to express task-based data-dependent algorithms for irregular datasets using TTG, and how to integrate these task-based algorithms inside existing applications.<br><br>The tutorial is available to everyone, and participants from any background are welcome to attend. A basic knowledge of C++ and templates will be helpful for participants who wish to try the hands-on. |

| | | | |
|---|---|---|---|
| Flux: Next-Generation Resource Management for Exascale Workflows and Job Scheduling | Thomas Scogland, Daniel Milroy, Tapasya Patki, James Corbett and Jae-Seung Yeom | Workflows<br>Resource management<br>Scheduling | Modern scientific workflows require complex interactions among physics simulations, analysis and visualization tools, data stores, and more. These requirements have outpaced the capabilities of traditional HPC resource management and scheduling (RMS) systems. Various scalability and programming effort challenges have already begun to appear, and several ad-hoc approaches to workflow management have come into existence. Exascale platforms will present significantly greater challenges and such ad-hoc solutions will no longer be sufficient. In this tutorial, we introduce Flux, a next-generation resource management framework that supports diverse scientific workflows, system scheduling on Exascale-class supercomputers such as the upcoming El Capitan and complex resources such as power and IO. With the help of practical demonstrations, attendees will learn about easy-to-use, portable solutions for scientific workflows with the Flux API as well as about hierarchical, user-level scheduling techniques that can be used for high-throughput computing and ensemble management.<br><br>Flux is an open-source, next-generation resource management framework that addresses this technology gap and expands the scheduler's view beyond the single dimension of nodes. It provides support for management of complex workflows as well as diverse resources such as power, network, IO with ease. The Flux infrastructure is designed to be fully hierarchical in order to expose high levels of scheduler parallelism while allowing for workload customization and enforcement of multi-level constraints through its flexible graph-based resource model. Most current HPC scheduling frameworks, such as SLURM or PBSPro, are not designed to support complex workflows and rely on a centralized daemon, resulting in scalability bottlenecks and limited throughput. This is an important thrust area for exascale, making this topic highly relevant for ECP. With the help of compelling use cases, practical demonstrations, and hands-on exercises, attendees will be able to:<br>•Learn how Flux can significantly reduce scripting effort and scheduling complexity,<br>•Understand and apply the Flux API to manage complex scientific workflows in a portable manner with a hands-on demo (Docker and/or AWS)<br>•Understand user-level hierarchical scheduling and apply such user-level schedulers to manage their ensembles and science workflows<br>•Understand how Flux can be installed and set up with the help of detailed examples<br>•Understand how Flux can be integrated to manage their specific workflow (support for debugging and testing will be provided),<br>•Understand the graph-based data model and how to write customized scheduling plugins<br>•Develop awareness for future directions and roadmap for Flux<br><br>Rough agenda:<br>Minutes:topic |