Overview of ECP Software Technology

Michael A. Heroux, Sandia National Laboratories
Director of Software Technology

E4S at NERSC 2022, August 25, 2022
Outline

• ECP, Briefly
• Establishing software ecosystems
• Developing software for GPU systems
ECP in a Nutshell
ECP’s holistic approach uses co-design and integration to achieve exascale computing

Performant mission and science applications at scale

Application Development (AD)
- Develop and enhance the predictive capability of applications critical to DOE
  - 24 applications
    National security, energy, Earth systems, economic security, materials, data
  - 6 Co-Design Centers
    Machine learning, graph analytics, mesh refinement, PDE discretization, particles, online data analytics

Software Technology (ST)
- Deliver expanded and vertically integrated software stack to achieve full potential of exascale computing
  - 70 unique software products
    spanning programming models and run times, math libraries, data and visualization

Hardware and Integration (HI)
- Integrated delivery of ECP products on targeted systems at leading DOE HPC facilities
  - 6 US HPC vendors
    focused on exascale node and system design; application integration and software deployment to Facilities
Rajeev Thakur, Programming Models and Runtimes (2.3.1)
Rajeev is a senior computer scientist at ANL and most recently led the ECP Software Technology focus area. His research interests are in parallel programming models, runtime systems, communication libraries, and scalable parallel I/O. He has been involved in the development of open-source software for large-scale HPC systems for over 20 years.

Jeff Vetter, Development Tools (2.3.2)
Jeff is a computer scientist at ORNL, where he leads the Future Technologies Group. He has been involved in research and development of architectures and software for emerging technologies, such as heterogeneous computing and nonvolatile memory, for HPC for over 15 years.

Sherry Li, Math Libraries (2.3.3)
Sherry is a senior scientist at Berkeley Lab. She has over 20 years of experience in high-performance numerical software, including development of SuperLU and related linear algebra algorithms and software.

Jim Ahrens, Data and Visualization (2.3.4)
Jim is a senior research scientist at the Los Alamos National Laboratory (LANL) and an expert in data science at scale. He started and actively contributes to many open-source data science packages including ParaView and Cinema.

Todd Munson, Software Ecosystem and Delivery (2.3.5)
Todd is a computational scientist in the Math and Computer Science Division of ANL. He has nearly 20 years of experience in high-performance numerical software, including development of PETSc/TAO and project management leadership in the ECP CODAR project.

Kathryn Mohror, NNSA ST (2.3.6)
Kathryn is Group Leader for the CASC Data Analysis Group at LLNL. Her work focuses on I/O for extreme scale systems, scalable performance analysis and tuning, fault tolerance, and parallel programming paradigms. She is a 2019 recipient of the DOE Early Career Award.
ECP ST has six technical areas

Programming Models & Runtimes
- Enhance and get ready for exascale the widely used MPI and OpenMP programming models (hybrid programming models, deep memory copies)
- Development of performance portability tools (e.g., Kokkos and Raja)
- Support alternate models for potential benefits and risk mitigation: PGAS (UPC++/GASNet), task-based models (Legion, PaRSEC)
- Libraries for deep memory hierarchy and power management

Development Tools
- Continued, multifaceted capabilities in portable, open-source LLVM compiler ecosystem to support expected ECP architectures, including support for F18
- Performance analysis tools that accommodate new architectures, programming models, e.g., PAPI, Tau

Math Libraries
- Linear algebra, iterative linear solvers, direct linear solvers, integrators and nonlinear solvers, optimization, FFTs, etc
- Performance on new node architectures; extreme strong scalability
- Advanced algorithms for multi-physics, multiscale simulation and outer-loop analysis
- Increasing quality, interoperability, complementarity of math libraries

Data and Visualization
- I/O via the HDF5 API
- Insightful, memory-efficient in-situ visualization and analysis – Data reduction via scientific data compression
- Checkpoint restart

Software Ecosystem
- Develop features in Spack necessary to support all ST products in E4S, and the AD projects that adopt it
- Development of Spack stacks for reproducible turnkey deployment of large collections of software
- Optimization and interoperability of containers on HPC systems
- Regular E4S releases of the ST software stack and SDKs with regular integration of new ST products

NNSA ST
- Open source NNSA Software projects
- Projects that have both mission role and open science role
- Major technical areas: New programming abstractions, math libraries, data and viz libraries
- Cover most ST technology areas
- Subject to the same planning, reporting and review processes

Area Leads:
- Rajeev Thakur
- Jeff Vetter
- Sherry Li
- Jim Ahrens
- Todd Munson
- Kathryn Mohror

ECP ST Director: Mike Heroux
ECP ST Deputy Director: L.C. McInnes
### ST L4 Leads

- **WBS**
- **Name**
- **PIS**
- **PCs - Project Coordinators**

#### 2.3 Software Technology

<table>
<thead>
<tr>
<th>WBS</th>
<th>WBS Name</th>
<th>CAM/PI</th>
<th>PC</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.3</td>
<td>Programming Models &amp; Runtimes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.3.1.01</td>
<td>PMR SDK</td>
<td>Shende, Sameer</td>
<td></td>
</tr>
<tr>
<td>2.3.1.07</td>
<td>Exascale MPI (MPICH)</td>
<td>Guo, Yanfei</td>
<td>Guo, Yanfei</td>
</tr>
<tr>
<td>2.3.1.08</td>
<td>Legion</td>
<td>McCormick, Pat</td>
<td>McCormick, Pat</td>
</tr>
<tr>
<td>2.3.1.09</td>
<td>PaRSEC</td>
<td>Bosilca, George</td>
<td>Carr, Earl</td>
</tr>
<tr>
<td>2.3.1.14</td>
<td>Pagoda: UPC++/GASNet for Lightweight Communication and Global Address Space Support</td>
<td>Hargrove, Paul</td>
<td>Hargrove, Paul</td>
</tr>
<tr>
<td>2.3.1.16</td>
<td>SICM</td>
<td>Graham, Jonathan</td>
<td>Turton, Terry</td>
</tr>
<tr>
<td>2.3.1.17</td>
<td>OMPI-X</td>
<td>Bernholdt, David</td>
<td>Grundhofer, Alicia</td>
</tr>
<tr>
<td>2.3.1.18</td>
<td>RAJA/Kokkos</td>
<td>Trott, Christian Robert</td>
<td>Trujillo, Gabrielle</td>
</tr>
<tr>
<td>2.3.1.19</td>
<td>Argo: Low-level resource management for the OS and runtime</td>
<td>Beckman, Pete</td>
<td>Gupta, Rinku</td>
</tr>
</tbody>
</table>

#### 2.3.2 Development Tools

<table>
<thead>
<tr>
<th>WBS</th>
<th>WBS Name</th>
<th>CAM/PI</th>
<th>PC</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.3.2.01</td>
<td>Development Tools Software Development Kit</td>
<td>Miller, Barton</td>
<td>Tim Haines</td>
</tr>
<tr>
<td>2.3.2.06</td>
<td>Exa-PAPI++: The Exascale Performance Application Programming Interface with Modern C++</td>
<td>Dongarra, Jack</td>
<td>Jagode, Heike</td>
</tr>
<tr>
<td>2.3.2.08</td>
<td>Extending HPC Toolkit to Measure and Analyze Code Performance on Exascale Platforms</td>
<td>Mellor-Crummey, John</td>
<td>Meng, Xiaozhu</td>
</tr>
<tr>
<td>2.3.2.10</td>
<td>PROTEAS-TUNE</td>
<td>Vetter, Jeff</td>
<td>Hornick, Mike</td>
</tr>
<tr>
<td>2.3.2.11</td>
<td>SOLLVE: Scaling OpenMP with LLVM for Exascale</td>
<td>Chandrasekaran, Sunita</td>
<td>Orysayev, Dossay</td>
</tr>
<tr>
<td>2.3.2.12</td>
<td>FLANG</td>
<td>McCormick, Pat</td>
<td>Perry-Holby, Alexis</td>
</tr>
</tbody>
</table>

#### 2.3.3 Mathematical Libraries

<table>
<thead>
<tr>
<th>WBS</th>
<th>WBS Name</th>
<th>CAM/PI</th>
<th>PC</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.3.3.01</td>
<td>Extreme-scale Scientific xSDK for ECP</td>
<td>Yang, Ulrike</td>
<td>Yang, Ulrike</td>
</tr>
<tr>
<td>2.3.3.06</td>
<td>Preparing PETSc/TAO for Exascale</td>
<td>Munson, Todd</td>
<td>Munson, Todd</td>
</tr>
<tr>
<td>2.3.3.07</td>
<td>STRUMPACK/SuperLU/FXT: sparse direct solvers, preconditioners, and FFT libraries</td>
<td>Li, Sherry</td>
<td>Li, Sherry</td>
</tr>
<tr>
<td>2.3.3.12</td>
<td>Enabling Time Integrators for Exascale Through SUNDIALS/Hydra</td>
<td>Woodward, Carol</td>
<td>Woodward, Carol</td>
</tr>
<tr>
<td>2.3.3.13</td>
<td>CLOVER: Computational Libraries Optimized Via Exascale Research</td>
<td>Dongarra, Jack</td>
<td>Carr, Earl</td>
</tr>
<tr>
<td>2.3.3.14</td>
<td>ALExa: Accelerated Libraries for Exascale/ForTrilinos</td>
<td>Prokopenko, Andrey</td>
<td>Grundhofer, Alicia</td>
</tr>
<tr>
<td>2.3.3.15</td>
<td>Sake: Solvers and Kernels for Exascale</td>
<td>Rajamanickam, Siva</td>
<td>Trujillo, Gabrielle</td>
</tr>
</tbody>
</table>

#### 2.3.4 Data and Visualization

<table>
<thead>
<tr>
<th>WBS</th>
<th>WBS Name</th>
<th>CAM/PI</th>
<th>PC</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.3.4.01</td>
<td>Data and Visualization Software Development Kit</td>
<td>Atkins, Chuck</td>
<td>Bagha, Neelam</td>
</tr>
<tr>
<td>2.3.4.09</td>
<td>ADIOS Framework for Scientific Data on Exascale Systems</td>
<td>Klassy, Scott</td>
<td>Hornick, Mike</td>
</tr>
<tr>
<td>2.3.4.10</td>
<td>DataLib: Data Libraries and Services Enabling Exascale Science</td>
<td>Ross, Rob</td>
<td>Ross, Rob</td>
</tr>
<tr>
<td>2.3.4.13</td>
<td>ECP/VTK-m</td>
<td>Moreland, Kenneth</td>
<td>Moreland, Kenneth</td>
</tr>
<tr>
<td>2.3.4.14</td>
<td>VeloC: Very Low Overhead Transparent Multilevel Checkpoint/Restart/Sz</td>
<td>Cappello, Franck</td>
<td>Ehling, Scott</td>
</tr>
<tr>
<td>2.3.4.15</td>
<td>ExaIO - Delivering Efficient Parallel I/O on Exascale Computing Systems with HDF5 and Unify</td>
<td>Byna, Suren</td>
<td>Bagha, Neelam</td>
</tr>
<tr>
<td>2.3.4.16</td>
<td>ALPINE: Algorithms and Infrastructure for In Situ Visualization and Analysis/ZFP</td>
<td>Ahrens, James</td>
<td>Turton, Terry</td>
</tr>
</tbody>
</table>

#### 2.3.5 Software Ecosystem and Delivery

<table>
<thead>
<tr>
<th>WBS</th>
<th>WBS Name</th>
<th>CAM/PI</th>
<th>PC</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.3.5.01</td>
<td>Software Ecosystem and Delivery Software Development Kit</td>
<td>Munson, Todd</td>
<td>Munson, Todd</td>
</tr>
<tr>
<td>2.3.5.09</td>
<td>SW Packaging Technologies</td>
<td>Gamblin, Todd</td>
<td>Gamblin, Todd</td>
</tr>
<tr>
<td>2.3.5.10</td>
<td>ExaWorks</td>
<td>Laney, Dan</td>
<td>Laney, Dan</td>
</tr>
</tbody>
</table>

#### 2.3.6 NNSA ST

<table>
<thead>
<tr>
<th>WBS</th>
<th>WBS Name</th>
<th>CAM/PI</th>
<th>PC</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.3.6.01</td>
<td>LANL ATDM</td>
<td>Mahrer, Kathryn</td>
<td>Mahrer, Kathryn</td>
</tr>
<tr>
<td>2.3.6.02</td>
<td>LLNL ATDM</td>
<td>Mike Lang</td>
<td>Vandenbusch, Tanya Marie</td>
</tr>
<tr>
<td>2.3.6.03</td>
<td>SNL ATDM</td>
<td>Becky Springmeyer</td>
<td>Gamblin, Todd</td>
</tr>
</tbody>
</table>

---

### ECP ST Stats

- **250 staff**
- **70 products**
- **35 L4 subprojects**
- **30 universities**
- **9 DOE labs**
- **6 technical areas**
- **1 of 3 ECP focus areas**
ECP Software Technology works on products that apps need now and in the future

**Key themes:**
- **Focus:** GPU node architectures and advanced memory & storage technologies
- **Create:** New high-concurrency, latency tolerant algorithms
- **Develop:** New portable (Nvidia, Intel, AMD GPUs) software product
- **Enable:** Access and use via standard APIs

**Software categories:**
- **Next generation established products:** Widely used HPC products (e.g., MPICH, OpenMPI, PETSc)
- **Robust emerging products:** Address key new requirements (e.g., Kokkos, RAJA, Spack)
- **New products:** Enable exploration of emerging HPC requirements (e.g., SICM, zfp, UnifyCR)

<table>
<thead>
<tr>
<th>Example Products</th>
<th>Engagement</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPI – Backbone of HPC apps</td>
<td>Explore/develop MPICH and OpenMPI new features &amp; standards</td>
</tr>
<tr>
<td>OpenMP/OpenACC – On-node parallelism</td>
<td>Explore/develop new features and standards</td>
</tr>
<tr>
<td>Performance Portability Libraries</td>
<td>Lightweight APIs for compile-time polymorphisms</td>
</tr>
<tr>
<td>LLVM/Vendor compilers</td>
<td>Injecting HPC features, testing/feedback to vendors</td>
</tr>
<tr>
<td>Perf Tools - PAPI, TAU, HPCToolkit</td>
<td>Explore/develop new features</td>
</tr>
<tr>
<td>Math Libraries: BLAS, sparse solvers, etc.</td>
<td>Scalable algorithms and software, critical enabling technologies</td>
</tr>
<tr>
<td>IO: HDF5, MPI-IO, ADIOS</td>
<td>Standard and next-gen IO, leveraging non-volatile storage</td>
</tr>
<tr>
<td>Viz/Data Analysis</td>
<td>ParaView-related product development, node concurrency</td>
</tr>
</tbody>
</table>

Legacy: A stack that enables performance portable application development on leadership platforms
Exascale Systems – Primary targets for ECP Software Teams

- ECP libraries & tools migrating to GPU platforms
- Target AMD, Intel and Nvidia (Perlmutter) devices
- Growing support for Arm/SVE in the same stack
- Mature MPI/CPU stack also robust and evolving
- Eye toward specialized devices, e.g., dataflow
- Legacy:
  - A stack to support application portability
  - Across many different distributed systems with
  - Multiple kinds of devices (GPUs, CPUs, etc)

Perlmutter is an important target and vehicle for our ECP work. It is essential for progress and delivery!
The Growing Complexity of Scientific Application Software Stacks
Challenges

As our software gets more complex, it is getting harder to install tools and libraries correctly in an integrated and interoperable software stack.
THE NUMBER OF ECP SOFTWARE TECHNOLOGY PROJECT DEPENDENCIES FOR EACH ECP APPLICATION PROJECT (ANONYMIZED)

- Critical
- Important
- Interested

<table>
<thead>
<tr>
<th></th>
<th>Total</th>
<th>Avg</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Critical</td>
<td>359</td>
<td>12.0</td>
<td>40</td>
</tr>
<tr>
<td>Important</td>
<td>198</td>
<td>6.6</td>
<td>24</td>
</tr>
<tr>
<td>Interest</td>
<td>141</td>
<td>4.7</td>
<td>11</td>
</tr>
</tbody>
</table>
Integration: AD Teams Depend Heavily on ST Software to Meet KPPs

- **Kokkos**
  - nanoBragg code ported from Nvidia to AMD GPUs with minimal effort

- **ExaFEL**

- **ExaWind**
  - hypre solve performance on AMD GPUs 30-40% faster than Summit

- **ADIOS**
  - ADIOS enables in-memory coupling between GENE and XGC

- **WDMApp**

- **hypre**

Slide courtesy of Andrew Siegel and Erik Draeger
Interesting Themes Arising from 2021 AD Assessment

- Sparse solver progress/research challenges
- Evolving OpenMP offload performance
- Co-maturation of vendor compilers, software stack
- ST and CD integration success stories
- Maturity of performance analysis tools
- Network performance

Slide courtesy of Andrew Siegel and Erik Draeger
A Sampler of Products

• No two project alike
• Some personality driven
• Some community driven
• Small, medium, large
Takeaways from product sampler

• Wide range of products and teams: libs, tools, small personality-driven, large community-driven
• Varied user base and maturity: widely used, new, emerging
• Variety of destinations: direct-to-user, facilities, community stacks, vendors, facilities, combo of these
• Wide range of dev practices and workflows from informal to formal
• Wide range of tools: GitHub, GitLab, Doxygen, Readthedocs, CMake, autotools, etc.

• Question at this point might (should?) be:
  – Why are you trying to make a portfolio from this eclectic assortment of products?

• Answer:
  – Each product team charged with challenging tasks:
    • Provide capabilities for next-generation leadership platforms
    • Address increasing software quality expectations
    • While independently developed, product compatibility and complementarity improvements matter
  – Working together on these frontiers is better than going alone
Takeaways from software complexity

- The ECP software ecosystem is truly a complex system, not just complicated
- Plan, execute, track and assess. Repeat
- Challenges are emergent: technical, sociological, and cognitive
Responding to complexity: Software Ecosystem via Platforms
Software Platforms: “Working in Public” Nadia Eghbal

- Platforms in the software world are digital environments that intend to improve the value, reduce the cost, and accelerate the progress of the people and teams who use them.

- Platforms can provide tools, workflows, frameworks, and cultures that provide a (net) gain for those who engage.

- Eghbal Platforms:

<table>
<thead>
<tr>
<th>HIGH CONTRIBUTOR GROWTH</th>
<th>LOW CONTRIBUTOR GROWTH</th>
</tr>
</thead>
<tbody>
<tr>
<td>Federations (e.g., Rust)</td>
<td>Clubs (e.g., Astropy)</td>
</tr>
<tr>
<td>Stadiums (e.g., Babel)</td>
<td>Toys (e.g., ssh-chat)</td>
</tr>
</tbody>
</table>

About Platforms and ECP

• The ECP is commissioned to provide new scientific software capabilities on the frontier of algorithms, software and hardware

• The ECP provides platforms to foster collaboration and cooperation as we head into the frontier:
  – **E4S**: a comprehensive portfolio of ECP-sponsored products and dependencies
  – **SDKs**: Domain-specific collaborative and aggregate product development of similar capabilities
Delivering an open, hierarchical software ecosystem
More than a collection of individual products

Levels of Integration | Product | Source and Delivery
--- | --- | ---
• Build all SDKs | E4S | Source: ECP E4S team; Non-ECP Products (all dependencies)
Delivery: spack install e4s; containers; CI Testing
• Build complete stack
• Assure core policies
• Build, integrate, test

• Group similar products | SDKs | Source: SDK teams; Non-ECP teams (policy compliant, spackified)
Delivery: Apps directly; spack install sdk; future: vendor/facility
• Make interoperable
• Assure policy compliant
• Include external products

• Standard workflow | ST Products | Source: ECP L4 teams; Non-ECP Developers; Standards Groups
Delivery: Apps directly; spack; vendor stack; facility stack
• Existed before ECP

ECP ST Open Product Integration Architecture

ECP ST Individual Products
Extreme-scale Scientific Software Stack (E4S)

- **E4S**: HPC software ecosystem – a curated software portfolio
- A **Spack-based** distribution of software tested for interoperability and portability to multiple architectures
- Available from *source, containers, cloud, binary caches*
- Leverages and enhances SDK interoperability thrust
- Not a commercial product – an open resource for all
- Growing functionality: May 2022: E4S 22.05 – 100+ full release products

Community Policies
Commitment to SW quality

Curated collection
The end of dependency hell

Turnkey stack
A new user experience

DocPortal
Single portal to all E4S product info

Quarterly releases
Release 22.2 – February

Build caches
10X build time improvement

Post-ECP Strategy
LSSw, ASCR Task Force

Also includes other products, e.g.,
AI: PyTorch, TensorFlow, Horovod
**Co-Design**: AMReX, Cabana, MFEM
E4S DocPortal

- Single point of access
- All E4S products
- Summary Info
  - Name
  - Functional Area
  - Description
  - License
- Searchable
- Sortable
- Rendered daily from repos

All we need from the software team is a repo URL + up-to-date meta-data files

https://e4s-project.github.io/DocPortal.html
Goal: All E4S product documentation accessible from single portal on E4S.io (working mock webpage below)

https://e4s-project.github.io/DocPortal.html
Policies: Version 1

https://e4s-project.github.io/policies.html

- **P1:** Spack-based Build and Installation
- **P2:** Minimal Validation Testing
- **P3:** Sustainability
- **P4:** Documentation
- **P5:** Product Metadata
- **P6:** Public Repository
- **P7:** Imported Software
- **P8:** Error Handling
- **P9:** Test Suite

SDK lead: Jim Willenbring (SNL)

---

- Enhance sustainability and interoperability
- Serve as membership criteria for E4S
  - Membership is not required for *inclusion* in E4S
  - Also includes forward-looking draft policies
- Modeled after xSDK community policies
- Multi-year effort led by SDK team
  - Included representation from across ST
  - Multiple rounds of feedback incorporated from ST leadership and membership

---

E4S Community Policies: A commitment to quality improvement

- Enhance sustainability and interoperability
- Serve as membership criteria for E4S
  - Membership is not required for *inclusion* in E4S
- Also includes forward-looking draft policies
- Modeled after xSDK community policies
- Multi-year effort led by SDK team
  - Included representation from across ST
  - Multiple rounds of feedback incorporated from ST leadership and membership

---

SDK lead: Jim Willenbring (SNL)
Request for E4S Policy Status Drove Software Improvements

- L4 Project reviews required gap assessment against E4S Policies
- But no requirement to increase compatibility
- However, teams responded by reducing gaps
- On the right:
  - Flurry of E4S Validation Test Suite PRs prior to reviews
  - Other low hanging fruit changes made too
## E4S and SDKs as platforms are providing tremendous value

<table>
<thead>
<tr>
<th>Activity</th>
<th>SDKs</th>
<th>E4S</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planning</td>
<td>Transparent and collaborative requirements, analysis and design, delivery – better plans, less effort, improved complementarity</td>
<td>Campaign-based portfolio planning coordinated with Facilities, vendors, community ecosystem, non-DOE partners</td>
</tr>
<tr>
<td>Implementation</td>
<td>Leverage shared knowledge, infrastructure, best practices</td>
<td>ID and assist product teams with cross-cutting issues</td>
</tr>
<tr>
<td>Cultivating Community</td>
<td>Within a specific technical domain: Portability layers, LLVM coordination, sparse solvers, etc.</td>
<td>Across delivery and deployment, with software teams, facilities’ staff, with non-DOE users in industry, US agencies</td>
</tr>
<tr>
<td>Resolving issues, sharing solutions</td>
<td>Performance bottlenecks and tricks, coordinated packaging and use of substrate, e.g., Desul for RAJA and Kokkos</td>
<td>Build system bugs and enhancements, protocols for triage, tracking &amp; resolution, leverage across &amp; beyond DOE</td>
</tr>
<tr>
<td>Improving quality</td>
<td>Shared practice improvement, domain-specific quality policies, reduced incidental differences and redundancies, per-commit CI testing of portfolio</td>
<td>Portfolio-wide quality policies with assessment process and quality improvement efforts, documentation portal, portfolio testing on many platforms not available to developers. Address supply chain needs</td>
</tr>
<tr>
<td>Path-finding</td>
<td>Collaborative exploration and development of leading-edge tools and processes</td>
<td>Exploration and development of leading-edge packaging and distribution tools and workflows that provide capabilities and guidance for others</td>
</tr>
<tr>
<td>Training</td>
<td>Collaborative content creation and curation, coordinated training events for domain users, deep, problem-focused solutions using multiple products</td>
<td>Portfolio installation and use, set up of build caches, turnkey and portable installations, container and cloud instances</td>
</tr>
<tr>
<td>Developer experience</td>
<td>Increased community interaction, increased overhead (some devs question value), improved R&amp;D exploration, e.g., variable precision</td>
<td>Low-cost product visibility via doc portal, wide distribution via E4S as from-source/pre-installed/container environment</td>
</tr>
<tr>
<td>User experience</td>
<td>Improve multi-product use, better APIs through improved design, easier understanding of what to use when</td>
<td>Rapid access to latest stable feature sets, installation on almost any HPC system, leadership to laptop</td>
</tr>
<tr>
<td>Scientific Software R&amp;D</td>
<td>Shared knowledge of new algorithmic advances, licensing, build tools, and more</td>
<td>Programmatic cultivation of scientific software R&amp;D not possible at smaller scales</td>
</tr>
<tr>
<td>Community development</td>
<td>Attractive and collaborative community that attracts junior members to join, establishes multi-institutional friendships &amp; careers</td>
<td>Programmatic cultivation of community through outreach and funded opportunities that expand the sustainable membership possibilities</td>
</tr>
</tbody>
</table>

The SDK and E4S platforms provide compelling value for modest cost in ways that become more important going forward.
Expanding the Value and Impact of Software Ecosystems Going Forward
Pre-E4S User Support Model

App teams work with library/tool teams they know, mostly local

App teams and facilities support staff port and debug app code

Facilities support staff have difficulty finding support from library/tool teams except from local teams

Non-DOE users find it very difficult to use DOE libraries and tools. No support beyond basic usage

Industry and Other Agency users
E4S Phase 1 Support Model – Old relationships plus DOE E4S

DOE App Developers and Facilities Users

DOE Library and Tool Developers

DOE E4S Team

DOE Facilities User Support Staff

DOE E4S Team enables a portfolio approach:
• Integrated delivery/support of libs/tools
• Single point of contact for planning and issues

Industry and Other Agency users

Non-DOE users find it very difficult to use DOE libraries and tools. No support beyond basic usage

App teams and facilities support staff port and debug app code

Facilities support staff have difficulty finding support from library/tool teams except from local teams

App teams work with library/tool teams they know, typically local
E4S Phase 2 Support Model – Previous plus commercial E4S

**DOE App Developers and Facilities Users**

- App teams work with library/tool teams they know, locally.

**DOE Library and Tool Developers**

- DOE E4S Team enables a portfolio approach:
  - Integrated delivery/support of libraries/tools
  - Single point of contact for planning and issues

**DOE E4S Team**

- DOE E4S Team facilitates a portfolio approach:
  - Integrated delivery/support of libraries/tools
  - Single point of contact for planning and issues

**DOE Facilities User Support Staff**

- Facilities support staff have difficulty finding support from library/tool teams except from local teams.

**Commercial E4S Team**

- Close interaction:
  - DOE team in charge of strategy/policy
  - Commercial team handles support

**First of a kind interactions:**

- Industry/agencies can acquire support
- Shared costs and benefits with DOE

**Industry and Other Agency users**

- Non-DOE users find it very difficult to use DOE libraries and tools. No support beyond basic usage.
Expanding the Scope of Cost and Benefit Sharing for DOE Software Libraries and Tools

<table>
<thead>
<tr>
<th>Support Phase</th>
<th>Primary Scope</th>
<th>Primary Cost and Benefit Sharing Opportunities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-E4S</td>
<td>Local facility</td>
<td><strong>Local costs and benefits:</strong> Prior to ECP and E4S, libraries and tools were typically strongly connected to the local facility: ANL libs and tools at ALCF, LBL at NERSC, LLNL at Livermore Computing, etc.</td>
</tr>
<tr>
<td>+ ECP E4S</td>
<td>All DOE facilities</td>
<td><strong>DOE complex-shared costs and benefits:</strong> ECP requires, and E4S enables, interfacility availability and use of libs across all facilities: First-class support of ANL libs and tools at other facilities, etc.</td>
</tr>
<tr>
<td>+ Commercial E4S</td>
<td>DOE facilities, other US agencies, industry, and more</td>
<td><strong>Universal shared costs and benefits:</strong> Commercial support of E4S expands cost and benefit sharing to non-DOE entities: DOE costs are lower, software hardening more rapid. US agencies, industry and others can contract for support, gaining sustainable use of E4S software and contributing to its overall support.</td>
</tr>
</tbody>
</table>
Software Sustainability Activities
Key observation: We are scientists, problem solvers. Let’s use science to address our challenges!

Now: Improved SW environments (Jupyter), integration of software specialists as team members, data mining of repos

Next: Research Software Science
- Use scientific method to understand, improve development & use of software for research.
- Incorporate cognitive & social sciences.

[Diagram showing Team Skills Over Time]

Social & Cognitive Specialists
+ Data & SW Specialists
+ Math & CS Specialists
Domain Science Specialists

https://bssw.io/blog_posts/research-software-science-a-scientific-approach-to-understanding-and-improving-how-we-develop-and-use-software-for-research
First-of-a-kind US DOE Workshop

- The Science of Scientific-Software Development and Use
  - Dec 13 – 16, 2021
  - [https://www.orau.gov/SSSDU2021](https://www.orau.gov/SSSDU2021)

- Workshop Brochure available:
  - [https://doi.org/10.2172/1846008](https://doi.org/10.2172/1846008)

- Workshop Report in progress:
  - 3 Priority Research Directions
  - 3 Crosscutting Themes
SSSDU Priority Research Directions

• **PRD1: Develop methodologies and tools to comprehensively improve team-based scientific software development and use**  Focus: Team Impact
  - **Key question:** What practices, processes, and tools can help improve the development, sustainment, evolution, and use of scientific software by teams?

• **PRD2: Develop next-generation tools to enhance developer productivity and software sustainability**  Focus: Developer Impact
  - **Key questions:** How can we create and adapt tools to improve developer effectiveness and efficiency, software sustainability, and support for the continuous evolution of software? How can we support and encourage the adoption of such tools by developers?

• **PRD3: Develop methodologies, tools, and infrastructure for trustworthy software-intensive science**  Focus: Societal Impact
  - **Key questions:** How can we facilitate and encourage effective and efficient reuse of data and software from third parties while ensuring the integrity of our software and the resulting science? How can we provide flexible environments that “bake in” the tracking of software, provenance, and experiment management required to support peer review and reproducibility?
SSSDU Crosscutting Themes

• **Theme 1:** We need to consider both human and technical elements to better understand how to improve the development and use of scientific software.

• **Theme 2:** We need to address urgent challenges in workforce recruitment and retention in the computing sciences with growth through expanded diversity, stable career paths, and the creation of a community and culture that attract and retain new generations of scientists.

• **Theme 3:** Scientific software has become essential to all areas of science and technology, creating opportunities for expanded partnerships, collaboration, and impact.
Takeaways from Expanding Impact in the Future

• Introduction of commercial support for E4S users makes broad benefit & cost sharing possible
• Other agencies & industry can use E4S with confidence because they can acquire support
• The pursuit of effective and efficient scientific software can itself be informed by science
Developing software for GPU systems
Heterogeneous accelerated-node computing

**Accelerated node computing:** Designing, implementing, delivering, & deploying agile software that effectively exploits heterogeneous node hardware

- Execute on the largest systems … AND on today and tomorrow’s laptops, desktops, clusters, …

- We view *accelerators* as any compute hardware specifically designed to accelerate certain mathematical operations (typically with floating point numbers) that are typical outcomes of popular and commonly used algorithms. We often use the term GPUs synonymously with accelerators.

---

**Diagram credit:** Andrew Siegel
Kokkos/RAJA

• Two distinct products: Kokkos and RAJA
  – Both originate in NNSA
  – RAJAs main funding/usage in NNSA
  – Kokkos gets half its funding from NNSA, but >70% of users outside of NNSA

• Learn from and leverage each other’s work
  – Desul – common atomics library
  – Memory management – different strategies

• Other options: OpenMP, vendor-specific (CUDA, HIP, SYCL)
New algorithms highlights – batched computations, mixed precision

The “coopetition” model:

Step 1: Collaborative design space exploration

Step 2: Adaptation and implementation in each library
Batched Sparse Linear Algebra
Phase 1 Implementation

Scope and objectives

• Establish goals and needs for batched sparse linear algebra implementations
• Iterate over interface design choices for relevant ECP applications and their implementation options
• Limit memory usage and transfers for Phase 1 implementations
• Deliver performance that maximizes the compute throughput and/or attained memory bandwidth

Impact

• Enable batched sparse linear algebra routines on GPUs
• Allow input from ECP applications, numerical library developers, and hardware vendors
• Provide new interoperability layers so that applications can easily use sparse batched solvers and preconditioners

Speedup of Ginkgo batched iterative solvers over dgbsv

• Initial results using matrices from the XGC framework of the WDMApp ECP project
• Speedup for 5 Picard iterations using batched BiCGStab on 3 different GPUs over the banded solver on CPU
• The results compare against the current best solution with the required solver functionality on Skylake (CPU)

Project accomplishment

• Batched band Phase 1 implementation
• Batched sparse iterative and direct Phase 1 implementation
• Preparations for inclusion of batched functions in xSDK libraries and ECP applications
• Progress report on batched sparse linear algebra Phase 1 implementation

Deliverables

The report is available at [https://confluence.exascaleproject.org/display/STMS05/xSDK+Project+Documents](https://confluence.exascaleproject.org/display/STMS05/xSDK+Project+Documents) in the file “Milestone 45 Report_Batched Sparse LA Phase 1 Implementation” – Ask Piotr Luszczek for copy
The opportunity: Low-precision arithmetic is fast (and dangerous)

- We currently witness
  - the integration of low precision special function units into HPC hardware (NVIDIA Tensor Core, AMD Matrix Engine, etc.),
  - a widening gap between compute power and memory bandwidth,
  - and the increasing adoption of low precision floating point formats (fp16, bf16, etc.).

- … the US Exascale Computing Project decided for the aggressive step of building a multiprecision focus effort to take on the challenge of designing and engineering novel algorithms exploiting the compute power available in low precision and adjusting the communication format to the application specific needs.
Step 1: Concurrent exploration of the algorithm and software space

- **In cross-laboratory expert teams, we focus on:**
  - Mixed precision dense direct solvers (MAGMA and SLATE);
  - Mixed precision sparse direct solvers (SuperLU);
  - Mixed precision multigrid (on a theoretical level and in hypre);
  - Mixed precision FFT (heFFTE);
  - Mixed precision preconditioning (Ginkgo, Trilinos);
  - Separating the arithmetic precision from the memory precision (Ginkgo);
  - Mixed precision Krylov solvers (theoretical analysis, Ginkgo, Trilinos);

- **Mixed precision algorithms acknowledge and boost the GPU usage**
  - Algorithm development primarily focuses on GPU hardware (*Summit, Frontier*);
  - Latest evaluations on **NVIDIA A100** (Perlmutter), **AMD MI100** (Spock), **Intel Gen9 GPU**

- **Integrating mixed precision technology as production-ready implementation into ECP software products allows for the smooth integration into ECP applications.**
Step 2: Incorporate lessons learned into library ecosystem

For library interoperability and mixed precision usage:

- PETSc develops an abstraction layer to device solvers (vendor libraries, Kokkos Kernels, etc.) that allows flexible composition of Krylov solves in mixed-precision;

- hypre already supports the compilation in different precisions and work now focuses on compiling multiple precisions at a time to compose algorithms out of routines running in different precision formats;

- Ginkgo makes the “memory accessor” integration-ready for other software libraries;

- Kokkos and KokkosKernels implements support for compiling in IEEE754 half precision;

- SLATE contains mixed precision algorithms and templates the working precision; and

- MAGMA compiles in different precisions (z,c,d,s).
Status of early-access system experience

Excellent progress toward Exascale readiness and a lot more to do
Performance portability

- Portability strategy:
  - Strategy 1: Isolate performance-impacting code to select kernels, write own CUDA, HIP, SYCL
  - Strategy 2: Product uses Kokkos and RAJA as primary portability layers
  - Blend 1 & 2: Provide both

- Notes:
  - No ST products use OpenMP directly for GPU portability but
  - Kokkos and RAJA have OpenMP backends as an option

<table>
<thead>
<tr>
<th>Package</th>
<th>NVIDIA GPU</th>
<th>AMD GPU</th>
<th>Intel GPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArborX</td>
<td>support (Kokkos)</td>
<td>support (Kokkos)</td>
<td>in progress (Kokkos-SYCL backend)</td>
</tr>
<tr>
<td>DTK</td>
<td>support (Kokkos)</td>
<td>support (Kokkos)</td>
<td>in progress (Kokkos-SYCL backend)</td>
</tr>
<tr>
<td>Ginkgo</td>
<td>support (CUDA)</td>
<td>support (HIP)</td>
<td>support (DPC++)</td>
</tr>
<tr>
<td>heFFTe</td>
<td>support (CUDA)</td>
<td>support (HIP)</td>
<td>support (DPC++)</td>
</tr>
<tr>
<td>hypre</td>
<td>support (CUDA, RAJA, Kokkos)</td>
<td>support (HIP)</td>
<td>in progress (DPC++)</td>
</tr>
<tr>
<td>libEnsemble</td>
<td>supports apps running on GPUs</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>MAGMA</td>
<td>support (CUDA)</td>
<td>support (HIP)</td>
<td>planned</td>
</tr>
<tr>
<td>MFEM</td>
<td>support (CUDA)</td>
<td>support (HIP)</td>
<td>support (DPC++)</td>
</tr>
<tr>
<td>PETSc</td>
<td>support (CUDA</td>
<td>Kokkos)</td>
<td>support (HIP</td>
</tr>
<tr>
<td>SLATE</td>
<td>support (CUDA)</td>
<td>support (HIP)</td>
<td>in progress (DPC++)</td>
</tr>
<tr>
<td>STRUMPACK</td>
<td>support (CUDA)</td>
<td>support (HIP)</td>
<td>in progress (SYCL, oneAPI)</td>
</tr>
<tr>
<td>Sundials</td>
<td>support (CUDA, RAJA)</td>
<td>support (HIP, RAJA)</td>
<td>support (SYCL, oneAPI, RAJA)</td>
</tr>
<tr>
<td>SuperLU</td>
<td>support (CUDA)</td>
<td>support (HIP)</td>
<td>in progress (DPC++, oneAPI)</td>
</tr>
<tr>
<td>Tasmanian</td>
<td>support (CUDA)</td>
<td>support (HIP)</td>
<td>support (DPC++), but not in spack</td>
</tr>
<tr>
<td>Trilinos</td>
<td>support (Kokkos)</td>
<td>support (Kokkos)</td>
<td>in progress (Kokkos-SYCL backend)</td>
</tr>
</tbody>
</table>
The E4S Two-Step

• Step 1:
  – Migrate existing MPI-CPU code on top of E4S:
    • All E4S libraries & tools compile & run well on CPU architectures, including multi-threading & (improving) vectorization
    • Pick a performance portability approach (as described above)
    • Rewrite your loops for parallel portability, e.g., rewrite in Kokkos or RAJA
    • Link against E4S CPU versions of relevant libraries
  – Potential benefits:
    • Migrating to E4S on a stable computing platform, easy to migrate incrementally and detect execution diffs
    • Single build via Spack
    • Potential for using build caches (10x rebuild time improvement)
    • Single point of access to documentation
    • Increased quality of user experience via E4S support, E4S and SDK quality commitments
    • Preparation for Step 2…
The E4S Two-Step

• Step 2: Turn on GPU build
  – Builds with GPU backends (especially if using Kokkos or RAJA)
  – Transition to GPU is a debugging and adaptation exercise
  – Track growth in E4S GPU capabilities as E4S products improve GPU offerings

• Consider interactions with E4S commercial support team
  – Pay someone for support
  – Get advice on product choices
    • DOE teams generally can’t give you good advice on which solver or IO library to use
      – Like asking Microsoft and Apple to tell whether to purchase a PC or Mac
GPU Efforts Summary

- One legacy of ECP & E4S will be a SW stack that is portable across Nvidia, AMD, and Intel GPUs
- Porting to modern GPUs requires almost everything to be done on the GPUs
- Common refactoring themes:
  - Async under collectives
  - Batch execution
  - Pre-allocation and highly concurrent assembly: Sparse matrix assembly via COO format with atomics
- Two+hybrid portability models are used:
  - **Use portability layers**: Kokkos, RAJA or (eventually) OpenMP w target offload (OpenACC?)
  - **Isolate & and custom write**: Isolate perf-portable kernels and write your own CUDA, HIP, SYCL backend
  - **Hybrid**: Use portability layers, customize key kernels only
- Explore low-precision arithmetic: Substantial benefit (and risks)
- Rely more on third-party reusable libraries and tools.
Summary

• Using a portfolio-based approach for HPC software is about going together vs going alone.
• While products vary greatly, we all face the same frontiers: Evolving demands and systems.
• Success on the frontier is important for all HPC configurations: leadership to laptop.
• The new and evolving E4S and SDK platforms enable better, faster and cheaper, in net.
• A collective approach, E4S, enables new relationships with facilities, vendors, apps, industry.
• Potential NERSC user interests from ECP libraries and tools efforts:
  – Spack – can be transformative itself, independent of E4S.
  – Latest MPI, IO capabilities – available in E4S first.
  – Kokkos/RAJA portability layers – lessons learned, starting point for your own layer, direct use.
  – Lessons learned in new algorithms for highly-concurrent nodes.
  – Longer term: leverage E4S and SDKs for better/faster/cheaper use of open source.
  – Longer term: Social/cognitive aspects of technical software teams?
Thank you

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.

Thank you to all collaborators in the ECP and broader computational science communities. The work discussed in this presentation represents creative contributions of many people who are passionately working toward next-generation computational science.