The Extreme-scale Scientific Software Stack (E4S) and its Promise for the Exascale Computing Era

Michael A. Heroux, Sandia National Laboratories
Director of Software Technology, US Exascale Computing Project

ScalA19: 10th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems
November 18, 2019
ECP Context for E4S
ECP’s three technical areas have the necessary components to meet national goals

**Performant mission and science applications @ scale**

- **Aggressive RD&D Project**
  - Application Development (AD)
  - Develop and enhance the predictive capability of applications critical to the DOE
  - **24 applications** including national security, to energy, earth systems, economic security, materials, and data

- **Mission apps & integrated S/W stack**
  - 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

- **Deployment to DOE HPC Facilities**
  - 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

- **Hardware tech advances**
ECP software technologies overview

### 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
ECP Software Technology Leadership Team

Mike Heroux, Software Technology Director
Mike has been involved in scientific software R&D for 30 years. His first 10 were at Cray in the LIBSCI and scalable apps groups. At Sandia he started the Trilinos and Mantevo projects, is author of the HPCG benchmark for TOP500, and leads productivity and sustainability efforts for DOE.

Jonathan Carter, Software Technology Deputy Director
Jonathan has been involved in the support and development of HPC applications for chemistry, the procurement of HPC systems, and the evaluation of novel computing hardware for over 25 years. He currently a senior manager in Computing Sciences at Berkeley Lab.

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.

Lois Curfman McInnes, Math Libraries (2.3.3)
Lois is a senior computational scientist in the Mathematics and Computer Science Division of ANL. She has over 20 years of experience in high-performance numerical software, including development of PETSc and leadership of multi-institutional work toward sustainable scientific software ecosystems.

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.

Rob Neely, NNSA ST (2.3.6)
Rob is an Associate Division Leader in the Center for Applied Scientific Computing (CASC) at LLNL, chair of the Weapons Simulation & Computing Research Council, and lead for the Sierra Center of Excellence. His efforts span applications, CS research, platforms, and vendor interactions.
We work on products applications need now and into the future

Key themes:
• Exploration/development of new algorithms/software for emerging HPC capabilities:
• High-concurrency node architectures and advanced memory & storage technologies.
• Enabling access and use via standard APIs.

Software categories:
• The next generation of well-known and widely used HPC products (e.g., MPICH, OpenMPI, PETSc)
• Some lesser used but known products that address key new requirements (e.g., Kokkos, RAJA, Spack)
• New products that 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>
### ECP ST Subprojects

**- WBS**
**- Name**
**- PIs**
**- Project Managers (PMs)**

### ECP ST Stats

**- 33 L4 subprojects**
**- 10 PI/PC same**
**- 23 PI/PC different**

<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>Software Technology</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.3.1</td>
<td>Programming Models &amp; Runtimes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.3.1.01</td>
<td>PMR SDK</td>
<td>Thakur, Rajeev</td>
<td></td>
</tr>
<tr>
<td>2.3.1.07</td>
<td>Exascale MPI (MPICH)</td>
<td>Shende, Sameer</td>
<td>Shende, Sameer</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>Bosilica, 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>Baden, Scott</td>
<td>Hargrove, Paul (and PI)</td>
</tr>
<tr>
<td>2.3.1.16</td>
<td>SICM</td>
<td>Lang, Michael</td>
<td>Vigil, Brittny</td>
</tr>
<tr>
<td>2.3.1.17</td>
<td>OMPI-X</td>
<td>Bernholdt, David</td>
<td>TBD PMA Through ORNL PMO</td>
</tr>
<tr>
<td>2.3.1.18</td>
<td>RAJA/Kokkos</td>
<td>Trott, Christian</td>
<td>Trott, Christian</td>
</tr>
<tr>
<td>2.3.1.19</td>
<td>Argos: Low-level resource management for the OS and runtime</td>
<td>Beckman, Pete</td>
<td>Gupta, Rinku</td>
</tr>
<tr>
<td>2.3.2</td>
<td>Development Tools</td>
<td>Vetter, Jeff</td>
<td></td>
</tr>
<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 HPCToolkit to Measure and Analyze Code Performance on Exascale Platforms</td>
<td>Mellor-Crummey, John</td>
<td>Mellor-Crummey, John</td>
</tr>
<tr>
<td>2.3.2.10</td>
<td>PROTEAS-TUNE</td>
<td>Vetter, Jeff</td>
<td>Glassbrook, Dick</td>
</tr>
<tr>
<td>2.3.2.11</td>
<td>SOLLVE: Scaling OpenMP with LLVM for Exascale</td>
<td>Chapman, Barbara</td>
<td>Kong, Martin</td>
</tr>
<tr>
<td>2.3.2.12</td>
<td>FLANG</td>
<td>McCormick, Pat</td>
<td>Perry-Holby, Alexis</td>
</tr>
<tr>
<td>2.3.3</td>
<td>Mathematical Libraries</td>
<td>McInnes, Lois</td>
<td></td>
</tr>
<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>Smith, Barry</td>
<td>Munson, Todd</td>
</tr>
<tr>
<td>2.3.3.07</td>
<td>STRUMPACK/SuperLU/FFTX: sparse direct solvers, preconditioners, and FFT libraries</td>
<td>Li, Xiaoye</td>
<td>Li, Xiaoye</td>
</tr>
<tr>
<td>2.3.3.12</td>
<td>Enabling Time Integrators for Exascale Through SUNDIALS/ Hypre</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>Turner, John</td>
<td>TBD PMA Through ORNL PMO</td>
</tr>
<tr>
<td>2.3.4</td>
<td>Data and Visualization</td>
<td>Ahrens, James</td>
<td></td>
</tr>
<tr>
<td>2.3.4.01</td>
<td>Data and Visualization Software Development Kit</td>
<td>Atkins, Chuck</td>
<td>Atkins, Chuck</td>
</tr>
<tr>
<td>2.3.4.09</td>
<td>ADIOS Framework for Scientific Data on Exascale Systems</td>
<td>Klasky, Scott</td>
<td>TBD PMA Through ORNL PMO</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>VelociC: 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 Byna, Suren</td>
<td>Bagha, Neelam</td>
<td></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>
<tr>
<td>2.3.5</td>
<td>Software Ecosystem and Delivery</td>
<td>Munson, Todd</td>
<td></td>
</tr>
<tr>
<td>2.3.5.01</td>
<td>Software Ecosystem and Delivery Software Development Kit</td>
<td>Willenbring, James M</td>
<td>Willenbring, James M</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>2.3.6</td>
<td>NNSA ST</td>
<td>Neely, Rob</td>
<td></td>
</tr>
<tr>
<td>2.3.6.01</td>
<td>LANL ATDM</td>
<td>Mike Lang</td>
<td>Vandenbusch, Tanya Marie</td>
</tr>
<tr>
<td>2.3.6.02</td>
<td>LBNL ATDM</td>
<td>Becky Springmeyer</td>
<td>Gamblin, Todd</td>
</tr>
<tr>
<td>2.3.6.03</td>
<td>SNL ATDM</td>
<td>Jim Stewart</td>
<td>Trujillo, Gabrielle</td>
</tr>
</tbody>
</table>
Iterative, incremental delivery of a modular, interoperable, and deployable software stack
Delivering a Modular, Interoperable, and Deployable Software Stack

<table>
<thead>
<tr>
<th>Objectives</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Lower barrier to use ST products</td>
</tr>
<tr>
<td>• Lower barrier to enable Facilities to install all or parts</td>
</tr>
<tr>
<td>• Enable interoperability between ST products</td>
</tr>
<tr>
<td>• Enable uniform APIs where possible</td>
</tr>
<tr>
<td>• Establish an open, hierarchical software architecture to enable collaboration with other US Agencies and international HPC software institutions.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Challenges</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Large diverse group of ST products</td>
</tr>
<tr>
<td>• Different project management styles</td>
</tr>
<tr>
<td>• Lack of initial drivers for integration</td>
</tr>
<tr>
<td>• Complex software ecosystem with many players.</td>
</tr>
</tbody>
</table>
ECP ST Planning Process: Hierarchical, three-phase, cyclical

**Baseline**
- **FY20–23 Baseline Plan**
  - High level Definitions
  - Q2 FY19 start
  - FY20 Base plan
  - FY21–23 planning packages

**Annual Refinement**
- **FY Refine Baseline Plan**
  - As Needed
  - Basic activity definitions
  - 6 months prior to FY
  - 4–6 P6 Activities/year
  - Each activity:
    - % annual budget
    - Baseline start/end
    - High level description

**Per Activity**
- **Detailed Plan**
  - Complete activity definitions
  - 8 weeks prior to start
  - High-fidelity description
  - Execution strategy
  - Completion criteria
  - Personnel details

**Two-level Change Control**
- Changes to Cost, Scope, and Schedule
  - **Minor**
    - Lightweight Review in Jira, L3 and L2 leads
  - **Major**
    - Change Control Board Review, ECP leadership
    - Variance Recorded in Jira
  - Proceed with Execution
# KPP-3 Definition

<table>
<thead>
<tr>
<th>KPP ID</th>
<th>Description of Scope</th>
<th>Threshold KPP</th>
<th>Objective KPP</th>
<th>Verification Action/Evidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>KPP-1</td>
<td>Performance of scientific and national security applications relative to today’s</td>
<td>50% of selected applications achieve Figure of merit* improvement ≥50</td>
<td>100% of selected applications achieve Figure of merit improvement stretch goal</td>
<td>Independent assessment of measured results and report that threshold goal is met</td>
</tr>
<tr>
<td></td>
<td>performance</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>KPP-2</td>
<td>Broaden the reach of exascale science and mission capability</td>
<td>50% of selected applications can execute their challenge problem*</td>
<td>100% of selected applications can execute their challenge problem stretch goal</td>
<td>Independent assessment of mission application readiness</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>KPP-3</td>
<td>Productive and Sustainable Software Ecosystem</td>
<td>Software teams meet 50% of their weighted impact goals*</td>
<td>Software teams meet 100% of their weighted impact stretch goals</td>
<td>Independent assessment verifying threshold goal is met</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>KPP-4</td>
<td>Enrich the HPC Hardware Ecosystem</td>
<td>Vendors meet 80% of all the PathForward milestones</td>
<td>Vendors meet 100% of all the PathForward milestones</td>
<td>Independent assessment of the impact and timeliness of PathForward milestones</td>
</tr>
</tbody>
</table>
KPP-3: Focus on capability integration

• **Capability:** Any significant product functionality, including existing features adapted to the pre-exascale and exascale environments, that can be integrated into a client environment.

• **Capability Integration:** Complete, sustainable integration of a significant product capability into a client environment in a pre-exascale environment (tentative score) and in an exascale environment (confirmed score).

• **Product:** Capabilities are integrated into primary products chosen from a product dictionary.

• **Primary products examples:**
  – MPI is commonly known by users. MPICH and OpenMPI both provide implementations of that product.
  – Fortran is a product. Flang is a particular Fortran product. LLVM is a backend for some Fortran compilers.
  – FFT is a product. FFTX, FFT-ECP provide FFT capabilities through interchangeable interfaces.
  – C++ is a product. Clacc provides capabilities for Clang, as does LLVM.

• **Details:** See Confluence page export.
E4S Overview
Levels of Integration

ECP ST Individual Products

- Standard workflow
- Existed before ECP

ST Products

Source: ECP L4 teams; Non-ECP Developers; Standards Groups
Delivery: Apps directly; spack; vendor stack; facility stack

ECP ST Open Product Integration Architecture

SDKs

- Group similar products
- Make interoperable
- Assure policy compliant
- Include external products

Source: ECP SDK teams; Non-ECP Products (policy compliant, spackified)
Delivery: Apps directly; spack install sdk; future: vendor/facility

E4S

- Build all SDKs
- Build complete stack
- Containerize binaries

Source: ECP E4S team; Non-ECP Products (all dependencies)
Delivery: spack install e4s; containers; CI Testing
Extreme-scale Scientific Software Stack (E4S)

- **E4S**: A Spack-based distribution of ECP ST and related and dependent software tested for interoperability and portability to multiple architectures
- Provides distinction between SDK usability / general quality / community and deployment / testing goals
- Will leverage and enhance SDK interoperability thrust

- Oct 2018: E4S 0.1 - 24 full, 24 partial release products
- Jan 2019: E4S 0.2 - 37 full, 10 partial release products
- Nov 2019: E4S 1.0 - 50 full, 5 partial release products

Lead: Sameer Shende (U Oregon)
E4S 1.0 Full Release: 50 ECP Packages and all dependencies

- Adios
- AML
- Argobots
- Bolt
- Caliper
- Darshan
- Dyninst
- Faodel
- Flecsi
- Gasnet
- GEOPM
- Gotcha
- HDF5
- HPCToolkit
- Hypre
- Kokkos
- Legion
- Libnrm
- Libquo
- Magma
- Mercury
- MFEM
- MPICH
- MPIFileUtils
- Ninja
- OpenMPI
- PAPI
- Papyrus
- Parallel
- netCDF
- PDT
- PETSc
- Qthreads
- Raja
- Rempi
- SCR
- Spack
- Strumpack
- Sundials
- SuperLU
- SZ
- Tasmanian
- TAU
- TAU
- Trilinos
- Turbine
- Umpire
- UnifyFS
- UPC++ Veloc
- Zfp

All ST products will be released through E4S

Partial (in-progress) Release: ASCENT, Catalyst, Flang, libEnsemble, LLVM, Visit
ECP ST Product Dictionary

Created by Mike Heroux, last modified by Vivek Kale on 2019-10-05

The ECP Software Technology (ST) Product Dictionary is the official list of publicly recognized names to which ECP ST efforts contribute. While ST teams use an expanded product namespace, the list on this page indicates the eventual access point for ST product development efforts.

Official Product Dictionary
• Standardizes Product Definitions
• Enables consistent AD-CD-ST dependencies

This table lists in **bold** only those products that are typically recognizable to users. We call these **primary products**. Examples:

1. MPI is commonly known by users. MPICH and OpenMPI both provide implementations of that product.
2. Fortran is a product. Flang is a particular Fortran product. LLVM is a backend for some Fortran compilers.
3. FFT is a product. FTX, FFT-ECP provide FFT capabilities through interchangeable interfaces.
4. C++ is a product. Clacc provides capabilities for Clang, as does LLVM.

**Subproducts** are listed underneath with the primary product name as a prefix.

In some cases, a product may be listed as a primary product even if it is not widely recognized by the user community. In this case, the product developer needs to be able to address these additional criteria:

1. The product is intended for stand-alone delivery and not be included in some other product in the future.
2. There is a credible sustainability path for the product and the development team provides some evidence that it can support the product in a sustainable way, including staffing and funding.

**Deployment scope** is meant to give others a sense of how widely integrated a product is in the HPC ecosystem and how far along it is in maturing toward usability. The deployment scope categories are:

- **Broad**: Widely used in the HPC ecosystem, expected to be available and usable by many applications.
- **Moderate**: Some use in applications but not ubiquitous.
- **Experimental**: Still under development and used by collaborators and friendly users.

https://confluence.exascaleproject.org/display/1ST/ECP+ST+Product+Dictionary
### 70 Official Products
- Will change over time
- Establishes delivery plan

<table>
<thead>
<tr>
<th>Product</th>
<th>URL</th>
<th>Description/Notes</th>
<th>Deployment Scope</th>
<th>Technical Area</th>
<th>Point of Contact</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. ADIOS</td>
<td><a href="https://github.com/ornladios/ADIOS2">https://github.com/ornladios/ADIOS2</a></td>
<td>I/O and data management library for storage I/O, interconnects, and file system.</td>
<td>Broad</td>
<td>Data &amp; Viz</td>
<td>@Scott Klasky</td>
</tr>
<tr>
<td>2. AID</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AID: STAT</td>
<td><a href="https://github.com/LLNL">https://github.com/LLNL</a></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AID: Archer</td>
<td><a href="https://github.com/PRL">https://github.com/PRL</a></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AID: FLIT</td>
<td><a href="https://github.com/PRL">https://github.com/PRL</a></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>66. Vendor Stack</td>
<td></td>
<td>ECP ST design, development and demonstration efforts that are integrated into one or more of the Vendor software stacks</td>
<td>Experimental</td>
<td>Software Ecosystem</td>
<td>TBD</td>
</tr>
<tr>
<td>67. VeloC</td>
<td><a href="https://github.com/ECP-VeloC/VELOC">https://github.com/ECP-VeloC/VELOC</a></td>
<td>Scalable checkpoint-restart library.</td>
<td>Broad</td>
<td>Data &amp; Viz</td>
<td>@Franck Cappello</td>
</tr>
<tr>
<td>68. VTK-m</td>
<td><a href="http://m.vtk.org">http://m.vtk.org</a></td>
<td>Parallel on-node visualization toolkit.</td>
<td>Broad</td>
<td>Data &amp; Viz</td>
<td>@Kenneth Moreland</td>
</tr>
<tr>
<td>69. xSDK</td>
<td><a href="https://xsdk.info">https://xsdk.info</a></td>
<td>Math libraries meta product combining the most popular HPC math libraries into a compatible collection.</td>
<td>Moderate</td>
<td>Math libraries</td>
<td>@Ulrike Meier Yang</td>
</tr>
<tr>
<td>70. zfp</td>
<td><a href="https://github.com/LLNL/zfp">https://github.com/LLNL/zfp</a></td>
<td>In-memory data compression library.</td>
<td>Moderate</td>
<td>Data &amp; Viz</td>
<td>@Peter Lindstrom, @Terry Turton</td>
</tr>
</tbody>
</table>

[https://confluence.exascaleproject.org/display/1ST/ECP+ST+Product+Dictionary](https://confluence.exascaleproject.org/display/1ST/ECP+ST+Product+Dictionary)
# Partial List of Jira Dependency Issues

<table>
<thead>
<tr>
<th>Key</th>
<th>Summary</th>
<th>Assignee</th>
<th>Reporter</th>
<th>Status</th>
<th>Resolution</th>
<th>Created</th>
</tr>
</thead>
<tbody>
<tr>
<td>INT-691</td>
<td>PETSc/TAO ↔ Spack</td>
<td>Unassigned</td>
<td>Todd Munson</td>
<td>APPROVED</td>
<td>Approved</td>
<td>2019-09-10</td>
</tr>
<tr>
<td>INT-686</td>
<td>HDF5 ↔ PETSc/TAO</td>
<td>Unassigned</td>
<td>Todd Munson</td>
<td>APPROVED</td>
<td>Approved</td>
<td>2019-09-10</td>
</tr>
<tr>
<td>INT-690</td>
<td>PETSc/TAO ↔ xSDK</td>
<td>Unassigned</td>
<td>Todd Munson</td>
<td>APPROVED</td>
<td>Approved</td>
<td>2019-09-10</td>
</tr>
<tr>
<td>INT-687</td>
<td>MPI-IO ↔ PETSc/TAO</td>
<td>Unassigned</td>
<td>Todd Munson</td>
<td>APPROVED</td>
<td>Approved</td>
<td>2019-09-10</td>
</tr>
<tr>
<td>INT-689</td>
<td>OpenMP ↔ PETSc/TAO</td>
<td>Unassigned</td>
<td>Todd Munson</td>
<td>APPROVED</td>
<td>Approved</td>
<td>2019-09-10</td>
</tr>
<tr>
<td>INT-677</td>
<td>libEnsemble ↔ xSDK</td>
<td>Unassigned</td>
<td>Todd Munson</td>
<td>APPROVED</td>
<td>Approved</td>
<td>2019-09-10</td>
</tr>
<tr>
<td>INT-681</td>
<td>BLAS ↔ PETSc/TAO</td>
<td>Unassigned</td>
<td>Todd Munson</td>
<td>APPROVED</td>
<td>Approved</td>
<td>2019-09-10</td>
</tr>
</tbody>
</table>
Jira Dependency Issue Type

- Official ST and AD Product lists enable rigorous dependency management.
- New Jira Dependency issue type.
- Explicit dependency connections between AD/CD/ST products.

![Jira Dependency Issue View](image)
ECP SW Technology
Software Architecture –
Spack Software
Distribution System
Spacc enables Software distribution for HPC

- Spack automates the build and installation of scientific software
- Packages are templated, so that users can easily tune for the host environment

No installation required: clone and go

```bash
$ git clone https://github.com/spack/spack
$ spack install hdf5
```

Simple syntax enables complex installs

```bash
$ spack install hdf5@1.10.5
$ spack install hdf5@1.10.5 %clang@6.0
$ spack install hdf5@1.10.5 +threadssafe
$ spack install hdf5@1.10.5 cppflags="-O3 -g3"
$ spack install hdf5@1.10.5 target=haswell
$ spack install hdf5@1.10.5 +mpi ~mpich@3.2
```

- Ease of use of mainstream tools, with flexibility needed for HPC tuning
- Major victories:
  - ARES porting time on a new platform was reduced from 2 weeks to 3 hours
  - Deployment time for 1,300-package stack on Summit supercomputer reduced from 2 weeks to a 12-hour overnight build
  - Used by teams across ECP to accelerate development
Spack is being used on many of the top HPC systems

- Official deployment tool for the U.S. Exascale Computing Project
- 7 of the top 10 supercomputers
- High Energy Physics community
  - Fermilab, CERN, collaborators
- Astra (Sandia)
- Fugaku (Japanese National Supercomputer Project)
Spack is used worldwide!

Over 3,400 software packages
Over 2,000 monthly active users
Over 400 contributors (and growing)

Active users of Spack documentation site for one month
https://spack.readthedocs.io
Spack strategy is to enable exascale software distribution on both bare metal and containers

1. **New capabilities to make HPC packaging easy and automated**
   - Optimized builds and package binaries that exploit the hardware
   - Workflow automation for facilities, developers, and users
   - Strong integration with containers as well as facility deployments

2. **Develop exascale enhancements to container runtimes**
   - Understand and automate performance/portability tradeoffs
   - Optimized containers & build automation for exascale HPC centers
   - Enable portability from laptops to exascale

3. **Outreach to users**
   - Ongoing working groups, Best practice guides
   - Tutorials, workshops, BOFs for Spack and Containers

4. **Collaboration across ECP**
   - Work with HI and other areas to build service infrastructure
   - Facilitate curation of packages through E4S and facilities
   - Ongoing ECP user support
Spacc heavily involved in the ECP CI project.

- We have added security features to the open source GitLab product.
  - Integration with center identity management
  - Integration with schedulers like SLURM, LSF

- We are democratizing testing at Livermore Computing
  - Users can run tests across 30+ machines by editing a file
  - Previously, each team had to administer own servers

- ECP sites are deploying GitLab CI for users
  - All HPC centers can leverage these improvements
  - NNSA labs plan to deploy common high-side CI infrastructure
  - We are developing new security policies to allow external open source code to be tested safely on key machines
Integration and Interoperability: E4S on AWS

- E4S AWS public image ami-063e830287b86155c (US-West-2 Oregon) has following container runtimes:
  - Docker
  - Shifter
  - Singularity
  - Charliecloud
- Spack with base PMR components
- E4S full featured Singularity image
  - (exascaleproject/sdk:AHM19)
- Used in ISC-HPC 2019 tutorials
- Used as base image for NASA GEOS-Chem E4S public image
- Resources provided by AWS AI/ML team
Reproducible, Customizable Container Builds & Spack Mirrors

- E4S provides base images and recipes for building Docker containers based on SDKs
  - Git: [https://github.com/UO-OACISS/e4s](https://github.com/UO-OACISS/e4s)
  - Base images released (September 2019):
    - UBI 7.6 (RHEL Universal Binary Image for container builds) for x86_64
    - Centos 7.6 for x86_64
    - Ubuntu 18.04 for x86_64
    - UBI 7.6 (RHEL) for ppc64le

- E4S provides build caches for Spack for native bare-metal as well as container builds based installation of ST products
  - Build caches: [https://oaciss.uoregon.edu/e4s](https://oaciss.uoregon.edu/e4s)
    - The build cache model can be extended to target platforms, and can be managed by facilities staff when appropriate.
E4S Build Cache Binaries

Spack E4S Build Cache

Last updated: 20-Sep-2019 11:07 PST

Click on one of the packages below to see a list of all available variants.

1148 Spack binaries in the build cache

Search

adios2@2.4.0 arml@0.1.0 argobots@1.0.rc1 arpack-ng@3.7.0 autconf@2.69 automake@1.16.1 axl@0.1.1 bdf@0.1.0.5 binutils@2.32 bison@3.0.5 bmi@develop bolt@1.0.rc1 boost@1.70.0 butterfylpack@1.0.1 bzip2@1.0.8 c-blosc@1.17.0 cairo@1.16.0 caliper@2.0.1 cinch@develop cmake@3.15.1 cmake@3.15.3 cuda@10.0.130 cuda@10.1.243 curl@7.63.0 darshan-runtime@3.1.7 darshan-util@3.1.7 diffutils@3.7 double-conversion@2.0.1 dircmp@1.1.0
dyninst@10.1.0

Click on the full spec link to find out more.

<table>
<thead>
<tr>
<th>Link</th>
<th>Arch</th>
<th>OS</th>
<th>Compiler</th>
<th>Created</th>
<th>Full Hash</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full Spec</td>
<td>x86_64</td>
<td>centos7</td>
<td>gcc@7.3.0</td>
<td>18-Sep-2019 19:07</td>
<td>m66bcmvvlkyl5%38qg6mmux7myuiers</td>
</tr>
<tr>
<td>Full Spec</td>
<td>x86_64</td>
<td>centos7</td>
<td>gcc@7.3.0</td>
<td>18-Sep-2019 19:11</td>
<td>v2wlu3j78x84n8x2mcmm5oi76o</td>
</tr>
<tr>
<td>Full Spec</td>
<td>x86_64</td>
<td>centos7</td>
<td>gcc@7.3.0</td>
<td>15-Sep-2019 22:08</td>
<td>czwmd656m2c9m547o7ajjx</td>
</tr>
<tr>
<td>Full Spec</td>
<td>x86_64</td>
<td>centos7</td>
<td>gcc@7.3.0</td>
<td>15-Sep-2019 22:13</td>
<td>nolxytynw5gqeo3%p3qjajm2c5653k</td>
</tr>
<tr>
<td>Full Spec</td>
<td>x86_64</td>
<td>centos7</td>
<td>gcc@7.3.0</td>
<td>18-Sep-2019 19:16</td>
<td>lko6qpcl58pwxw8lthx6k3spxfylyx</td>
</tr>
<tr>
<td>Full Spec</td>
<td>x86_64</td>
<td>centos7</td>
<td>gcc@7.3.0</td>
<td>18-Sep-2019 19:21</td>
<td>c255c3765g758uai5w5wcbnwqjybgfz</td>
</tr>
</tbody>
</table>

eflutis@0.176 environment-modules@4.3.0 er@0.0.3 expat@2.2.5 faodel@1.19061 findutils@4.6.0 flatcc@0.5.3 flecsi@develop flex@2.6.4 font-util@1.3.2
fontconfig@2.12.3 fontsproto@2.1.3 freetype@2.9.1 gasnet@2019.3.0 gcc@7.3.0 gdbm@1.18.1 gettext@0.19.8.1 git@2.21.0 glib@2.35.1 glm@0.9.7.1
GPL3proto@1.4.17 gmp@6.1.2 googletest@1.8.1 gotcha@0.10.2 gotcha@1.0.2 gperf@3.0.4 harfbuzz@2.3.1 hdf5@1.10.5 help2man@1.47.8 hpctoolkit@2019.08.14
hwloc@1.11.11 hpyre@2.16.0 hpyre@2.17.0 icu4c@64.1 inputproto@2.3.2 intel-mkl@2019.3.199 intel-tbb@2019.4 intel-xed@2019.03.03 isl@0.18 jsoncpp@1.9.1

https://oaciss.uoregon.edu/e4s
Reproducible Container Builds using E4S Base Images

- PMR SDK base image (UBI 7.6) has Spack build cache mirror and GPG key installed.
- Base image has GCC and MPICH configured for MPICH ABI level replacement (with system MPI).
- Customized container build using binaries from E4S Spack build cache for fast deployment.
- No need to rebuild packages from the source code.
- Same recipe for container and native bare-metal builds with Spack!
Spack Build Caches from E4S Base Images

- x86_64 build cache
  - 914 binaries
  - 40 GB on disk

- IBM Power 9 (ppc64le) build cache
  - 101 binaries
  - 2.6 GB on disk
  - early stages of effort
  - Initial ARM 64 build cache is underway
ECP SW Technology
Software Architecture –
SDKs
Software Development Kits (SDKs): Key delivery vehicle for ECP
A collection of related software products (packages) where coordination across package teams improves usability and practices, and foster community growth among teams that develop similar and complementary capabilities

• **Domain scope**
  Collection makes functional sense

• **Interaction model**
  How packages interact; compatible, complementary, interoperable

• **Community policies**
  Value statements; serve as criteria for membership

• **Meta-infrastructure**
  Invokes build of all packages (Spack), shared test suites

• **Coordinated plans**
  Inter-package planning. Augments autonomous package planning

• **Community outreach**
  Coordinated, combined tutorials, documentation, best practices

---

**ECP ST SDKs: Grouping similar products for collaboration & usability**

Programming Models & Runtimes Core
Tools & Technologies
Compilers & Support
Math Libraries (xSDK)
Viz Analysis and Reduction
Data mgmt., I/O Services & Checkpoint/Restart

“Unity in essentials, otherwise diversity”
xSDK compatible package: Must satisfy mandatory xSDK policies:

M1. Support xSDK community GNU Autoconf or CMake options.
M2. Provide a comprehensive test suite.
M3. Employ user-provided MPI communicator.
M4. Give best effort at portability to key architectures.
M5. Provide a documented, reliable way to contact the development team.
M6. Respect system resources and settings made by other previously called packages.
M7. Come with an open source license.
M8. Provide a runtime API to return the current version number of the software.
M9. Use a limited and well-defined symbol, macro, library, and include file name space.
M10. Provide an accessible repository (not necessarily publicly available).
M11. Have no hardwired print or IO statements.
M12. Allow installing, building, and linking against an outside copy of external software.
M13. Install headers and libraries under <prefix>/include/ and <prefix>/lib/.
M14. Be buildable using 64 bit pointers. 32 bit is optional.
M15. All xSDK compatibility changes should be sustainable.
M16. The package must support production-quality installation compatible with the xSDK install tool and xSDK metapackage.

Also recommended policies, which currently are encouraged but not required:

R1. Have a public repository.
R2. Possible to run test suite under valgrind in order to test for memory corruption issues.
R3. Adopt and document consistent system for error conditions/exceptions.
R4. Free all system resources it has acquired as soon as they are no longer needed.
R5. Provide a mechanism to export ordered list of library dependencies.
R6. Provide versions of dependencies.

xSDK member package: Must be an xSDK-compatible package, and it uses or can be used by another package in the xSDK, and the connecting interface is regularly tested for regressions.

We welcome feedback. What policies make sense for your software?

https://xsdk.info/policies
xSDK version 0.4.0: December 2018

Each xSDK member package uses or can be used with one or more xSDK packages, and the connecting interface is regularly tested for regressions.

Tested on key machines at ALCF, NERSC, OLCF, also Linux, Mac OS X

Impact: Improved code quality, usability, access, sustainability
Foundation for work on performance portability, deeper levels of package interoperability

December 2018
- 17 math libraries
- 2 domain components
- 16 mandatory xSDK community policies
- Spack xSDK installer

Domain components
- Reacting flow, etc.
- Reusable.

Libraries
- Solvers, etc.
- Interoperable.

Frameworks & tools
- Doc generators.
- Test, build framework.

SW engineering
- Productivity tools.
- Models, processes.

Extreme-Scale Scientific Software Development Kit (xSDK)
xSDK version 0.5: November 2019
(21 math libs, 2 domain-specific packages)

- AMReX
- ButterflyPACK
- DTK
- deal.ii
- Ginkgo
- hypre
- libEnsemble
- MAGMA
- MFEM
- Omega_h
- PETSc/TAO
- PHIST
- PLASMA
- preCICE
- PUMI
- SLEPc
- STRUMPACK
- SUNDIALS
- SuperLU
- Tasmanian
- Trilinos
- Pflotran
- Alquimia

Notes:

- Growth:
  - 5 in release 0.1.
  - 7 in 0.2
  - 9 in 0.3
  - 19 in 0.4
  - 23 in 0.5

- You do not need to build all packages.
- We build and test all packages.
- Any subset is guaranteed to build if using the same build parameters, platforms.
- Similar builds should work or require less effort for success.
SDK “Horizontal” Grouping: Key Quality Improvement Driver

Horizontal (vs Vertical) Coupling
- Common substrate
- Similar function and purpose
  - e.g., compiler frameworks, math libraries
- Potential benefit from common Community Policies
  - Best practices in software design and development and customer support
- Used together, but not in the long vertical dependency chain sense
- Support for (and design of) common interfaces
  - Commonly an aspiration, not yet reality

Horizontal grouping:
- Assures X=Y.
- Protects against regressions.
- Transforms code coupling from heroic effort to turnkey.
Development of xSDK Community Policies

- Community policies now available on Github for revision control and to preserve history: https://github.com/xsdk-project/xsdk-community-policies
- Established process on how to change or add policies
- Newest release: 0.5.0

Also recommended policies, which currently are encouraged but not required:

**R1.** Have a public repository.
**R2.** Possible to run test suite under valgrind in order to test for memory corruption issues.
**R3.** Adopt and document consistent system for error conditions/exceptions.
**R4.** Free all system resources it has acquired as soon as they are no longer needed.
**R5.** Provide a mechanism to export ordered list of library dependencies.
**R6.** Provide versions of dependencies.
**R7.** Have README, SUPPORT, LICENSE, and CHANGELGLOG file in top directory.

Changes in 0.5.0:
- New recommended policy R7
- Dropped the requirement to detect MPI 2 features in M3
- Made various editorial changes in M5, M13, M15, and R2 for clarification or to fix typos.

xSDK compatible package: Must satisfy mandatory xSDK policies:

- M1. Support xSDK community GNU Autoconf or CMake options.
- M2. Provide a comprehensive test suite.
- M3. Employ user-provided MPI communicator.
- M4. Give best effort at portability to key architectures.
- M5. Provide a documented, reliable way to contact the development team.
- M6. Respect system resources and settings made by other previously called packages.
- M7. Come with an open source license.
- M8. Provide a runtime API to return the current version number of the software.
- M9. Use a limited and well-defined symbol, macro, library, and include file name space.
- M10. Provide an accessible repository (not necessarily publicly available).
- M11. Have no hardwired print or IO statements.
- M12. Allow installing building, and linking against an outside copy of external software.
- M13. Install headers and libraries under <prefix>/include and <prefix>/lib/.
- M14. Be buildable using 64-bit pointers. 32-bit is optional.
- M15. All xSDK compatibility changes should be sustainable.
- M16. The package must support production-quality installation compatible with the xSDK install tool and xSDK metapackage.

We welcome feedback. What policies make sense for your software?

https://xsdk.info/policies
Integration of Community Policies

- Potential new xSDK members will fill out compatibility form: https://github.com/xsdk-project/xsdk-policy-compatibility
- xSDK team will check for compatibility and approve if compatible
- Designing and implementing xSDK policy checker to automate the policy compliance checking.
  - Implemented checkers for a few policy rules. Examples:
    - M3: Employ user-provided MPI Communicator: Policy checker lists the file names that contain MPI_COMM_WORLD.
    - M7: Come with an open source license. Policy checker search license files at the top directly of the source and scan the headers of source code files.
Processes for xSDK release and delivery

• 2-level release process
  – xSDK
    • Ensure and test compatibility of mostly independent package releases
  – xSDK member packages
    • Achieve compatibility with xSDK community policies prior to release
      – https://github.com/xsdk-project/xsdk-policy-compatibility
    • Have a Spack package
    • Port to target platforms
    • Provide user support

• Obtaining the latest release: https://xsdk.info/releases

• Draft xSDK package release process checklist:
  – https://docs.google.com/document/d/16y2bL1RZg8wke0vY8c97ssvhRYNez34Q4QGg4LoIEUk/edit?usp=sharing
There are 5 L4 projects to define and/or enhance SDKs

- Each L3 area has an L4 project devoted to SDK definition and coordination across the portfolio
- Software ecosystem L4 project focuses on packaging
  - Spack
  - Containers
- Strong coordination with HI Software Deployment projects
- Drives milestone-based planning
ECP ST SDKs will span all technology areas

**Motivation:** Properly chosen cross-team interactions will build relationships that support interoperability, usability, sustainability, quality, and productivity within ECP ST.

**Action Plan:** Identify product groupings where coordination across development teams will improve usability and practices, and foster community growth among teams that develop similar and complementary capabilities.

<table>
<thead>
<tr>
<th>PMR Core (17)</th>
<th>Compilers and Support (7)</th>
<th>Tools and Technology (11)</th>
<th>xSDK (16)</th>
<th>Visualization Analysis and Reduction (9)</th>
<th>Data mgmt, I/O Services, Checkpoint restart (12)</th>
<th>Ecosystem/E4S at-large (12)</th>
</tr>
</thead>
<tbody>
<tr>
<td>QUO</td>
<td>openarc</td>
<td>TAU</td>
<td>hypre</td>
<td>ParaView</td>
<td>SCR</td>
<td>mpiFileUtils</td>
</tr>
<tr>
<td>Papyrus</td>
<td>Kitsune</td>
<td>HPCToolkit</td>
<td>FleSCI</td>
<td>Catalyst</td>
<td>FAODEL</td>
<td>TriBITS</td>
</tr>
<tr>
<td>SICM</td>
<td>LLVM</td>
<td>Dyninst Binary Tools</td>
<td>MFEM</td>
<td>VTK-m</td>
<td>ROMIO</td>
<td>MarFS</td>
</tr>
<tr>
<td>Legion</td>
<td>CHiLL autotuning comp</td>
<td>Gotcha</td>
<td>Kokkoskernels</td>
<td>SZ</td>
<td>Mercury (Mochi suite)</td>
<td>G UFI</td>
</tr>
<tr>
<td>Kokkos (support)</td>
<td>LLVM openMP comp</td>
<td>Caliper</td>
<td>Trilinos</td>
<td>zfp</td>
<td>HDF5</td>
<td>Intel GEOPM</td>
</tr>
<tr>
<td>RAJA</td>
<td>OpenMP V &amp; V</td>
<td>PAPI</td>
<td>SUNDIALS</td>
<td>VisIt</td>
<td>Parallel netCDF</td>
<td>BEE</td>
</tr>
<tr>
<td>CHAI</td>
<td>Flang/LLVM Fortran comp</td>
<td>Program Database Toolkit</td>
<td>PETSc/TAO</td>
<td>VisIt</td>
<td>libEnsemble</td>
<td></td>
</tr>
<tr>
<td>ParRSEC*</td>
<td></td>
<td>Search (random forests)</td>
<td>libEnsemble</td>
<td>ASCENT</td>
<td>Cinema</td>
<td></td>
</tr>
<tr>
<td>DARMA</td>
<td></td>
<td>Siboka</td>
<td>STRUMPACK</td>
<td>Darshan</td>
<td>UnifyCR</td>
<td></td>
</tr>
<tr>
<td>GASNet-EX</td>
<td></td>
<td>C2C</td>
<td>SuperLU</td>
<td>VeloC</td>
<td>IOSS</td>
<td></td>
</tr>
<tr>
<td>Qthreads</td>
<td></td>
<td>Sonar</td>
<td>ForTrilinos</td>
<td>NRM</td>
<td>ArgoContainers</td>
<td></td>
</tr>
<tr>
<td>BOLT</td>
<td></td>
<td></td>
<td>SLATE</td>
<td>ArgoContainers</td>
<td></td>
<td>Spack</td>
</tr>
<tr>
<td>UPC++</td>
<td></td>
<td></td>
<td>MAGMA</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MPICH</td>
<td></td>
<td></td>
<td>DTK</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Open MPI</td>
<td></td>
<td></td>
<td>Tasmanian</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Umpire</td>
<td></td>
<td></td>
<td>TuckerMPI</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Legend:**
- **PMR**
- **Tools**
- **Math Libraries**
- **Data and Vis**
- **Ecosystems and delivery**
Data SDK: HDF5 API Initiative

- Adding The HDF Group (THG) to the Data & Vis SDK project to assist in various aspects of improving the use of the HDF5 API
  - Support and training
    - Direct support and training will be provided to ECP applications and ST projects to assist in the performant use of HDF5
  - Compatible APIs with other I/O libraries
    - DataLib (pnetcdf) and ADIOS will develop interfaces compatible with the HDF5 API, possibly through the HDF5 Virtual Object Layer
    - THG will provide support to the I/O projects to assist in achieving this compatibility

- Both Kitware and THG will identify the I/O usage patterns w.r.t. metadata operations, bulk data operations, access patterns.
  - This will help identify usage patterns that can be improved on by applications and ST projects
  - Also will help identify usage patterns that HDF5 can work to optimize for.
SDK Summary

• SDKs will help reduce complexity of delivery:
  – Hierarchical build targets.
  – Distribution of software integration responsibilities.

• New Effort: Started in April 2018, fully established in August 2018.

• Extending the SDK approach to all ECP ST domains.
  – SDKs create a horizontal coupling of software products, teams.
  – Create opportunities for better, faster, cheaper – pick all three.

• First concrete effort: Spack target to build all packages in an SDK.
  – Decide on good groupings.
  – Not necessarily trivial: Version compatibility issues, Coordination of common dependencies.

• Longer term:
  – Establish community policies, enhance best practices sharing.
  – Provide a mechanism for shared infrastructure, testing, training, etc.
  – Enable community expansion beyond ECP.
E4S Documentation Initiative
ECP Software Product Documentation Goals

• Provide a single online location for accurate product descriptions for ECP software products.

• Derived requirements:
  – Sustainable: Must be integrated into software team workflows.
  – Incremental: Must build on community approaches to providing this kind of information.
  – Extensible: Must be usable by any open source software team.

• Strategy:
  – Use the open source community approach of specially-name files in software repositories.
  – Adopt commonly used file names when available.
  – Identify new information items not already being requested.
  – Develop new special file names for information beyond what is already captured.
  – Create web-based raking tool to capture information from product repositories and present in summary form on a webpage.
Status of efforts

• The math libraries “SDK” effort (xSDK) has developed a recommended policy for member product teams to provide the following files in the root directory of the default branch of the product repository:
  – README[.md],
  – SUPPORT[.md],
  – LICENSE[.md],
  – CHANGELOG[.md]
Next phase: Test process, determine other special files

- Beyond the README, SUPPORT, LICENSE and CHANGELOG files, what information do we need from each product?

- Examples from https://github.com/kmindi/special-files-in-repository-root/blob/master/README.md are:
  - CONTRIBUTING – how to contribute
  - CONTRIBUTORS – who contributed
  - ACKNOWLEDGMENTS
  - CODE_OF_CONDUCT
  - ISSUE_TEMPLATE, PULL_REQUEST_TEMPLATE
  - CODEOWNERS
Some recent new ideas for expansion

- Structure requirements for special files, e.g., README has a certain structure.
- README has status panels: CI passing, etc.
- Pointer to Spack recipe.
- SUPPORTEDPLATFORMS – Describe platforms, especially leadership platforms.
- Where is the product pre-installed? Facility and vendor stacks, Linux distros, etc.
- Pointer to usage stats: NERSC makes theirs’ available, ORNL generates them but keeps internal.
- Documentation: Could be incorporated into README file, if brief. Or README could point to DOCUMENTATION file, which would contain documentation, point to detailed documentation or both.
- Could have an FAQ: Primary text with some content not found in the product special files, e.g., information about how Spack supports modules generation.
- VERSION - Contains the current product version number and the corresponding hash in the repo.
Putting it all together: Products + SDKs + E4S

• In ECP, teams increasingly need to ensure that their libraries and components work together
  – Historically, HPC codes used very few dependencies
• Now, groups of teams work together on small releases of “Software Development Kits”
• SDKs will be rolled into a larger, periodic release.

E4S
ECP-wide software release
https://e4s.io
Preparing for Exascale Platforms
Department of Energy (DOE) Roadmap to Exascale Systems

An impressive, productive lineup of *accelerated node* systems supporting DOE’s mission

<table>
<thead>
<tr>
<th>Pre-Exascale Systems [Aggregate Linpack (Rmax) = 323 PF!]</th>
<th>First U.S. Exascale Systems</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>2012</strong></td>
<td><strong>2016</strong></td>
</tr>
<tr>
<td>Titan (9)</td>
<td></td>
</tr>
<tr>
<td>ORNL</td>
<td></td>
</tr>
<tr>
<td>Cray/AMD/NVIDIA</td>
<td></td>
</tr>
<tr>
<td>Mira (21)</td>
<td></td>
</tr>
<tr>
<td>ANL</td>
<td></td>
</tr>
<tr>
<td>IBM BG/Q</td>
<td></td>
</tr>
<tr>
<td>Theta (24)</td>
<td></td>
</tr>
<tr>
<td>ANL</td>
<td></td>
</tr>
<tr>
<td>Cray/Intel KNL</td>
<td></td>
</tr>
<tr>
<td>Cori (12)</td>
<td></td>
</tr>
<tr>
<td>LBNL</td>
<td></td>
</tr>
<tr>
<td>Cray/Intel Xeon/KNL</td>
<td></td>
</tr>
<tr>
<td>Summit (1)</td>
<td></td>
</tr>
<tr>
<td>ORNL</td>
<td></td>
</tr>
<tr>
<td>IBM/NVIDIA</td>
<td></td>
</tr>
<tr>
<td>Sequoia (10)</td>
<td></td>
</tr>
<tr>
<td>LLNL</td>
<td></td>
</tr>
<tr>
<td>IBM BG/Q</td>
<td></td>
</tr>
<tr>
<td>Trinity (6)</td>
<td></td>
</tr>
<tr>
<td>LANL/SNL</td>
<td></td>
</tr>
<tr>
<td>Cray/Intel Xeon/KNL</td>
<td></td>
</tr>
<tr>
<td>Sierra (2)</td>
<td></td>
</tr>
<tr>
<td>LLNL</td>
<td></td>
</tr>
<tr>
<td>IBM/NVIDIA</td>
<td></td>
</tr>
<tr>
<td>Aurora</td>
<td></td>
</tr>
<tr>
<td>ANL</td>
<td></td>
</tr>
<tr>
<td>Intel/Cray</td>
<td></td>
</tr>
<tr>
<td>Crossroads</td>
<td></td>
</tr>
<tr>
<td>LLNL</td>
<td></td>
</tr>
<tr>
<td>TBD</td>
<td></td>
</tr>
<tr>
<td>Perlmutter</td>
<td></td>
</tr>
<tr>
<td>LBNL</td>
<td></td>
</tr>
<tr>
<td>Cray/AMD/NVIDIA</td>
<td></td>
</tr>
<tr>
<td>First U.S. Exascale Systems</td>
<td></td>
</tr>
<tr>
<td>Frontier</td>
<td></td>
</tr>
<tr>
<td>ORNL</td>
<td></td>
</tr>
<tr>
<td>TBD</td>
<td></td>
</tr>
<tr>
<td>Perlmutter</td>
<td></td>
</tr>
<tr>
<td>LBNL</td>
<td></td>
</tr>
<tr>
<td>Cray/AMD/NVIDIA</td>
<td></td>
</tr>
<tr>
<td>Aurora</td>
<td></td>
</tr>
<tr>
<td>ANL</td>
<td></td>
</tr>
<tr>
<td>Intel/Cray</td>
<td></td>
</tr>
<tr>
<td>Perlmutter</td>
<td></td>
</tr>
<tr>
<td>LBNL</td>
<td></td>
</tr>
<tr>
<td>Cray/AMD/NVIDIA</td>
<td></td>
</tr>
<tr>
<td>First U.S. Exascale Systems</td>
<td></td>
</tr>
<tr>
<td>Frontier</td>
<td></td>
</tr>
<tr>
<td>ORNL</td>
<td></td>
</tr>
<tr>
<td>TBD</td>
<td></td>
</tr>
<tr>
<td>Perlmutter</td>
<td></td>
</tr>
<tr>
<td>LBNL</td>
<td></td>
</tr>
<tr>
<td>Cray/AMD/NVIDIA</td>
<td></td>
</tr>
<tr>
<td>Aurora</td>
<td></td>
</tr>
<tr>
<td>ANL</td>
<td></td>
</tr>
<tr>
<td>Intel/Cray</td>
<td></td>
</tr>
<tr>
<td>Perlmutter</td>
<td></td>
</tr>
<tr>
<td>LBNL</td>
<td></td>
</tr>
<tr>
<td>Cray/AMD/NVIDIA</td>
<td></td>
</tr>
<tr>
<td>First U.S. Exascale Systems</td>
<td></td>
</tr>
<tr>
<td>Frontier</td>
<td></td>
</tr>
<tr>
<td>ORNL</td>
<td></td>
</tr>
<tr>
<td>TBD</td>
<td></td>
</tr>
<tr>
<td>Perlmutter</td>
<td></td>
</tr>
<tr>
<td>LBNL</td>
<td></td>
</tr>
<tr>
<td>Cray/AMD/NVIDIA</td>
<td></td>
</tr>
<tr>
<td>Aurora</td>
<td></td>
</tr>
<tr>
<td>ANL</td>
<td></td>
</tr>
<tr>
<td>Intel/Cray</td>
<td></td>
</tr>
<tr>
<td>Perlmutter</td>
<td></td>
</tr>
<tr>
<td>LBNL</td>
<td></td>
</tr>
<tr>
<td>Cray/AMD/NVIDIA</td>
<td></td>
</tr>
<tr>
<td>First U.S. Exascale Systems</td>
<td></td>
</tr>
<tr>
<td>Frontier</td>
<td></td>
</tr>
<tr>
<td>ORNL</td>
<td></td>
</tr>
<tr>
<td>TBD</td>
<td></td>
</tr>
<tr>
<td>Perlmutter</td>
<td></td>
</tr>
<tr>
<td>LBNL</td>
<td></td>
</tr>
<tr>
<td>Cray/AMD/NVIDIA</td>
<td></td>
</tr>
<tr>
<td>Aurora</td>
<td></td>
</tr>
<tr>
<td>ANL</td>
<td></td>
</tr>
<tr>
<td>Intel/Cray</td>
<td></td>
</tr>
<tr>
<td>Perlmutter</td>
<td></td>
</tr>
<tr>
<td>LBNL</td>
<td></td>
</tr>
<tr>
<td>Cray/AMD/NVIDIA</td>
<td></td>
</tr>
</tbody>
</table>
Some Observations on Accelerator Programming

• Wait until you are ready.
  – A lot of flux in software stacks.
  – Let others (us) work on stabilizing the ecosystem.

• Start with a concrete, robust development environment.
  – Many people start with CUDA, even if the target is different.
  – Portability abstractions, e.g., Kokkos, can be reasonable too.

• Expose massive concurrency, hiding massive latency
  – GPUs are multi tera-operation per second devices.
  – Need a lot of uninterrupted work to hide overheads.

• Note: Designing for GPUs will also generally improve CPU performance.
  – Modern CPUs benefit a lot from concurrency.
  – CPU memory systems perform better with more concurrent requests.
Working out Our On-node Parallel Programming Strategies

• CUDA/HIP/SYCL/OpenCL – Explicit parallel threading.
  – We made the transition to using CUDA. Supporting AMD, Intel challenging.

• OpenMP – Markup-based approach
  – Need high tolerance for markup, performance portability uncertain.

• Kokkos – parallel\_for/\_reduce/\_scan patterns
  – User provides loop body, C++ functor/lambda.

• Side note on C – has long been a parallel programming language.
  – Speculative execution, branch prediction
  – “C is not a low-level language”, David Chisnall.
<table>
<thead>
<tr>
<th>Package</th>
<th>NVIDIA GPU</th>
<th>AMD GPU</th>
<th>Intel Xe</th>
</tr>
</thead>
<tbody>
<tr>
<td>DTK</td>
<td>support (Kokkos)</td>
<td>in progress (Kokkos)</td>
<td>in progress (Kokkos)</td>
</tr>
<tr>
<td>Ginkgo</td>
<td>support (CUDA)</td>
<td>in progress (HIP)</td>
<td>no support</td>
</tr>
<tr>
<td>hypre</td>
<td>support (CUDA/OMP4.5/RAJA/Kokkos)</td>
<td>no support</td>
<td>no support</td>
</tr>
<tr>
<td>MAGMA</td>
<td>support (CUDA)</td>
<td>support (OpenCL)</td>
<td>in progress (OpenCL/SYCL)</td>
</tr>
<tr>
<td>MFEM</td>
<td>support (CUDA/OCCA/RAJA)</td>
<td>support (HIP/OCCA)</td>
<td>in progress (OCCA/OpenCL)</td>
</tr>
<tr>
<td>PETSc</td>
<td>support (CUDA)</td>
<td>support (ViennaCL/OpenCL)</td>
<td>no support</td>
</tr>
<tr>
<td>STRUMPACK</td>
<td>in progress (CUDA)</td>
<td>no support</td>
<td>no support</td>
</tr>
<tr>
<td>SUNDIALS</td>
<td>support (CUDA/RAJA/OMP4.5)</td>
<td>no support</td>
<td>in progress (OMP4.5)</td>
</tr>
<tr>
<td>SuperLU</td>
<td>support (CUDA)</td>
<td>no support</td>
<td>no support</td>
</tr>
<tr>
<td>Tasmanian</td>
<td>support (CUDA)</td>
<td>no support</td>
<td>no support</td>
</tr>
<tr>
<td>Trilinos</td>
<td>support (Kokkos)</td>
<td>in progress (Kokkos)</td>
<td>in progress (Kokkos)</td>
</tr>
</tbody>
</table>
How E4S and OpenHPC Work Together

- **OpenHPC**
  - Provides installation and base components for installing and running cluster to supercomputer
  - Provides generic binaries for all systems (working on targeted builds)

- **E4S**
  - Built from scratch for your system
  - Focus on upper layers of system software stack (libraries, runtimes, etc.)
  - Targeted for capability-class machines

Spack, MPI, RMs, base installation
Builds for exascale machines, Scaled and additional submissions to TSC
1st Workshop on NSF and DOE High Performance Computing Tools

Dates: July 10-11, 2019
Location: 312 Lillis, University of Oregon, Eugene, OR 97403

Workshop Agenda

Wednesday, July 10, 2019: 312 Lillis, University of Oregon

8:00am - 8:45am: Registration

8:45am - 9:15am: Welcome, introductions, David Conover (VPRI, UO), Vipin Chaudhary (Program Director, CISE/OAC, NSF), Jonathan Carter (Deputy Director, Software Technology, ECP, and Deputy for Science, LBL) [slides] [video]

9:15am -
10:30am -
11:00am -
11:00am -
12:30pm -
2:00pm - 3:30pm: Spack tutorial - Todd Gamblin and Greg Becker (LLNL) [slides] [video]
2:00pm - 3:30pm: Spack tutorial - Todd Gamblin and Greg Becker (LLNL) [video]

3:30 pm -
4:00pm -
4:00pm - 5:30pm: Spack tutorial - Todd Gamblin and Greg Becker (LLNL) [video 1] [video 2] [video 3] [video 4] [video 5]

6:30pm - 9:30pm: Working dinner at the Jordan Schnitzer Museum of Art, UO

Thursday, July 11, 2019: 312 Lillis, University of Oregon

9:00am - 9:30am: Unifying Software Distribution in ECP - Todd Gamblin (LLNL) [slides] [video]
9:00am - 9:30am: Containers for HPC - Andrew Youngs (Sandia National Laboratories, NM) [slides] [video]
9:30am - 10:00am: Software deployment at DOE facilities - David Montoya (Los Alamos National Laboratory, NM) [slides] [video]
10:00am - 10:30am: E4S - Sameer Shende (University of Oregon) [slides] [video]
10:30am - 11:00am: Break
11:00am - 11:30am: Overview of software architecture - Todd Gamblin, LLNL
11:30am - 12:30pm: Hands-on, applying Spack to applications.
12:30pm - 2:00pm: Lunch: 440 Lillis
2:00pm - 3:30pm: Hands-on, applying Spack to applications.
3:30pm - 4:00pm: Break
4:00pm - 5:00pm: Hands-on, Spack and E4S.
5:00pm - 6:15pm: Closing remarks, planning for the next workshop - Jonathan Carter (Lawrence Berkeley National Laboratory)
6:30pm: Dinner at Excelsior Inn

https://oaciss.uoregon.edu/NSFDOE19/agenda.html
Extending Collaborations

• E4S/SDK collaborations make sense with:
  – HPC open source development projects:
    • deal.II (NSF math library),
    • PHIST (DLR Germany library).
  – Application teams in search of open source software foundation:
    • NSF science teams.
    • NOAA, others.
  – Commercial open source packagers/distributors
    • OpenHPC.
    • HPC systems vendors.
  – HPC systems facilities:
    • SDSC, TACC, others.
E4S: Building on top of previous efforts

- E4S did not emerge from nothing.
- Leveraging the work of many others.
- HPC Linux: Work done at U of Oregon, and at ParaTools.
- IDEAS-Classic: xSDK – the original SDK continuing under ECP.
- Spack – Pre-dates E4S.
E4S Wrap Up
### Software Technology Ecosystem

**Levels of Integration**

<table>
<thead>
<tr>
<th>Product</th>
<th>Source</th>
<th>Delivery</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECP ST Individual Products</td>
<td>ECP L4 teams; Non-ECP Developers; Standards Groups</td>
<td>Apps directly; spack; vendor stack; facility stack</td>
</tr>
<tr>
<td>ST Products</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDKs</td>
<td>ECP SDK teams; Non-ECP Products (policy compliant, spackified)</td>
<td>Apps directly; spack install sdk; future: vendor/facility</td>
</tr>
<tr>
<td>E4S</td>
<td>ECP E4S team; Non-ECP Products (all dependencies)</td>
<td>spack install e4s; containers; CI Testing</td>
</tr>
</tbody>
</table>

**Levels of Integration**

- Group similar products
- Make interoperable
- Assure policy compliant
- Include external products
- Build all SDKs
- Build complete stack
- Containerize binaries

**Levels of Integration**

- Standard workflow
- Existed before ECP

**ECP ST Open Product Integration Architecture**
## E4S Summary

### What E4S is not

- A closed system taking contributions only from DOE software development teams.
- A monolithic, take-it-or-leave-it software behemoth.
- A commercial product.
- A simple packaging of existing software.

### What E4S is

- Extensible, open architecture software ecosystem accepting contributions from US and international teams.
- Framework for collaborative open-source product integration.
- A full collection of compatible software capabilities and a manifest of a la carte selectable software capabilities.
- Vehicle for delivering high-quality reusable software products in collaboration with others.
- The conduit for future leading edge HPC software targeting scalable next-generation computing platforms.
- A hierarchical software framework to enhance (via SDKs) software interoperability and quality expectations.
Final Remarks

• Software Development Kits (SDKs) provide a new software coordination mechanism:
  – Do not significantly increase the number of software products in the HPC ecosystem.
  – Provide intermediate build, install and test targets for E4S (reducing complexity).
  – Enable development teams to improve and standardize look-and-feel, best practices, policies.
  – Improve interoperability, interchangeability of similar products.
  – Fosters communities of developers in a cooperative/competitive environment.
  – **Provides integration point for SDK-compatible, non-ECP products.**

• The Extreme-scale Scientific Software Stack (E4S) provides a complete HPC software stack:
  – Does not significantly increase the number of software products in the HPC ecosystem.
  – Provides a coordinated approach to building, installing and testing HPC software.
  – Tests some products with a subset of configurations on a subset of platforms.
  – Improves stability of the ST stack so that any subset is more stable.
  – **Provides a complete software stack, including non-ECP products.**

• **The SDK/E4S architecture is open:**
  – Enables light-weight coordinated collaboration among open source software teams.
  – ECP seeks collaboration: libraries/tools and SDKs, facilities and E4S.