The US Exascale Project Software Stack: Why It Matters to You

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

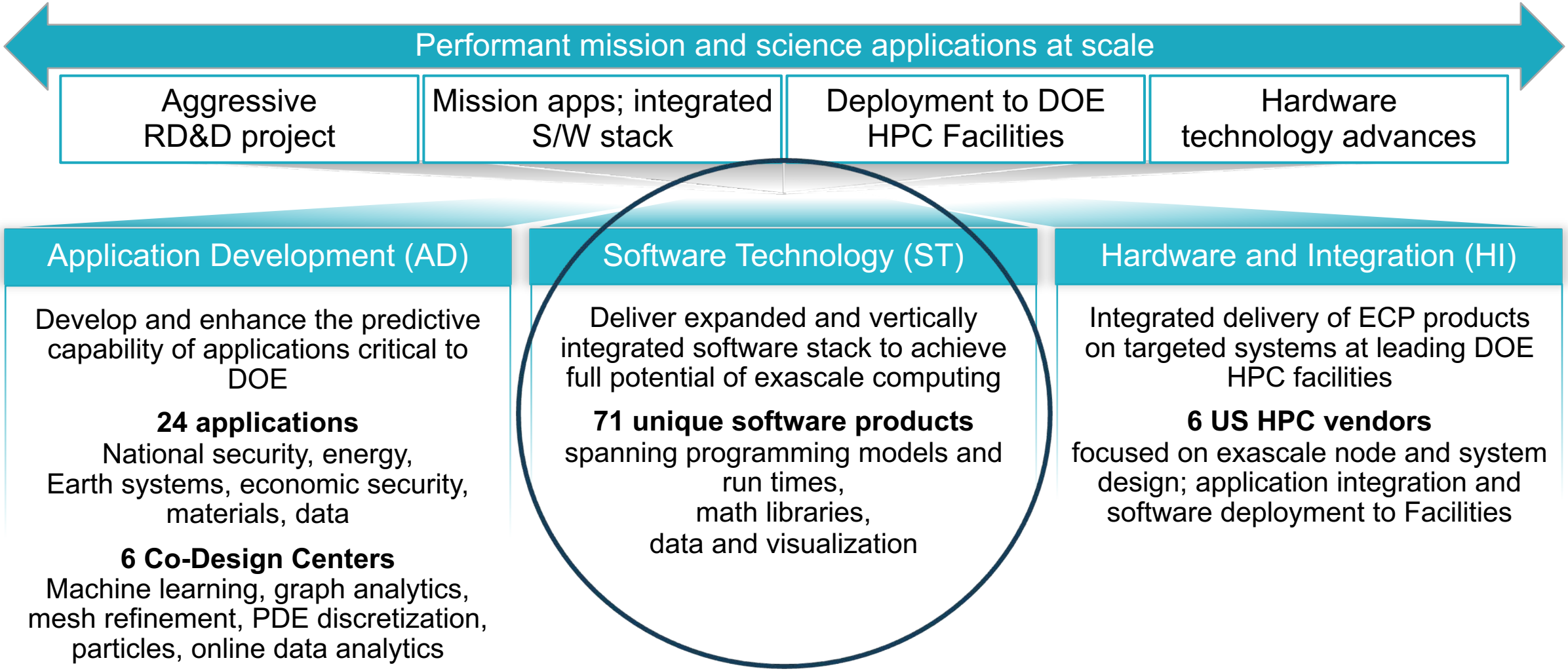
RTE Seminar, June 14, 2021

NNSA National Nuclear Security Administration

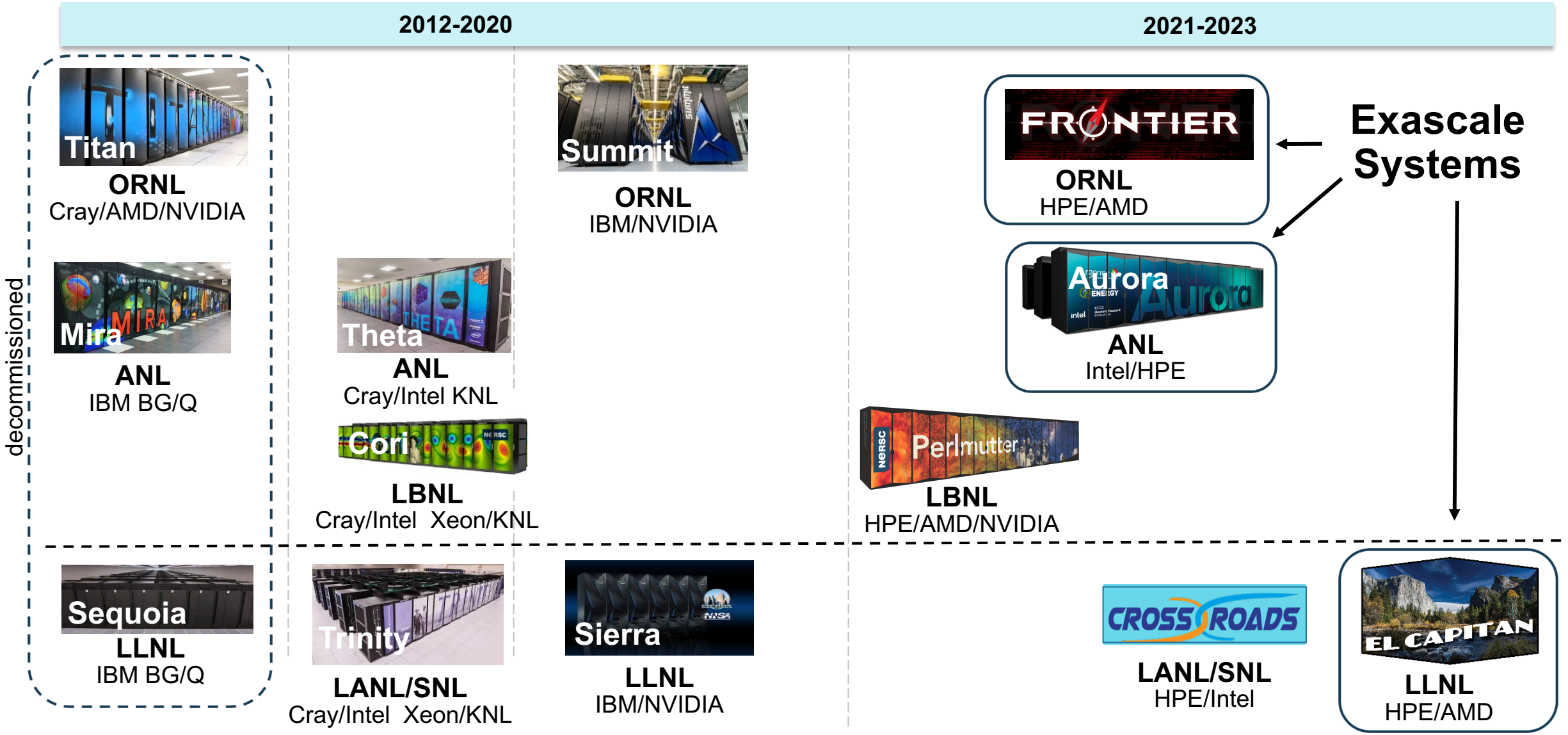
U.S. DEPARTMENT OF ENERGY | Office of Science

Brief Intro to the Exascale Computing Project (ECP)

ECP's holistic approach uses co-design and integration to achieve exascale computing



DOE HPC Roadmap to Exascale Systems



Brief Intro to ECP Software Technology (ST) Focus Area

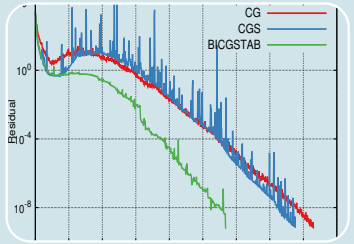
ECP **Software Technology (ST)**

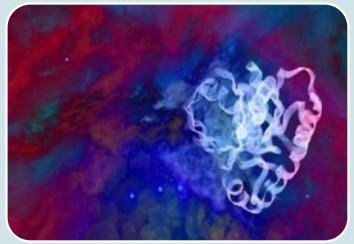
Goal

Build a comprehensive, coherent software stack that enables application developers to productively develop highly parallel applications that effectively target diverse exascale architectures

- Prepare SW stack for scalability with massive on-node parallelism
- Extend existing capabilities when possible, develop new when not
- Guide, and complement, and integrate with vendor efforts
- Develop and deliver high-quality and robust software products

ECP ST has six technical areas





National Nuclear Security Administration

E4S

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

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

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

We work on products applications need now and into 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)

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

Progress toward Exascale readiness

SLATE port to AMD and Intel platforms

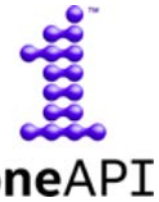
ECP WBS	2.3.3.13 CLOVER (SLATE)
PI	Jack Dongarra, UTK
Members	UTK

Scope and objectives

- SLATE is a distributed, GPU-accelerated, dense linear algebra library, intended to replace ScaLAPACK
- SLATE covers parallel BLAS, linear system solvers, least squares, eigensolvers, and the SVD

Port to AMD and Intel

- SLATE and BLAS++ now support all three major GPU platforms



Impact

- Initially supported NVIDIA's cuBLAS for use on current machines like Summit
- Can now use AMD's rocBLAS in preparation for Frontier, and Intel's oneMKL in preparation for Aurora
- Other projects can also leverage BLAS++ for portability

Accomplishment

- Refactored SLATE to use BLAS++ as portability layer
- Ported BLAS++ to AMD rocBLAS and Intel oneMKL

Deliverables Report: https://www.icl.utk.edu/publications/swan-016
Code in git repos: bitbucket.org/icl/slate/ and bitbucket.org/icl/blaspp/

Kokkos: Support and AMD Functionality.

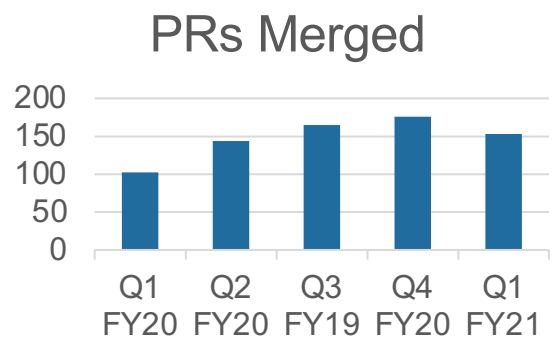
ECP WBS	2.3.6.03 – SNL ATDM ST
PI	Christian Trott, SNL
Members	SNL

Scope and objectives

- Kokkos provides the C++ based based Programming Model for Performance Portability for Sandia and many applications at partner institutions
- The goal is to enable single source applications and libraries to simply recompile for new architectures including Exascale Platforms.

Support and Development

- Slack continues to be primary support vehicle
- Regular meetings with NNSA customers for progress updates held
- Continue >50 PRs merged per month



AMD Support

- Support for everything KokkosKernels and Trilinos need
- Provided changes for Trilinos to enable Krylov solver
  - Tpetra, Belos fully compile
  - Tpetra >95% of tests pass
- Still running into known AMD bug, reported mid 2020

Kokkos Update and Maintenance

- Release 3.3 rolled
  - Near Feature Complete Support for HIP
- Added support for Intel OneAPI compiler
- Added support for Fujitsu ARM A64FX and Fujitsu compiler
- Improved Spack support

Deliverables Kokkos: https://github.com/kokkos/kokkos
Slack: https://kokkosteam.slack.com

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525.

FY21: FFT: Application-specific FFT optimizations and integration within Copa and ExaAM

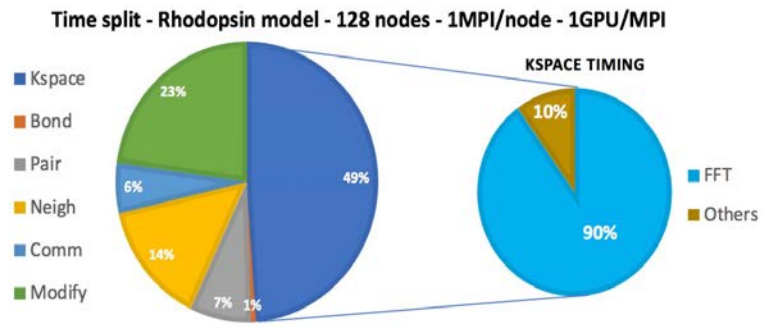
ECP WBS	2.3.3.13 FFT-ECP
PI	Jack Dongarra, (UTK – ICL)
Members	Stan Tomov (UTK – ICL), Alan Ayala (UTK – ICL), Miroslav Stoyanov (ORNL)

Scope and objectives

- Design and implement a sustainable FFT library for Exascale platforms
- Define consistent FFT-ECP APIs for FFTs on Exascale systems to help key ECP applications that need FFT functionalities to run at exascale
- FY21 plan: develop application-specific FFTs, optimizations, and integration in ECP applications; add HIP and DPC++ backends to support AMD and Intel GPUs;
- Milestone driver: Implement multidimensional FFTs and optimizations in heFFTe for applications where the input data is purely real.

heFFTe 2.1 Release

- Profile on running LAMMPS Rhodopsin benchmark with FFTMPI using 128 Summit nodes on a 1024^3 FFT;
- heFFTe with cuFFT backend accelerates FFT 2X compared to FFTMPI and 25% the entire application



Impact

- Developed application-specific FFT optimizations and integration within Copa and ExaAM applications;
- Provide ECP applications acceleration for their FFT computations on various GPU-accelerated heterogeneous architectures with GPUs from Nvidia, AMD, and Intel;
- FFT-ECP stakeholders are application developers, e.g., LAMMPS and HACC, Copa and ExaAM, as well as ECP vendors where heFFTe enables FFT applications to run more efficiently on current and upcoming platforms.

Project accomplishment

- Implemented application-specific FFT optimizations and tuning for systems with Nvidia and AMD GPUs;
- Added Intel GPU support;
- Released heFFTe 2.1 featuring with new application-specific optimizations, tuning, and added Intel GPU support;
- heFFTe Integration and acceleration within CoPA projects and ExaAM/Meumapps.

Deliverables heFFTe 2.1 http://icl.utk.edu/fft/ multidimensional FFTs and application-specific optimizations with added Intel GPUs support. Relation projects: ECP LAMMPS, HACC, CoPA, Cabana, Alpine, FFTX, SLATE, xSDK, and MAGMA (http://icl.cs.utk.edu/magma/) ExaWind, EMPIRE/PIC/PICSAR, WarpX, ExaSky, LaticeQCD/MILC, EXAALT, ExaAM, QMCPACK, NWChemEx

Document on Performance Evaluation of Solvers in *hypre* 2.20.0

ECP WBS	WBS 2.3.3.12
PI	Carol Woodward, LLNL
Members	LLNL
Milestone Lead	Ulrike Meier Yang, LLNL

Scope and objectives

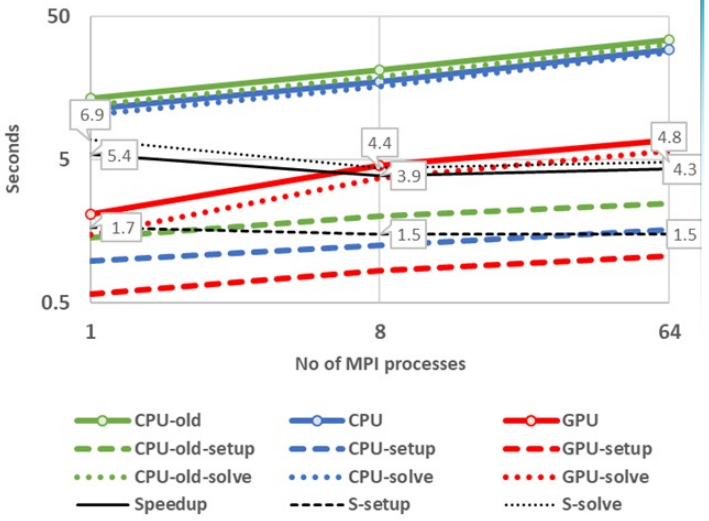
- This project focuses on enhancements for *hypre* and SUNDIALS in preparation for exascale systems
- Goals for *hypre* include increasing GPU-enabled portions as well as portability.
- This milestone evaluates and analyzes the GPU and CPU performance of structured and unstructured solvers in hypre for various problems on Lassen and Summit.

Impact

- Linear systems are an important part of many application codes, and often make up a large portion of their execution times.
- Efficient linear solvers are crucial for ECP applications, and any improvements in performance and memory usage positively impact the applications.

Comparing hypre's GPU and CPU performance

Weak scaling study of AMG-PCG applied to an unstructured problem on a crooked pipe on Lassen using 1, 2, or 16 nodes. 'CPU' uses same parameters as 'GPU' including a newly designed interpolation. 'CPU-old' uses the old Interpolation. Presented are setup, solve and total times, including Speedups (CPU/GPU)

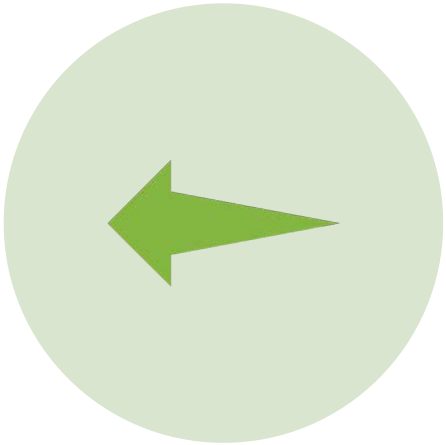


Project accomplishment

- Measured GPU and CPU performance of hypre's structured and unstructured solvers on a variety of problems on Lassen and Summit.
- Analyzed and summarized the results in a document that is available on confluence.

Deliverables The document is available at https://confluence.exascaleproject.org/display/STLM12/Software+Documents in file 'Performance Evaluation of hypre Solvers.pdf'

Key Product Development Takeaways



ECP ST teams are creating new algorithms that effectively expose and exploit massive on-node parallelism, in addition to MPI

ECP ST products are expanding support to all GPU architectures: Nvidia, AMD, Intel

Application teams are increasingly relying on ECP ST products to get performance & portability

Getting portable performance via E4S products

Challenge: How can I port my code effectively and efficiently to diverse and emerging architectures?

- Goals:
  - Get performance
    - Get all or most of the potential performance on a platform (varies with specific situation)
    - Get on the commodity performance curve: Porting to next similar system, say 2X faster, your code is about 2X faster
  - Get portability
    - Minimize how much special code needs to be written for each target platform
    - Can be done by using portability layers, language features, libraries that provide functionality across many systems
- Examples:
  - Use Kokkos to write your parallel loops:
    - Enables performance across multiple platforms by compiling with a backend that transforms your loops for the target
    - Targets can be Intel CPU, Nvidia GPU, AMD GPU, Intel GPU, Arm SVE, future parallel devices
  - Use PETSc to solve large sparse linear systems:
    - PETSc runs well on CPUs and GPUs, adapting algorithms and implementations behind the scenes
    - Note: Assembling the sparse linear system for GPU systems needs to be done on the GPU, using, e.g., Kokkos

Writing *your* code for portable performance

OpenMP

- An open standard
- Target offload supports GPUs
- Commonly used by Fortran codes, uncommon for C++

Cuda/HIP/SYCL

- Vendor specific, esp CUDA
- HIP portable in principle, but really driven by AMD
- SYCL portable in principle, but really driven by Intel

Kokkos/RAJA

- Kokkos uses C++ template meta-programming, widely used, lots of training and documentation
- RAJA more modular design (e.g., loop vs memory management), fundamental to LLNL ecosystem

Using libraries for portable performance

Dense Lin Alg

- Vendors typically provide, e.g., MKL
- ECP efforts provide alternative for reference and design ideas

FFTs

- Vendors provide building blocks, e.g., 1D
- Many apps have their own 3D framework
- heFFTe provides new portable 3D library emphasizing internode scalability

Sparse Lin Alg

- Strong tradition for DOE
- Sparse direct: SuperLU/STRUMPACK
- Sparse iterative: PETSc, Trilinos/KokkosKernels
- Apps will need to move problem construction to GPU

Addressing IO Bottlenecks

HDF5

- Continued evolution for modern platforms

ADIOS

- Alternative, customizable library
- Also becoming available via HDF5 API

Data compression

- VeloC/SZ
- ZFP
- Libraries that support *in situ* compression

The Extreme-Scale Scientific Software Stack (E4S) and Software Development Kits (SDKs)

Core questions E4S is addressing

How can new ECP software capabilities be effectively and efficiently integrated and sustained?

- ECP success requires development, delivery and use of new GPU capabilities in 70 products
- Requires coordination of versioning, integration, testing, debugging, interaction with vendors and facilities
- Requires access to new documentation
- Requires focus on high quality

How can E4S build upon, leverage and extend existing capabilities and activities?

- Using Spack for product installation, leveraging growing Spack capabilities
- Making E4S available via containers, cloud platforms
- Providing integration pathways to multiple destinations: from-source, LLVM, vendor stacks, facilities, etc

How can E4S become a sustainable, open, collaborative software ecosystem for HPC?

- Hierarchical, open architecture to accept and manage community contributions
- Defined processes for community engagement within DOE, with other US agencies, industry, international partners
- Delivering the value proposition of the ecosystem vs each app managing its dependencies

ECP applications rely on ST products across all technical areas

24 ECP 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

Consider ECP software technologies needed by 5 ECP applications:

ExaWind: Turbine Wind Plant Efficiency

Harden wind plant design and layout against energy loss susceptibility; higher penetration of wind energy

Lead: NREL
DOE EERE

Subsurface: Carbon Capture, Fossil Fuel Extraction, Waste Disposal

Reliably guide safe long-term consequential decisions about storage, sequestration, and exploration

Lead: LBNL
DOE BES, EERE, FE, NE

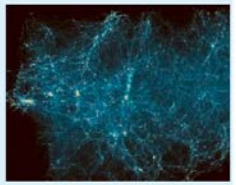
WDMApp: High-Fidelity Whole Device Modeling of Magnetically Confined Fusion Plasmas

Prepare for ITER experiments and increase ROI of validation data and understanding; prepare for beyond-ITER devices

Lead: PPPL
DOE FES

ExaSky: Cosmological Probe of the Standard Model of Particle Physics

Unravel key unknowns in the dynamics of the Universe: dark energy, dark matter, and inflation

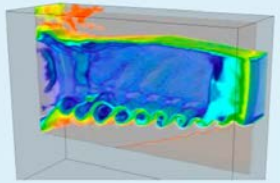


Lead: ANL
DOE HEP

The MARBL Multi-physics Code

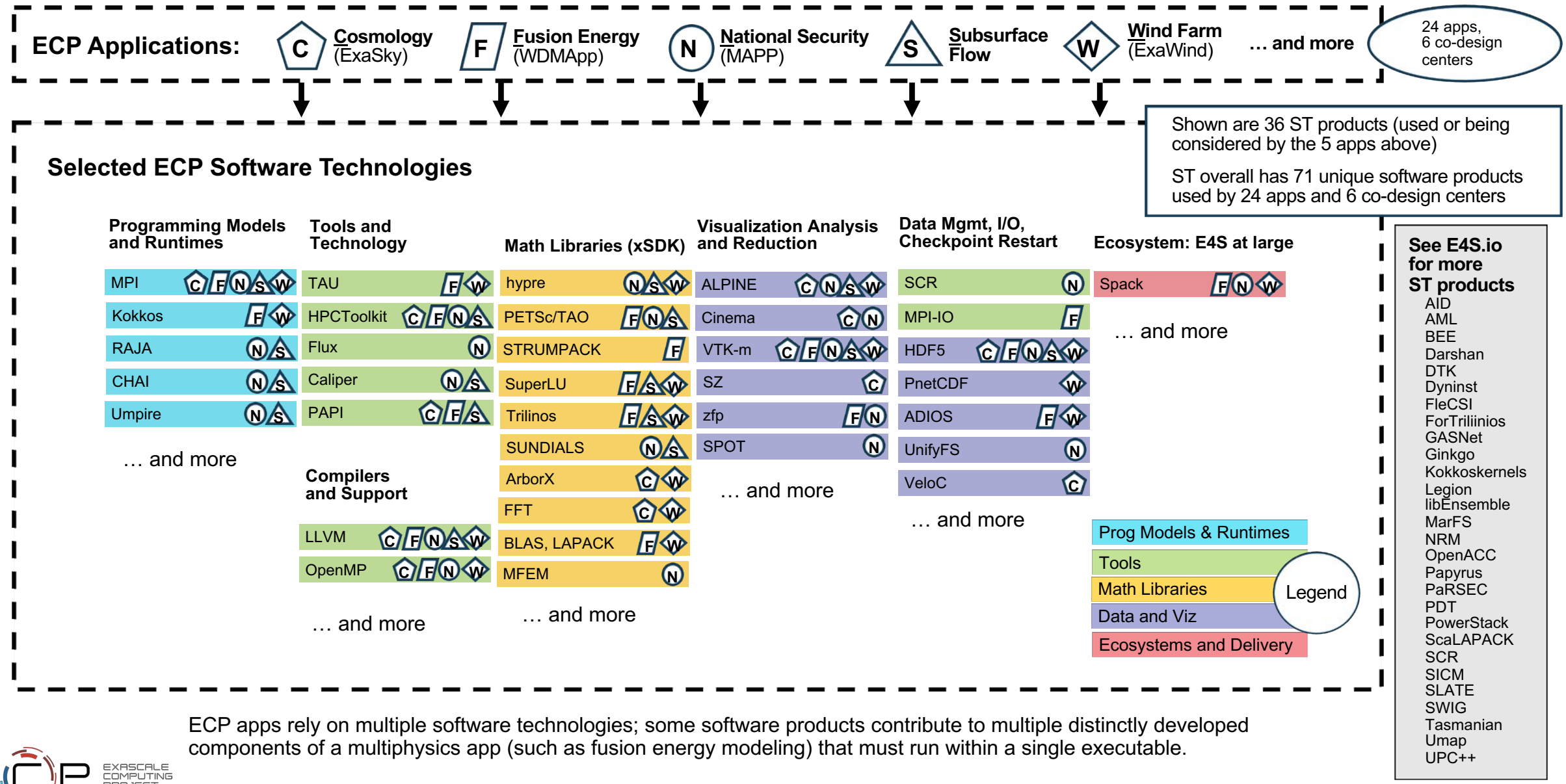
Multi-physics simulations of high energy-density physics and focused experiments driven by high-explosive, magnetic or laser based energy sources

- Magneto-radiation-hydrodynamics at the exascale
- Next-generation pulsed power / ICF modeling
- High-order numerical methods



Lead: LLNL

ECP applications require consistency across the software stack



ECP apps rely on multiple software technologies; some software products contribute to multiple distinctly developed components of a multiphysics app (such as fusion energy modeling) that must run within a single executable.

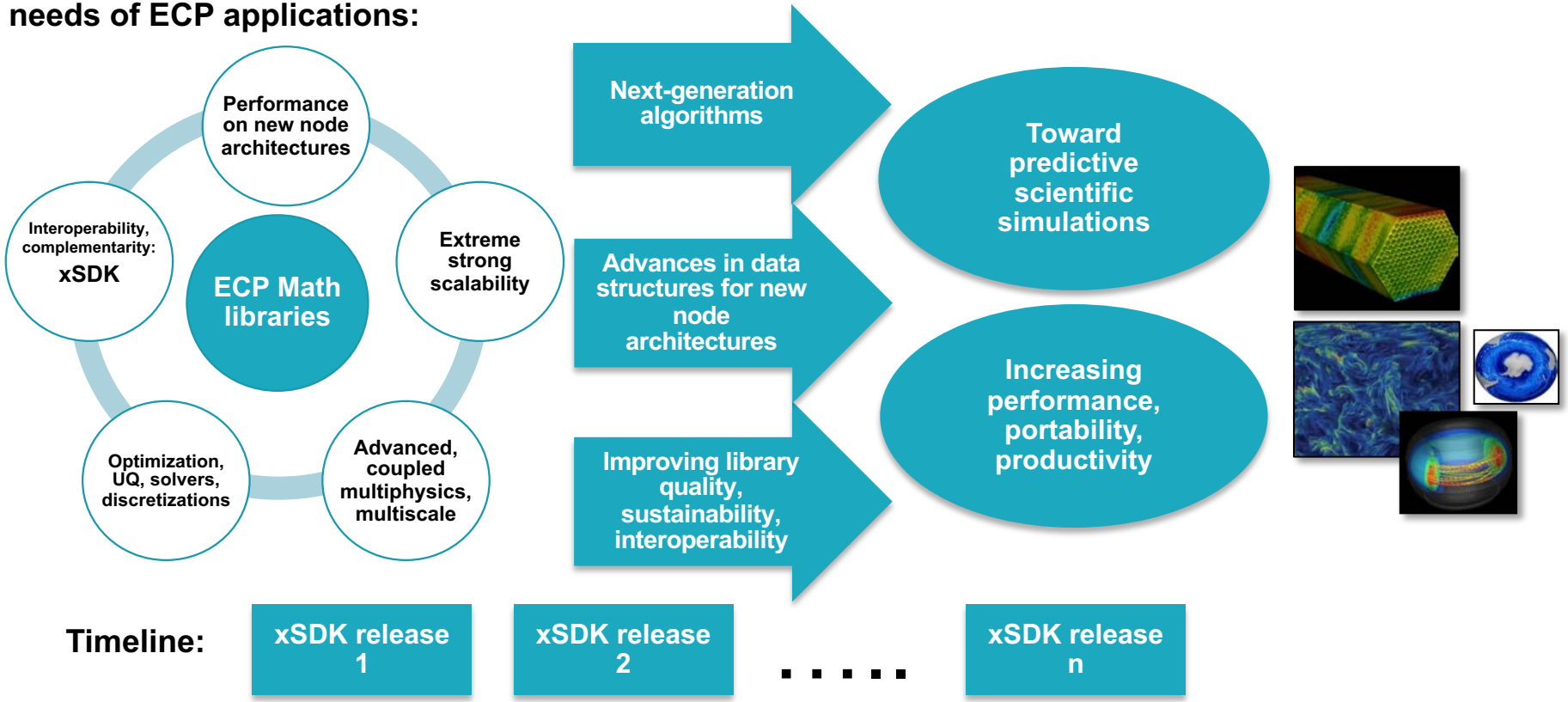
xSDK: Primary delivery mechanism for ECP math libraries' continual advancements toward predictive science

xSDK release 0.6.0 (Nov 2020)

- hypre
- PETSc/TAO
- SuperLU
- Trilinos
- AMReX
- ButterflyPACK
- DTK
- Ginkgo
- heFFTe
- libEnsemble
- MAGMA
- MFEM
- Omega_h
- PLASMA
- PUMI
- SLATE
- Tasmanian
- SUNDIALS
- Strumpack
- Alquimia
- PFLOTRAN
- deal.II
- preCICE
- PHIST
- SLEPc

} from the broader community

As motivated and validated by the needs of ECP applications:



Timeline: xSDK release 1 → xSDK release 2 → . . . . . → xSDK release n

Ref: xSDK: Building an Ecosystem of Highly Efficient Math Libraries for Exascale, **SIAM News**, Jan 2021

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
- 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
- Feb 2020: E4S 1.1 - 61 full release products
- Nov 2020: E4S 1.2 (aka, 2020.10) - 67 full release products
- Feb 2021: E4S 21.02 - 67 full release, 4 partial release
- May 2021: E4S 21.05 – 87 full release products

https://e4s.io

Lead: Sameer Shende
(U Oregon)

Also include other products .e.g.,
AI: PyTorch, TensorFlow, Horovod
Co-Design: AMReX, Cabana

Delivering an open, hierarchical software ecosystem

More than a collection of individual products

Levels of Integration	Product	Source and Delivery

ECP ST Open Product Integration Architecture

- Build all SDKs
- Build complete stack
- Assure core policies
- Build, integrate, test

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

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

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

- Standard workflow
- Existed before ECP

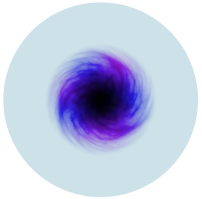
Source: ECP L4 teams; Non-ECP Developers; Standards Groups
Delivery: Apps directly; spack; vendor stack; facility stack

ECP ST Individual Products

E4S: Better quality, documentation, testing, integration, delivery, building & use

Delivering HPC software to facilities, vendors, agencies, industry, international partners in a brand-new way

Community Policies
Commitment to software quality



DocPortal
Single portal to all E4S product info

Portfolio testing
Especially leadership platforms

Curated collection
The end of dependency hell

Quarterly releases
Release 1.2 – November

Build caches
10X build time improvement

Turnkey stack
A new user experience

https://e4s.io

E4S Strategy Group
US agencies, industry, international

E4S Community Policies

E4S Community Policies V1.0 Released

What is E4S?

The Extreme-scale Scientific Software Stack (E4S) is a community effort to provide open source software packages for developing, deploying and running scientific applications on high-performance computing (HPC) platforms. E4S provides from-source builds and containers of a broad collection of HPC software packages.

Purpose

E4S exists to accelerate the development, deployment and use of HPC software, lowering the barriers for HPC users. E4S provides containers and turn-key, from-source builds of more than 80 popular HPC products in programming models, such as MPI; development tools such as HPCToolkit, TAU and PAPI; math libraries such as PETSc and Trilinos; and Data and Viz tools such as HDF5 and Paraview.

Approach

By using Spack as the meta-build tool and providing containers of pre-built binaries for Docker, Singularity, Shifter and CharlieCloud, E4S enables the flexible use and testing of a large collection of reusable HPC software packages.

E4S Community Policies Version 1

A Commitment to Quality Improvement

- Will serve as membership criteria for E4S
  - Membership is not required for *inclusion* in E4S
  - Also includes forward-looking draft policies
- Purpose: enhance sustainability and interoperability
- Topics cover building, testing, documentation, accessibility, error handling and more
- Multi-year effort led by SDK team
  - Included representation from across ST
  - Multiple rounds of feedback incorporated from ST leadership and membership
- Modeled after xSDK Community Policies
- https://e4s-project.github.io/policies.html

P1 *Spack-based Build and Installation* Each E4S member package supports a scriptable *Spack* build and production-quality installation in a way that is compatible with other E4S member packages in the same environment. When E4S build, test, or installation issues arise, there is an expectation that teams will collaboratively resolve those issues.

P2 *Minimal Validation Testing* Each E4S member package has at least one test that is executable through the E4S validation test suite (*https://github.com/E4S-Project/testsuite*). This will be a post-installation test that validates the usability of the package. The E4S validation test suite provides basic confidence that a user can compile, install and run every E4S member package. The E4S team can actively participate in the addition of new packages to the suite upon request.

P3 *Sustainability* All E4S compatibility changes will be sustainable in that the changes go into the regular development and release versions of the package and should not be in a private release/branch that is provided only for E4S releases.

P4 *Documentation* Each E4S member package should have sufficient documentation to support installation and use.

P5 *Product Metadata* Each E4S member package team will provide key product information via metadata that is organized in the *E4S DocPortal* format. Depending on the filenames where the metadata is located, this may require *minimal setup*.

P6 *Public Repository* Each E4S member package will have a public repository, for example at GitHub or Bitbucket, where the development version of the package is available and pull requests can be submitted.

P7 *Imported Software* If an E4S member package imports software that is externally developed and maintained, then it must allow installing, building, and linking against a functionally equivalent outside copy of that software. Acceptable ways to accomplish this include (1) forsaking the internal copied version and using an externally-provided implementation or (2) changing the file names and namespaces of all global symbols to allow the internal copy and the external copy to coexist in the same downstream libraries and programs. This pertains primarily to third party support libraries and does not apply to key components of the package that may be independent packages but are also integral components to the package itself.

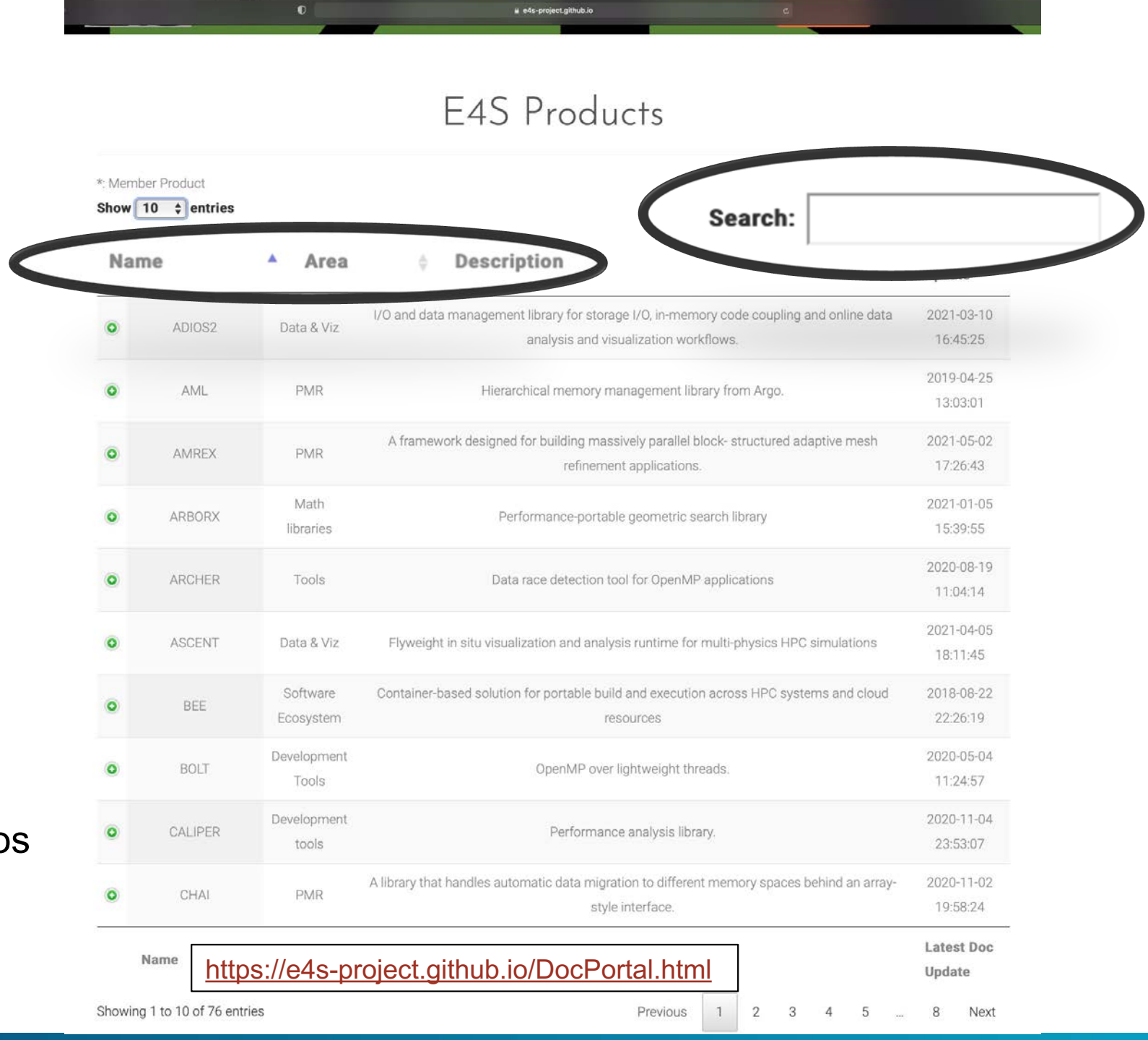
P8 *Error Handling* Each E4S member package will adopt and document a consistent system for signifying error conditions as appropriate for the language and application. For e.g., returning an error condition or throwing an exception. In the case of a command line tool, it should return a sensible exit status on success/failure, so the package can be safely run from within a script.

P9 *Test Suite* Each E4S member package will provide a test suite that does not require special system privileges or the purchase of commercial software. This test suite should grow in its comprehensiveness over time. That is, new and modified features should be included in the suite.

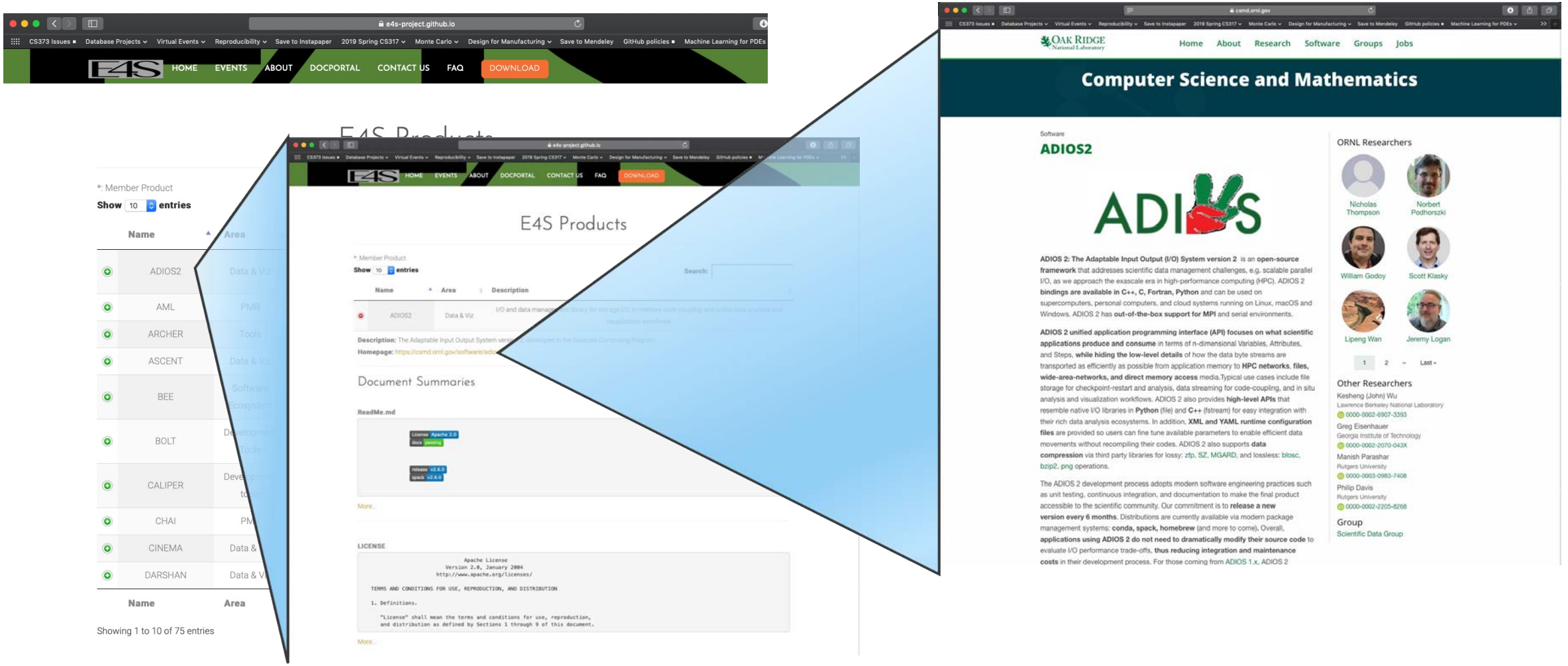
E4S DocPortal

E4S DocPortal

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



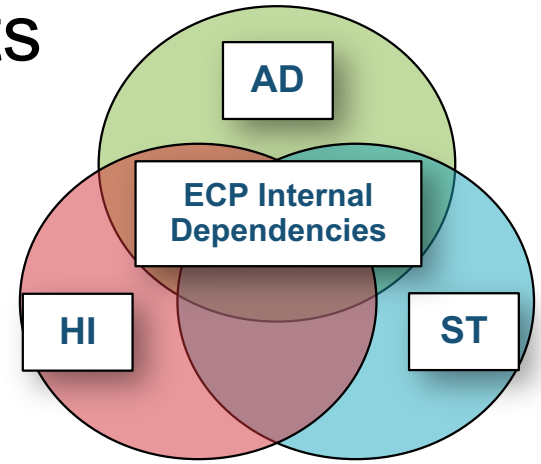
Goal: All E4S product documentation accessible from single portal on E4S.io (working mock webpage below)



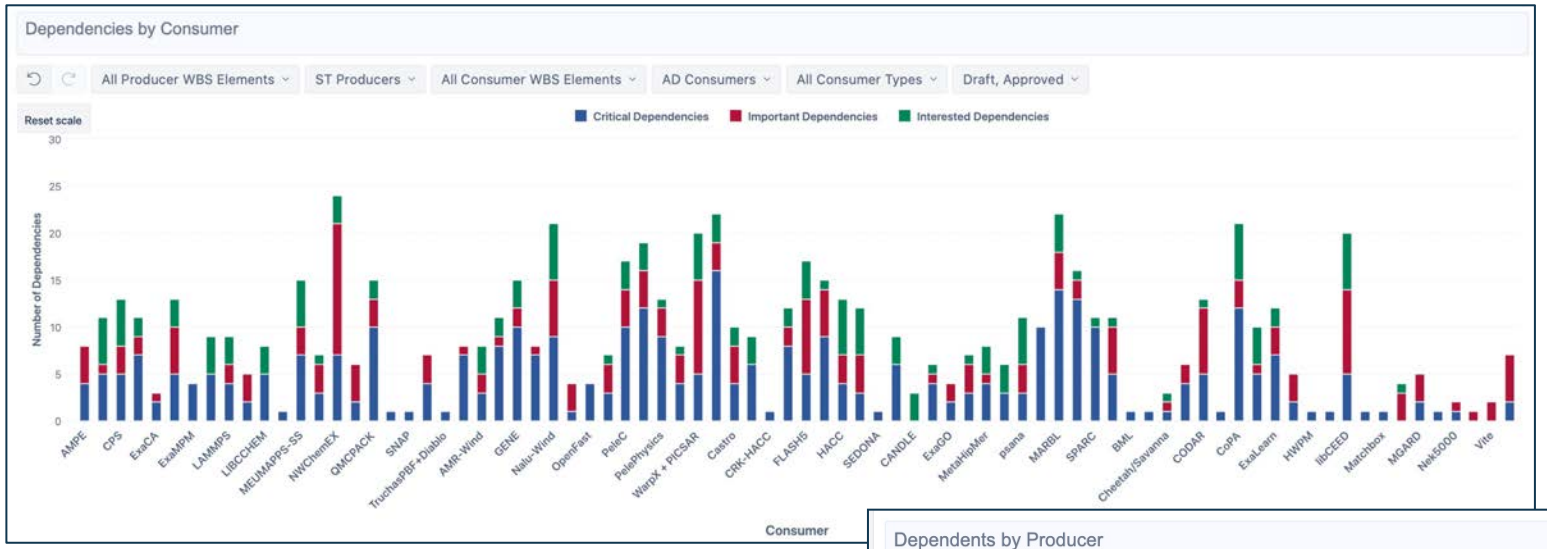
https://e4s-project.github.io/DocPortal.html

Using E4S

ECP apps (AD) are primary consumers of ST products

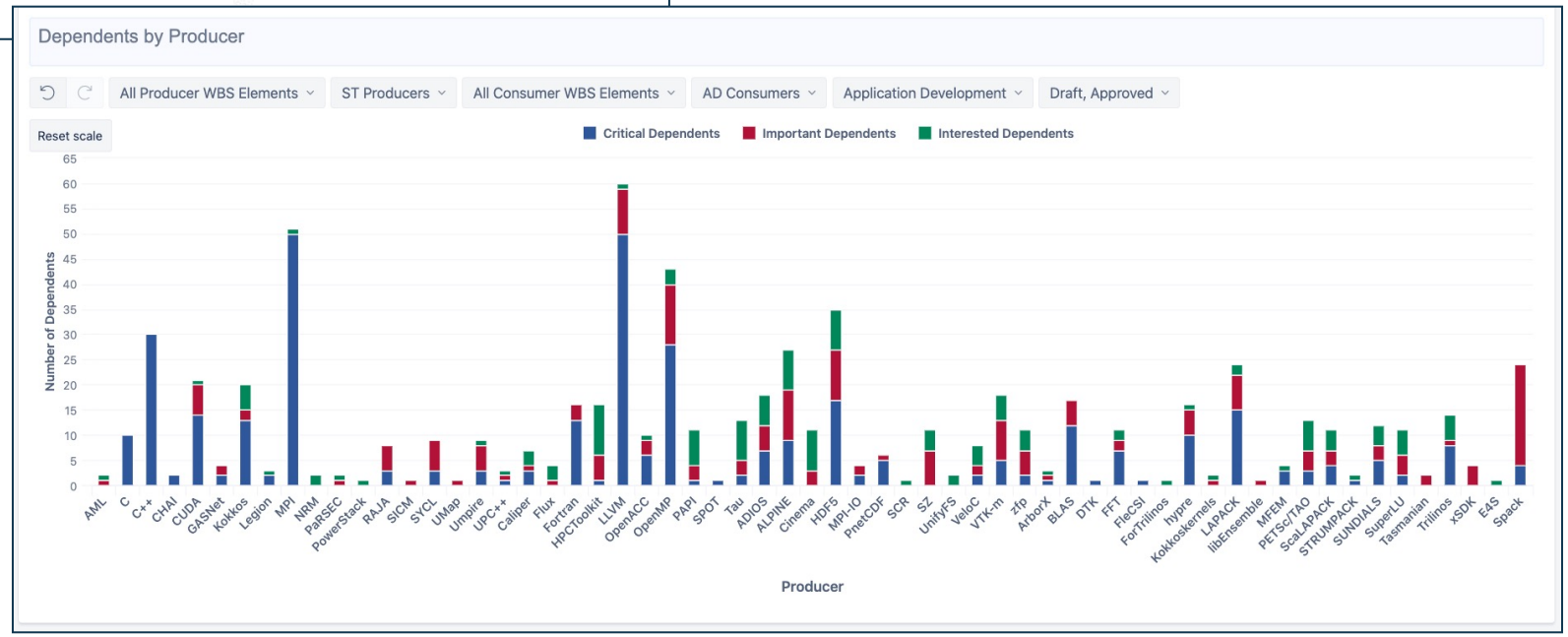


Dependency Database

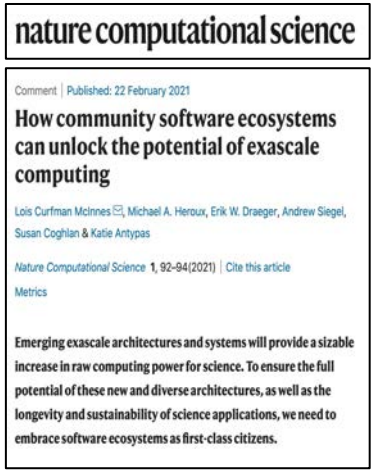


View by AD consumers

View by ST producers



https://dx.doi.org/10.1038/s43588-021-00033-y



nature computational science

Comment | Published: 22 February 2021

How community software ecosystems can unlock the potential of exascale computing

Lois Curfman McInnes ✉, Michael A. Heroux, Erik W. Draeger, Andrew Siegel, Susan Coghlan & Katie Antypas

Nature Computational Science 1, 92–94(2021) | Cite this article

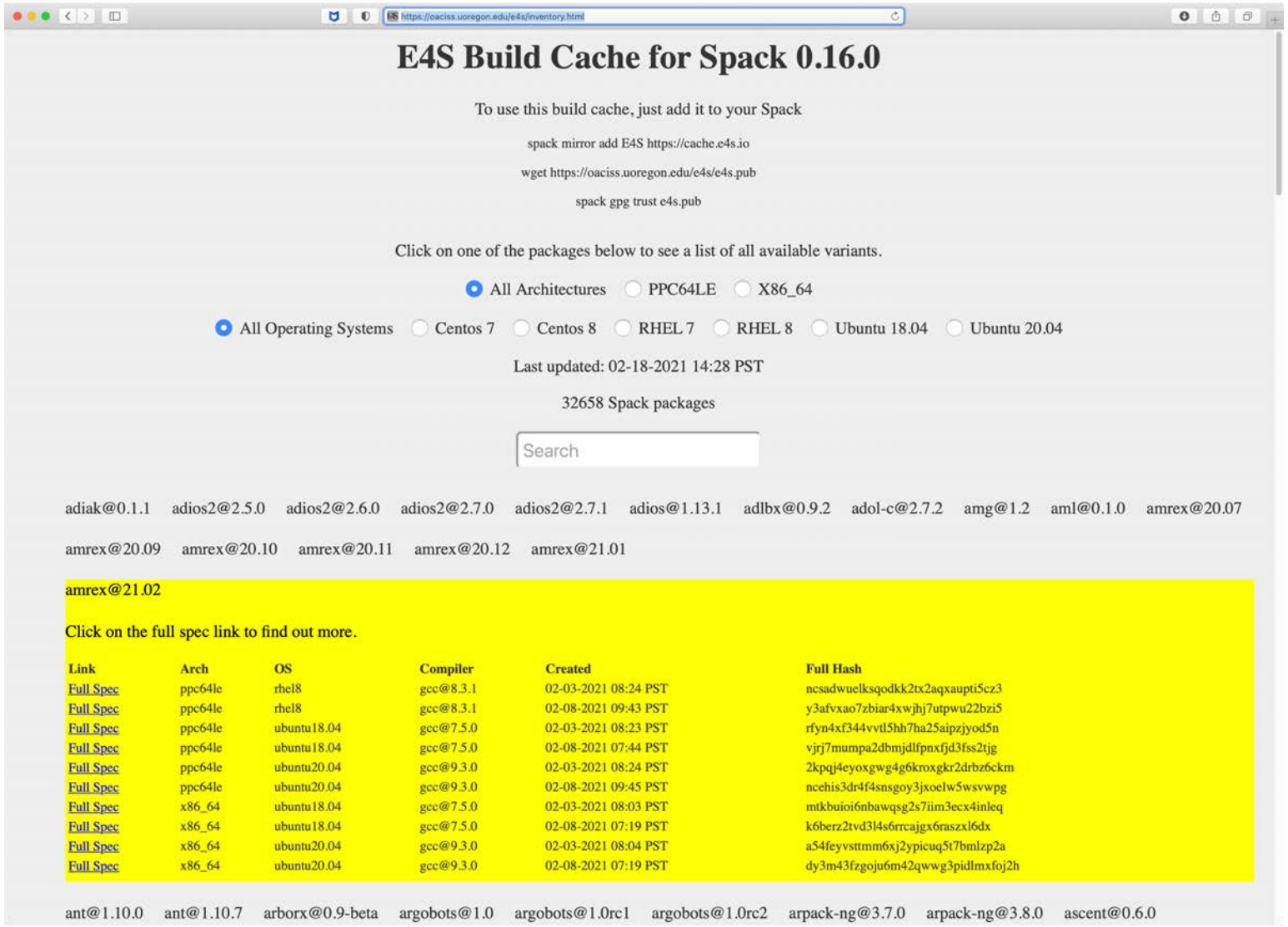
Metrics

Emerging exascale architectures and systems will provide a sizable increase in raw computing power for science. To ensure the full potential of these new and diverse architectures, as well as the longevity and sustainability of science applications, we need to embrace software ecosystems as first-class citizens.

Spack

- E4S uses the Spack package manager for software delivery
- Spack provides the ability to specify versions of software packages that are and are not interoperable.
- Spack is a build layer for not only E4S software, but also a large collection of software tools and libraries outside of ECP ST.
- Spack supports achieving and maintaining interoperability between ST software packages.

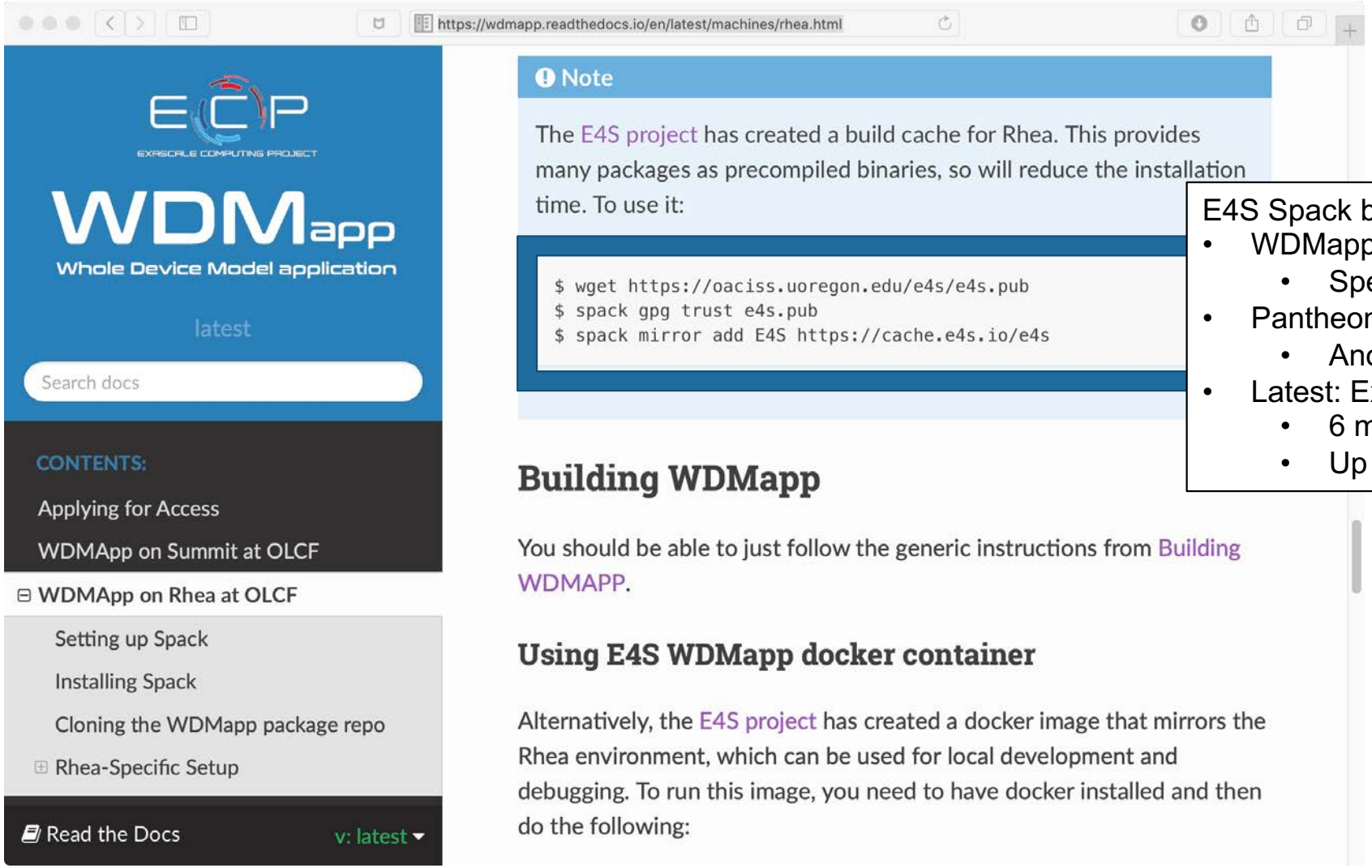
E4S: Spack Build Cache



- 32,000+ binaries
- S3 mirror
- No need to build from source code!
- Speeds up installations 10x

- https://oaciss.uoregon.edu/e4s/inventory.html

WDMApp: Speeding up bare-metal installs using E4S build cache



E4S Spack build cache:

- WDMapp added E4S mirror
  - Speedup: 10X
- Pantheon: 10X
  - Another 10X via “smoother” installs
- Latest: ExaWind (Nalu-Wind)
  - 6 minutes with build cache
  - Up to 4 hours without

- https://wdmapp.readthedocs.io/en/latest/machines/rhea.html

E4S Community Engagement

Opportunities via E4S

- E4S enables portfolio strategy for ASCR R&D software delivery:
  - Facilities: Robust planning, delivery, integration and testing at Facilities
  - Community: MPI Forum, C++, OpenMP, LLVM
  - Vendor: Coordinated integration into vendor software stacks
  - Users: Turnkey delivery of capabilities to DOE program offices, US agencies, industry, international partners
- E4S provides incentives and support for high-quality research software products
  - Community policies: Drives quality by explicit expectations and clear view of gaps
  - SDKs for community interaction: Build awareness and collaboration across independent teams
  - Transparency: E4S DocPortal, build, test, integration shows quality (good or poor) of a product
- E4S provides direct path for software teams to reach users and other stakeholders
  - Example: ArborX is brand new geometric search library
    - Part of E4S, available at DocPortal, tested regularly on many platforms
    - Installed anywhere E4S is installed, users can count on it being there
    - Without E4S: ArborX would take years to become visible and available
  - Availability and adoption timeline reduced from years (or never) to months

Joining E4S

- Process:
- Pre-req: Must make sense
- L0: E4S Spackified
- L1: Listed in DocPortal
- L2: Satisfy policies

E4S represents a growing community of HPC software products. Please contact us if you would like your product to be a part of E4S.

Pre-requisite: Justification for being included. The product must have some value or strong potential value to the HPC community. In other words, it must make sense for the product to belong to E4S.

Level 0: Be listed in the E4S Spack installation script and be buildable in all E4S target environments. The list of Spack recipes for E4S is here.

Level 1: Be present in the DocPortal. To achieve this, we need a URL to your main repo to add to this list.

Level 2: Satisfy all E4S community policies. Full satisfaction is not required at this time. We are still establishing a process. Note that derived requirements from E4S member packages are not required to satisfy community policies as long as they do destabilize E4S builds or portability.

Created for The Extreme-scale Scientific Software Stack (E4S) Project by Michael A. Heroux

Attribution - Derived from a design by Quentin Petit

Broader Community Engagement

The Second Extreme-scale Scientific Software Stack Forum (E4S Forum)
September 24th, 2020, Workshop at EuroMPI/USA'20

- Presenters from 11 institutions, 6 non-DOE
- 70 participants
  - DOE Labs, NASA
  - AMD
  - HLRS, CSCS

- E4S: The Extreme-scale Scientific Software Stack for Collaborative Open Source Software, Michael Heroux, Sandia National Laboratories
- Title: Practical Performance Portability at CSCS, **Ben Cumming, CSCS**
- Title: An Overview of High Performance Computing and Computational Fluid Dynamics at NASA, **Eric Nielsen, NASA Langley**
- Towards An Integrated and Resource-Aware Software Stack for the EU Exascale Systems, **Martin Schulz, Technische Universität München**
- Spack and E4S, Todd Gamblin, LLNL
- Rocks and Hard Places – Deploying E4S at Supercomputing Facilities, Ryan Adamson, Oak Ridge Leadership Computing Facility
- Advances in, and Opportunities for, LLVM for Exascale, Hal Finkel, Argonne National Laboratory
- Kokkos: Building an Open Source Community, Christian Trott, SNL
- Experiences in Designing, Developing, Packaging, and Deploying the MVAPICH2 Libraries in Spack, **Hari Subramoni, Ohio State University**
- Software Needs for Frontera and the NSF Leadership Class Computing Facility – the Extreme Software Stack at the Texas Advanced Computing Center, **Dan Stanzione, TACC**
- Building an effective ecosystem of math libraries for exascale, Ulrike Yang
- Towards Containerized HPC Applications at Exascale, Andrew Younge, Sandia
- E4S Overview and Demo, Sameer Shende, University of Oregon
- The Supercomputer “Fugaku” and Software, programming models and tools, **Mitsuhisa Sato, RIKEN Center for Computational Science (R-CCS), Japan**

E4S provides a natural collaboration vehicle for interacting within DOE, with other US agencies, industry and international partners

E4S summary

What E4S is not	What E4S is
• A closed system taking contributions only from DOE software development teams.	• Extensible, open architecture software ecosystem accepting contributions from US and international teams. • Framework for collaborative open-source product integration for ECP & beyond, including AI and Quantum.
• A monolithic, take-it-or-leave-it software behemoth.	• Full collection if compatible software capabilities **and** • Manifest of a la carte selectable software capabilities.
• A commercial product.	• Vehicle for delivering high-quality reusable software products in collaboration with others. • New entity in the HPC ecosystem enabling first-of-a-kind relationships with Facilities, vendors, other DOE program offices, other agencies, industry & international partners.
• A simple packaging of existing software.	• Hierarchical software framework to enhance (via SDKs) software interoperability and quality expectations. • Conduit for future leading edge HPC software targeting scalable computing platforms.

Looking Forward

Lessons learned from E4S/ECP ST to carry forward

- Deliver DOE reusable software as a portfolio
  - E4S value is already more than the sum of its parts
  - Community policies drive quality, membership
  - DocPortal, testing, containerization, cloud, build caches, modules, etc., greatly improve access & usability
  - Poor performing products are ID'ed, then improved or removed
- E4S is ready to extend to next-generation software and hardware needs
  - AI/ML products already in portfolio, ready for any new products
  - Quantum, FPGA, neuromorphic devices likely to be accelerators
    - From a macro software architecture, similar to GPUs
    - Software for these devices can and should be part of the same stack for holistic HPC environment
- DOE software as a portfolio is a first-class entity in the ecosystem
  - E4S planning, executing, tracking, assessing is peer collaboration with Facilities, program offices, vendors, etc
  - E4S can become a perennial asset for DOE/ASCR as part of its mission impact within and beyond DOE

E4S sustainability

Challenges

- ECP has a robust tailored 413.3b project management infrastructure
- Transitioning & adapting this infrastructure is essential for post-ECP success
- Funding models, portfolio management, org structure are particularly critical

Opportunities

- A sustainable software ecosystem for HPC software from DOE & broader community
- Payoff if done right: better, faster and cheaper – get all three

E4S Expansion – Base Scope & Gaps

Within base scope

Making a high-quality HPC product portfolio through tools, processes, and transparency

Community policies: Improve product quality upstream, shepherd membership growth

DocPortal: Provide easy access to product documentation

Portfolio testing: Protecte against regressions, prepare for new platforms

Curated collection: Maintain version compatibility across products

Turnkey stack via quarterly releases: Provide functionality via Spack, containers, clouds

Gaps not in base

Features that are a significant departure from core mission needs

Sustained support of new customers (without specific collaborative funding)

Activities related to commercial software enterprise

Ongoing support of a maintenance-only product (no longer funded for R&D)

Need: Business models for the gaps

Final points

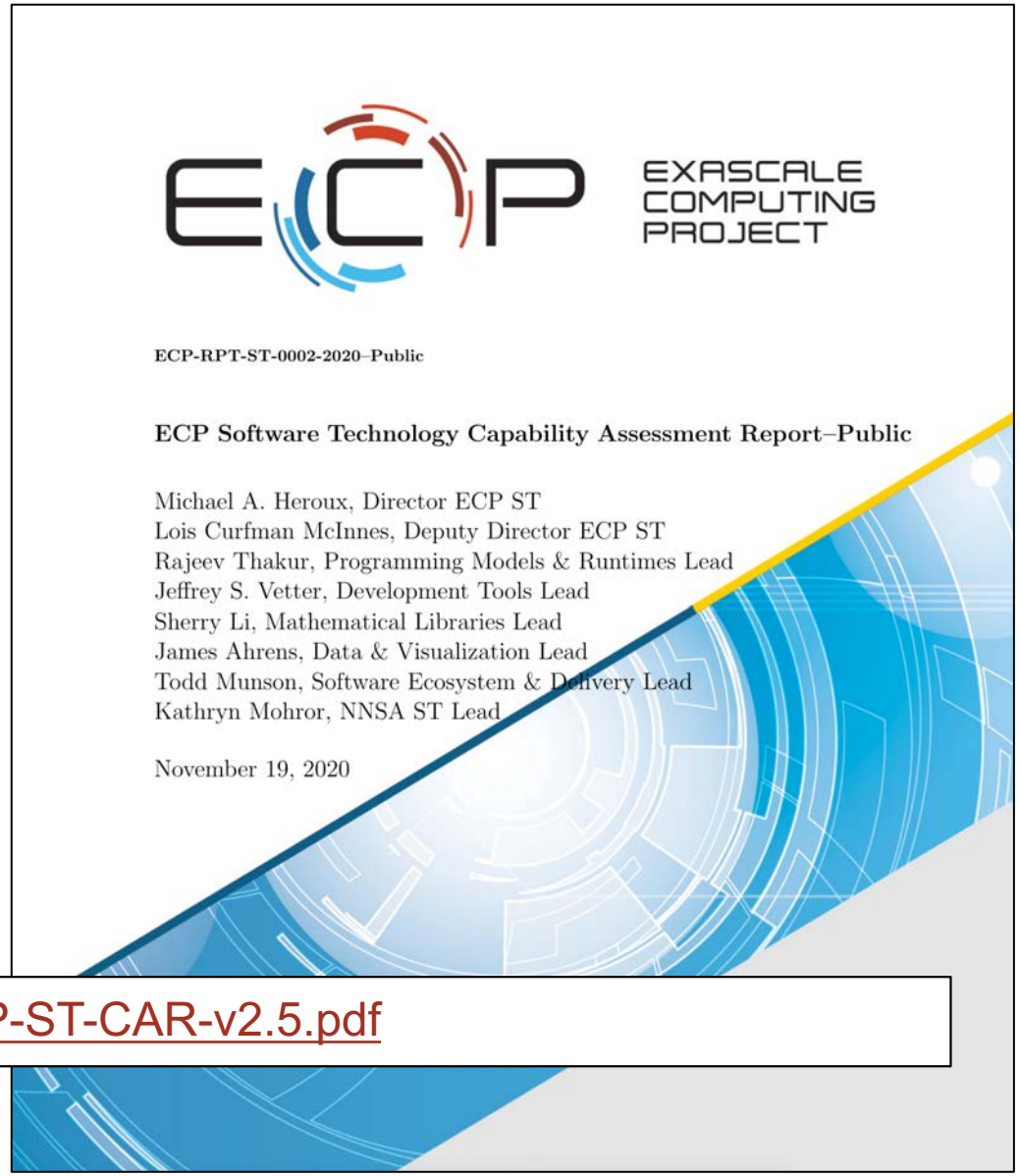
- E4S is a curated software stack with quality improvement incentives, moving toward turnkey use
- With DOE program managers ECP is starting
  - Software ecosystem sustainability planning
  - E4S strategic plan (will include monthly townhalls)
- We believe
  - E4S has reduced important gaps that limit usefulness of DOE software for industry
  - But some gaps remain
- Next steps:
  - Better characterize these gaps
  - Explore models to further reduce and close gaps
  - Plan and execute toward sustainability

Some opportunities for interactions

- E4S is ready for app teams to use now
  - Curated, version-managed collection of many libraries & tools app teams use
  - Turnkey builds, containers & cloud builds, Spack build cache: Can dramatically improve productivity
  - Full E4S suite available for non-GPU platforms (CPU-based clusters)
  - Many E4S product work on Nvidia GPUs, growing set of capabilities for Intel, AMD GPUs, some Arm/SVE
- Would love to engage new software teams
- Another opportunity:
  - 2021 Collegeville Workshop on Scientific Software – Software Teams
  - https://collegeville.github.io/CW21/
- Thank you!

ST Capability Assessment Report (CAR)

- Tiered discussion of ECP Software Technology structure, strategy, status and plans
- From high-level overview to details about each team's activities and next steps
- Produced about twice a year
- Includes gap analyses
- E4S scope updated for emerging needs



https://www.exascaleproject.org/wp-content/uploads/2021/01/ECP-ST-CAR-v2.5.pdf

Thank you

https://www.exascaleproject.org

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.

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.