

Accelerating HPC Impact: GPUs, Software Ecosystems, People, Trust



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

NASA Virtual Visit, April 7, 2022

Outline

- ECP, Briefly
- Establishing software ecosystems
- Building an HPC community for the future
- Developing software for GPU systems
- Building trust in computations

ECP in a Nutshell



ECP by the numbers

Takeway: ECP is a big gnarly project!

7
YEARS
\$1.8B

A seven-year, \$1.8B R&D effort that launched in 2016

6
CORE DOE
LABS

Six core DOE National Laboratories: Argonne, Lawrence Berkeley, Lawrence Livermore, Oak Ridge, Sandia, Los Alamos

- Staff from most of the 17 DOE national laboratories take part in the project

3
FOCUS
AREAS

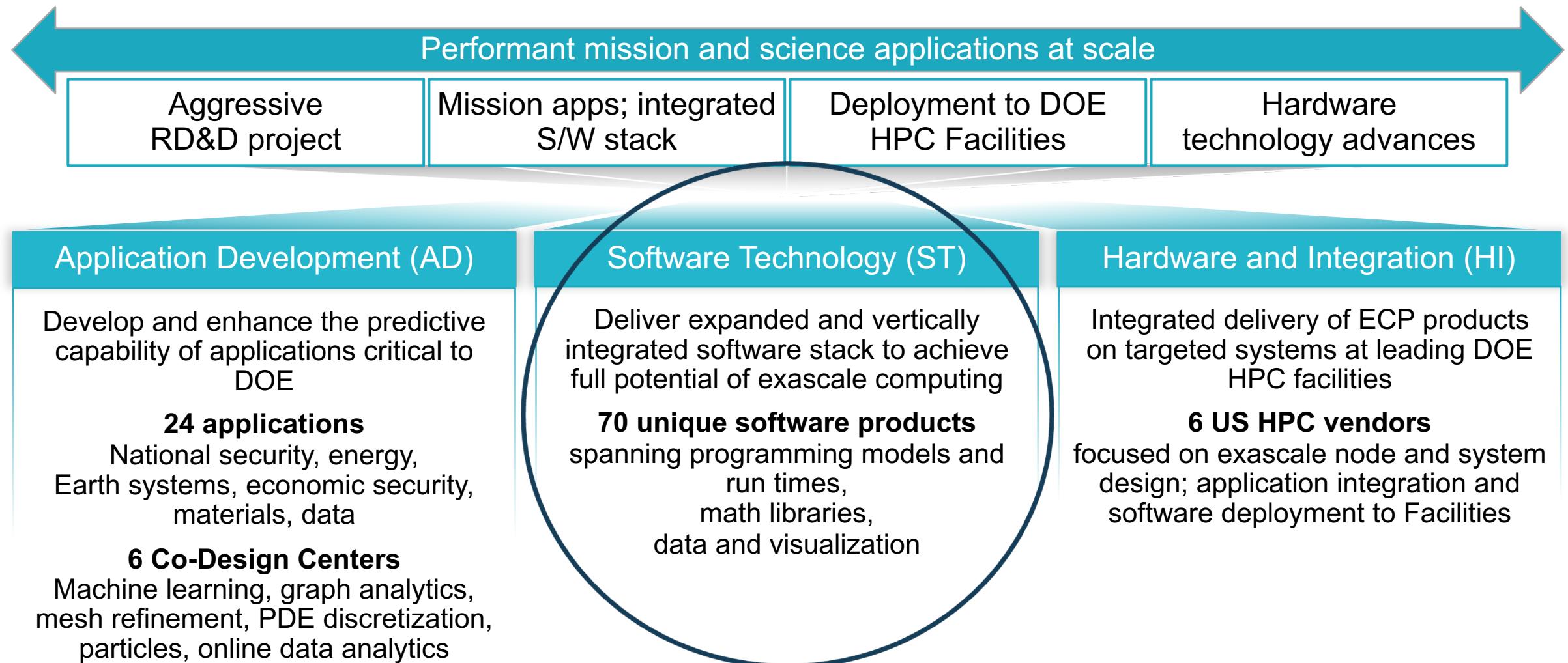
Three technical focus areas: Hardware and Integration, Software Technology, Application Development supported by a Project Management Office

80+
R&D TEAMS
1000
RESEARCHERS

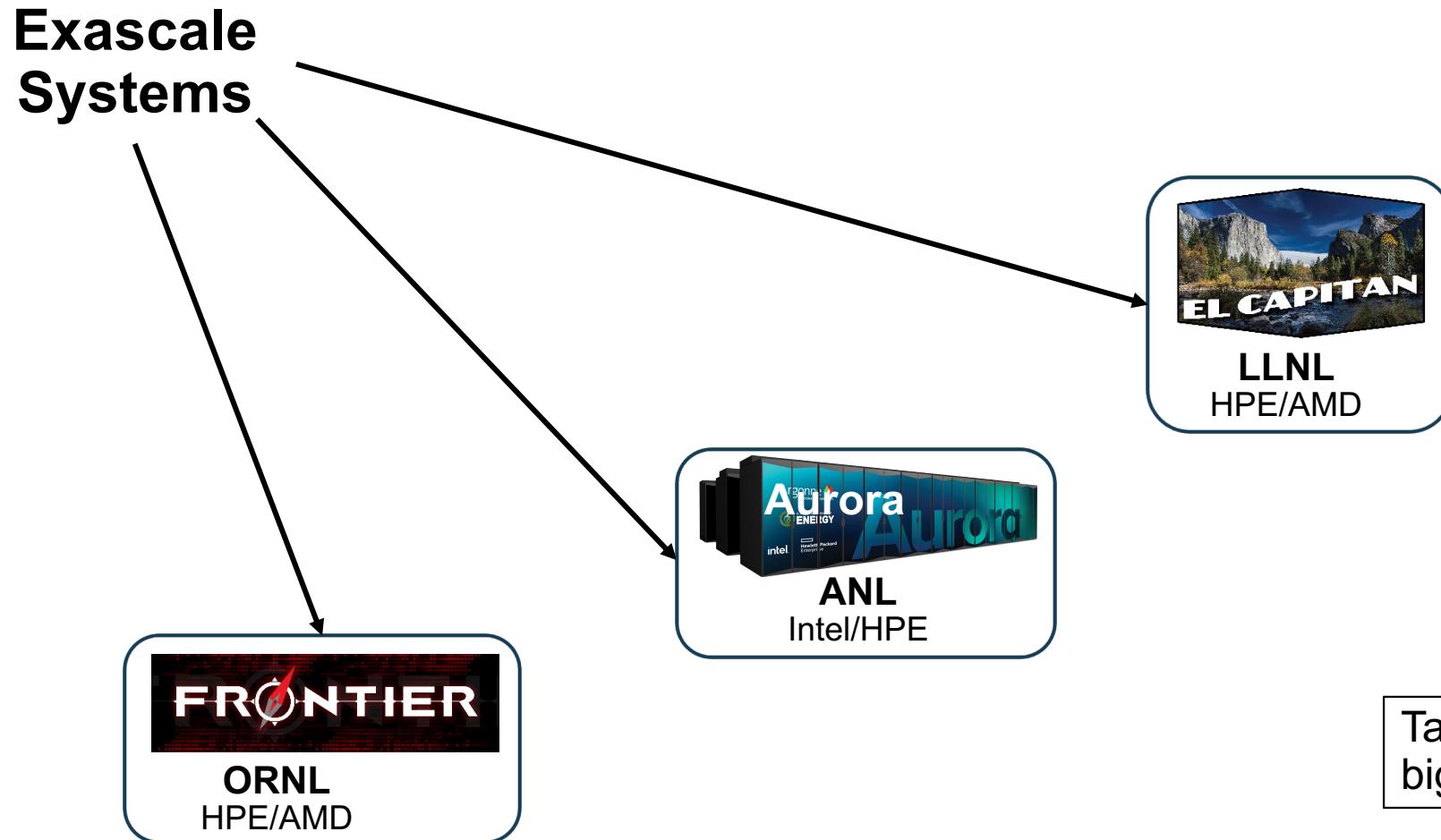
More than 80 top-notch R&D teams

Hundreds of consequential milestones delivered on schedule and within budget since project inception

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



Exascale Systems – Primary targets for ECP Software Teams



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.



Lois Curfman McInnes, Software Technology Deputy Director

Lois is a senior computational scientist in the Mathematics and Computer Science Division of ANL. She has over 20 years of experience in HPC numerical software, including development of PETSc and leadership of multi-institutional work toward sustainable scientific software ecosystems.



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.



Xiaoye (Sherry) Li, Math Libraries (2.3.3)

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



Jim Ahrens, Data and Visualization (2.3.4)

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



Todd Munson, Software Ecosystem and Delivery (2.3.5)

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

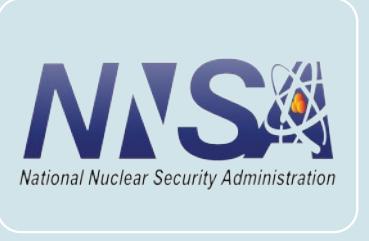
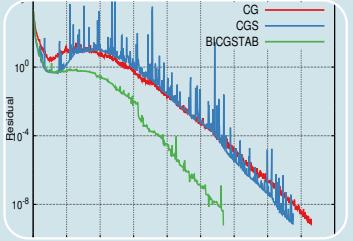
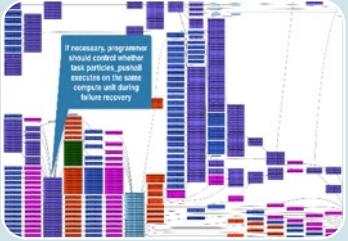


Kathryn Mohror, NNSA ST (2.3.6)

Kathryn is Group Leader for the CASC Data Analysis Group at LLNL. Her work focuses on I/O for extreme scale systems, scalable performance analysis and tuning, fault tolerance, and parallel programming paradigms. She is a 2019 recipient of the DOE Early Career Award.

ECP ST has six technical areas

ECP ST Director: Mike Heroux
ECP ST Deputy Director: L.C. McInnes



Programming Models & Runtimes

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

Development Tools

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

Math Libraries

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

Data and Visualization

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

Software Ecosystem

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

NNSA ST

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

Area Leads:

Rajeev Thakur

Jeff Vetter

Sherry Li

Jim Ahrens

Todd Munson

Kathryn Mohror

ST L4 Leads

- WBS
- Name
- PIs
- PCs - Project Coordinators

ECP ST Stats

- 250 staff
- 70 products
- 35 L4 subprojects
- 30 universities
- 9 DOE labs
- 6 technical areas
- 1 of 3 ECP focus areas

WBS	WBS Name	CAM/PI	PC
2.3	Software Technology	Heroux, Mike, McInnes, Lois	
2.3.1	Programming Models & Runtimes	Thakur, Rajeev	
2.3.1.01	PMR SDK	Shende, Sameer	Shende, Sameer
2.3.1.07	Exascale MPI (MPICH)	Guo, Yanfei	Guo, Yanfei
2.3.1.08	Legion	Mccormick, Pat	Mccormick, Pat
2.3.1.09	PaRSEC	Bosilca, George	Carr, Earl
2.3.1.14	Pagoda: UPC++/GASNet for Lightweight Communication and Global Address Space Support	Hargrove, Paul	Hargrove, Paul
2.3.1.16	SICM	Graham, Jonathan	Turton, Terry
2.3.1.17	OMPI-X	Bernholdt, David	Grundhoffer, Alicia
2.3.1.18	RAJA/Kokkos	Trott, Christian Robert	Trujillo, Gabrielle
2.3.1.19	Argo: Low-level resource management for the OS and runtime	Beckman, Pete	Gupta, Rinku
2.3.2	Development Tools	Vetter, Jeff	
2.3.2.01	Development Tools Software Development Kit	Miller, Barton	Tim Haines
2.3.2.06	Exa-PAPI++: The Exascale Performance Application Programming Interface with Modern C++	Dongarra, Jack	Jagode, Heike
2.3.2.08	Extending HPCToolkit to Measure and Analyze Code Performance on Exascale Platforms	Mellor-Crummey, John	Meng, Xiaozhu
2.3.2.10	PROTEAS-TUNE	Vetter, Jeff	Hornick, Mike
2.3.2.11	SOLLVE: Scaling OpenMP with LLVM for Exascale	Chandrasekaran, Sunita	Oryspayev, Dossay
2.3.2.12	FLANG	McCormick, Pat	Perry-Holby, Alexis
2.3.3	Mathematical Libraries	Li, Sherry	
2.3.3.01	Extreme-scale Scientific xSDK for ECP	Yang, Ulrike	Yang, Ulrike
2.3.3.06	Preparing PETSc/TAO for Exascale	Munson, Todd	Munson, Todd
2.3.3.07	STRUMPACK/SuperLU/FFTX: sparse direct solvers, preconditioners, and FFT libraries	Li, Sherry	Li, Sherry
2.3.3.12	Enabling Time Integrators for Exascale Through SUNDIALS/ Hypre	Woodward, Carol	Woodward, Carol
2.3.3.13	CLOVER: Computational Libraries Optimized Via Exascale Research	Dongarra, Jack	Carr, Earl
2.3.3.14	ALExa: Accelerated Libraries for Exascale/ForTrilinos	Prokopenko, Andrey	Grundhoffer, Alicia
2.3.3.15	Sake: Solvers and Kernels for Exascale	Rajamanickam, Siva	Trujillo, Gabrielle
2.3.4	Data and Visualization	Ahrens, James	
2.3.4.01	Data and Visualization Software Development Kit	Atkins, Chuck	Bagha, Neelam
2.3.4.09	ADIOS Framework for Scientific Data on Exascale Systems	Klasky, Scott	Hornick, Mike
2.3.4.10	DataLib: Data Libraries and Services Enabling Exascale Science	Ross, Rob	Ross, Rob
2.3.4.13	ECP/VTK-m	Moreland, Kenneth	Moreland, Kenneth
2.3.4.14	VeloC: Very Low Overhead Transparent Multilevel Checkpoint/Restart/Sz	Cappello, Franck	Ehling, Scott
2.3.4.15	ExaIO - Delivering Efficient Parallel I/O on Exascale Computing Systems with HDF5 and Unify	Byna, Suren	Bagha, Neelam
2.3.4.16	ALPINE: Algorithms and Infrastructure for In Situ Visualization and Analysis/ZFP	Ahrens, James	Turton, Terry
2.3.5	Software Ecosystem and Delivery	Munson, Todd	
2.3.5.01	Software Ecosystem and Delivery Software Development Kit	Willenbring, James M	Willenbring, James M
2.3.5.09	SW Packaging Technologies	Gamblin, Todd	Gamblin, Todd
2.3.5.10	ExaWorks	Laney, Dan	Laney, Dan
2.3.6	NNSA ST	Mohror, Kathryn	
2.3.6.01	LANL ATDM	Mike Lang	Vandenbusch, Tanya Marie
2.3.6.02	LLNL ATDM	Becky Springmeyer	Gamblin, Todd
2.3.6.03	SNL ATDM	Jim Stewart	Trujillo, Gabrielle

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

Establishing Software Ecosystems



The Growing Complexity of Scientific Application Software Stacks

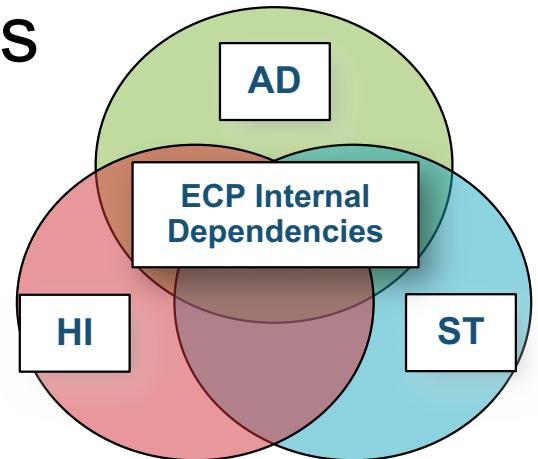


Challenges

- As our software gets more complex, it is getting harder to install tools and libraries correctly in an integrated and interoperable software stack.

ECP apps (AD) are primary consumers of ST products

Dependency Database



View by ST producers

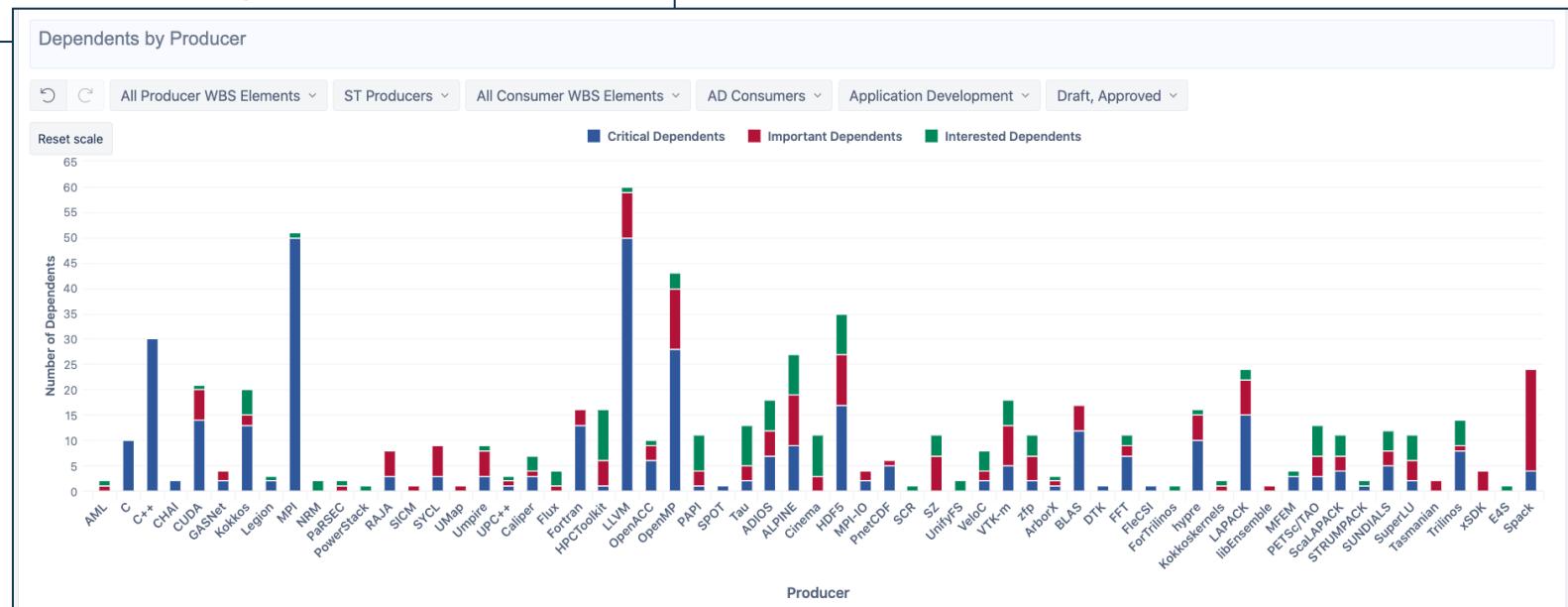
View by AD consumers

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

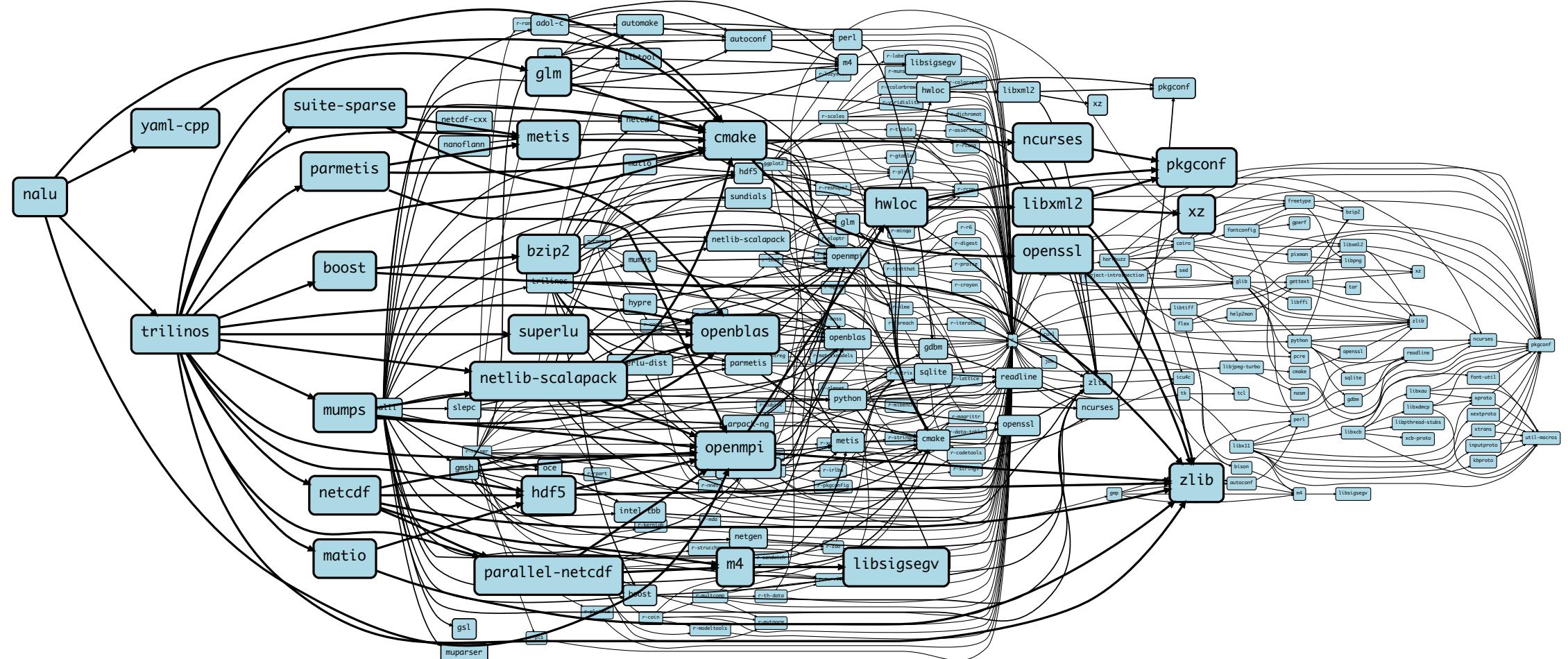
nature computational science

Comment | Published: 22 February 2021

How community software ecosystems can unlock the potential of exascale computing

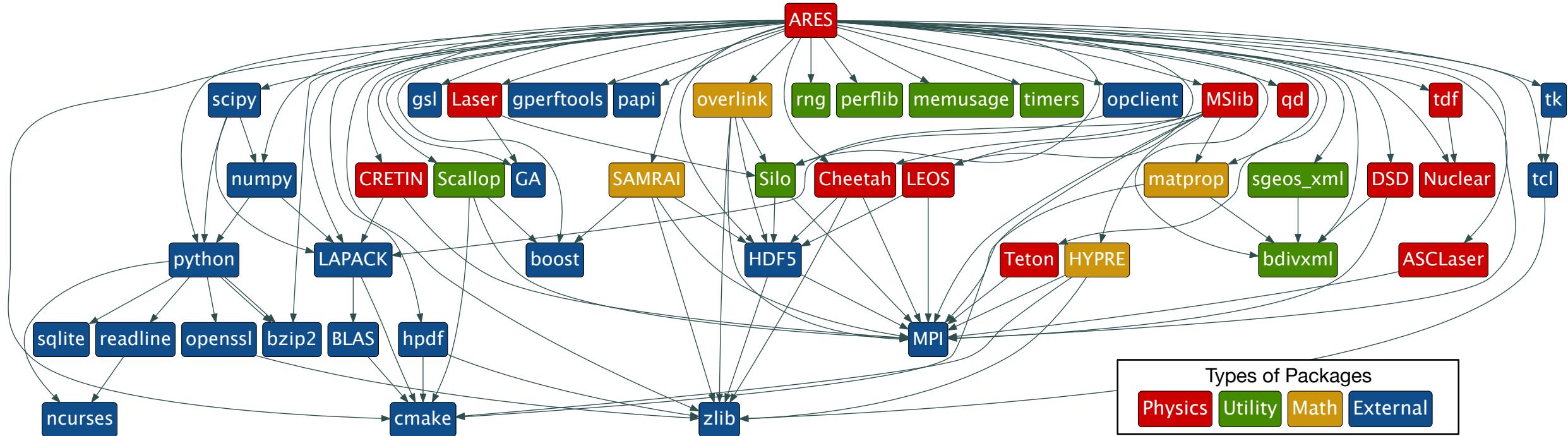


Scientific software is becoming extremely complex



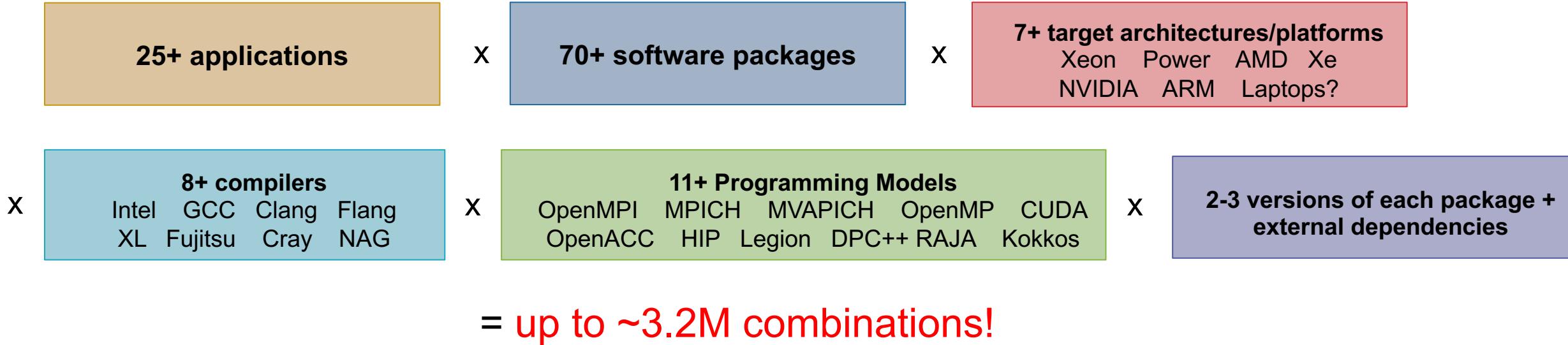
Nalu: Generalized Unstructured Mesh Movement for Parallel Meshing, Migrating Mesh Flow

Even proprietary codes are based on many open source libraries



- Half of this DAG is external (blue); *more than half* of it is open source
- Nearly *all* of it needs to be built specially for HPC to get the best performance

The Exascale Computing Project is building an entire ecosystem



- Every application has its own stack of dependencies.
- Developers, users, and facilities dedicate (many) FTEs to building & porting.
- Often trade reuse and usability for performance.

We must make it easier to rely on others' software!

How to install software on a supercomputer

1. Download all 16 tarballs you need
 2. Start building!

configure **make**

configure **make**

תְּמִימָנָה וְמִבְּרָא כְּמַלְכֵנוּ :

make

A vertical ruler scale from 0 to 10 cm, with major markings every 1 cm and minor markings every 1 mm.

configure **make**

make install **configure** **make**

卷之三

卷之三

卷之三

```
click make make install
```

make install

3. Run code
4. **Segfault!?**
5. Start over...

A Sampler of Products

MPICH is a high performance portable implementation of the **Interface (MPI)** standard.



OpenMP[®]



- No two project alike
- Some personality driven
- Some community driven
- Small, medium, large



Takeaways from product sampler

- Wide range of products and teams: libs, tools, small personality-driven, large community-driven
- Varied user base and maturity: widely used, new, emerging
- Variety of destinations: direct-to-user, facilities, community stacks, vendors, facilities, combo of these
- Wide range of dev practices and workflows from informal to formal
- Wide range of tools: GitHub, GitLab, Doxygen, Readthedocs, CMake, autotools, etc.
- Question at this point might (should?) be:
 - Why are you trying to make a portfolio from this eclectic assortment of products?
- Answer:
 - Each product team charged with challenging tasks:
 - Provide capabilities for next-generation leadership platforms
 - Address increasing software quality expectations
 - While independently developed, product compatibility and complementarity improvements matter
 - Working together on these frontiers is better than going alone

Takeaways from software complexity

- The ECP software ecosystem is truly a complex system, not just complicated
- Plan, execute, track and assess. Repeat
- Challenges are emergent: technical, sociological, and cognitive

Responding to complexity: Software Ecosystem via Platforms



Software Platforms: “Working in Public” Nadia Eghbal



- Platforms in the software world are digital environments that intend to improve the value, reduce the cost, and accelerate the progress of the people and teams who use them
- Platforms can provide tools, workflows, frameworks, and cultures that provide a (net) gain for those who engage
- Eghbal Platforms:

	HIGH USER GROWTH	LOW USER GROWTH
HIGH CONTRIBUTOR GROWTH	Federations (e.g., Rust)	Clubs (e.g., Astropy)
LOW CONTRIBUTOR GROWTH	Stadiums (e.g., Babel)	Toys (e.g., ssh-chat)

About Platforms and ECP

- The ECP is commissioned to provide new scientific software capabilities on the frontier of algorithms, software and hardware
- The ECP provides platforms to foster collaboration and cooperation as we head into the frontier:
 - **E4S:** a comprehensive portfolio of ECP-sponsored products and dependencies
 - **SDKs:** Domain-specific collaborative and aggregate product development of similar capabilities

Delivering an open, hierarchical software ecosystem

More than a collection of individual products

Levels of Integration

Product

Source and Delivery

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

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

- Standard workflow
- Existed before ECP



SDKs



ECP ST Open Product Integration Architecture

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

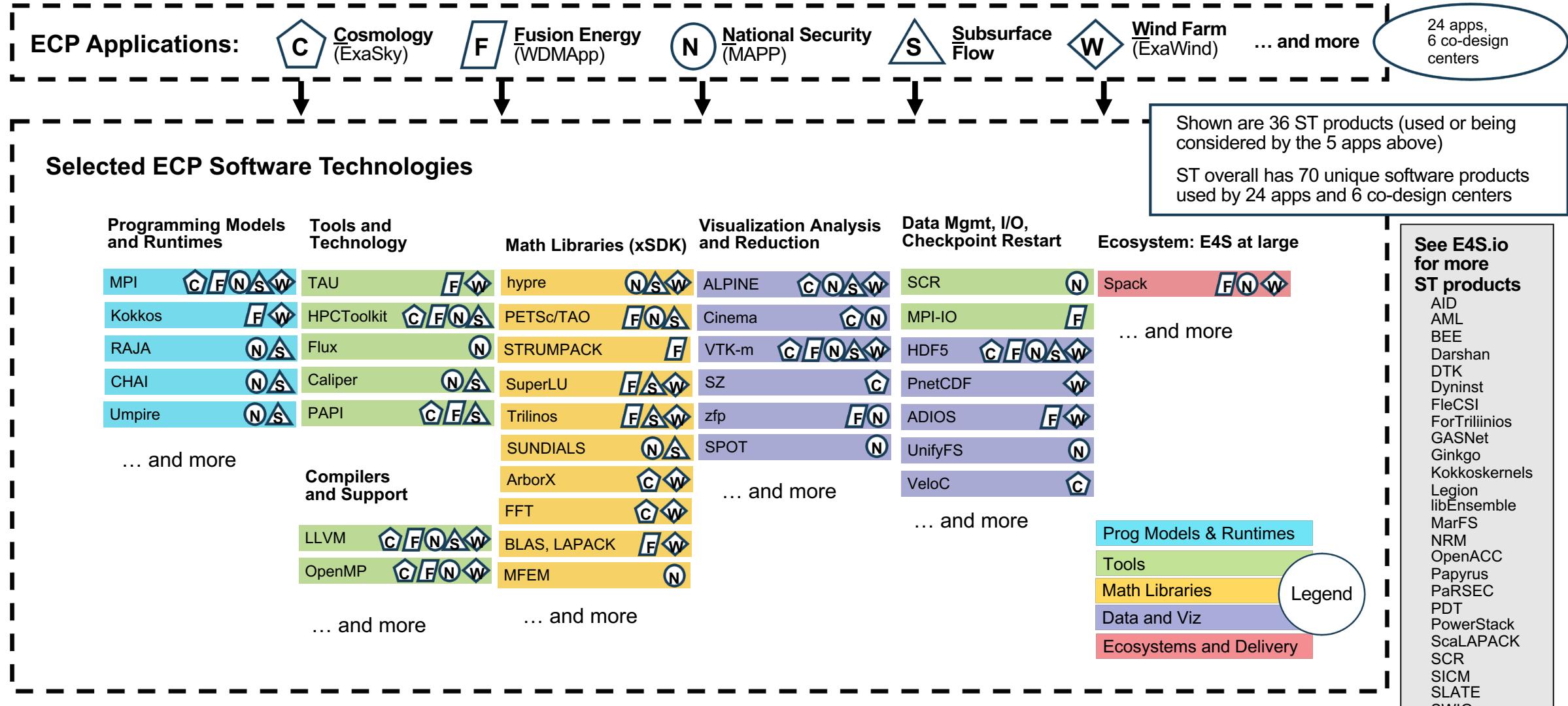
Source: SDK teams; Non-ECP teams (policy compliant, spackified)
Delivery: Apps directly; spack install sdk; future: vendor/facility

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

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.
- <https://spack.io>

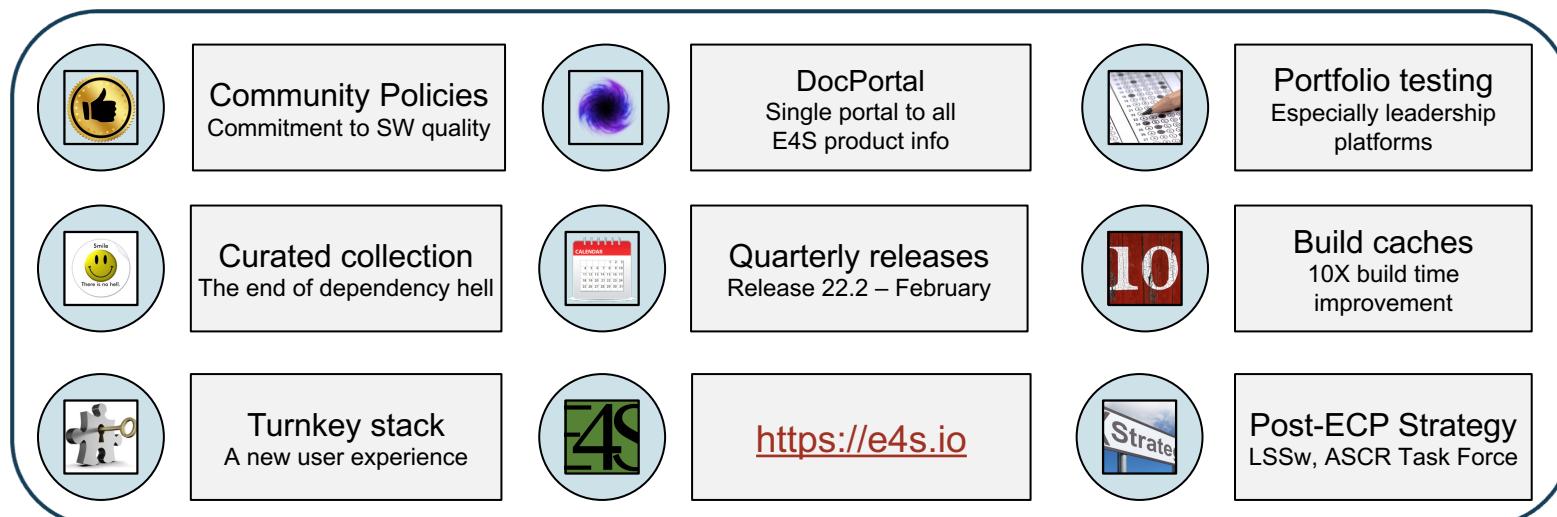
ECP applications require consistency across the software stack



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.

Extreme-scale Scientific Software Stack (E4S)

- E4S: HPC software ecosystem – a curated software portfolio
- A **Spack-based** distribution of software tested for interoperability and portability to multiple architectures
- Available from **source, containers, cloud, binary caches**
- Leverages and enhances SDK interoperability thrust
- Not a commercial product – an open resource for all
- Growing functionality: Feb 2022: E4S 22.02 – 100 full release products



<https://spack.io>

Spack lead: Todd Gamblin (LLNL)



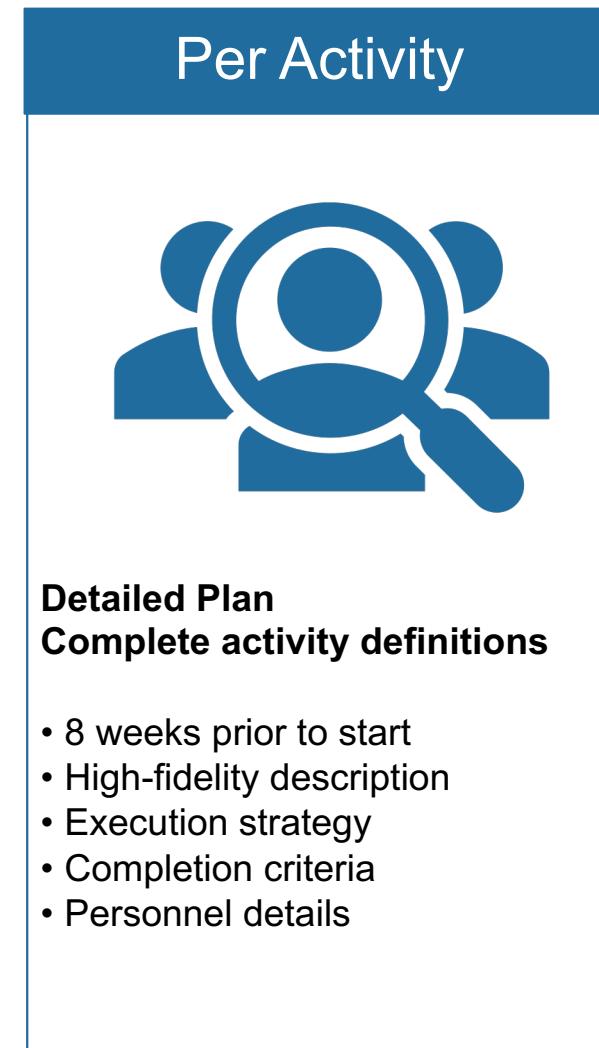
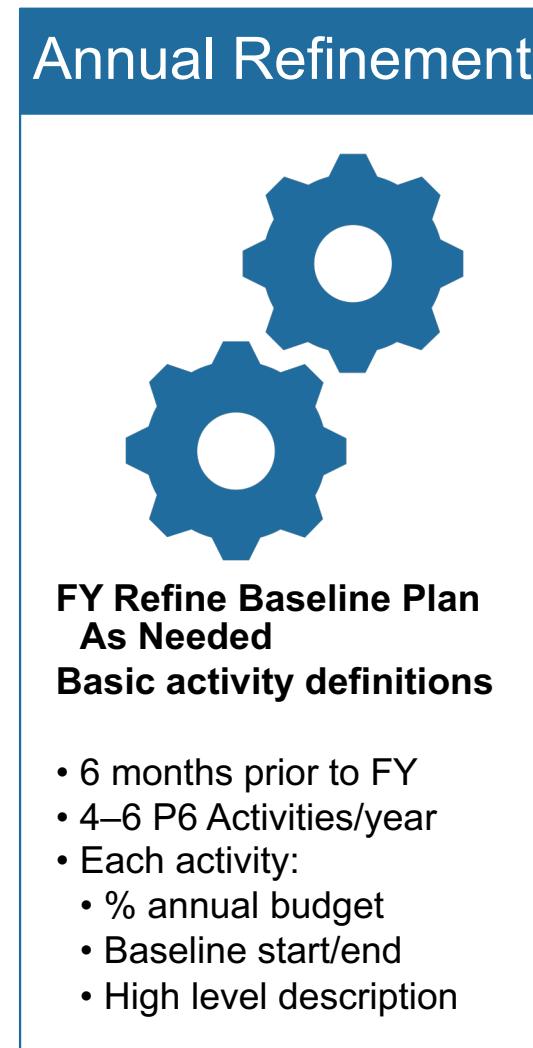
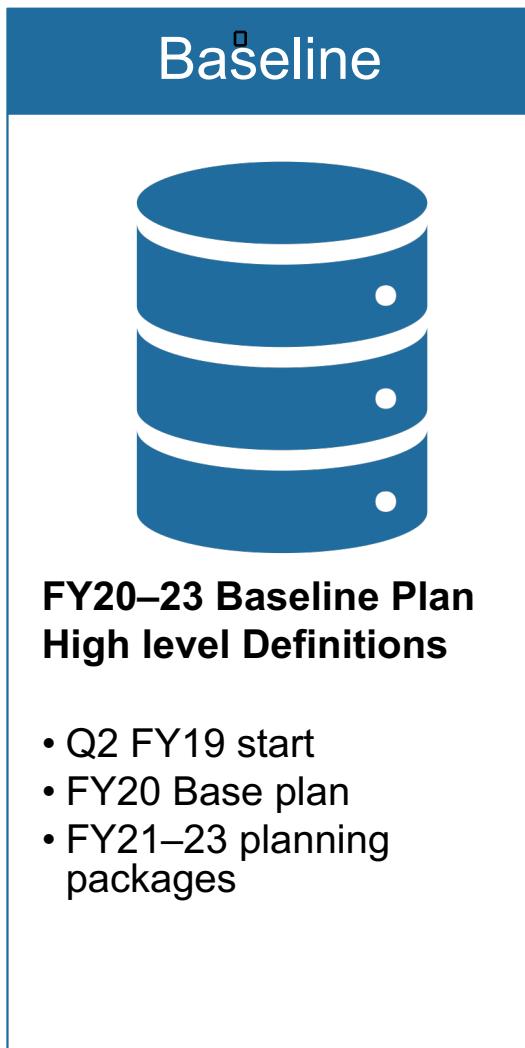
<https://e4s.io>

E4S lead: Sameer Shende (U Oregon)



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

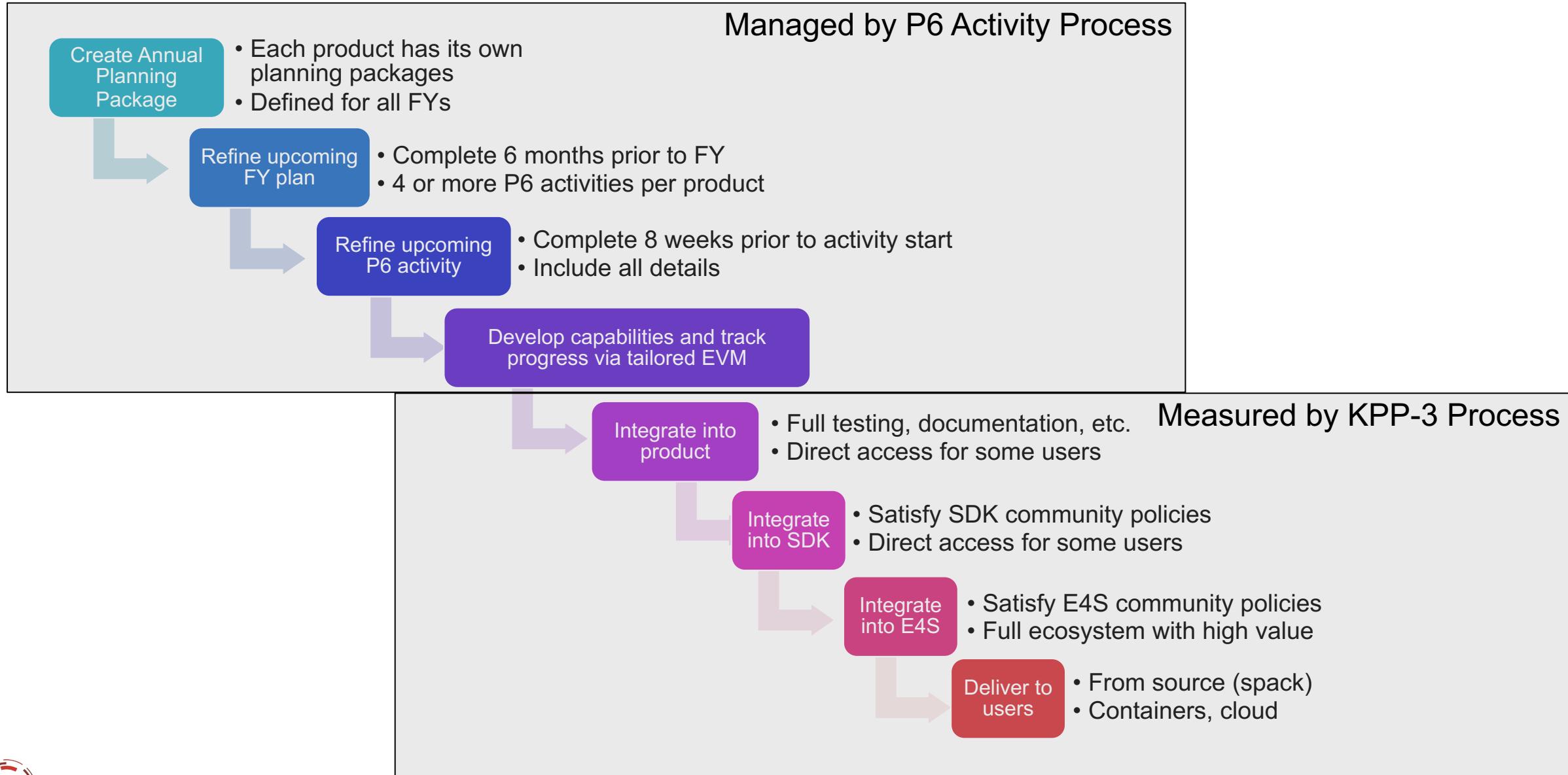
ECP ST Planning Process: Hierarchical, three-phase, cyclical



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
 - Approximately 1 FTE-year cost
 - Single capability intended to be broadly useful
- **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)
 - Integration typically includes availability via E4S
 - 1 point for a capability integrated into a particular client environment for a particular platform
 - Example: New PETSc solver integrated into WDMApp on Frontier – 1 point

ECP ST Lifecycle summary



E4S and SDKs as platforms are providing tremendous value

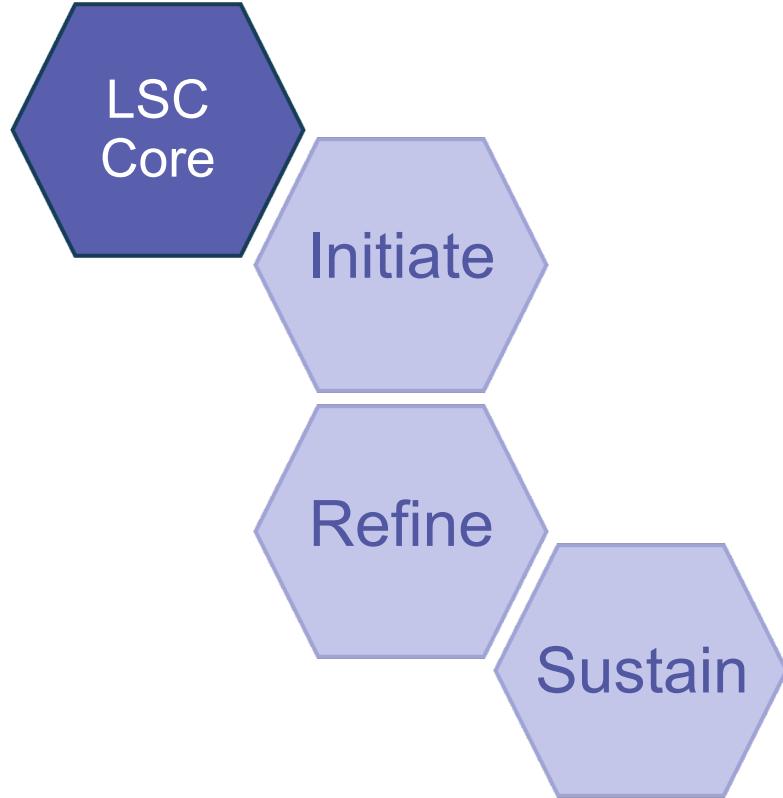
Activity	SDKs	E4S
Planning	Transparent and collaborative requirements, analysis and design, delivery – better plans, less effort, improved complementarity	Campaign-based portfolio planning coordinated with Facilities, vendors, community ecosystem, non-DOE partners
Implementation	Leverage shared knowledge,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

LSC Disclaimer

- The following descriptions are notional
- We are actively engaging stakeholders to evolve our approach
- Goal: Be ready to execute a plan for post-ECP software efforts

ECP's Evolving Vision for Software Sustainability

A Leadership Software Center (LSC) with core efforts + “sprint-like” campaigns



Leadership Software Campaigns

Starting point: ECP's Extreme-Scale Scientific Software Stack, **E4S**, a Spack-based distribution of software tested for interoperability and portability to multiple architectures (e4s.io)

	LSC-1 FY 2024-26	LSC-2 FY 2027-29	LSC-3 FY 2030-32
Next phase core SW	✓	✓	✓
Establish AI/ML SDK	✓		
Next phase AI/ML		✓	✓
Scope Edge SDK	✓		
Establish Edge SDK		✓	
Next phase Edge			✓
Scope Quantum SDK		✓	
Establish Quantum SDK			✓
Contingency	✓	✓	✓

A component of our response to a Feb 2021 IPR Recommendation: *Identify long term options for supporting and evolving the software ecosystem developed and used throughout the ECP project.*

LSC Execution Approach

Plan, Execute, Track, Assess Lifecycle

- All activities governed by phased development process
- Executed as “campaigns”: LSC-1, LSC-2, ...
- Tailored agile approach
- Hierarchical approach:
 - Multi-year baseline as campaign
 - Refine annually
 - Add fidelity per milestone at “last responsible moment”

Change Management Process:

- Changes from campaign base plan managed by a process
- Any changes to cost, scope and schedule
- Explicit review process determined by degree of change
- Change control process assures lightweight transparency
- Objective: **Always do most important work at any time**

Capability Integration Strategy

DOE software products have four primary integration targets:

- **Vendors:** Specific HPC enhancements, integrated into system vendor stacks
- **Community SW:** C++, Fortran, LLVM
- **Facilities:** Tuned open-source SW for key platforms
- **Direct to apps:** Application teams download and build
- **Note:** Some products are available via 2 – 3 of the above targets

LSC goals:

- **Establish and ensure quality standards** for LSC product development and delivery
- Assure that funded projects **develop and deliver** to one or more integration targets
- **Track and assess** integration status of new capabilities

Building an HPC Community for the Future





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

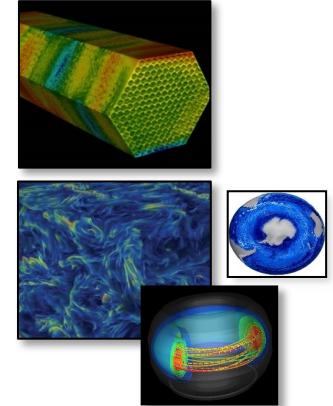
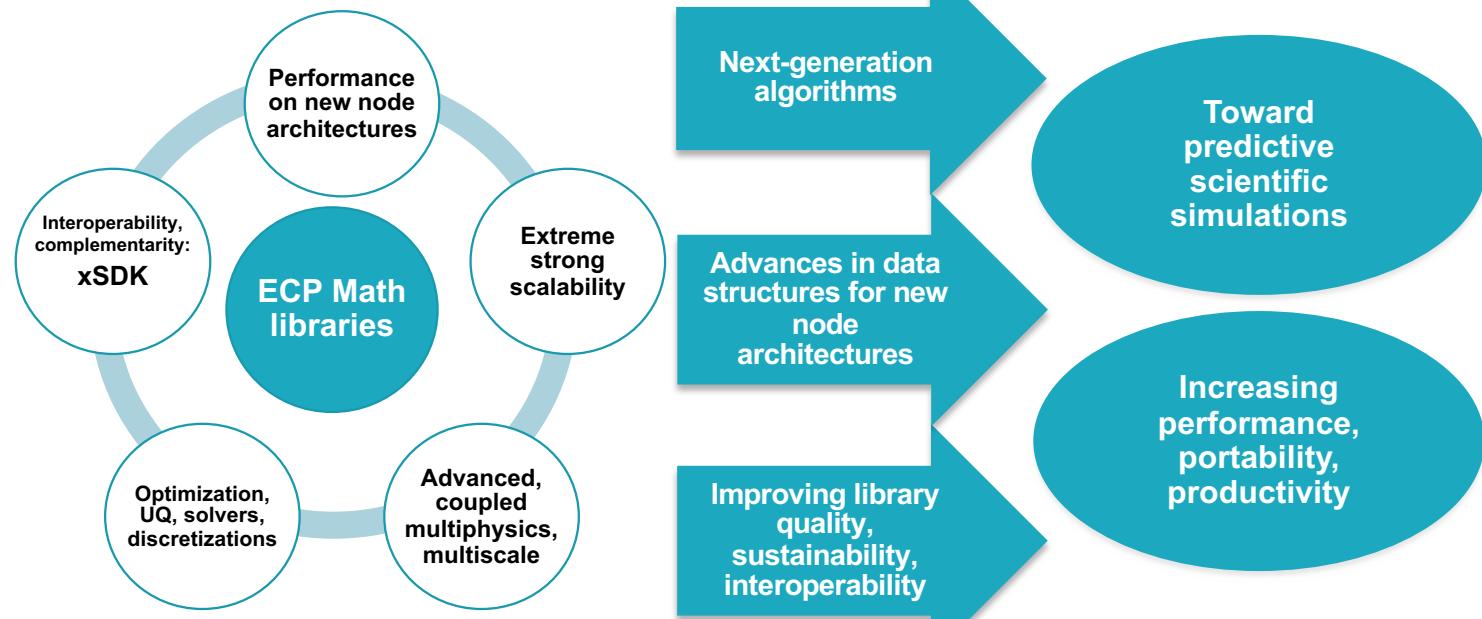
xSDK release 0.6.0 (Nov 2020)

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:



Ulrike Yang, PI

xSDK4ECP: Project strategy

Goals: Create a value-added aggregation of ECP mathematics libraries, to increase the combined usability, standardization and interoperability of these libraries, as needed to support large-scale multiphysics and multiscale problems.

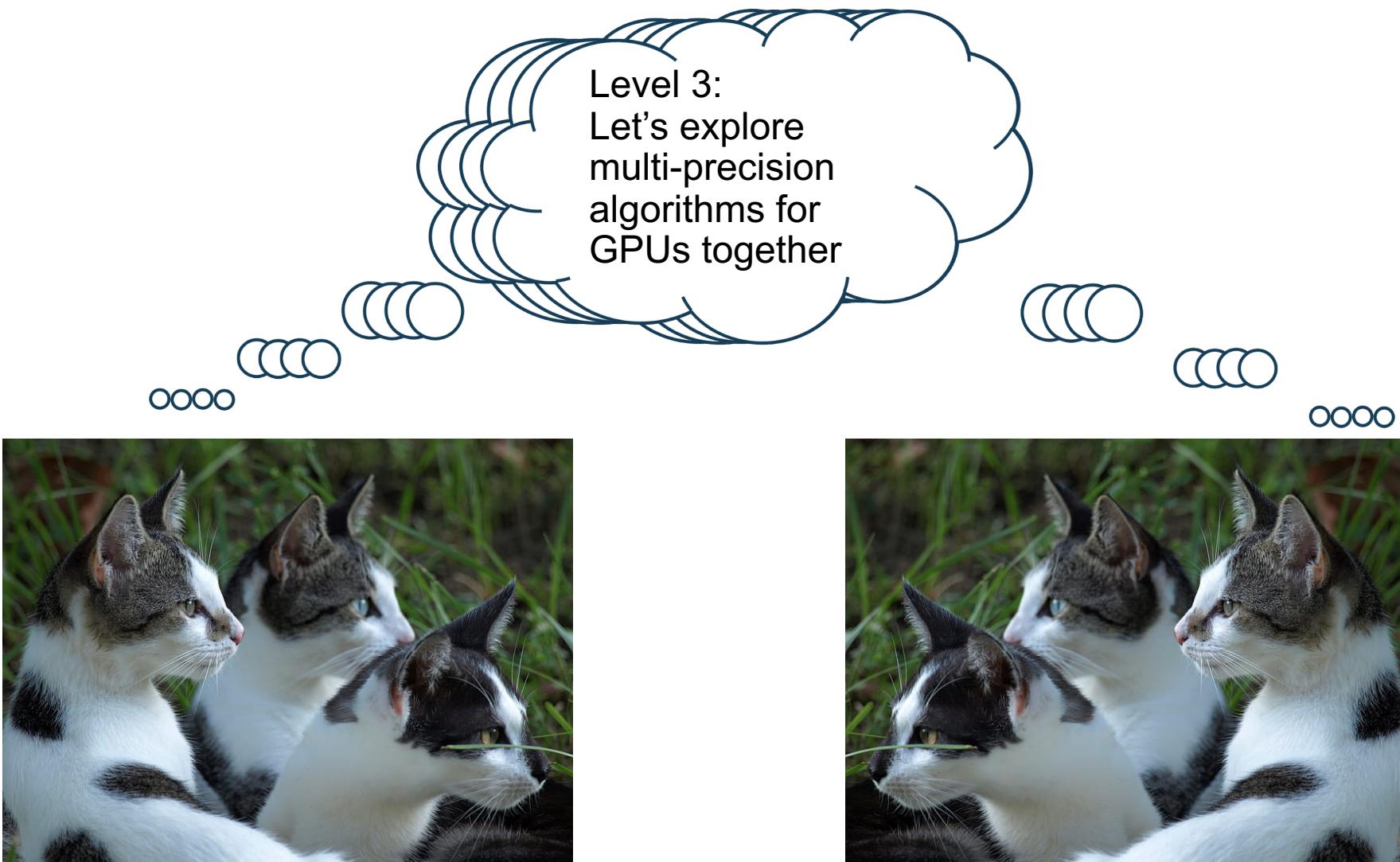
Project Team	(FY20)
Aaron Fisher	Natalie Beams
Ahmad Abdelfattah	Nick Higham
Asim YarKhan	Osni Marques
Barry Smith	Piotr Luszczek
Boyana Norris	Robert Falgout
Carol Woodward	Samuel Knight
Cody Balos	Sarah Osborn
Damien Lebrun-Grandie	Satish Balay
Daniel Arndt	Scott Kruger
Daniel Osei-Kuffuor	Sherry Li
Erik Boman	Siva Rajamanickam
Erin Carson	Stan Tomov
Gerald Ragghianti	Sri Pranesh
Hengrui Lu	Stephen Hudson
Hartwig Anzt	Stuart Slattery
Ichi Yamazaki	Terry Cojean
Jack Dongarra	Thomas Grützmacher
Jamie Finney	Tobias Ribisel
Jennifer Loe	Tomas Gergelits
Jim Demmel	Tzanio Kolev
Jim Willenbring	Ulrike Meier Yang
Keita Teranishi	Veselin Dobrev
Lois Curfman McInnes	Viktor Reshniak
Mark C. Miller	Wenjun Ge
Mark Gates	Yang Liu
Mike Heroux	Younghyun Cho
Miroslav Stoyanov	...

Scope:

- Development of xSDK community policies for improved software quality and combined build infrastructure
- Coordinated releases of complete xSDK with testing, documentation, packaging and deployment
- Development of new interoperability layers between xSDK members
- Outreach to ECP (and general) community through surveys, summary reports, tutorials, presentations, articles, etc
- Subprojects: adaptive execution (LBNL, UCB), code quality (ANL, U of Oregon)
- Multiprecision effort (lead: H. Anzt), see 2nd part of review

An SDK Maturity Model or the Benefits of Coop-etition

Scenario: Two Product Teams in the Same SDK (e.g., math libs SDK – xSDK)



- Establish coop-etition:
 - Lower-cost comparison of products, increased incentives for improvement
 - Encourages SDK participation: learn from each other, be in the know
- Lead to community growth:
 - Humanizes the other teams
 - Exposes opportunities to share strengths
- Retain autonomy of SDK member teams
 - Each team makes its own informed decisions
 - Better decisions from shared study of new ideas
- Challenges
 - Coordination has overhead
 - Poor habits can spill over (but so can good ones)
- Bottom line: SDKs as we define them:
 - Are platforms to support open, collaborative scientific discovery across teams
 - Make sharing and cooperation, which are fundamental to science, easier to realize

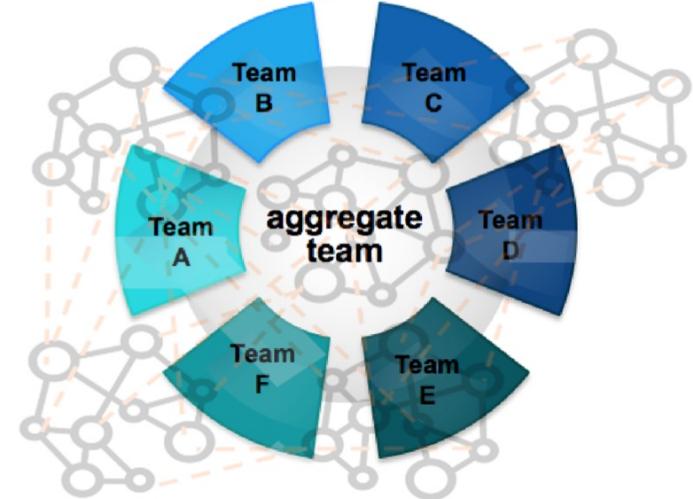
Takeaways from SDKs

ECP: A “team of teams”

An aggressive research, development and deployment project, focused on delivery of mission-critical applications, an integrated software stack, and exascale hardware technology advances

Multilayered collaboration across the ECP community

- Ref: [Scaling productivity and innovation on the path to exascale with a “team of teams” approach](#), E. Raybourn et al, 2019



Networked teams, at scale. Multidisciplinary expertise, such as:

Research software engineers (RSEs)

Computational scientists
and engineers

Data scientists and
engineers

Performance engineers

Cognitive and
social scientists

Applications scientists

Computer scientists

Applied mathematicians

Project
coordinators

Stakeholders

And more ...

Advancing scientific productivity through better scientific software

Science through computing is only as good as the software that produces it.

1 Customize and curate methodologies

- Target scientific software productivity and sustainability
- Use workflow for best practices content development



2 Incrementally and iteratively improve software practices

- Determine high-priority topics for improvement and track progress
- *Productivity and Sustainability Improvement Planning (PSIP)*

3 Establish software communities

- Determine community policies to improve software quality and compatibility
- Create Software Development Kits (SDKs) to facilitate the combined use of complementary libraries and tools

4 Engage in community outreach

- Broad community partnerships
- Collaboration with computing facilities
- Webinars, tutorials, events
- *WhatIs* and *HowTo* docs
- Better Scientific Software site (<https://bssw.io>)

IDEAS-ECP team

<https://www.ideas-productivity.org>



Ann Almgren (LBNL)



Ross Bartlett (SNL)



David E. Bernholdt (ORNL)
Institutional PI



Anshu Dubey (ANL)
Institutional PI



Lawrence Livermore National Laboratory



O UNIVERSITY OF OREGON



Elsa Gonsiorowski (LLNL)
Institutional PI



Patricia Grubel (LANL)



Rinku Gupta (ANL)



Rebecca Hartman-Baker (LBNL)
Computing Facility Liaison



Mark Miller (LLNL)



J. David Moulton (LANL)
Institutional PI



Hai Ah Nam (LBNL)



Boyana Norris (U. Oregon)
Institutional PI



Michael Heroux (SNL)
Lead Co-PI



Axel Huebl (LBNL)



Lena Lopatina (LANL)
Computing Facility Liaison



Rose Lynch (ANL)



Elaine Raybourn (SNL)
Institutional PI



Katherine Riley (ANL)
Computing Facility Liaison



David Rogers (ORNL)
Computing Facility Liaison



Jean Shuler (LLNL)
Computing Facility Liaison



Addi Malviya Thakur (ORNL)



Osni Marques (LBNL)
Institutional PI



Lois Curfman McInnes (ANL)
Lead Co-PI



Reed Milewicz (SNL)



Ben Sims (LANL)



Deborah Stevens (ANL)



Greg Watson (ORNL)



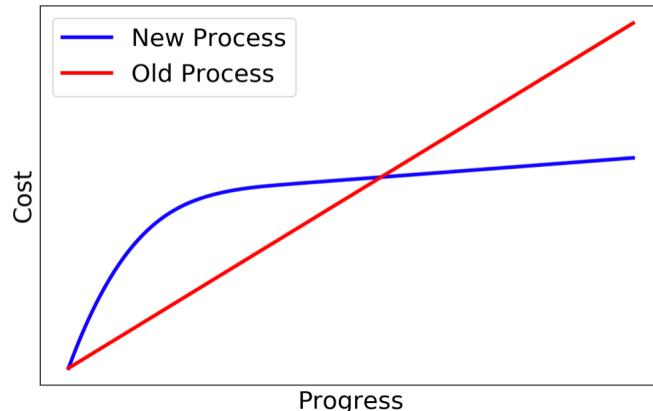
Jim Willenbring (SNL)

IDEAS-ECP Alumni

- Satish Balay (ANL)
- Lisa Childers (ANL)
- Todd Gamblin (LLNL)
- Judy Hill (ORNL)
- Steve Hudson (ANL)
- Christoph Junghans (LANL)
- Alicia Klinvex (SNL)
- Shannon Lindgren (ANL)
- Jared O'Neal (ANL)
- Michele Rosso (LBNL)
- Barry Smith (ANL)
- Louis Vernon (LANL)
- Paul Wolfenbarger (SNL)

Productivity and sustainability improvement planning: Recent successes with PSIP on HDF5

<https://bssw.io/psip>

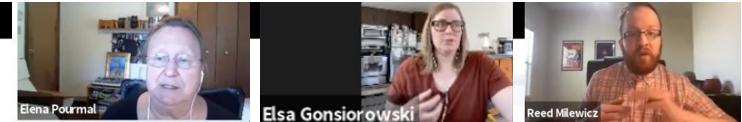


"The PSIP project had an immediate impact on our community. With the GitHub move we see increasing amounts of small but very valuable contributions to make HDF5 code and documentation better." — **Elena Pourmal, Director of Engineering, The HDF Group**

HDF5 improvement goals - achieved by using PSIP progress tracking cards (PTC)

- Modernize processes for handling [documentation \(PTC\)](#)
- Move HDF5 from a THG managed Bitbucket instance to [GitHub \(PTC\)](#)
- Define and adopt a set of consistent [coding standards \(PTC\)](#)

Conclusion



PSIP allows you to realize process improvements with minimal disruption to any current development.

- *By now you should understand ...*
- A practice that can help your team mitigate technical risk and develop software with confidence. (PSIP)
- How to identify topics for improvement by rating your project
- Progress tracking cards (PTC)
- Online resources such as RateYourProject and the PTC Catalog
- Integrating PTCs into your projects



Enabling Software Quality

<https://bssw.io/psip/>



IDEAS Outreach

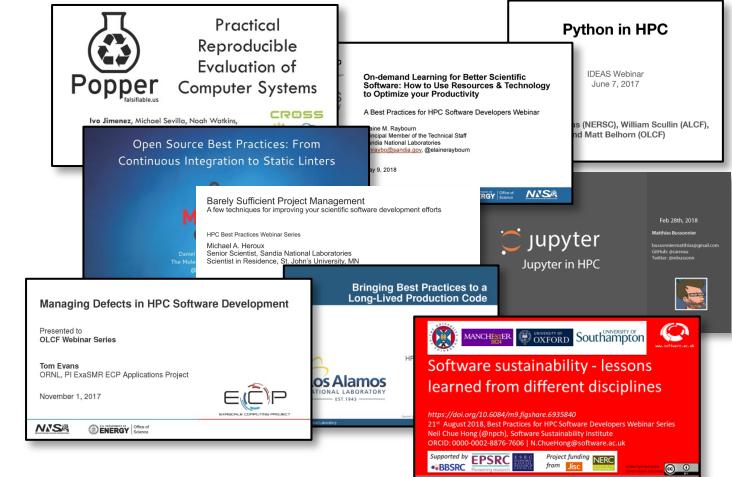
Lead: David Bernholdt

Better Scientific Software Tutorials

- Covering issues of developer productivity, software sustainability and reliability, with a special focus on the challenges of complex, large-scale HPC
 - software design, agile methodologies, Git workflows, reproducibility, software testing, continuous integration testing, refactoring, and more
- <https://bssw-tutorial.github.io>
- Recent venues
 - Supercomputing (2016-2021)
 - SEAS's Improving Scientific Software (2021)
 - ECP Annual Meeting (2017-2021)
 - ISC (2017-2019, 2021), ATPESC (2016-2021)

BSSw Tutorial
@ SC21, Nov 15

Mailing list to follow IDEAS-led events (webinars, panels, BOFs, etc.): <http://eepurl.com/cQCyJ5>



Webinar Series: Best Practices for HPC Software Developers (HPC-BP)

- Covering topics in software development and HPC
- <https://ideas-productivity.org/events/hpc-best-practices-webinars>
- Lead: Osni Marques
- Presented by the community to the community
- Monthly series, since May 2016 (offered live and archived)
 - [Best Practices for HPC Software Developers: The First Five Years of the Webinar Series](#), O. Marques and D. Bernholdt, Oct 2021



IDEAS Outreach

Lead: David Bernholdt

Technical Meetings and Birds of a Feather Sessions

- Creating opportunities to talk about software development, productivity, and sustainability
- <https://ideas-productivity.org/events>
- Minisymposia
 - SIAM CSE, SIAM PP (2015-2022), PASC (2018, 2019)
 - Ref: [A Look at Software-Focused Topics at SIAM CSE21](#), March 2021
- Thematic poster sessions
 - SIAM CSE (2017, 2019, 2021)
- BOF sessions
 - Software Engineering and Reuse in Modeling, Simulation and Data Analytics for Science and Engineering
 - Supercomputing (2015-2021), ISC (2019)
- [Collegeville Workshop Series on Scientific Software](#),
 - Ref: [Software Team Experiences and Challenges](#), K. Beattie et al, Oct 2021

Panel Series: Performance Portability & ECP

- Lead: Anshu Dubey (2020 series). Refs:
 - [Performance Portability in the Exascale Computing Project: Exploration Through a Panel Series](#), A. Dubey et al, IEEE CiSE, Sept 2021
 - SIAM CSE21 minisymposium: <https://doi.org/10.6084/m9.figshare.c.5321441>
 - Minisymposium accepted for ECCOMAS 2022

Panel Series: Strategies for Working Remotely

- Exploring strategies for working remotely, with emphasis on how HPC teams can be effective and efficient in long-term hybrid settings
- <https://www.exascaleproject.org/strategies-for-working-remotely>
- Lead: Elaine Raybourn
- Quarterly series, since April 2020 (offered live and archived)
- Ref: [Why We Need Strategies for Working Remotely: The ECP Panel Series](#), E. Raybourn, SC20 State of the Practice, Nov 2020



BSSw Fellowship: Meet the Fellows

Meet Our Fellows

The BSSw Fellowship program gives recognition and funding to leaders and advocates of high-quality scientific software. Meet the Fellows and Honorable Mentions and learn more about how they impact Better Scientific Software.

[Fellowships Overview](#)
[Apply](#)
[Meet Our Fellows](#)
[BSSw Fellowship FAQ](#)

Community Growth

2018 - 2021

2018 Class

Fellows



Jeffrey Carver
University of Alabama

Improving code quality through modern peer code review



Ivo Jimenez
University of California, Santa Cruz

Enabling reproducible research through automated computational experimentation



Daniel S. Katz
University of Illinois at Urbana-Champaign, National Center for Supercomputing Applications

Giving software developers long overdue credit through principles for software citation



Andrew Lumsdaine
Pacific Northwest National Laboratory, University of Washington, Northwest Institute for Advanced Computing

Giving efficient use of modern C++ for high-performance computing

2019 Class

Fellows



Rene Gassmoeller
University of California, Davis

Guiding your scientific software project from inception to long-term sustainability



Ignacio Laguna
Lawrence Livermore National Laboratory

Improving the reliability of scientific applications by analyzing and debugging floating-point software



Tanu Malik
DePaul University

Reducing technical debt in scientific software through reproducible containers

2020 Class

Fellows



Kyle Niemeyer
Oregon State University

Educating academics on best practices for developing research software



Nasir Eisty
University of Alabama

Automating testing in scientific software



Damien Rouson
Sustainable Horizons Institute, Sourcey Institute

Introducing agile scientific software development to underrepresented groups



Cindy Rubio-Gonzalez

University of California, Davis

Improving the reliability and performance of numerical software

2021 Class

Fellows



Marisol García-Reyes
Farallon Institute

Increasing accessibility of data & cloud technologies



Mary Ann Leung
Sustainable Horizons Institute

Increasing developer productivity and innovation through diversity



Chase Million
Million Concepts

Project management best practices for research software



Amy Roberts
University of Colorado Denver

Enabling collaboration through version control user stories

Honorable Mentions



Keith Beattie
Lawrence Berkeley National Laboratory

Computational Research Division, Computer Systems Engineer



Julia Stewart Lowndes
National Center for Ecological Analysis and Synthesis (NCEAS), UC Santa Barbara

Openscapes Director



Jonathan Madsen
Lawrence Berkeley National Laboratory

NERSC, Application Performance Specialist



Addi Thakur Malviya
Oak Ridge National Laboratory

Software Engineering Group, Group Leader



What is BSSw?

Community-based hub for sharing information on practices, techniques, and tools to improve developer productivity and software sustainability for computational science.

We want and *need* contributions from the community ...
Join us!

- **Types of content**
 - Informative articles
 - Curated links
 - Highlight other web-based content
 - Events
 - WhatIs, HowTo docs
 - Blog articles

BSSw.io
editor in chief:
Rinku Gupta

The screenshot shows the BSSw homepage. At the top, there's a dark blue header with the BSSw logo, "better scientific software", and a search icon. Below the header is a yellow banner with a navigation bar containing "Resources", "Blog", "Events", "About", and a magnifying glass icon. A blue arrow points from the "Blog" link in the banner down to the "Recent articles" section. Another blue arrow points from the "Receive our email digest" button in the banner down to the "Receive our email digest" button in the main content area. A third blue arrow points from the "Events" link in the banner down to the "Recent articles" section. In the center, there's a large image of a computer screen displaying a 3D visualization of a simulation. Overlaid on the image is a blue box containing the URL "https://bssw.io".

Receive our email digest

Recent articles

- [The Contributions of Scientific Software to Scientific Discovery](#), K. Keahey & R. Gupta
- [Software Team Experiences and Challenges](#), C. Balos, J. Brown, G. Chourdakis et al.
- [Performance Portability and the ECP Project](#), A. Dubey
- [Testing Non-Deterministic Research Software](#), N. Eisty,
- [What Does This Line Do? The Challenge of Writing a Well-Documented Code](#), M. Stoyanov

Better Scientific Software: 2020 Highlights



- [Unit Testing C++ with Catch](#), M. Dewing
- [The Art of Writing Scientific Software in an Academic Environment](#), H. Anzt
- [FLASH5 Refactoring and PSIP](#), A. Dubey & J. O'Neal
- [Software Sustainability in the Molecular Sciences](#), T. Windus & T.D. Crawford
- [Working Effectively with Legacy Code](#), R. Bartlett
- [Building Community through Software Policies](#), P. Luszczek & U.M. Yang
- [Continuous Technology Refreshment: An Introduction Using Recent Tech Refresh Experiences on VisIt](#), M. Miller & H. Auten

Leadership Scientific Software (defn)

- Libraries, tools and environments that
 - Contribute to scientific discovery and insight in
 - New and emerging computing environments
- Push the boundary of feasibility
 - Enabling
 - Larger scale, higher fidelity and greater integration of
 - Advanced computing ecosystems
- Does “leadership” limit the scope of discussion?
 - Yes, we are directly focused on non-commodity environments, but:
 - Still use laptops, desktops, CPU clusters as part of our development efforts
 - Many of our tools and libraries need to be available everywhere
 - Non-commodity focus does not mean we work only on non-commodity systems
- Focus is on efforts that include co-design of
 - **Computing platforms:** Modeling & simulation, AI/ML, edge: at scale
 - **System software:** Collaborative co-design with vendors
 - **Science-specific tools and libraries:** What we are developing for users

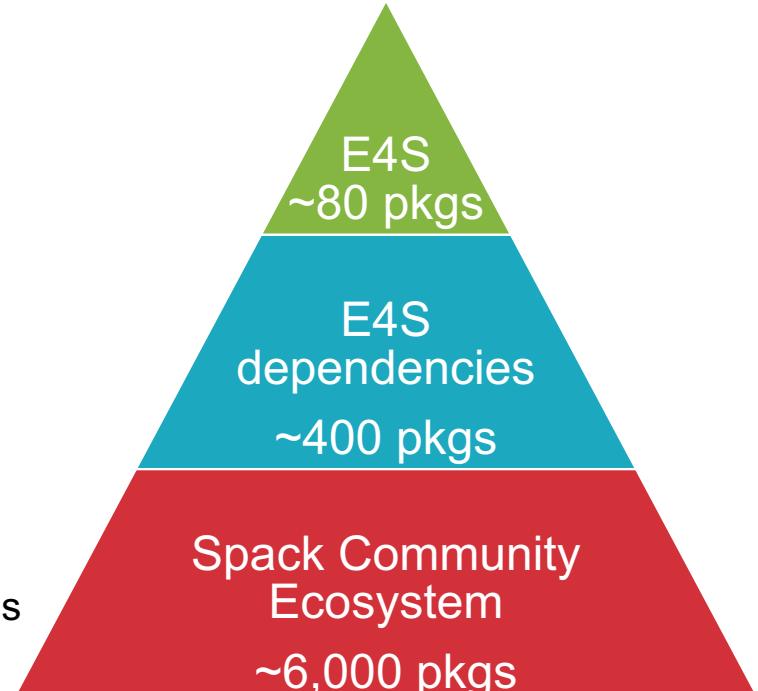


Figure from Todd Gamblin

Leadership Scientific Software Community Discussions

Leadership Scientific Software (LSSw) Portal <https://lssw.io>

The LSSw portal is dedicated to building community and understanding around the development and sustainable delivery of leadership scientific software

- LSSw Town Hall Meetings (ongoing)
 - 3rd Thursday each month, 3 – 4:30 pm Eastern US time
 - 100+ attendees at each meeting, sessions recorded
- Town Hall Topics
 - Meeting 1: Overview of the ECP Software Technology Focus Area
 - Meeting 2: Progress, impediments, priorities & gaps in LSSw panel
 - Meeting 3: Expanding the LSSw User Communities - Panel
 - Meeting 4: Expanding the LSSw Developer Communities – Panel
 - Meeting 5: Retrospective on Previous Meetings – Discussion
 - Meeting 6: Other HPC Software Ecosystems – Panel
 - Meeting 7: Expanding the Scope of What is Reusable – Panel

LSSw Town Halls – Forum for exploring future, building community

Topic: Progress, impediments, priorities and gaps in leadership scientific software

- Ann Almgren, Berkeley Lab, PI of the AMReX project
- Todd Gamblin, Lawrence Livermore National Lab, PI of the Spack project
- Paul Kent, Oak Ridge National Lab, PI of the QMCPACK project
- J. David Moulton, Los Alamos National Lab, PI of the IDEAS Watersheds project
- Todd Munson, Argonne National Lab, PI of the PETSc/TAO project

Themes:

- Improved SW quality and availability accelerates scientific discovery
- Maintaining SW workforce is essential through visible, sustained career paths
- Engaging, growing, & sustaining a user base is essential for viable products
- Regular testing & integration are essential for providing trusted SW components
- Complexity is growing in many dimensions, coordinated SW efforts can mitigate it

Topic: US Agency Use of DOE HPC Software

- Shawn Brown, Pittsburgh Supercomputing Center
- Jeff Durachta, NOAA
- Alice Koniges, University of Hawai'i Data Science Center
- Piyush Mehrotra, NASA
- Andrew Wissink, US Army

Themes:

- Open-source community-based software products are attractive resources
- Heterogeneous platforms (GPUs) represent a significant challenge for apps
- Lack of stable programming environments, transition costs are blockers for GPUs
- Spack is used or is on the radar for all panelist communities
- DOE math libs, perf tools, portability layers & E4S used or on the radar of most

Topic: Expanding Leadership Scientific Software Developer and User Communities

- Deb Agarwal, Berkeley Lab
- Anshu Dubey, Argonne National Laboratory
- Bill Hart, Sandia National Labs
- Addi Malviya-Thakur, Oak Ridge National Laboratory
- Katherine Riley, Argonne National Laboratory

Themes:

- All panelists support the expanded definition of leadership to include their domain
- New leadership definition enables holistic strategy for quality scientific SW
- The represented communities have much in common with HPC communities
- In future, HPC and these communities have emerging collaboration opportunities
- SW practices & tools from these communities can help HPC teams improve

Topic: Scientific Software Ecosystems

- Anita Carleton, CMU, SEI
- Theresa Windus, Iowa State, MoSSI
- Lorraine Hwang, UC Davis, CIG
- Elizabeth Sexton-Kennedy, Fermi Lab, HSF
- Andy Terrel, Xometry, NumFocus

Themes:

- Ecosystem membership criteria tend to be informal for most ecosystems.
- A product is welcome if it has a user community, funding & fits in the ecosystem
- Most ecosystems have a lean budget and live on soft funding
- A major contribution of ecosystems is training: developers, user, leaders
- With some exceptions, software quality criteria are not explicitly stated

LSSw Meeting 7: Thursday, April 21, 2022, 3 - 4:30 pm ET

Topic: Expanding the Scope of What is Reusable: A panel discussion

- **Description:** General-purpose reusable libraries and tools for scientific applications have been very successful. Math, I/O, viz and portable programming libraries and tools have been particularly valuable. Other, more application-specific, libraries and tools have also had some success, for example, the Co-Design Centers sponsored by the Exascale Computing Project, but have received less attention and can be more challenging to sustain. This month we have panelists to help explore expanding the kinds of functionality that can be encapsulated for reuse:
 - Angela Herring, LANL
 - Slaven Peles, ORNL
 - Andrew Salinger, SNL
 - Andrew Siegel, ANL
 - One more, TBD
- In opening remarks, panelists briefly address the following questions from their perspective:
 - Do you think there is value in designing, implementing, and delivering application-specific libraries, tools, and environments as reusable components?
 - What has worked and not worked well with past efforts in this area?
 - What are some near-term opportunities to componentize in your application area?
 - How could this kind of software collection be adapted and sustained?
- **Why attend:** To discuss the feasibility, strategies, and opportunities for expanding the scope of functionality that can be encoded in reusable components, libraries, and tools to better include more application-specific functionality.

<https://lssw.io>

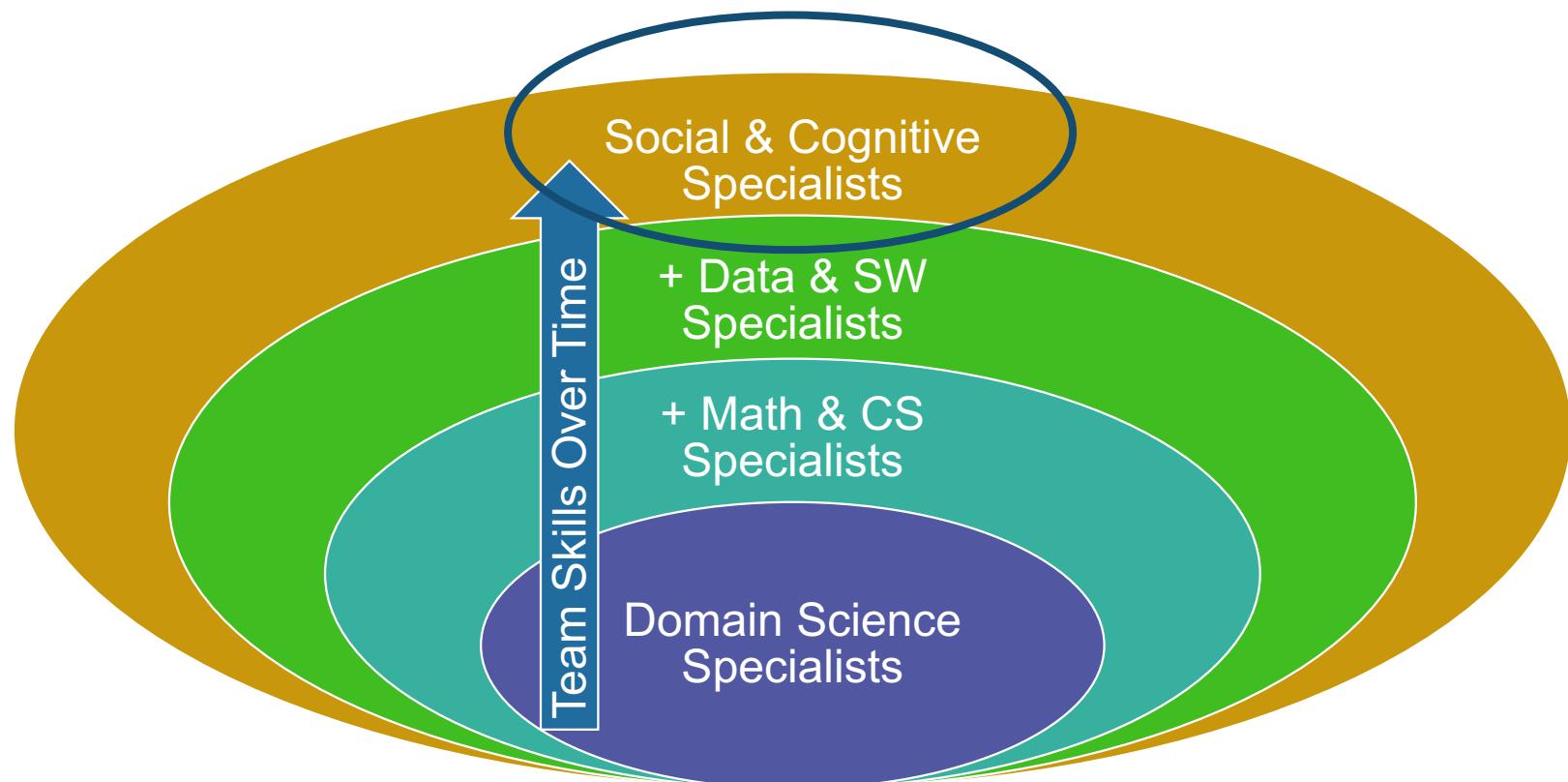
Expanding Software Team Skills: Research Software Science (RSS)

Key observation: We are scientists, problem solvers. **Use science to address our challenges!**

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

Next: **Research Software Science**

- Use scientific method to understand, improve development & use of software for research.
- **Incorporate cognitive & social sciences.**



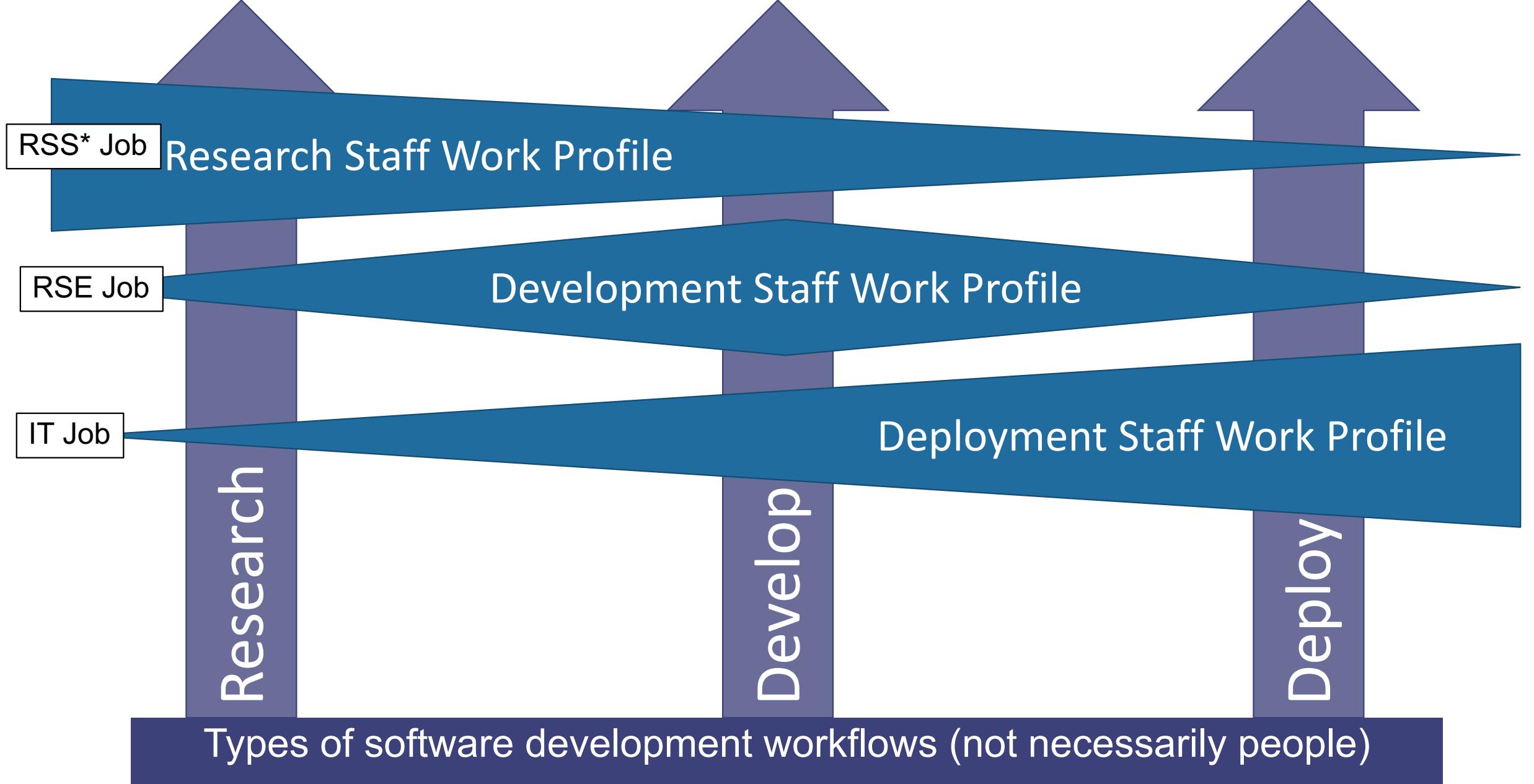
New DOE Area of Exploration: Research Software Science (RSS)

Workshop on Research Software Science

- Software is an increasingly important component in the pursuit of scientific discovery.
- Both its development and use are essential activities for many scientific teams.
- At the same time, very little scientific study has been conducted to understand, characterize, and improve the development and use of software for science.
- Research Software Science (RSS) is a frontier.
- *RSS: Using the scientific method to understand and improve how software is developed and used for research*

The screenshot shows the homepage of the workshop website. The header includes navigation links for Home, Agenda, Position Paper Submission, and Contacts, along with 'Register Now' and 'Already Registered?' buttons. The main content area features the title 'Workshop on the Science of Scientific-Software Development and Use', sponsored by the U.S. Department of Energy, Office of Advanced Scientific Computing Research. It also displays the dates 'December 13 - 15, 2021' and '12 - 5 PM Eastern Time'. A large list of statistics is presented below:

- 126 whitepaper submissions (at same time as RFI!)
- Whitepapers at: <https://www.orau.gov/SSSDU2021>
- 150 registrants (maxed), 100 registered observers
- 150 – 170 active participants across three days
- Goal: Leverage new skills applied to scientific SW



Why A Software Science Focus now: The “No CS” Scenario

Scenario: Suppose our research centers had no formally trained computer scientists and CS work had to be done by people who learned it on their own, or just happened to study a bit of CS as part of their other formal training. This situation is undesirable in three ways:

1. We have non-experts doing CS work, making them less available in their expertise
2. CS work takes a long time to complete compared to other work
3. We get suboptimal results and pay high ongoing maintenance cost

Replace "CS" with "Software" in this scenario and the situation describes scientific software today

Why focus on software science now:

- The role of software has become central to much of our work and the knowledge base is too sophisticated to rely only on non-experts
- Scientific software success depends on producing high-quality, sustainable software products
- Investing in software as a first-class pursuit improves the whole scientific ecosystem

Applying Social & Cognitive Science to Software Teams

- Reed Milewicz – Emerging research software scientist
 - Elaine Raybourn – Sandia social scientist
 - New scientific tools to study and improve developer productivity, software sustainability
 - Correlation: Happiness and connectedness

New Professional Role: *Research Software Scientist*

Talk to Me: A Case Study on Coordinating Expertise in Large-Scale Scientific Software Projects

Reed Milewicz and Elaine M. Raybourn
Sandia National Laboratories, 1611 Innovation Pkwy SE, Albuquerque, New Mexico 87123

scale collaborative scientific software projects typically possesses knowledge and expertise than any one person or communication of knowledge software. In creating and safeguarding software, we cannot attempt to scale up the awareness and have up the production and development directly address these challenges. We attempted by opportunity to develop better study of developers of software are technical.

people of different backgrounds and cultures, there must be close success, there must be close parties [2]. Unfortunately, we are projects, we are

Source: <https://arxiv.org/pdf/1809.06317.pdf>

Source: <https://arxiv.org/pdf/1809.06317.pdf>

Building Community Takeaways

- The design, construction and use of software for science is a multi-faceted endeavor
- Community building is essential:
 - IDEAS – training, incremental improvement planning
 - SDKs – accelerate design space exploration, create coop-etition
 - RSEs – national and international recognition for an important & previously unacknowledged community
 - BSSw – Portal & fellowships
 - LSSw – Platform for exploring next-gen software ecosystems and communities
 - RSS – Leveraging science for scientific software

Developing software for GPU systems



We work on products applications need now and into the future

Key themes:

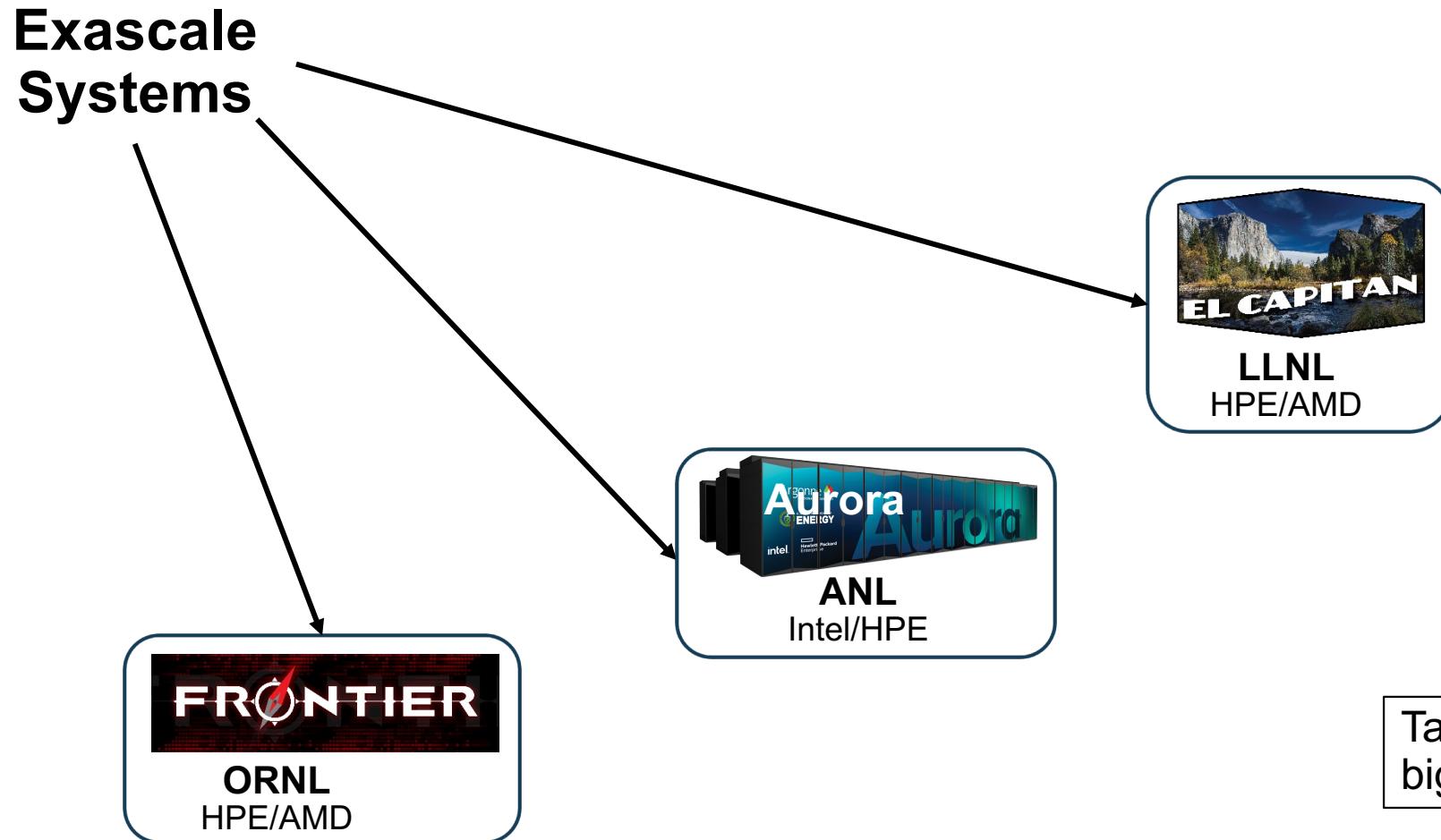
- Focus: GPU node architectures and advanced memory & storage technologies
- Create: New high-concurrency, latency tolerant algorithms
- Develop: New portable (Nvidia, Intel, AMD GPUs) software product
- Enable: Access and use via standard APIs

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

Exascale Systems – Primary targets for ECP Software Teams

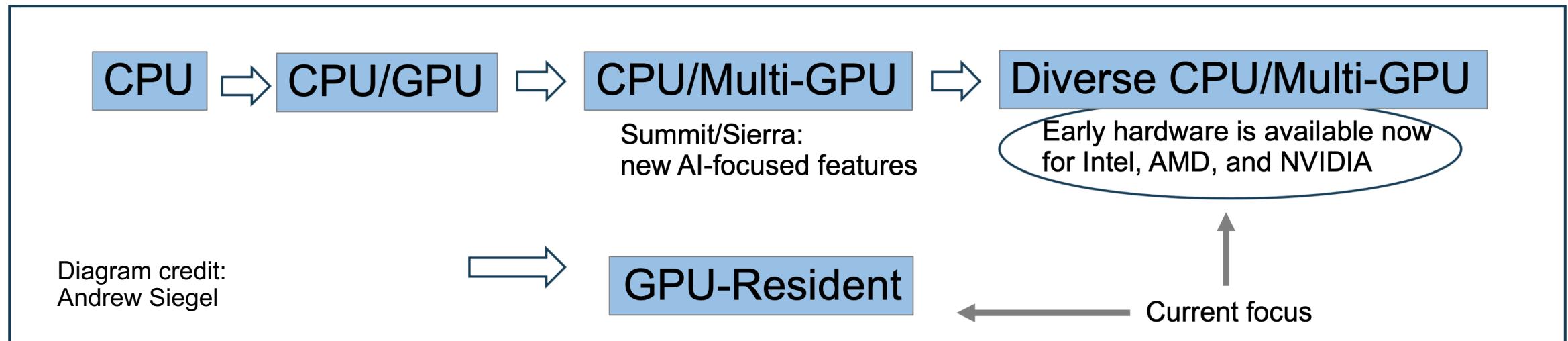


Heterogeneous accelerated-node computing

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

- Execute on the largest systems ... AND on today and tomorrow's laptops, desktops, clusters, ...
- We view *accelerators* as any compute hardware specifically designed to accelerate certain mathematical operations (typically with floating point numbers) that are typical outcomes of popular and commonly used algorithms. We often use the term GPUs synonymously with accelerators.

Text credit: Doug Kothe



Kokkos/RAJA

- Two distinct products: Kokkos and RAJA
 - Both originate in NNSA
 - RAJAs main funding/usage in NNSA
 - Kokkos gets half its funding from NNSA, but >70% of users outside of NNSA
- Worked on some shared low level capabilities
 - Atomics: reported last year on the development, Kokkos now replaces internal atomic operation implementation with that stand alone implementation
 - Tools: Kokkos-Tools interface was orthogonalized from Kokkos Core Programming model, RAJA is looking at using that

What is Kokkos?

- Ecosystem for portable and performant parallel programming (focus: on-node parallelism)
 - **Expanding solution** for common needs of modern science/engineering codes
- **Kokkos Core: C++ Programming Model for Performance Portability**
 - Goal: **Write algorithms once**, run everywhere (almost) optimally
 - Implemented as a template library on top of CUDA, OpenMP, HIP, SYCL, ...
 - Aligns with developments in the C++ standard
- **Kokkos Kernels: Numerical libraries with interfaces to vendor kernels**
 - Goal: Deliver high performance kernels that are portable
 - Optimized implementations for sparse/dense linear algebra and graph kernels
- **Open-Source Software:** <https://github.com/kokkos>
- Production use on all major HPC systems by dozens of teams: *Trinity, Summit, Sierra, Fugaku,*
- Many users at a wide range of institutions:



Max-Planck-Institut
für Plasmaphysik



Rensselaer



Kokkos Core Team



Sandia
National
Laboratories



BERKELEY LAB

- Kokkos Core works as an integrated multi-lab project
 - Developers from SNL, ORNL, ANL, LBL working as integrated team
- Labs leading backend work for machines at their centers
- Sandia contributed < 50% of all code changes in 2020/21



SYNERGY



U.S. DEPARTMENT OF
ENERGY

Office of
Science

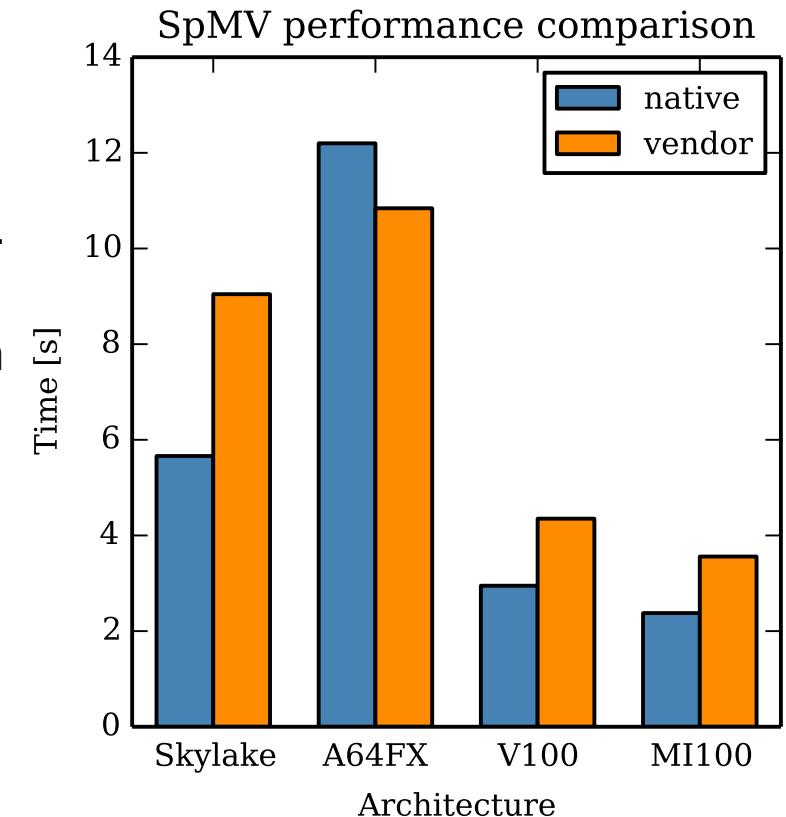
- Used by LANL + SNL ATDM
- Most SNL ASC apps using GPUs do so via Kokkos
- Shared development
- Common knowledge base
- More impact = more vendor influence
- Full support for machines of NNSA and OOS Labs
- Frontier/EI Capitan similar requirements
- **18 ECP Critical dependencies**
- Many open science apps use Kokkos
- >700 Kokkos Slack users
 - Only 30% from NNSA
 - 40% growth in the last 12 months

ECP users of Kokkos

- Extensively used across ECP projects
 - **Apps:** ExaWind, EXAALT, WDMApp, ExaAM, LatticeQCD, E3SM-MMF, SNL & LANL ATDM apps
 - **Codesign:** ExaGraph, CoPA
 - **Libraries & tools:** ALExa (ArborX and DTK), Kokkos Kernels, Trilinos, FleCSI

Kokkos Kernels Pre-Exascale Results

- Performance Results on Spock (Frontier early-access system)
 - Performance of native SpMV implementation in Kokkos Kernels against vendor TPLs (MKL, ArmPL, cuSPARSE and rocSPARSE) on four architectures demonstrate good portability.
 - Kokkos Kernels implementations strive to extract best performance on each architecture but also allow direct calls to vendor TPLs when possible or needed providing users with good baseline performance for most common linear algebra kernels.
 - Note that the results in figure to the right are subject to change depending on the matrix used for comparison, here two matrices representative of finite element/difference discretization were used



Kokkos Kernels is starting to generate results on pre-exascale systems

Kokkos Participation in ISO C++ Standards Committee

MDSPAN

- `mdspan` got LEWG (working group in charge of library design review) approval for C++23
 - Still needs wording review approval
- `mdspan` will replace guts of Kokkos View for managing data layouts
- Strong support from vendors
- C++ will finally have multi dimensional arrays like Fortran!
 - But much better: incorporates all the customization points from Kokkos
 - Layouts, Memory Access Traits, Can also be used for memory space typesafety
- Proposal: <https://wg21.link/P0009> Implementation: <https://github.com/kokkos/mdspan>

std::linalg

- Full BLAS for C++ but with mixed precision and flexible data layouts via `mdspan`
- Missed the deadline for C++23 due to limited committee review time, very likely for C++26
- Vendors are co-authors – collaborating with some on reference implementation

New algorithms highlights – mixed precision

The “coopetition” model:

Step 1: Collaborative design space exploration

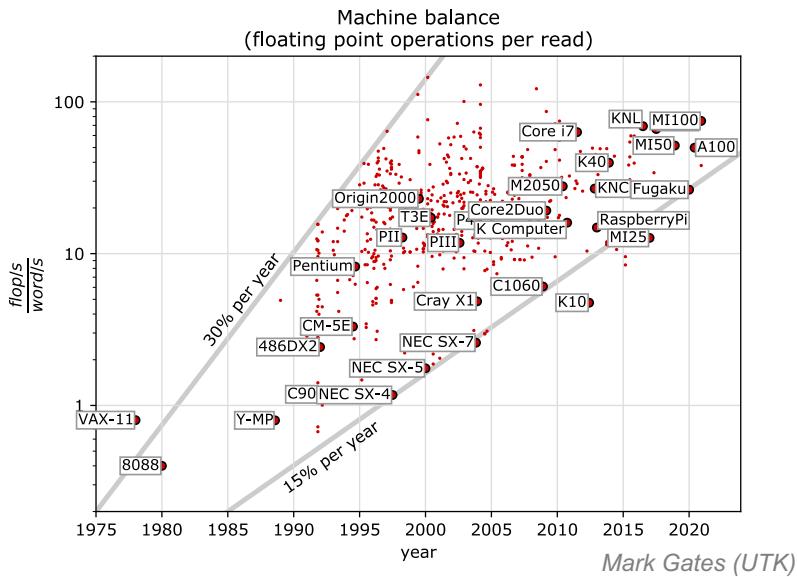
Step 2: Adaptation and implementation in each library



The opportunity: Low-precision arithmetic is fast (and dangerous)

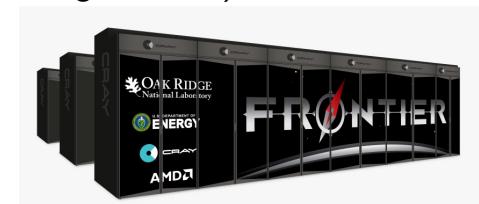
- We currently witness

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



NVIDIA A100 TENSOR CORE GPU SPECIFICATIONS
(SXM4 AND PCIE FORM FACTORS)

	A100 40GB PCIe	A100 80GB PCIe	A100 40GB SXM	A100 80GB SXM
FP64			9.7 TFLOPS	
FP64 Tensor Core			19.5 TFLOPS	
FP32			19.5 TFLOPS	
Tensor Float 32 (TF32)			156 TFLOPS 312 TFLOPS*	
BFLOAT16 Tensor Core			312 TFLOPS 624 TFLOPS*	
FP16 Tensor Core			312 TFLOPS 624 TFLOPS*	
INT8 Tensor Core			624 TOPS 1248 TOPS*	



Computation	MI100 (Peak)	MI250X (Peak)
MI200 Matrix FP64 vs. MI100 Vector FP64	11.5 TFLOPS	95.7 TFLOPS
MI200 Vector FP64 vs. MI100 Vector FP64	11.5 TFLOPS	47.9 TFLOPS
MI200 Matrix FP32 vs. MI100 Matrix FP32	46.1 TFLOPS	95.7 TFLOPS
MI200 Packed FP32 vs. MI100 Vector FP32	23.1 TFLOPS	95.7 TFLOPS
MI200 Vector FP32 vs. MI100 Vector FP32	23.1 TFLOPS	47.9 TFLOPS
MI200 Matrix FP16 vs. MI100 Matrix FP16	184.6 TFLOPS	383 TFLOPS
MI200 Matrix BF16 vs. MI100 Matrix BF16	92.3 TFLOPS	383 TFLOPS
MI200 Matrix INT8 vs. MI100 Matrix INT8	184.6 TOPS	383 TOPS

Nov 2009 Top500:

- Jaguar #1 system
- 1.75 petaflops/s
- FP64 (not FP16)

Step 1: Concurrent exploration of the algorithm and software space

- In cross-laboratory expert teams, we focus on:

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

- Mixed precision algorithms acknowledge and boost the GPU usage

- Algorithm development primarily focuses on GPU hardware (**Summit**, **Frontier**);
- Latest evaluations on **NVIDIA A100** (Perlmutter), **AMD MI100** (Spock), **Intel Gen9 GPU**

- Integrating mixed precision technology as **production-ready implementation into ECP software products** allows for the smooth integration into **ECP applications**.

Advances in Mixed Precision Algorithms: 2021 Edition

by the ECP Multiprecision Effort Team (Lead: Hartwig Anzt)

Ahmad Abdelfattah, Hartwig Anzt, Alan Ayala, Erik G. Boman, Erin Carson, Sébastien Cayrols, Terry Cojean, Jack Dongarra, Rob Falgout, Mark Gates, Thomas Grützmacher, Nicholas J. Higham, Scott E. Kruger, Sherry Li, Neil Lindquist, Yang Liu, Jennifer Loe, Piotr Luszczek, Pratik Nayak, Daniel Osei-Kuffuor, Sri Pranesh, Sivasankaran Rajamanickam, Tobias Ribizel, Barry Smith, Kasia Swirydowicz, Stephen Thomas, Stanimire Tomov, Yaohung M. Tsai, Ichi Yamazaki, Urike Meier Yang

August 28, 2021

TABLE OF CONTENTS

1 Dense Linear Algebra	3
1.1 The mix of precisions in a direct solver	3
1.2 Details of implementation	3
1.3 Experimental results	4
1.4 Enabling Mixed Precision Iterative Refinement Solvers on Spock	5
2 Eigen-Solvers	6
3 Mixed Precision Sparse Factorizations	8
3.1 Mixed Precision sparse LU and QR	8
3.2 Mixed Precision sparse direct solvers	9
4 Mixed Precision Krylov solvers	10
4.1 Mixed Precision GMRES With Iterative Refinement	10
4.1.1 Convergence and Kernel Speedup for GMRES vs GMRES-IR	10
4.1.2 Convergence and Kernel Speedup for Preconditioned GMRES vs GMRES-IR	11
4.2 Compressed Basis Krylov Solvers	12
4.3 s-step Lanczos and CG	16
4.4 Arnoldi-QR MGS-GMRES	18
4.5 Alternative Approaches	19
5 Mixed Precision Sparse Approximate Inverse Preconditioning	19
6 Mixed Precision Strategies for Multigrid	20
6.1 Mixed-precision algebraic multigrid	20
6.2 Enabling Mixed-Precision capabilities in hypre	23
6.3 Current status and future plans:	24
7 Mixed Precision FFT	25
7.1 Data compression to reduce communication	25
7.2 Approximate FFTs with speed-to-accuracy trade-offs	26
7.3 Towards mixed-precision MPI	27
8 Memory Accessor	28
9 Software featuring mixed- and multiprecision functionality	29
9.1 Ginkgo	29
9.2 Kokkos Core, Kokkos Kernels, and Trilinos Additions	31
9.3 MAGMA	31
9.4 heFFTe	31
9.5 PLASMA	31
9.6 PETSc	31
9.7 hypre	32

Step 2: Incorporate lessons learned into library ecosystem

For library interoperability and mixed precision usage:

- **PETSc** develops an abstraction layer to device solvers (vendor libraries, Kokkos Kernels, etc.) that allows flexible composition of Krylov solves in mixed-precision;
- **hypre** already supports the compilation in different precisions and work now focuses on compiling multiple precisions at a time to compose algorithms out of routines running in different precision formats;
- **Ginkgo** makes the “memory accessor” integration-ready for other software libraries;
- **Kokkos** and **KokkosKernels** implements support for compiling in IEEE754 half precision;
- **SLATE** contains mixed precision algorithms and templates the working precision; and
- **MAGMA** compiles in different precisions (z,c,d,s).



Status of early-access system experience

Excellent progress toward Exascale readiness and a lot more to do



EAS Experience Highlights

- 30 of 35 Review presentations specifically addressed Spock porting experiences. The other five:
 - PAPI – Fully engaged in AMD and Intel device preparation, just didn't mention Spock directly
 - SICM – Focused on low-level issues on local hardware
 - Packaging – Working on Summit, Perlmutter, containers for EAS are not as urgent
 - Flang – OpenMP target offload not available
 - Legion –Working on Nvidia (Summit) because of collaborations
- Kokkos and RAJA portability layers are the focus heavy vendor optimization and collaboration
 - Kokkos has 18 critical dependencies – largest count for non-vendor product
 - RAJA – 4 ECP critical dependencies and foundation for LLNL portability
- MPICH (the source for Exascale vendor MPI products) is also under heavy development
 - Shared HPE/Cray Slack channel, weekly dev calls with vendor partners
 - Numerous GPU features under development and optimization
- Less work on Intel right now, but critical layers are fully engaged as possible
- Math libraries represent largest collection of performance-sensitive products in ST
 - Lots of progress in the past year...

January 2022 ECP ST Math Libraries Accelerator Support Status

Package	NVIDIA GPU	AMD GPU	Intel GPU
ArborX	support (Kokkos)	support (Kokkos)	in progress (Kokkos-SYCL backend)
DTK	support (Kokkos)	support (Kokkos)	in progress (Kokkos-SYCL backend)
Ginkgo	support (CUDA)	support (HIP)	support (DPC++)
heFFTe	support (CUDA)	support (HIP)	support (DPC++)
hypre	support (CUDA, RAJA, Kokkos)	support (HIP)	in progress (DPC++)
KokkosKernels	support (Kokkos)	support (Kokkos)	in progress (Kokkos-SYCL backend)
libEnsemble	supports applications running on GPUs	N/A	N/A
MAGMA	support (CUDA)	support (HIP)	planned
MFEM	support (CUDA)	support (HIP)	support (DPC++)
PETSc/TAO	support (CUDA Kokkos)	support (HIP Kokkos)	in progress (DPC++ Kokkos-SYCL)
SLATE	support (CUDA)	support (HIP)	in progress (DPC++)
STRUMPACK	support (CUDA)	support (HIP)	in progress (SYCL, oneAPI)
Sundials	support (CUDA, RAJA)	support (HIP, RAJA)	support (SYCL, oneAPI, RAJA)
SuperLU	support (CUDA)	support (HIP)	in progress (DPC++, oneAPI)
Tasmanian	support (CUDA)	support (HIP)	support (DPC++), but not in spack
Trilinos	support (Kokkos)	support (Kokkos)	in progress (Kokkos-SYCL backend)

- All product teams are working across all three GPU architectures
- KPP-3 threshold efforts target Frontier, reason for further progress on AMD GPUs, relative to Intel

Performance portability

- Portability strategy:

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

- Notes:

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

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

GPU Efforts Summary

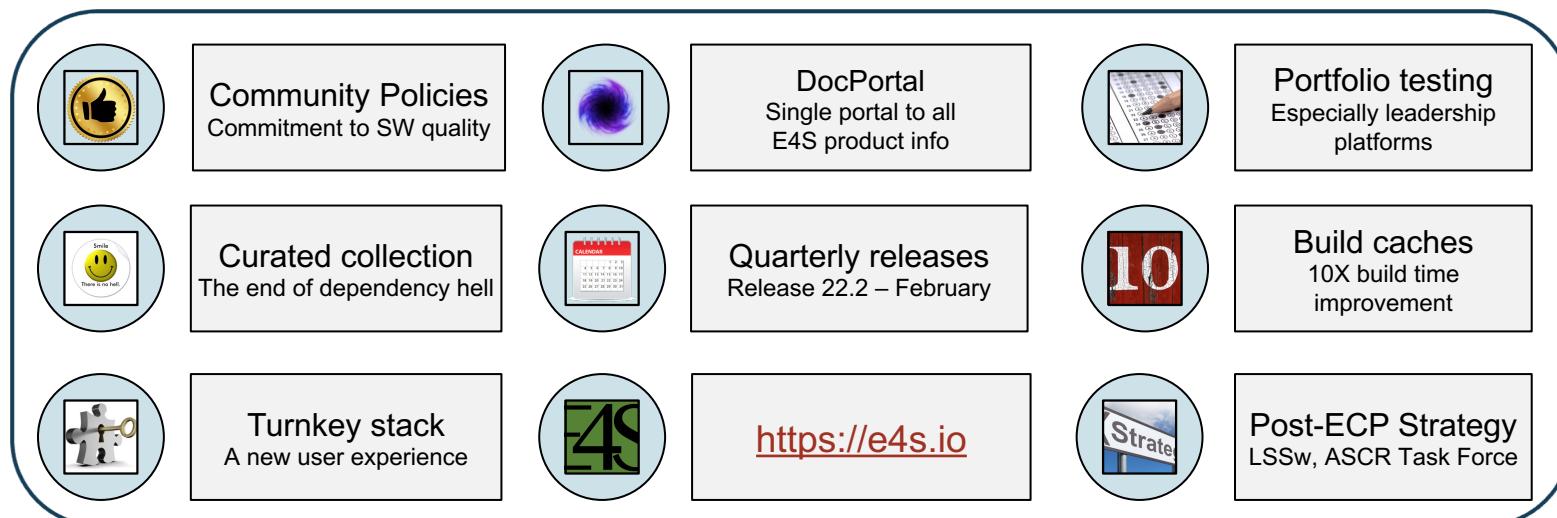
- One legacy of ECP & E4S will be a SW stack that is portable across Nvidia, AMD, and Intel GPUS
- Porting to modern GPUs requires almost everything to be done on the GPUs
- Two+hybrid portability models are used:
 - **Use portability layers:** Kokkos, RAJA or (eventually) OpenMP w target offload (OpenACC?)
 - **Isolate & and custom write:** Isolate perf-portable kernels and write your own CUDA, HIP, SYCL backend
 - **Hybrid:** Use portability layers, customize key kernels only
- Explore low-precision arithmetic: Substantial benefit (and risks)
- Rely more on third-party reusable libraries and tools.

Building Trust in Computations



Extreme-scale Scientific Software Stack (E4S)

- E4S: HPC software ecosystem – a curated software portfolio
- A **Spack-based** distribution of software tested for interoperability and portability to multiple architectures
- Available from **source, containers, cloud, binary caches**
- Leverages and enhances SDK interoperability thrust
- Not a commercial product – an open resource for all
- Growing functionality: Nov 2021: E4S 21.11 – 91 full release products



<https://spack.io>

Spack lead: Todd Gamblin (LLNL)



E4S

<https://e4s.io>



E4S lead: Sameer Shende (U Oregon)

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

E4S DocPortal

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

E4S Products

*: Member Product
Show 10 entries

Name	Area	Description	Latest Doc Update
ADIOS2	Data & Viz	I/O and data management library for storage I/O, in-memory code coupling and online data analysis and visualization workflows.	2021-03-10 16:45:25
AML	PMR	Hierarchical memory management library from Argo.	2019-04-25 13:03:01
AMREX	PMR	A framework designed for building massively parallel block-structured adaptive mesh refinement applications.	2021-05-02 17:26:43
ARBORX	Math libraries	Performance-portable geometric search library	2021-01-05 15:39:55
ARCHER	Software Ecosystem	Container-based solution for portable build and execution across HPC systems and cloud resources	2018-08-22 22:26:19
ASCENT	Data & Viz	High performance scientific visualization and analysis runtime for GPU accelerated simulation	2020-04-06 11:34:11
BEE	Development Tools	OpenMP over lightweight threads.	2020-05-04 11:24:57
BOLT	Development tools	Performance analysis library.	2020-11-04 23:53:07
CALIPER	PMR	A library that handles automatic data migration to different memory spaces behind an array-style interface.	2020-11-02 19:58:24
CHAI			

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

Name: <https://e4s-project.github.io/DocPortal.html>

Showing 1 to 10 of 76 entries

Previous 1 2 3 4 5 ... 8 Next

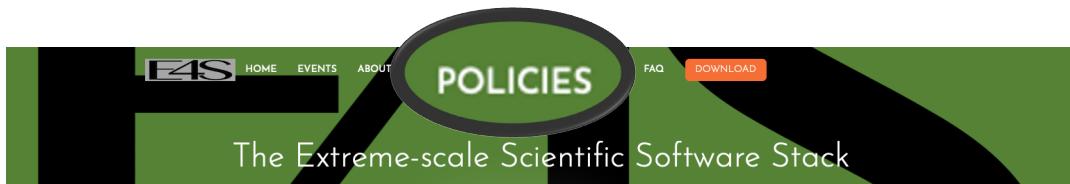
Goal: All E4S product documentation accessible from single portal on E4S.io (working mock webpage below)

The diagram illustrates the goal of creating a unified documentation portal for E4S products. It shows three overlapping web browser windows:

- Top-left window:** Shows the E4S Project GitHub page (<https://e4s-project.github.io>) with a sidebar listing various E4S products (e.g., ADIOS2, AML, ARCHER, ASCENT, BEE, BOLT, CALIPER, CHAI, CINEMA, DARSHAN) and their areas (e.g., Data & Viz, PMR, Tools, Software, Ecosystem, Development Tools).
- Middle window:** Shows a detailed view of the ADIOS2 product page on the E4S Project GitHub site. It includes a description of ADIOS2 as "The Adaptable Input Output System version 2", developed in the Exascale Computing Program. It lists document summaries (ReadMe.md, LICENSE), releases (v2.6.0, spec v2.6.0), and a license section (Apache License, Version 2.0).
- Right window:** Shows the ORNL Computer Science and Mathematics software page for ADIOS2. It features the ADIOS logo, a brief description of the software, and a list of ORNL researchers involved in the project, each with a profile picture and name.

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

E4S Community Policies: A commitment to quality improvement



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



SDK lead: Jim Willenbring (SNL)



Policies: Version 1

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

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

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.

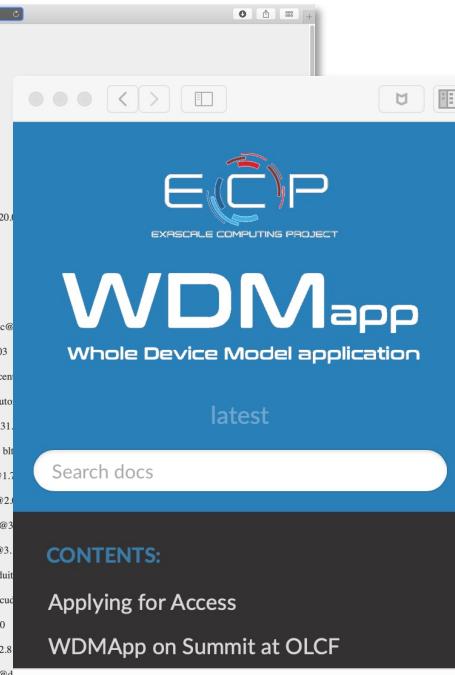
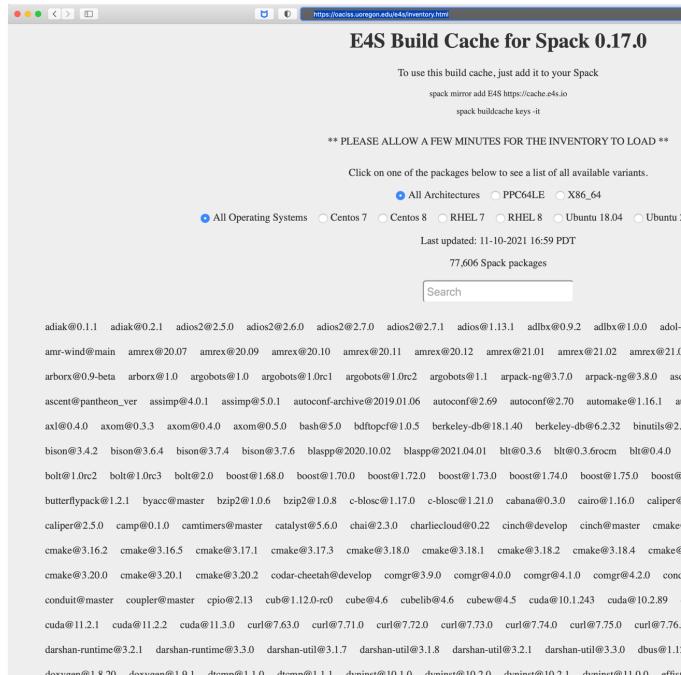
P7 Imported Software If an E4S member package imports software that is externally developed and maintained, then it must allow installing, building, and linking against a functionally equivalent outside copy of that software. Acceptable ways to accomplish this include (1) forswaking the internal copied 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.

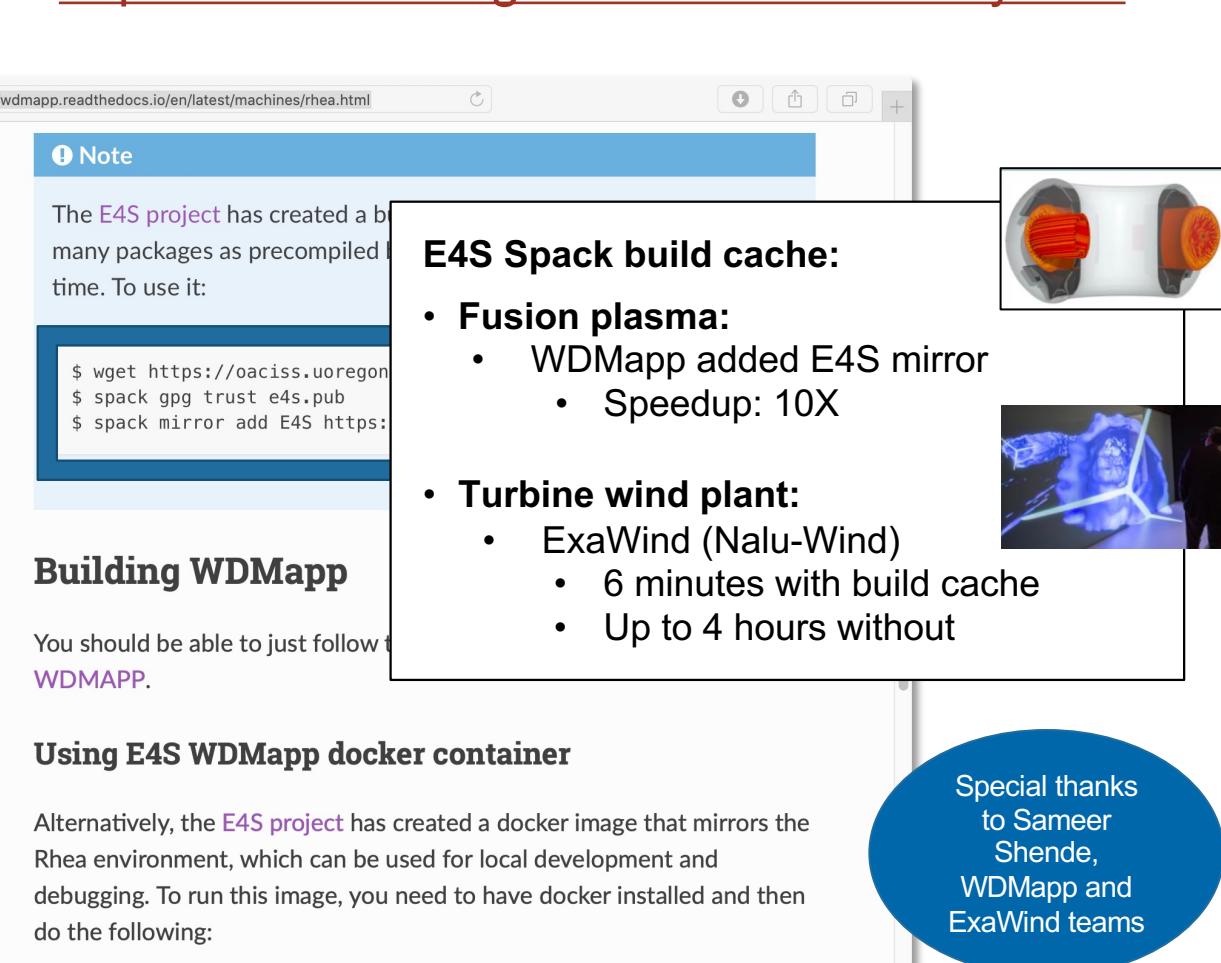
P9 Test Suite Each E4S member package will provide a test suite that does not require special system privileges or the purchase of commercial software. This test suite should grow in its comprehensiveness over time. That is, new and modified features should be included in the suite.

Speeding up bare-metal installs using the E4S build cache

<https://oaciss.uoregon.edu/e4s/inventory.html>



- 75,000+ binaries
- S3 mirror
- No need to build from source code!



E4S Spack build cache:

- **Fusion plasma:**
 - WDMapp added E4S mirror
 - Speedup: 10X
- **Turbine wind plant:**
 - ExaWind (Nalu-Wind)
 - 6 minutes with build cache
 - Up to 4 hours without

Building WDMapp

You should be able to just follow the [WDMAPP](#).

Using E4S WDMapp docker container

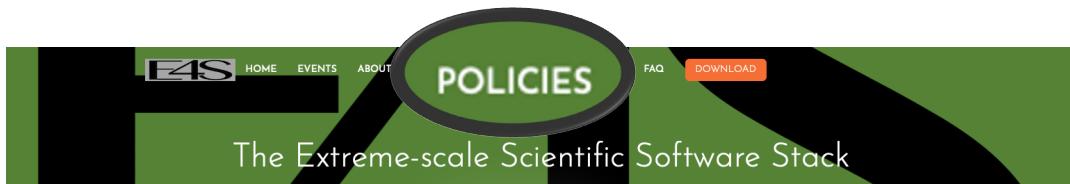
Alternatively, the [E4S project](#) has created a docker image that mirrors the Rhea environment, which can be used for local development and debugging. To run this image, you need to have docker installed and then do the following:

```
$ wget https://oaciss.uoregon.edu/e4s/pub/e4s.pub
$ spack gpg trust e4s.pub
$ spack mirror add E4S https://oaciss.uoregon.edu/e4s/mirror
```

Special thanks
to Sameer
Shende,
WDMapp and
ExaWind teams

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

E4S Community Policies: A commitment to quality improvement



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



SDK lead: Jim Willenbring (SNL)



Policies: Version 1

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

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

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) forswaking 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.

Request for E4S Policy Status Drove Improvement

- L4 Project reviews required gap assessment against E4S Policies
- But no requirement to increase compatibility
- However, teams responded by reducing gaps
- On the right:
 - Flurry of E4S “smoke test” PRs prior to reviews
 - Other low hanging fruit changes made too

Search or jump to... / Pull requests Issues Marketplace Explore

E4S-Project / testsuite Public

Code Issues Pull requests 1 Actions Projects Wiki Security Insights Settings

Filters is:pr is:closed

Clear current search query, filters, and sorts

1 Open ✓ 19 Closed Author Label

- Add umap validation test #25 by egreen77 was merged on Dec 2, 2021 · Changes requested
- adding cinema test #24 by dholgers was merged on Dec 1, 2021
- adds conduit validation tests #22 by cyrush was merged on Nov 30, 2021
- adds ascent cinema test #21 by cyrush was merged on Nov 30, 2021
- Add validation test for visit. #20 by brugger1 was merged on Nov 29, 2021
- improving butterflypack validation_tests by using cmake build #19 by liuyangzhan was merged on Nov 24, 2021 · Approved
- Update tasmanian heffte #18 by miktostyanov was merged on Nov 19, 2021 · Changes requested
- [WIP] Pull UPC++ test sources + Makefile from Spack rather than embedding #17 by bonachea was merged on Nov 15, 2021
- Add HPX tests #16 by msimberg was merged on Nov 18, 2021 · Changes requested
- remove omega-h tests #15 by cwsmit was closed on Nov 5, 2021
- In strumpack test, check return codes, solution #14 by pgphysels was merged on Nov 5, 2021
- Add validation test for Dyninst #13 by hainest was merged 6 days ago · Approved
- Add validation test for HPCToolkit #10 by scottkwarren was merged on Sep 22, 2021 · Changes requested
- Add Cabana validation test #9 by streeve was merged on Sep 22, 2021

Takeaways from community policies

- Community policies need to emerge from the community itself
- Policies should be versioned and refined over time
- Signaling quality expectation, accurately, even with low precision can drive quality improvement

ECP ST Products: Delivering for Deployment

*Developing and delivering new library and tool features
in a collaborative multi-phase lifecycle*



Software feature development lifecycle model for an ST product

		Phase		
		Incubate	Harden	Deliver
		Developers	Integrators	Users
Developers	Explore new algorithms & implementations	Refactor and merge exploratory code, complete tests, documentation	Promote to release branch, integrate into SDK and E4S	Finalize smoke tests, DocPortal, community policy, engage SD, facilities
Integrators	Collaborative work in a sandbox environment	Product team integrates feature	Product team promotes to release, collaborates with SDK/E4S team	Product team works with E4S team, SD and facilities staff
Users	Early collaborators, co-develop with app partners	Friendly app teams prepared to work with the ST team on debugging	App teams looking for latest stable features and staffed to incorporate new features	Apps looking for stable functionality in turnkey environment
Availability	Forked repo or local branch	Develop (pre-release) repo branch	Directly from main repo branch	Product release site, E4S, vendor, facility, community repo, or combo

Some Takeaways:

- AD users of ST products tend to engage in the incubate and harden phases of ST feature development
- First priority for ST teams must be engaging AD users
- First EAS/Exascale ports happen in incubation/harden phases
- For many ST features, sustainable deployment is not E4S (even though E4S inclusion is important)

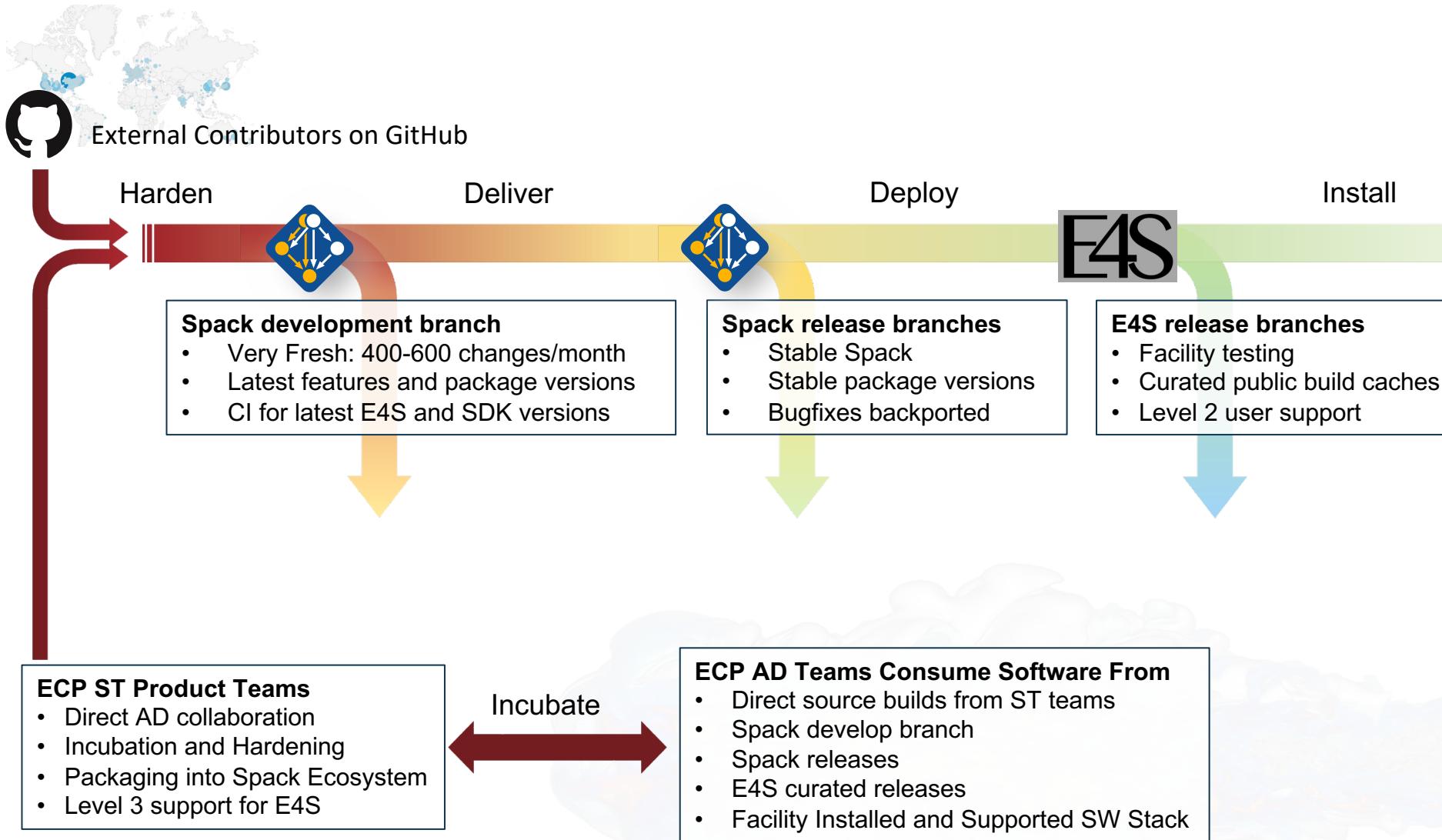
Software feature development lifecycle model: Examples

		Phase			
	Incubate	Harden	Deliver	Deploy	
Nalu-Wind & hypre	Nalu-Wind team forks hypre repo, prototypes GPU-friendly 2-stage Gauss-Seidel smoother consulting with hypre team, also fixes GPU-only bugs in other parts of hypre	hypre team ingests prototype GS smoother, provides coverage tests	Available to all hypre users	Coming soon	
SUNDIALS & Ginkgo	SUNDIALS team collaborates with Ginkgo team on interface for batched sparse solvers needed for Pele-C, others	Ginkgo refactors existing APIs, tests	Available to all Ginkgo users	Coming soon	
ALPINE & VTK-m	ALPINE team prototypes a distributed contour tree algorithm within their fork of VTK-m	VTK-m team merges into their develop branch for hardening	VTK-m promotes to release branch	In E4S V21.11	

Key points:

- Many of the most impactful ST capabilities are developed using this collaborative development pattern
- Incubation and hardening occur on the early-access systems since these systems are driving the need
- Ingesting for sustainable production use ends with promotion to the official release branch for delivery
- E4S integration occurs quarterly, several months after new features are available from product release branch

ECP Software Stream: Incubation to Installation



Facilities



Software Integration at Facilities

- Integration of Vendor Stack
- Local builds and testing
- System tuned configurations
- Local filesystem installation
- Local module scheme





Frank – Designed for Libs & Tools Developers

- Prep system for ECP libs & tools
- Access to latest non-NDA HW/SW
- Shared file system – 1 copy of SW
- Port to many device types at once
- Porting support from E4S team
- CI testing workhorse (500K builds)
- Next: Bare metal, BIOS-changing support for low-level software work

This is a list of all OACISS servers			
Name	Description	Processor	Local Network
			for systems of that type.
Primary login gateway			
Quad Cooper lake + Intel DG1			within the OACISS racks in the machine room. All OACISS systems automatically search .nic.oregon.edu for DNS, so only the
Quad Cooper lake + A100 (80GB)			systems (orthus, cerberus) are accessible by machines outside of nic.oregon.edu.
Cascade lake 6248 node			
AMD + 2 A100 (40GB)			
4 x 26c Xeon Gold 6438 @ 2.3GHz			100GbE + EDR
4 x 24c Xeon Platinum 8367HC @ 3.2GHz			R86.U10
4 x 24c Xeon Gold 6248R @ 2.9GHz			R86.U10
2 x 24c Epyc Rome 7402 @ 2.8GHz			100GbE + 2xEDR
2 x 24c Epyc Milan 7413 @ 2.6GHz			R85.U22
2 x 24c Epyc Milan 7413 @ 2.6GHz			R85.U26
2 x 14c Xeon E5-2660 v4 2.0GHz			100GbE
2 x 16c Xeon Gold 6226R @ 2.9GHz			R86.U6
2 x 20c Power9 @ 3.66GHz			10GbE + EDR
2 x 20c Power9 @ 3.66GHz			R86.U26
2 x 20c Power9 @ 3.66GHz			R86.U18
2 x 18c Xeon E5-2697 v4			100GbE
8c Xeon 4108 Silver @ 1.8GHz			R86.U35
2 x 14c Xeon E5-2680v4 @ 2.3GHz			40GbE + EDR
2 x 20c Power8 @ 3.5GHz			R85.U6
2 x 20c Power8 @ 3.5GHz			R85.U18
2 x 16c Power8 @ 2.1GHz			10GbE + 2xEDR
2 x 18c Xeon Gold 6110 @ 2.3GHz			R86.U22
2 x 22c Power9 @ 2.2GHz			R84.U44
2 x 22c Power9 @ 2.2GHz			R84.U29
Datacenter			
Compute			
Compute: Orthus	Quad Cooper lake + Intel DG1	2 x 8c Xeon E5-2667 v2 @ 3.3GHz	10GbE
Compute: Jupiter	Quad Cooper lake + A100 (80GB)	4 x 24c Xeon Gold 6438 @ 2.3GHz	100GbE + EDR
Compute: Saturn	Cascade lake 6248 node	4 x 26c Xeon Platinum 8367HC @ 3.2GHz	R86.U10
Compute: Reptar	AMD + 2 A100 (40GB)	4 x 24c Xeon Gold 6248R @ 2.9GHz	R86.U10
Compute: Ilyad	AMD + 2 MI50 + A100 (40GB)	2 x 24c Epyc Rome 7402 @ 2.8GHz	100GbE + 2xEDR
Compute: Gilgamesh	Intel + 2 AMD MI100 + MI50	2 x 24c Epyc Milan 7413 @ 2.6GHz	R85.U22
Compute: Instinct	A100 (80GB) + P100 + V100 GPU node	2 x 14c Xeon E5-2660 v4 2.0GHz	R85.U6
Compute: Voltar	IBM Power9 + 4 V100	2 x 16c Xeon Gold 6226R @ 2.9GHz	R86.U26
Compute: Cyclops	IBM Power9 + 4 V100	2 x 20c Power9 @ 3.66GHz	R86.18
Compute: Gorgon	IBM Power9	2 x 20c Power9 @ 3.66GHz	R86.U16
Compute: Medusa	IBM Power9	2 x 20c Power9 @ 3.66GHz	R86.U14
Compute: Typhon	IBM Power9	2 x 20c Power9 @ 3.66GHz	R86.U12
Compute: Delphi	IBM Power9	2 x 18c Xeon E5-2697 v4	100GbE
Compute: Aurora	IBM Power9	8c Xeon 4108 Silver @ 1.8GHz	R86.U35
Compute: Godzilla	IBM Power9	2 x 14c Xeon E5-2680v4 @ 2.3GHz	40GbE + EDR
Compute: Centaur	Intel + GV100	2 x 20c Power8 @ 3.5GHz	R85.U6
Compute: Minotaur	NEC SX-Aurora demo machine	2 x 20c Power8 @ 3.5GHz	R85.U20
Compute: Eagle	Intel DG1 + 2 x K80 node	2 x 16c Power8 @ 2.1GHz	R86.U24
Compute: Pegasus	IBM Power8 + 2 K80	2 x 18c Xeon Gold 6110 @ 2.3GHz	100GbE + EDR
Compute: Vina	IBM Power8 + 2 K80	2 x 22c Power9 @ 2.2GHz	R86.U22
Compute: Pike	IBM Power9 + 3 x T4	2 x 22c Power9 @ 2.2GHz	R84.U44
Compute: Cirrus-AIX.stor	Compute node	2 x 22c Power9 @ 2.2GHz	R84.U29
Compute: Cumulus-AIX.stor	Raptor Talos II	Drives 8K display in 472	
Compute: Nimbus-AIX.stor	Raptor Talos II + MI25	Secondary login gateway; Jetson/Nucs + NFS	
Compute: KNL_Grover	AIX machine	Intel NUCs (16)	
Compute: Axis cluster (axis1-8)	AIX machine	Tegra TX-1	
Visualization			Tegra TX-2
Visualization: Chymera	AIX machine	NVidia Tegra 3	
Visualization: Cerberus	AIX machine	ARM64 v8	
NUC cluster	Intel Phi system	M1 Mac	
Jetson ARM64 cluster	DL580 G7 nodes	Intel Xe	
Jetson ARM64 cluster		VLSI simulation node	
Compute: Xavier			
Compute: OD1K			
Compute: Omicron			
Compute: Sever			
Compute: Silicon			
Infrastructure			
Infrastructure: orion			
Infrastructure: mecha			
Infrastructure: newstorage			
Infrastructure: mnemosyne			
Infrastructure: lighthouse			

Wrapping It Up



ECP is a complex system

- ECP is specifically about preparing ~25 application code for Exascale computing systems and creating a sustainable underlying software stack, but it is much more
- The complexity of the ECP mission (cost, scope and schedule) require innovation, risk taking
- Requirements and solution emerge over time
- ECP is a big human experiment: Why I joined!

ECP goal: A sustainable, reusable software ecosystem

- ECP is driving the creation of a portfolio approach for reusable scientific software:
 - Available to you from laptops to supercomputers
 - Portable across CPU and GPU architectures
 - Available as open source for you to use, contribute to, and collaborate with
- Creating a future software organization that is a first-class citizen in the leadership computing ecosystem
- And You:
 - Consider using E4S: <https://e4s-project.github.io/download.html>
 - Consider contributing to E4S: <https://e4s-project.github.io/join.html>
 - Consider contributing to one of the SDKs, e.g.: <https://xsdk.info>

ECP goal: A better scientific software future

- Improving how we do our work
- Engaging a broader community
- And You:
 - Receive our BSSw email digest: <https://bssw.io/pages/receive-our-email-digest>
 - Contribute to BSSw.io: <https://bssw.io/pages/what-to-contribute-content-for-better-scientific-software>
 - Apply for 2023 BSSw Fellowship (summer 2022): <https://bssw.io/fellowship>
 - Attend:
 - HPC Best Practices Webinars: <https://ideas-productivity.org/events/hpc-best-practices-webinars>
 - Working Remotely Panels: <https://ideas-productivity.org/events/strategies-for-working-remotely-panels>
 - Our tutorials: <https://bssw-tutorial.github.io>

ECP goal: A sustainable software R&D community

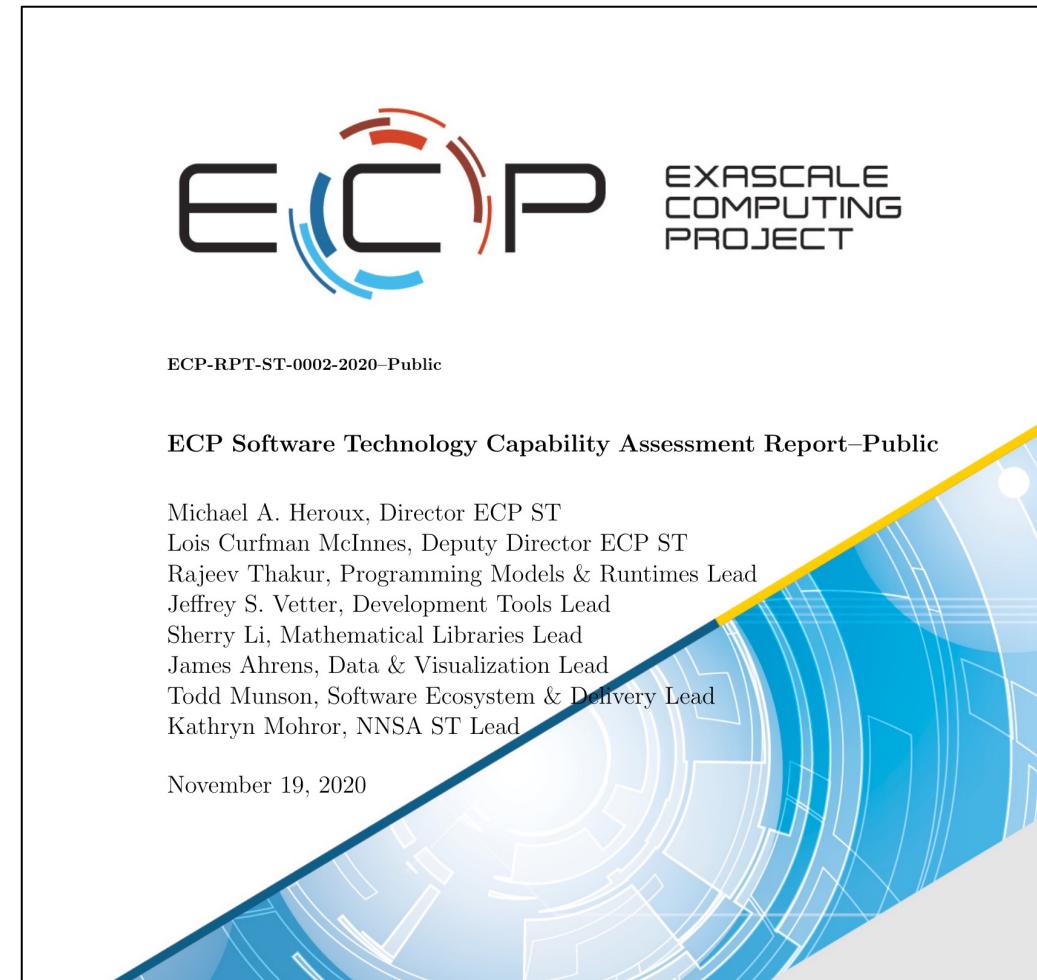
- Growing an R&D community of diverse contributors toward exploring and realizing leadership computing, generation to generation
- Exploring emerging algorithms, software, and computing platforms to deliver capabilities for solving emerging scientific problems
- And You:
 - Join the Leadership Scientific Software (LSSw) conversation: <https://lssw.io>
 - Slack workspace, mail lists, archive of previous town halls
 - Registration for next town hall
 - Track DOE efforts in Research Software Science (RSS)
 - Science of Scientific-Software Development and Use Workshop: <https://www.orau.gov/SSSDU2021>
 - Download position papers: https://www.orau.gov/support_files/2021SSDU/2021-Position-Papers.zip
 - Look for workshop report in late spring 2022

Summary

- Using a portfolio-based approach for HPC software is about going together vs going alone
- While products vary greatly, we all face the same frontiers: Evolving systems, quality, users
- Success on the frontier is important for all HPC configurations: leadership to laptop
- Progress is made by collaborating and sharing information
- The new and evolving E4S and SDKs platforms enable better, faster and cheaper, in net
- A collective approach, esp. E4S, enables new relationships with facilities, vendors, apps, others
- Improving how we develop & use software is an important scientific effort itself, as a community
- Via LSSw, we are planning for a sustainable future that builds on the progress ECP has enabled
- Via RSS, we are expanding scientific software team skills to include social and cognitive sciences

ST Capability Assessment Report (CAR)

- Products discussed here are presented with more detail and further citations.
- We classify ECP ST product deployment as broad, moderate, or experimental.
 - Broad and moderate deployment is typically suitable for collaboration.
 - Web links are available for almost all products.
 - 67 of 71 of ECP ST products are available as part of the Extreme-scale Scientific Software Stack (E4S) <http://e4s.io>.
 - CAR V3.0 appears in June



<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.



ECP Director: Doug Kothe
ECP Deputy Director: Lori Diachin

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.**



**Sandia
National
Laboratories**