Overview of ECP Software Technology

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

E4S at NERSC 2022, August 25, 2022

Outline

- ECP, Briefly
- Establishing software ecosystems
- Developing software for GPU systems

ECP in a Nutshell

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

Performant mission and science applications at scale

Aggressive RD&D project	Mission apps; integrated S/W stack	Deployment to DOE HPC Facilities	Hardware technology advances

Application Development (AD)

Develop and enhance the predictive capability of applications critical to DOE

24 applications
National security, energy, Earth systems, economic security, materials, data

6 Co-Design Centers
Machine learning, graph analytics, mesh refinement, PDE discretization, particles, online data analytics

Software Technology (ST)

Deliver expanded and vertically integrated software stack to achieve full potential of exascale computing

70 unique software products
spanning programming models and run times,
math libraries,
data and visualization

Hardware and Integration (HI)

Integrated delivery of ECP products on targeted systems at leading DOE HPC facilities

6 US HPC vendors
focused on exascale node and system design; application integration and software deployment to Facilities

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.

Xaioye (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.

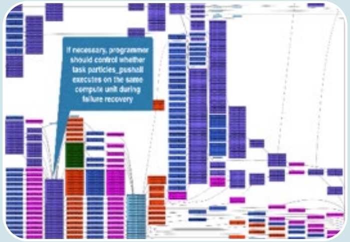
Jim Ahrens, Data and Visualization (2.3.4)
Jim is a senior research scientist at the Los Alamos National Laboratory (LANL) and an expert in data science at scale. He started and actively contributes to many open-source data science packages including ParaView and Cinema.

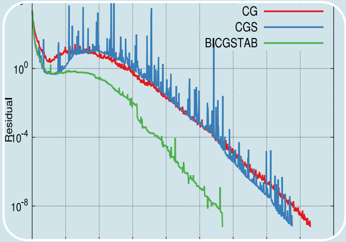
Todd Munson, Software Ecosystem and Delivery (2.3.5)
Todd is a computational scientist in the Math and Computer Science Division of ANL. He has nearly 20 years of experience in high-performance numerical software, including development of PETSc/TAO and project management leadership in the ECP CODAR project.

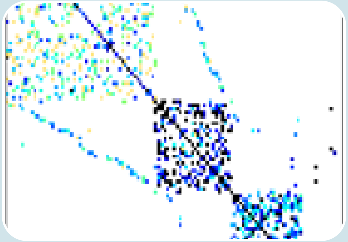
Kathryn Mohror, NNSA ST (2.3.6)
Kathryn is Group Leader for the CASC Data Analysis Group at LLNL. Her work focuses on I/O for extreme scale systems, scalable performance analysis and tuning, fault tolerance, and parallel programming paradigms. She is a 2019 recipient of the DOE Early Career Award.

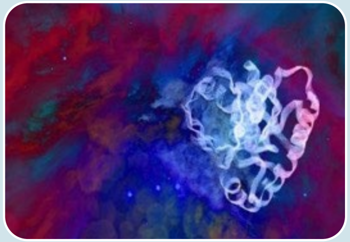
ECP ST has six technical areas

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









E4S

NNSA
National Nuclear Security Administration

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

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

ECP ST Stats

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

EXASCALE COMPUTING PROJECT

ECP Software Technology works on products that apps need now and in the future

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

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

Legacy: A stack that enables performance portable application development on leadership platforms

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

Exascale Systems

FRONTIER

ORNL
HPE/AMD

ANL
Intel/HPE

LLNL
HPE/AMD

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

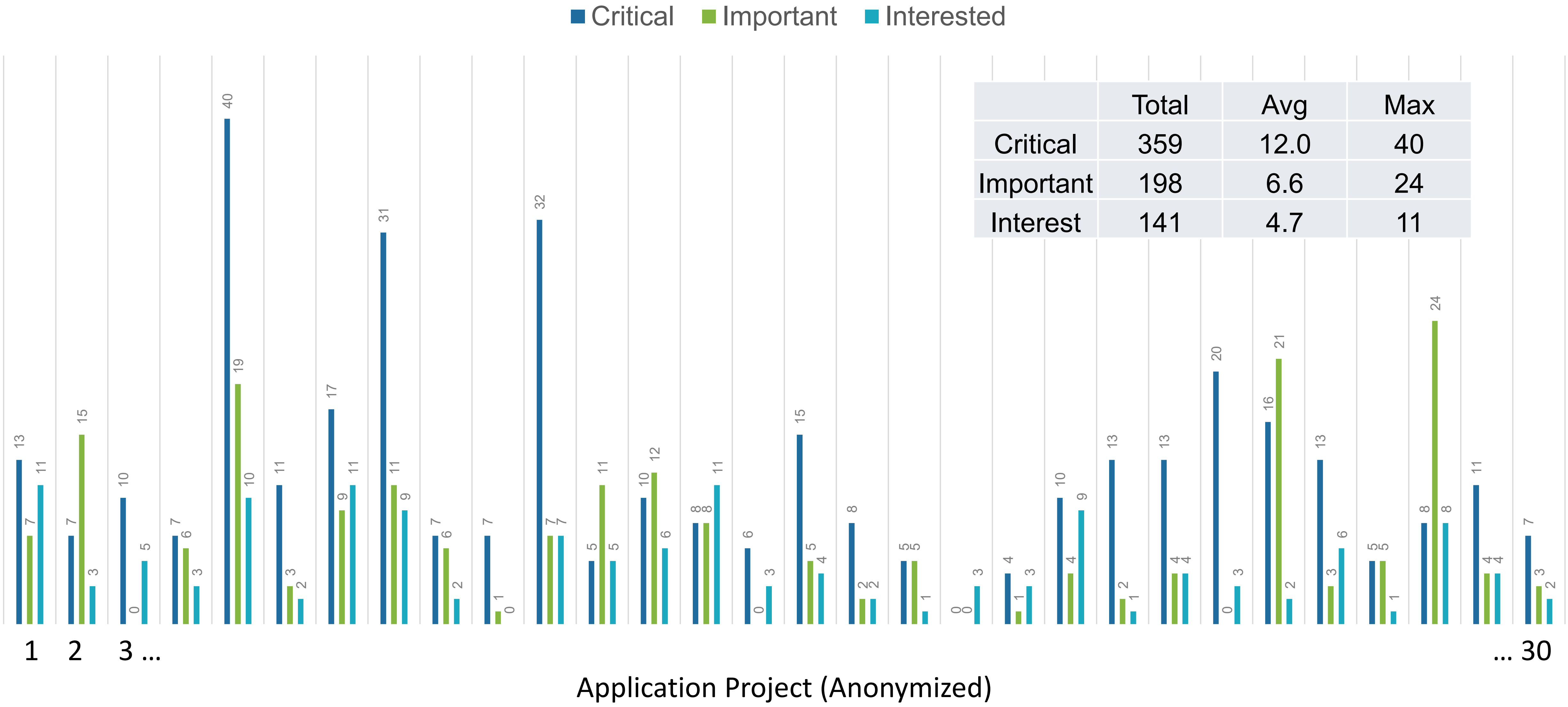
Perlmutter is an important target and vehicle for our ECP work. It is essential for progress and delivery!

The Growing Complexity of Scientific Application Software Stacks

Challenges

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

THE NUMBER OF ECP SOFTWARE TECHNOLOGY PROJECT DEPENDENCIES FOR EACH ECP APPLICATION PROJECT (ANONYMIZED)



	Total	Avg	Max
Critical	359	12.0	40
Important	198	6.6	24
Interest	141	4.7	11

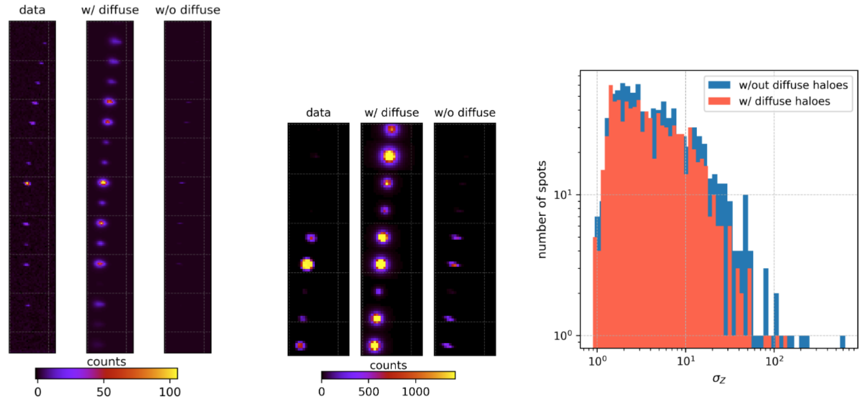
EXASCALE COMPUTING PROJECT

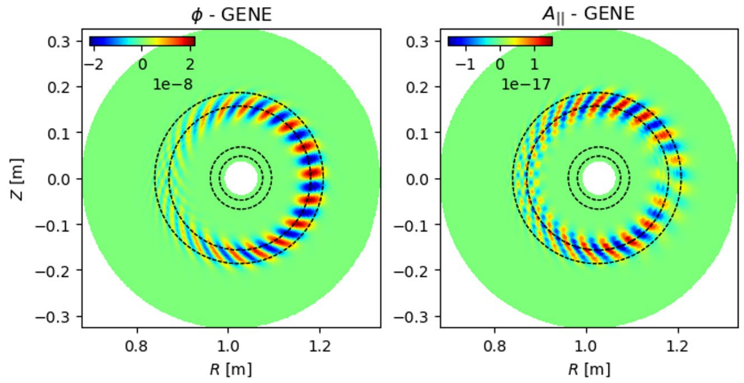
Integration: AD Teams Depend Heavily on ST Software to Meet KPPs

Kokkos

ExaFEL

nanoBragg code ported from Nvidia to AMD GPUs with minimal effort





ADIOS

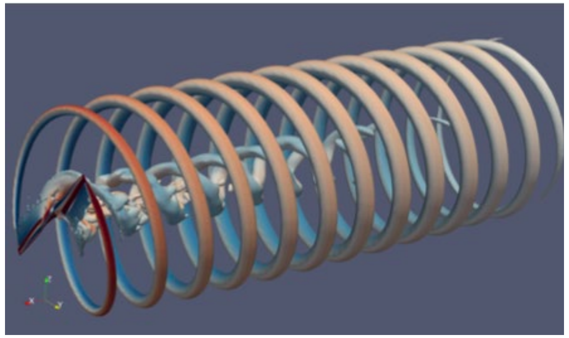
WDMApp

ADIOS enables in-memory coupling between GENE and XGC

hypre

ExaWind

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



Slide courtesy of Andrew Siegel and Erik Draeger

Interesting Themes Arising from 2021 AD Assessment

Sparse solver progress/research challenges

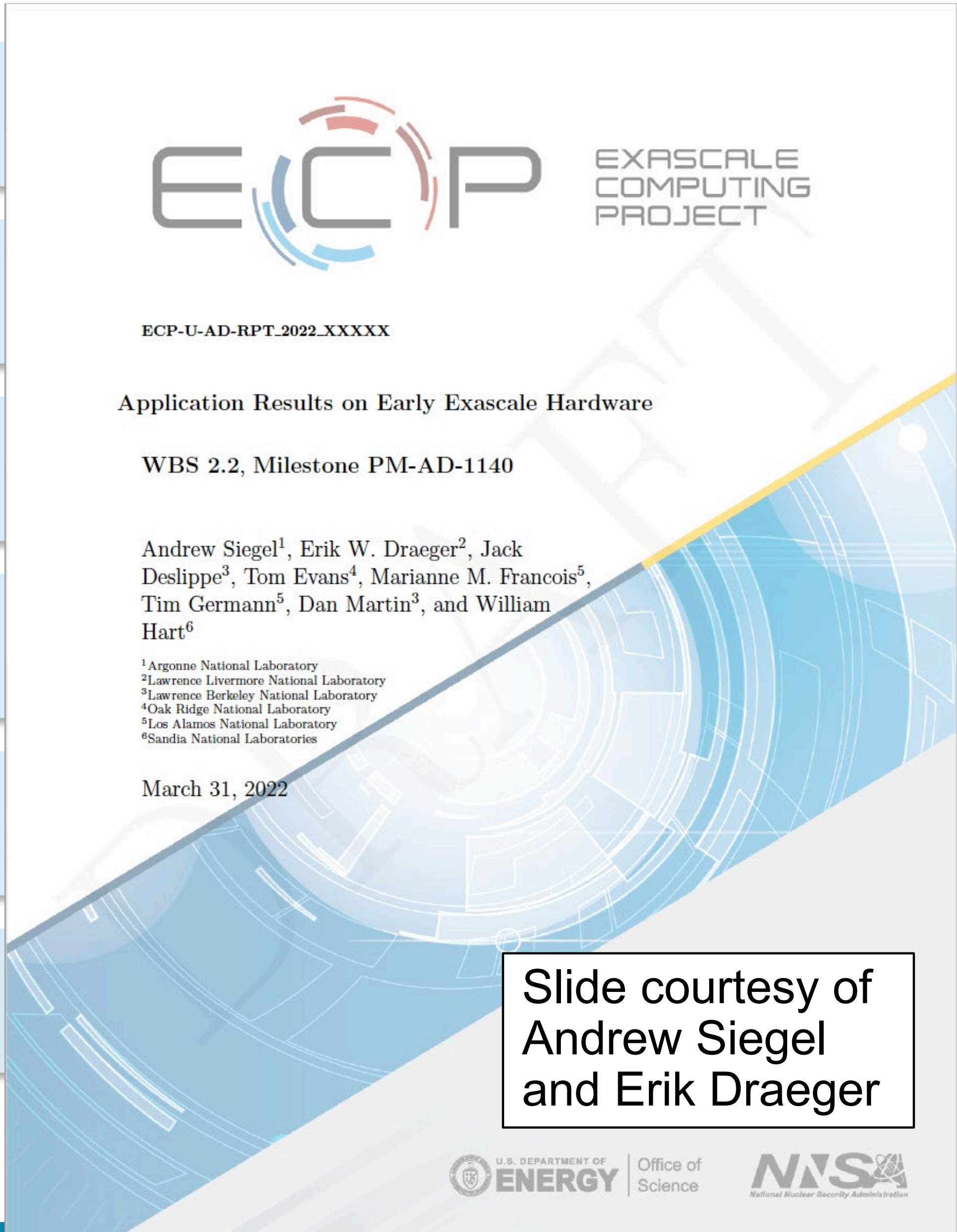
Evolving OpenMP offload performance

Co-maturation of vendor compilers, software stack

✓ ST and CD integration success stories

Maturity of performance analysis tools

Network performance



EXASCALE COMPUTING PROJECT

ECP-U-AD-RPT_2022_XXXXX

Application Results on Early Exascale Hardware

WBS 2.2, Milestone PM-AD-1140

Andrew Siegel[1], Erik W. Draeger[2], Jack Deslippe[3], Tom Evans[4], Marianne M. Francois[5], Tim Germann[5], Dan Martin[3], and William Hart[6]

[1]Argonne National Laboratory
[2]Lawrence Livermore National Laboratory
[3]Lawrence Berkeley National Laboratory
[4]Oak Ridge National Laboratory
[5]Los Alamos National Laboratory
[6]Sandia National Laboratories

March 31, 2022

Slide courtesy of Andrew Siegel and Erik Draeger

A Sampler of Products

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

Takeaways from product sampler

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

Takeaways from software complexity

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

Responding to complexity: Software Ecosystem via Platforms

Software Platforms: “Working in Public” Nadia Eghbal

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

	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)

Eghbal, Nadia. Working in Public: The Making and Maintenance of Open Source Software (p. 60). Stripe Press. Kindle Edition.

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	E4S	**Source:** ECP E4S team; Non-ECP Products (all dependencies) **Delivery:** spack install e4s; containers; CI Testing
• Group similar products • Make interoperable • Assure policy compliant • Include external products	SDKs	**Source:** SDK teams; Non-ECP teams (policy compliant, spackified) **Delivery:** Apps directly; spack install sdk; future: vendor/facility
• Standard workflow • Existed before ECP	ST Products	**Source:** ECP L4 teams; Non-ECP Developers; Standards Groups **Delivery:** Apps directly; spack; vendor stack; facility stack

ECP ST Open Product Integration Architecture

ECP ST Individual Products

Extreme-scale Scientific Software Stack (E4S)

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

Community Policies Commitment to SW quality	DocPortal Single portal to all E4S product info	Portfolio testing Especially leadership platforms
Curated collection The end of dependency hell	Quarterly releases Release 22.2 – February	Build caches 10X build time improvement
Turnkey stack A new user experience	https://e4s.io	Post-ECP Strategy LSSw, ASCR Task Force

Spack

https://spack.io

Spack lead: Todd Gamblin (LLNL)

E4S

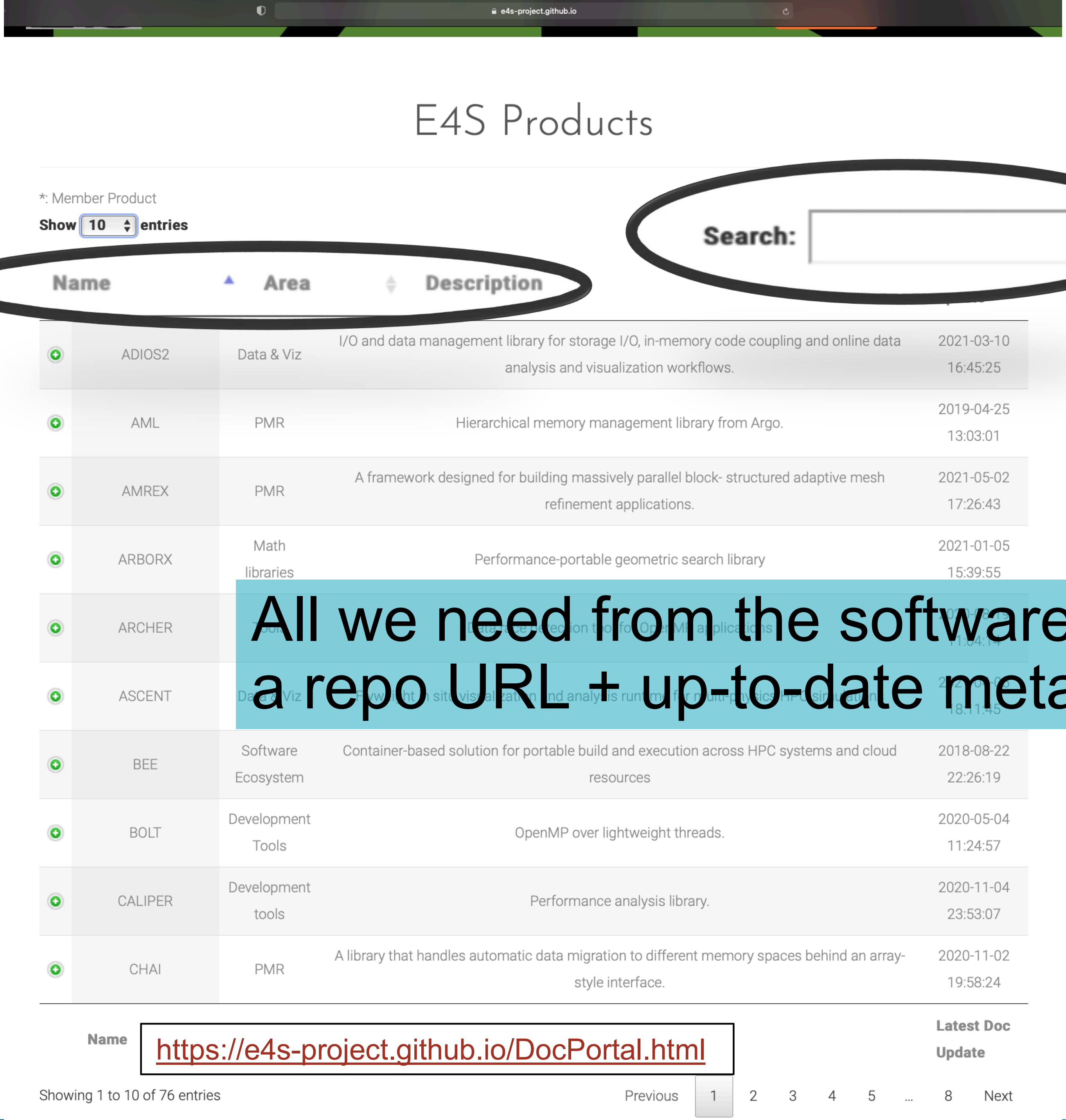
https://e4s.io

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

Search:

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	[illegible]	[illegible] detection tool for OpenMP applications	[illegible]
ASCENT	Data & Viz	[illegible]	[illegible] 18:11:45
BEE	Software Ecosystem	Container-based solution for portable build and execution across HPC systems and cloud resources	2018-08-22 22:26:19
BOLT	Development Tools	OpenMP over lightweight threads.	2020-05-04 11:24:57
CALIPER	Development tools	Performance analysis library.	2020-11-04 23:53:07
CHAI	PMR	A library that handles automatic data migration to different memory spaces behind an array-style interface.	2020-11-02 19:58:24
Name			Latest Doc Update

Showing 1 to 10 of 76 entries

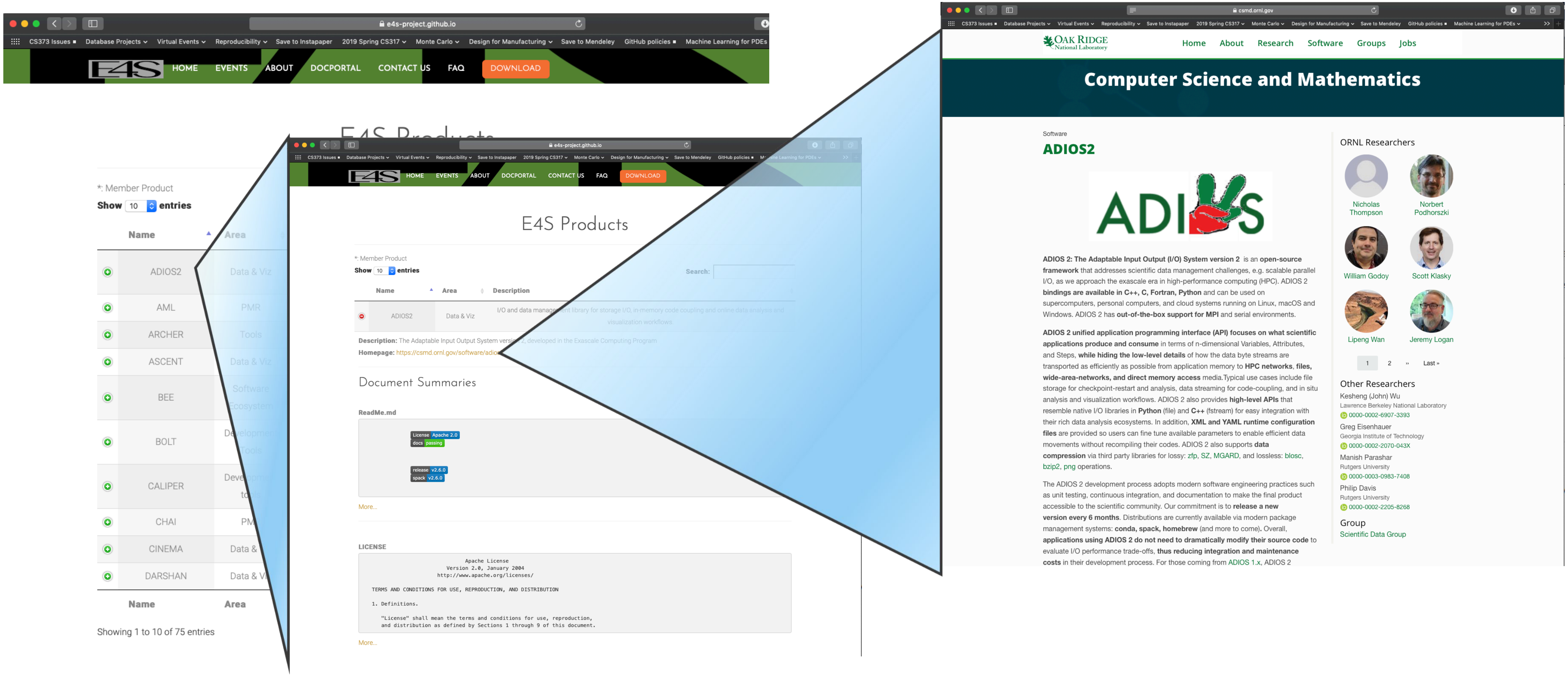
Previous 1 2 3 4 5 … 8 Next

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

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

EXASCALE COMPUTING PROJECT

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



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

E4S Community Policies: *A commitment to quality improvement*

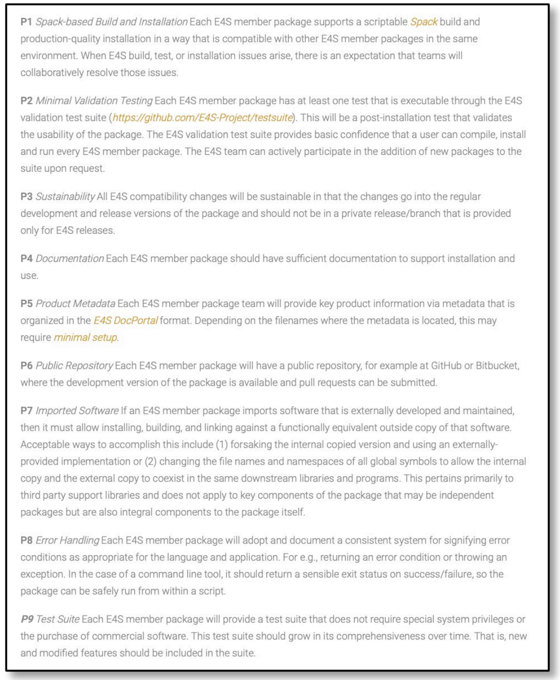
- Enhance sustainability and interoperability
- Serve as membership criteria for E4S
  - Membership is not required for *inclusion* in E4S
  - Also includes forward-looking draft policies
- Modeled after xSDK community policies
- Multi-year effort led by SDK team
  - Included representation from across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

SDK lead: Jim Willenbring (SNL)

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.

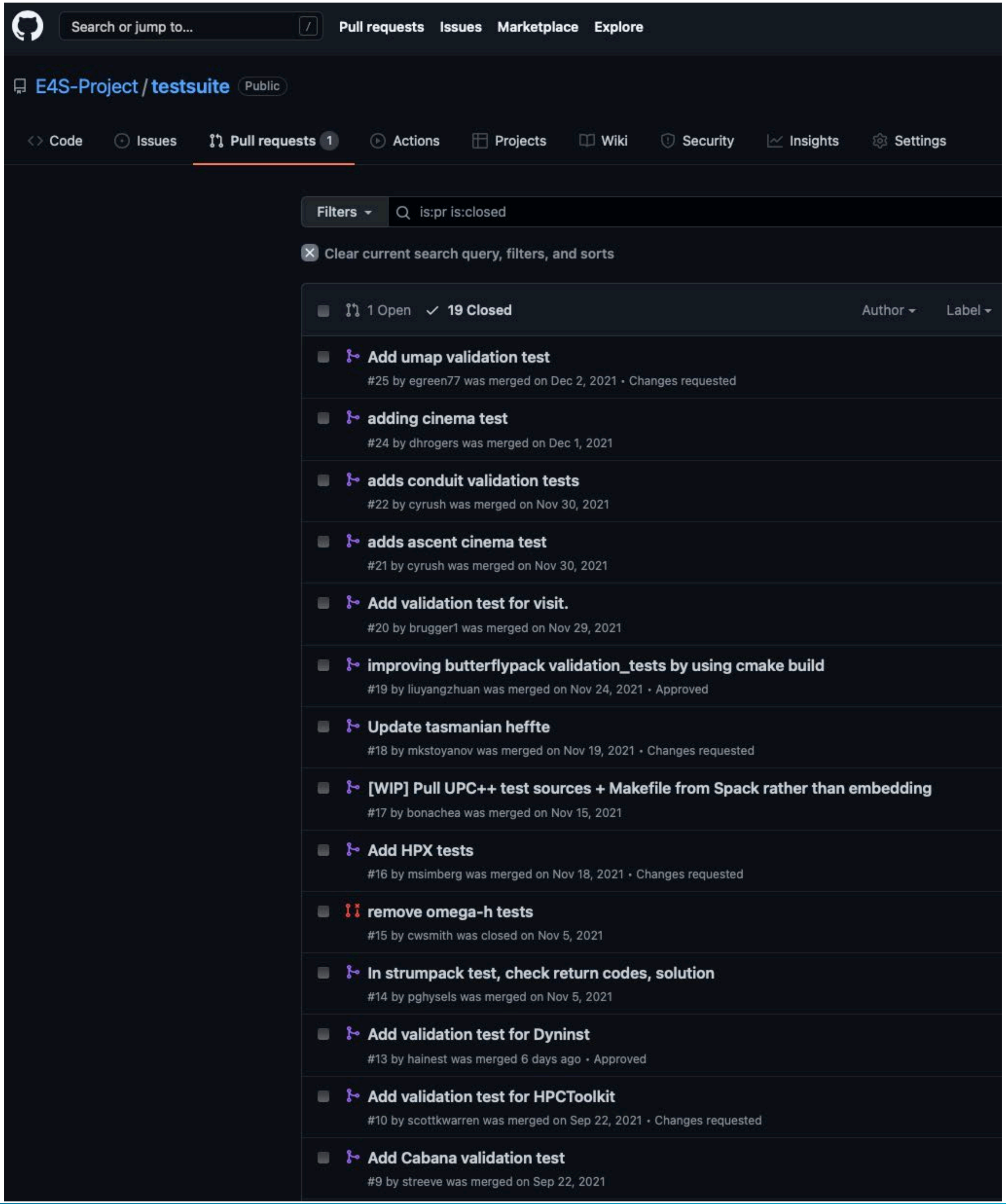
P7 *Imported Software* If an E4S member package imports software that is externally developed and maintained, then it must allow installing, building, and linking against a functionally equivalent outside copy of that software. Acceptable ways to accomplish this include (1) forsaking the internal copied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

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

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

Request for E4S Policy Status Drove Software Improvements

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



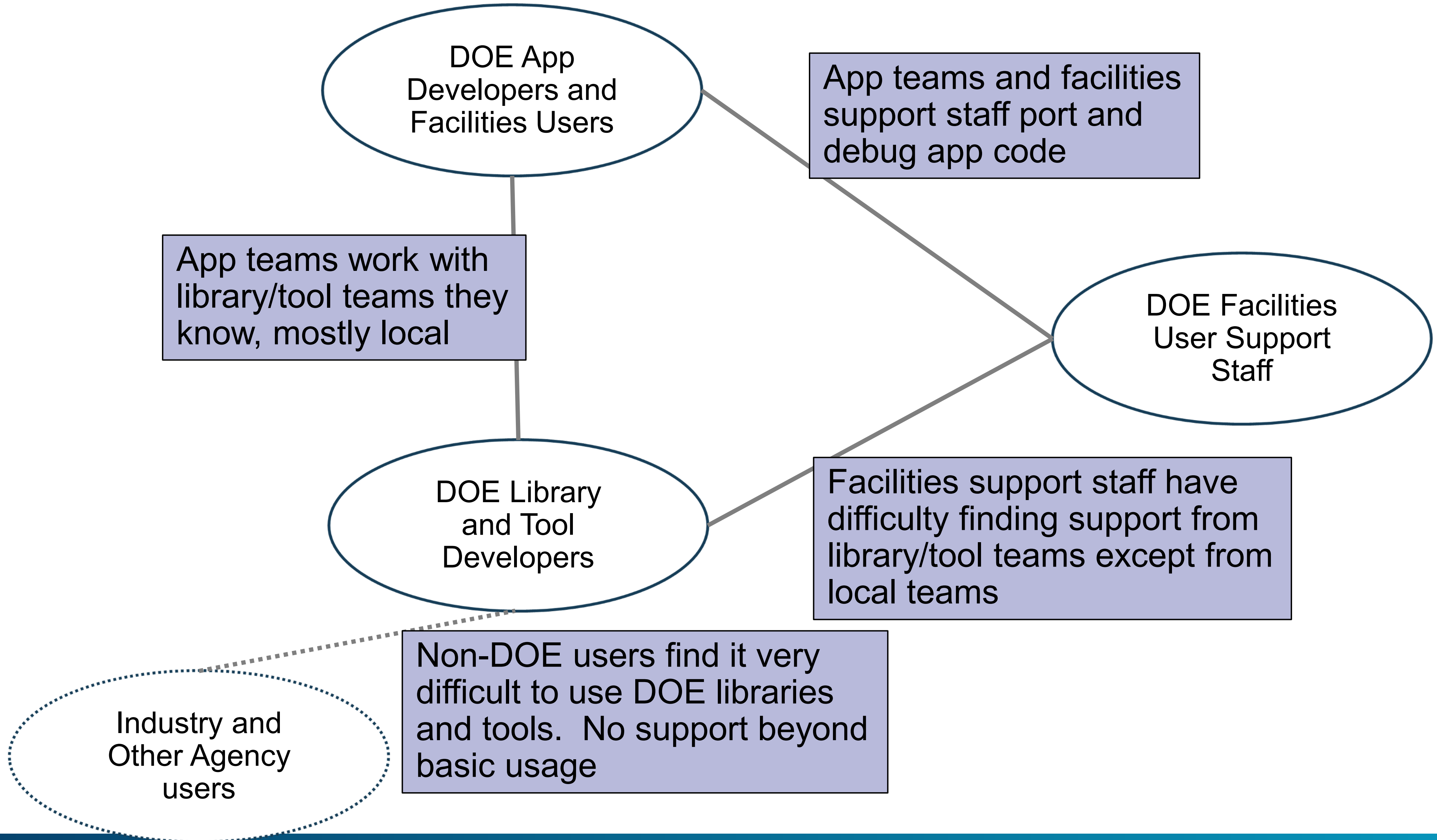
E4S and SDKs as platforms are providing tremendous value

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

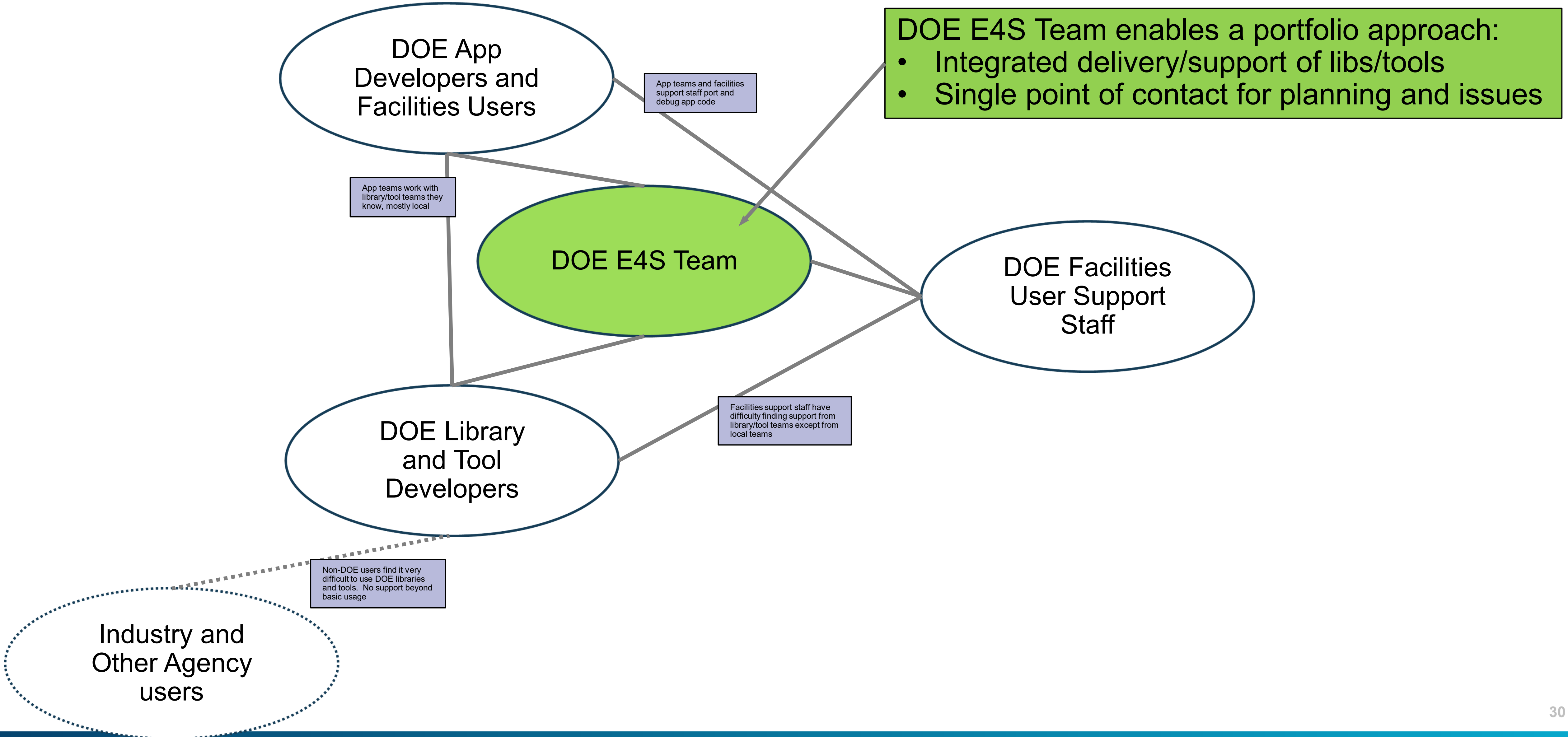
The SDK and E4S platforms provide compelling value for modest cost in ways that become more important going forward

Expanding the Value and Impact of Software Ecosystems Going Forward

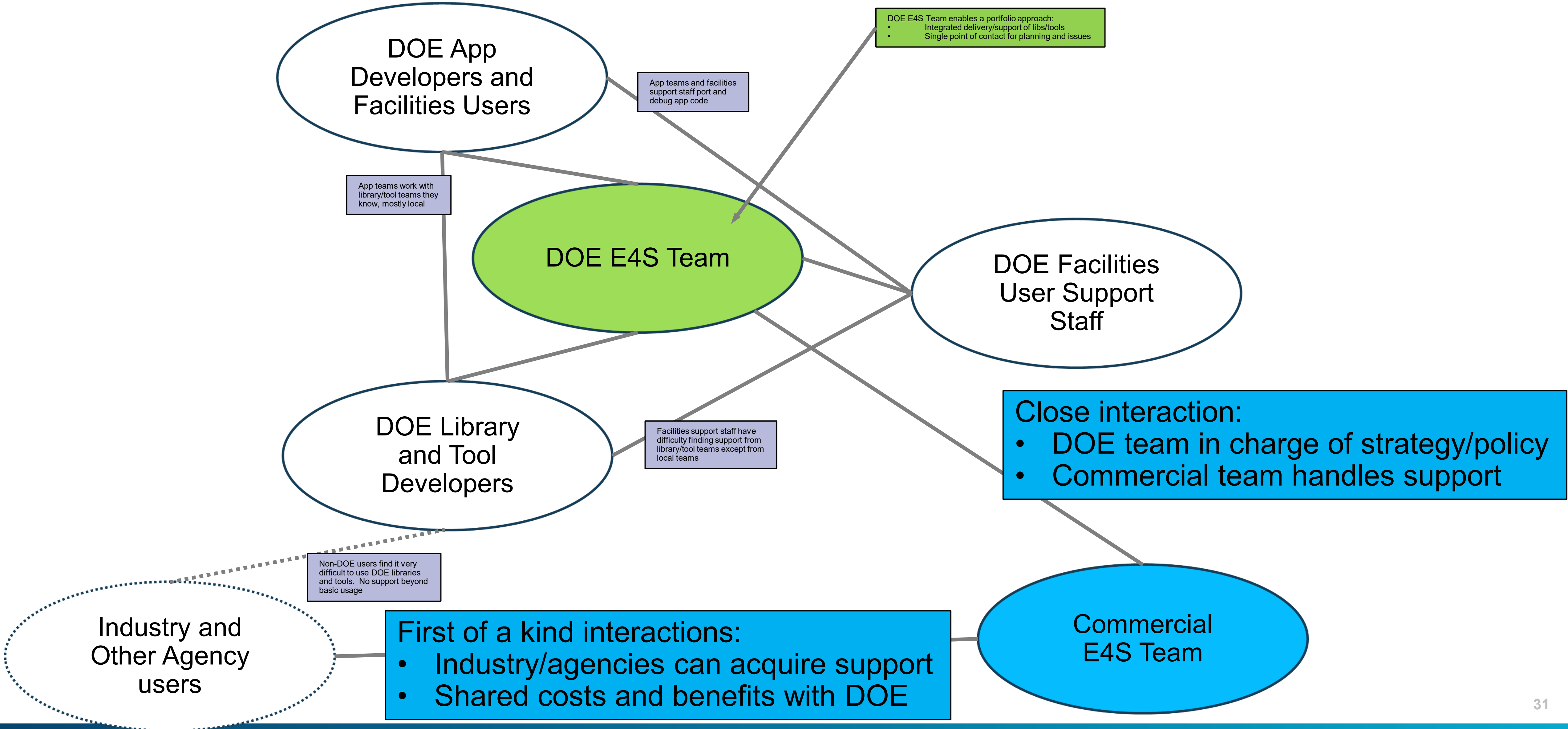
Pre-E4S User Support Model



E4S Phase 1 Support Model – Old relationships plus DOE E4S



E4S Phase 2 Support Model – Previous plus commercial E4S



Expanding the Scope of Cost and Benefit Sharing for DOE Software Libraries and Tools

Support Phase	Primary Scope	Primary Cost and Benefit Sharing Opportunities
Pre-E4S	Local facility	**Local costs and benefits:** Prior to ECP and E4S, libraries and tools were typically strongly connected to the local facility: ANL libs and tools at ALCF, LBL at NERSC, LLNL at Livermore Computing, etc.
+ ECP E4S	All DOE facilities	**DOE complex-shared costs and benefits:** ECP requires, and E4S enables, interfacility availability and use of libs across all facilities: First-class support of ANL libs and tools at other facilities, etc.
+ Commercial E4S	DOE facilities, other US agencies, industry, and more	**Universal shared costs and benefits:** Commercial support of E4S expands cost and benefit sharing to non-DOE entities: DOE costs are lower, software hardening more rapid. US agencies, industry and others can contract for support, gaining sustainable use of E4S software and contributing to its overall support.

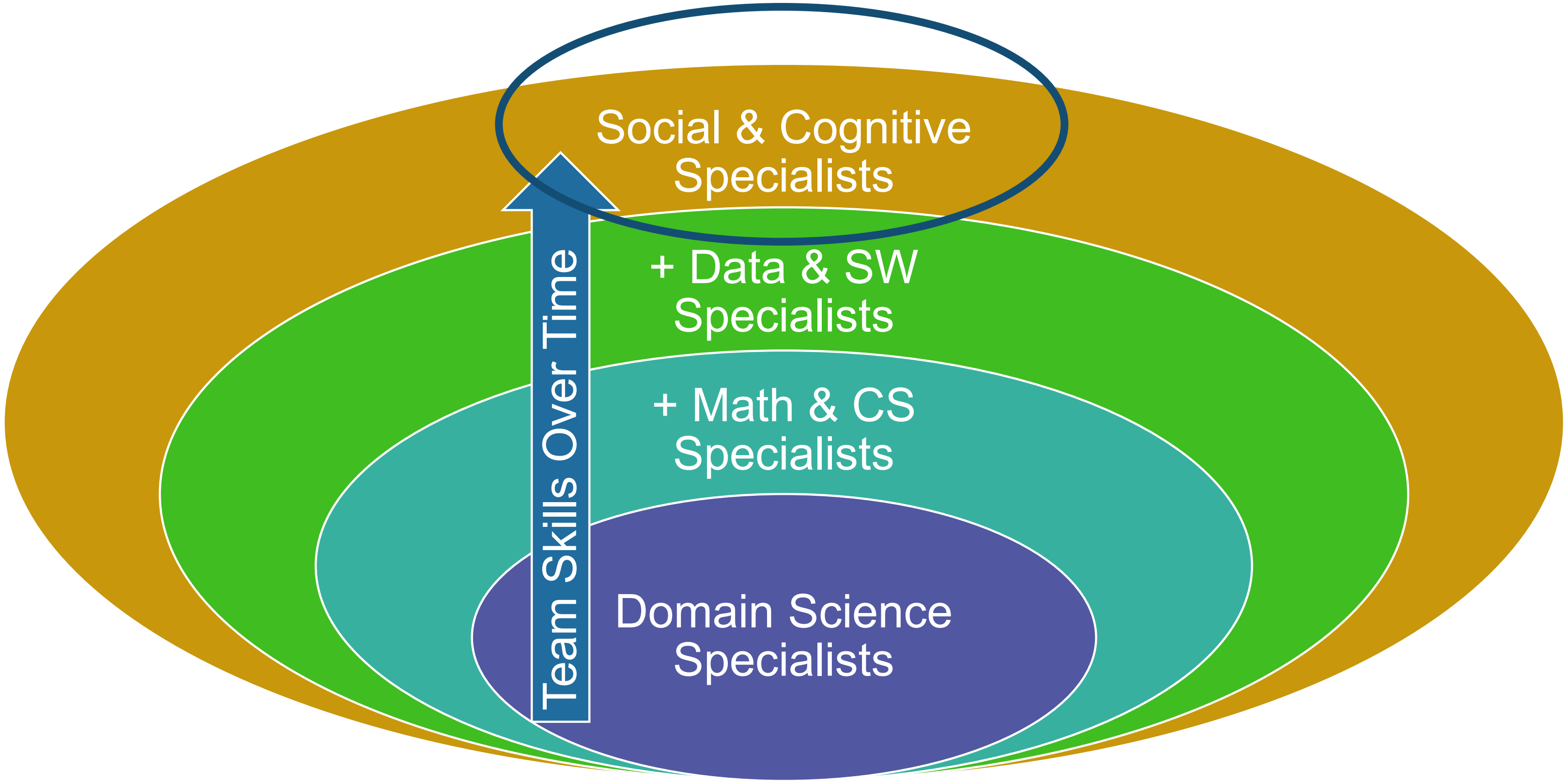
Software Sustainability Activities

Expanding Software Team Skills: Research Software Science (RSS)

Key observation: We are scientists, problem solvers. **Let's use science to address our challenges!**

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

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



https://bssw.io/blog_posts/research-software-science-a-scientific-approach-to-understanding-and-improving-how-we-develop-and-use-software-for-research

First-of-a-kind US DOE Workshop

- The Science of Scientific-Software Development and Use
  - Dec 13 – 16, 2021
  - https://www.orau.gov/SSSDU2021
- Workshop Brochure available:
  - https://doi.org/10.2172/1846008
- Workshop Report in progress:
  - 3 Priority Research Directions
  - 3 Crosscutting Themes

EXASCALE COMPUTING PROJECT

SSSDU Priority Research Directions

- **PRD1: Develop methodologies and tools to comprehensively improve team-based scientific software development and use** Focus: Team Impact
  - **Key question:** *What practices, processes, and tools can help improve the development, sustainment, evolution, and use of scientific software by teams?*
- **PRD2: Develop next-generation tools to enhance developer productivity and software sustainability** Focus: Developer Impact
  - **Key questions:** *How can we create and adapt tools to improve developer effectiveness and efficiency, software sustainability, and support for the continuous evolution of software? How can we support and encourage the adoption of such tools by developers?*
- **PRD3: Develop methodologies, tools, and infrastructure for trustworthy software-intensive science** Focus: Societal Impact
  - **Key questions:** *How can we facilitate and encourage effective and efficient reuse of data and software from third parties while ensuring the integrity of our software and the resulting science? How can we provide flexible environments that "bake in" the tracking of software, provenance, and experiment management required to support peer review and reproducibility?*

SSSDU Crosscutting Themes

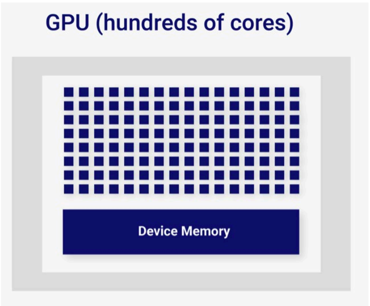
- **Theme 1:** We need to consider **both human and technical elements** to better understand how to improve the development and use of scientific software.
- **Theme 2:** We need to address urgent challenges in **workforce recruitment and retention** in the computing sciences with growth through **expanded diversity, stable career paths, and the creation of a community and culture** that attract and retain **new generations** of scientists.
- **Theme 3:** Scientific software has become essential to all areas of science and technology, creating opportunities for **expanded partnerships, collaboration, and impact**.

Takeaways from Expanding Impact in the Future

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

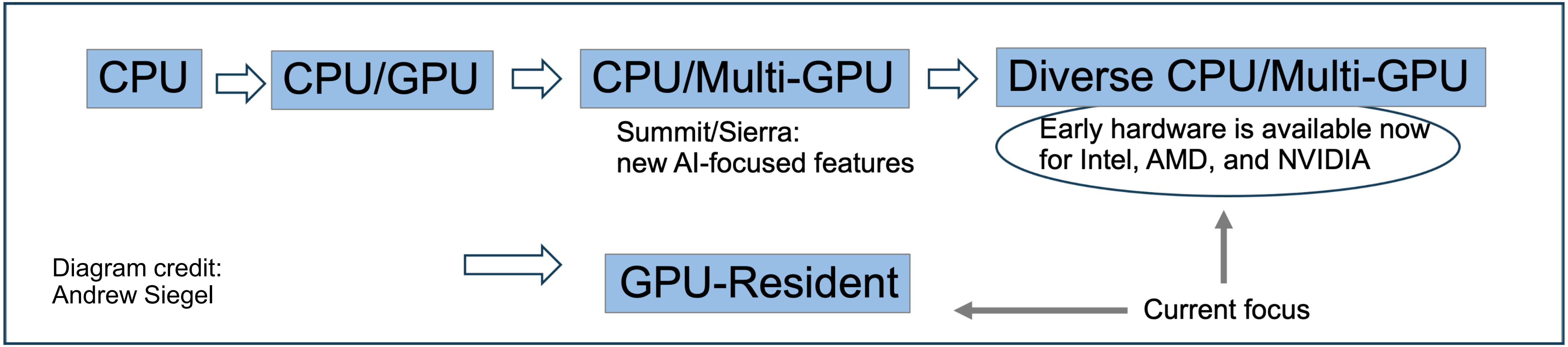
Developing software for GPU systems

Heterogeneous accelerated-node computing



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

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



EXASCALE COMPUTING PROJECT

Kokkos/RAJA

- Two distinct products: Kokkos and RAJA
  - Both originate in NNSA
  - RAJAs main funding/usage in NNSA
  - Kokkos gets half its funding from NNSA, but >70% of users outside of NNSA
- Learn from and leverage each other's work
  - Desul – common atomics library
  - Memory management – different strategies
- Other options: OpenMP, vendor-specific (CUDA, HIP, SYCL)

New algorithms highlights – batched computations, mixed precision

The “coopetition” model:

Step 1: Collaborative design space exploration

Step 2: Adaptation and implementation in each library

Batched Sparse Linear Algebra Phase 1 Implementation

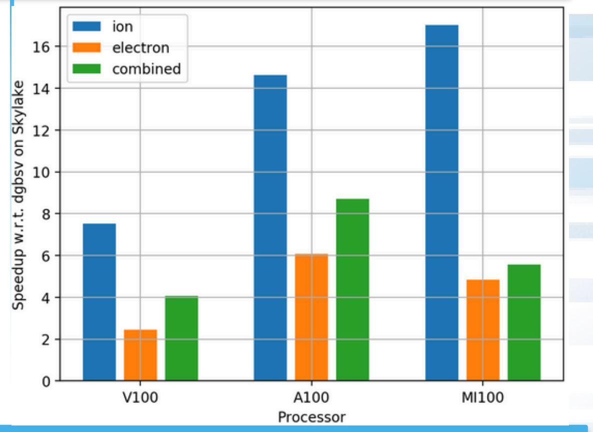
ECP WBS	WBS 2.3.3.01
PI	Ulrike Meier Yang, LLNL
Members	ANL, LBNL, LLNL, SNL, UC Berkeley, UTK
Milestone Lead	Piotr Luszczek, UTK

Scope and objectives

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

Speedup of Ginkgo batched iterative solvers over dgbsv

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



Impact

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

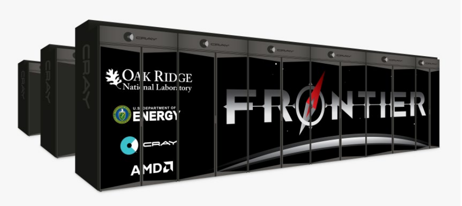
Project accomplishment

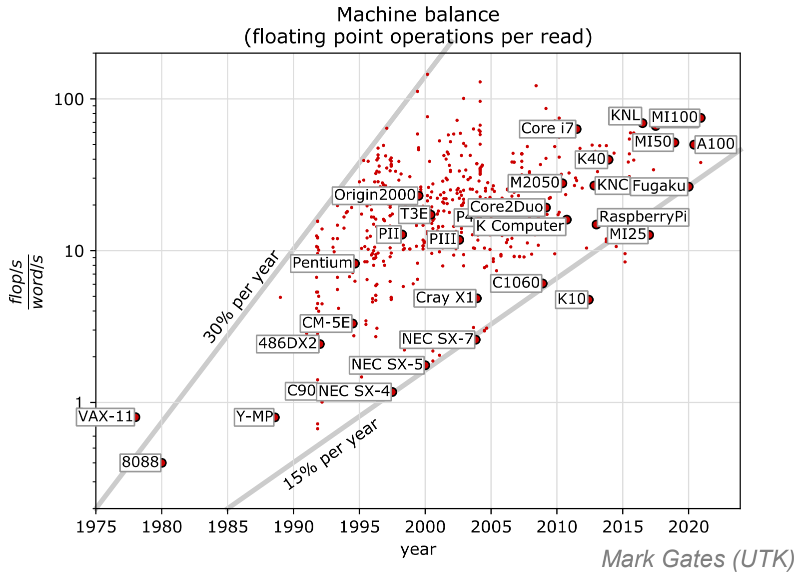
- Batched band Phase 1 implementation
- Batched sparse iterative and direct Phase 1 implementation
- Preparations for inclusion of batched functions in xSDK libraries and ECP applications
- Progress report on batched sparse linear algebra Phase 1 implementation

Deliverables The report is available at https://confluence.exascaleproject.org/display/STMS05/xSDK+Project+Documents in the file "Milestone 45 Report_Batched Sparse LA Phase 1 Implementation" – Ask Piotr Luszczek for copy

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

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





Mark Gates (UTK)

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

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

Nov 2009 Top500:
- Jaguar #1 system
- 1.75 petaflops/s
- FP64 (not FP16)

EXASCALE COMPUTING PROJECT

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, Sebastien 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 Alterative 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

ECP EXASCALE COMPUTING PROJECT

Step 2: Incorporate lessons learned into library ecosystem

For library interoperability and mixed precision usage:

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

Status of early-access system experience

Excellent progress toward Exascale readiness and a lot more to do

Performance portability

- Portability strategy:
  - Strategy 1: Isolate performance-impacting code to select kernels, write own CUDA, HIP, SYCL
  - Strategy 2: Product uses Kokkos and RAJA as primary portability layers
  - Blend 1 & 2: Provide both
  - Notes:
    - No ST products use OpenMP directly for GPU portability but
    - Kokkos and RAJA have OpenMP backends as an option

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)

The E4S Two-Step

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

The E4S Two-Step

- Step 2: Turn on GPU build
  - Builds with GPU backends (especially if using Kokkos or RAJA)
  - Transition to GPU is a debugging and adaptation exercise
  - Track growth in E4S GPU capabilities as E4S products improve GPU offerings
- Consider interactions with E4S commercial support team
  - Pay someone for support
  - Get advice on product choices
    - DOE teams generally can’t give you good advice on which solver or IO library to use
      - Like asking Microsoft and Apple to tell whether to purchase a PC or Mac

GPU Efforts Summary

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

Summary

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

Thank you

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.

EXASCALE COMPUTING PROJECT

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