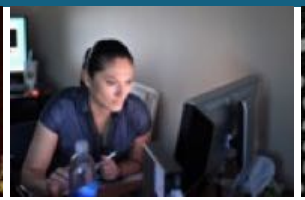
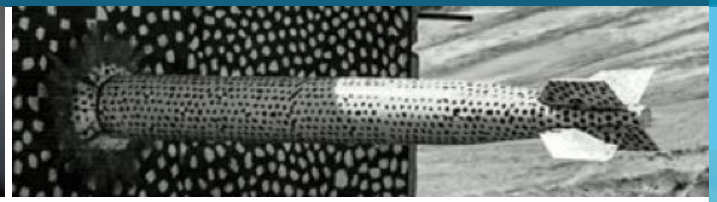
Leadership Scientific Software Trends from 2000 – 2040 Through the Lens of the Trilinos Project





PRESENTED BY

Michael Heroux

Sandia National Laboratories

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.

Brief history of my experience

- Started parallel programming with **Fortran and Q8 calls on CDC Cyber 205**
- Wrote Cray **LIBSCI** code for **BLAS, LAPACK, sparse solvers, FFTs** in **Fortran/CAL**
- Wrote code for industry apps **FIDAP, FLUENT, STAR-CD** for **vector/MPP** machines
- Founder of **Trilinos, Mantevo, HPCG** projects
- Founder of original **Kokkos** – initial use of **execution patterns**, **breaking of storage association**.
- Architect of **E4S, xSDK** – Exascale Computing Project (ECP) software ecosystems
- Director of Software Technology for ECP – Broad visibility into **70+ next-gen products**

Brief Trilinos History

2001 – 2009
2010 – 2016
2017 – Now

Some Trilinos History

Trilinos started in December 2001

- Fun fact: The first Trilinos commit was on Fri Dec 14 22:43:40 2001
- While the command `commit log --reverse` shows the first Trilinos commit was on Fri Feb 13 23:00:10 1998, this is a commit preserved from the partitioning package Zoltan that was integrated into Trilinos years later
- There are similar commits for the multigrid package ML

The "Tri" in Trilinos was determined by the intent for three packages, there are now 50+ packages

Trilinos phases:

- Started with the Epetra stack: MPI-only, double precision arithmetic, up to 2B equations
- New stack based on Tpetra: MPI+Kokkos, templated precisions, arbitrary problem size

Trilinos-Kokkos/KokkosKernels relationship:

- Kokkos started in Trilinos: Extracted to support users who don't need solvers, and those who do
- Kokkos and KokkosKernels snapshotted into Trilinos regularly

Motivation For Trilinos

- Sandia does LOTS of solver work.
- When I started at Sandia in May 1998:
  - Aztec was a mature package. Used in many codes.
  - FETI, PETSc, DSCPack, Spooles, ARPACK, DASPK, and many other codes were (and are) in use.
  - New projects were underway or planned in multi-level preconditioners, eigensolvers, non-linear solvers, etc…
- The challenges:
  - Little or no coordination was in place to:
    - Efficiently reuse existing solver technology.
    - Leverage new development across various projects.
    - Support solver software processes.
    - Provide consistent solver APIs for applications.
  - ASCI was forming software quality assurance/engineering (SQA/SQE) requirements:
    - Daunting requirements for any single solver effort to address alone.

Sandia National Laboratories

Evolving Trilinos Solution

- Trilinos[1] is an evolving framework to address these challenges:
  - Fundamental atomic unit is a *package*.
  - Includes core set of vector, graph and matrix classes (Epetra/Tpetra packages).
  - Provides a common abstract solver API (Thyra package).
  - Provides a ready-made package infrastructure (new_package package):
    - Source code management (cvs, bonsai).
    - Build tools (autotools).
    - Automated regression testing (queue directories within repository).
    - Communication tools (mailman mail lists).
  - Specifies requirements and suggested practices for package SQA.
- In general allows us to categorize efforts:
  - Efforts best done at the Trilinos level (useful to most or all packages).
  - Efforts best done at a package level (peculiar or important to a package).
  - **Allows package developers to focus only on things that are unique to their package.**

1. Trilinos loose translation: “A string of pearls”

Trilinos Strategic Goals

- **Scalable Solvers:** As problem size and processor counts increase, the cost of the solver will remain a nearly fixed percentage of the total solution time.
- **Hardened Solvers:** Never fail unless problem essentially unsolvable, in which case we diagnose and inform the user why the problem fails and provide a reliable measure of error.
- **Full Vertical Coverage:** Provide leading edge capabilities from basic linear algebra to transient and optimization solvers.

Algorithmic Goals

- **Universal Interoperability:** All Trilinos packages will be interoperable, so that any combination of solver packages that makes sense algorithmically will be possible within Trilinos.
- **Universal Solver RAS:** Trilinos will be:
  - Integrated into every major application at Sandia (**Availability**).
  - The leading edge hardened, efficient, scalable solutions for each of these applications (**Reliability**).
  - Easy to maintain and upgrade within the application environment (**Serviceability**).

Software Goals

Sandia National Laboratories

Trilinos Packages

- Trilinos is a collection of *Packages*.
- Each package is:
  - Focused on important and state-of-the-art algorithms in its problem regime.
  - Developed by a small team of domain experts.
  - Self-contained: No (or minimal) explicit dependencies on any other software packages (with some special exceptions).
  - Configurable/buildable/documented on its own.
- Sample packages: NOX, AztecOO, IFPACK.
- Special packages: Epetra, TSF, Teuchos.

Sandia National Laboratories

Greek Names

Day 1 of Package Life

- **CVS:** Each package is self-contained in Trilinos/package/ directory.
- **Bugzilla:** Each package has its own Bugzilla product.
- **Bonsai:** Each package is browsable via Bonsai interface.
- **Mailman:** Each Trilinos package, including Trilinos itself, has four mail lists:
  - package-checkins@software.sandia.gov
    - CVS commit emails. “Finger on the pulse” list.
  - package-developers@software.sandia.gov
    - Mailing list for developers.
  - package-users@software.sandia.gov
    - Issues for package users.
  - package-announce@software.sandia.gov
    - Releases and other announcements specific to the package.
- **New_package** (optional): Customizable boilerplate for
  - Autoconf/Automake/Doxygen/Python/Thyra/Epetra/TestHarness/Website

Sandia National Laboratories

Sample Package Maturation Process

Step	Example
Package added to CVS: Import existing code or start with new_package.	ML CVS repository migrated into Trilinos (July 2002).
Mail lists, Bugzilla Product, Bonsai database created.	ml-announce, ml-users, ml-developers, ml-checkins, ml-regression @software.sandia.gov created, linked to CVS (July 2002).
Package builds with configure/make, Trilinos-compatible	ML adopts Autoconf, Automake starting from new_package (June 2003).
Epetra objects recognized by package.	ML accepts user data as Epetra matrices and vectors (October 2002).
Package accessible via Thyra interfaces.	ML adaptors written for TSFCore_LinOp (Thyra) interface (May 2003).
Package uses Epetra for internal data.	ML able to generate Epetra matrices. Allows use of AztecOO, Amesos, Ifpack, etc. as smoothers and coarse grid solvers (Feb-June 2004).
Package parameters settable via Teuchos ParameterList	ML gets manager class, driven via ParameterLists (June 2004).
Package usable from Python (PyTrilinos)	ML Python wrappers written using new_package template (April 2005).

Startup Steps

Maturation Steps

Trilinos Interoperability Mechanisms

- M1: Package accepts user data as Epetra objects.
- M2: Package can be used via TSF abstract solver classes.
- M3: Package can use Epetra for private data.
- M4: Package accesses solver services via TSF interfaces.
- M5: Package builds under Trilinos `configure` scripts.

Interoperability Example: AztecOO

- AztecOO: Preconditioned Krylov Solver Package.
- Primary Developer: Mike Heroux.
- Minimal *explicit, essential* dependence on other Trilinos packages.
  - Uses abstract interfaces to matrix/operator objects.
  - Has independent configure/build process (but can be invoked at Trilinos level).
  - Sole dependence is on Epetra (but easy to work around).
- *Interoperable* with other Trilinos packages:
  - Accepts user data as Epetra matrices/vectors.
  - Can use Epetra for internal matrices/vectors.
  - Can be used via TSF abstract interfaces.
  - Can be built via Trilinos configure/build process.
  - Can provide solver services for NOX.
  - Can use IFPACK, ML or AztecOO objects as preconditioners.

Observations from Trilinos 2001 - 2009

Focus on creating a federation to address numerous stakeholder issues:

- Bringing independent teams together to address software quality requirements
- Provide community for inter-dependent development teams
- Provide a single collection of libraries for users
- Retain small team ability for name recognition, autonomy at local level
- Provide a large-scale product portfolio that sponsors can track, assess and talk about

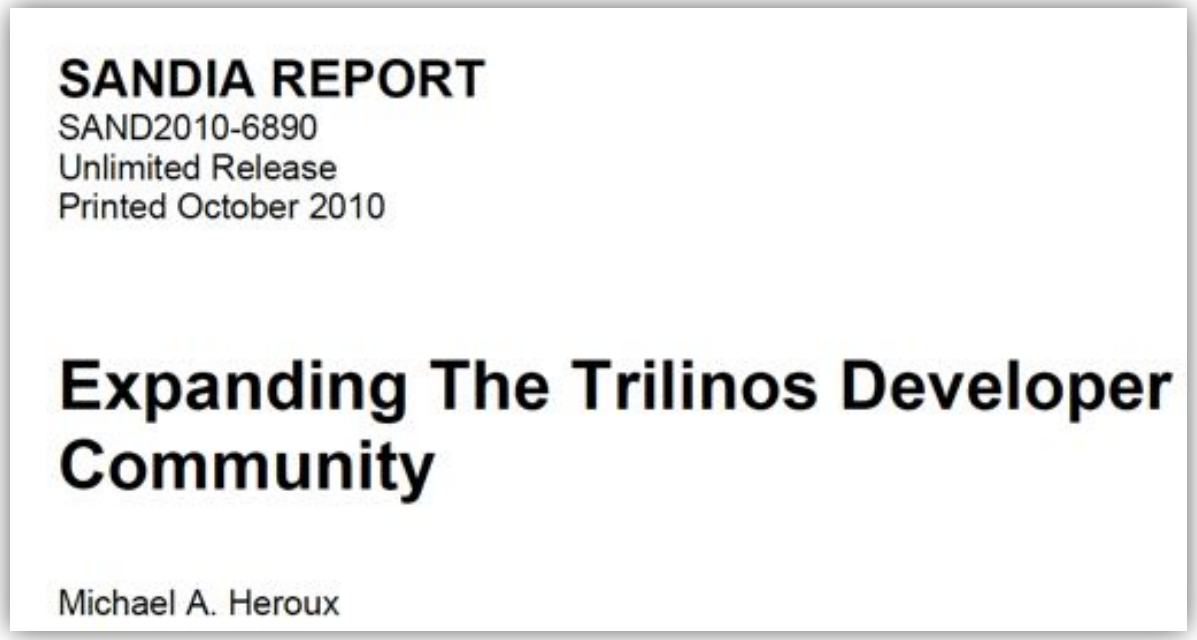
Provide software platform:

- Common tools, processes and infrastructure
- Interoperable components for each other to use
- Ready-made NewPackage to kickstart a new effort
- Technical engagement with application teams
- Common data services API via Epetra abstract classes (e.g., Epetra_Operator)

Many of these attributes have modern replacements:

- Kokkos/KokkosKernels/Tpetra
- GitHub repos, tools, workflows
- TriBITS/CMake and Spack

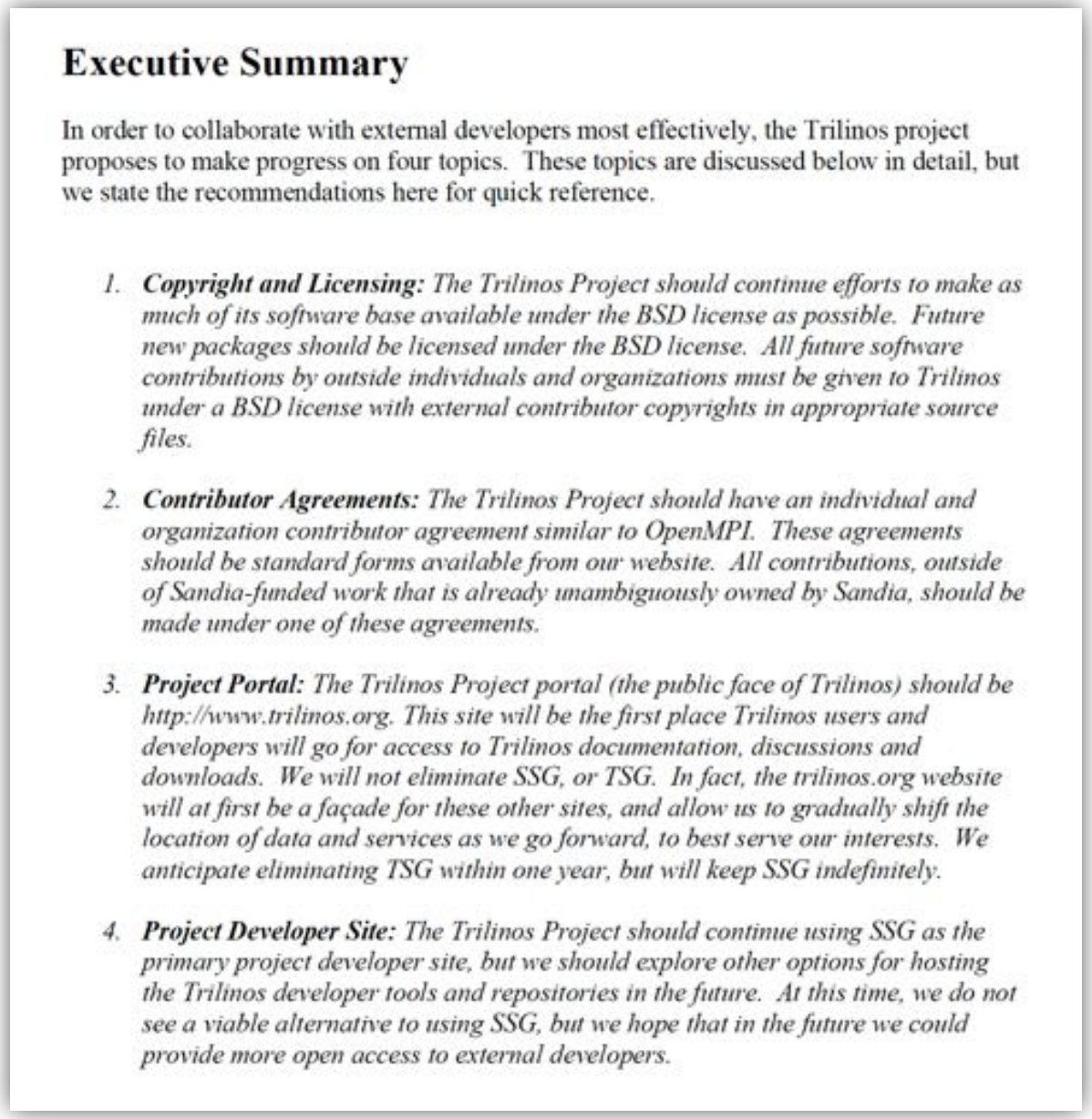
Expanding the Trilinos Developer Community



SANDIA REPORT
SAND2010-6890
Unlimited Release
Printed October 2010

Expanding The Trilinos Developer Community

Michael A. Heroux



Executive Summary

In order to collaborate with external developers most effectively, the Trilinos project proposes to make progress on four topics. These topics are discussed below in detail, but we state the recommendations here for quick reference.

1. ***Copyright and Licensing:*** *The Trilinos Project should continue efforts to make as much of its software base available under the BSD license as possible. Future new packages should be licensed under the BSD license. All future software contributions by outside individuals and organizations must be given to Trilinos under a BSD license with external contributor copyrights in appropriate source files.*

2. ***Contributor Agreements:*** *The Trilinos Project should have an individual and organization contributor agreement similar to OpenMPI. These agreements should be standard forms available from our website. All contributions, outside of Sandia-funded work that is already unambiguously owned by Sandia, should be made under one of these agreements.*

3. ***Project Portal:*** *The Trilinos Project portal (the public face of Trilinos) should be http://www.trilinos.org. This site will be the first place Trilinos users and developers will go for access to Trilinos documentation, discussions and downloads. We will not eliminate SSG, or TSG. In fact, the trilinos.org website will at first be a façade for these other sites, and allow us to gradually shift the location of data and services as we go forward, to best serve our interests. We anticipate eliminating TSG within one year, but will keep SSG indefinitely.*

4. ***Project Developer Site:*** *The Trilinos Project should continue using SSG as the primary project developer site, but we should explore other options for hosting the Trilinos developer tools and repositories in the future. At this time, we do not see a viable alternative to using SSG, but we hope that in the future we could provide more open access to external developers.*

2010 –Focus on transition to community project

- Permissive license for easier corporate interactions
- Contributor agreements for non-Sandia members
- Website with non-Sandia and non-gov root
- Open repository
- Tremendous effort and commitment to make real

The Transition to GitHub

Never migrated to SVN

[EXTERNAL] [Trilinos-developers] Trilinos officially on github

Trilinos-developers <trilinos-developers-bounces@trilinos.org>
on behalf of

Perschbacher, Brent M <bmpersc@sandia.gov>
Tue 11/17/2015 12:28 PM
To: trilinos-developers@trilinos.org <trilinos-developers@trilinos.org>

Hello all,
The Trilinos Framework team is pleased to announce that the move to Github has been completed! The main Trilinos repository has been moved to github.com/trilinos/Trilinos.git and it is ready for you to begin working on it. This repo was filtered to remove files which has made it history incompatible with any existing clones of Trilinos. It is recommended that you start from a fresh clone of Trilinos to avoid issues due to the history changes. This is not a requirement, but is the easiest transition path. If you have commits you need to get to github there are instructions on how to do this below.

Note that the packages that were split off into their own repos as part of this move are not quite ready yet. In the coming days we do expect to have them moved too and will announce when they are ready. The packages that will be in their own repo are: moocho, optika, Sundance, Ctrilinos, ForTrilinos, WebTrilinos, meros, and mesquite.

To access Trilinos you will need a Github account. If you haven't already please send your github user name to Jim so that he can invite you to the Trilinos project. If you do not already have an account you can sign up for free at: https://github.com/join. Keep in mind that Github's Terms of Service only allow one free account per person.

How to get Trilinos from github:
Github allows both https and ssh access. Both are equally valid and both have their pros and cons in a Sandia environment. SSH is what we have been using so it may be more natural to continue with it, but the choice can be made on an individual basis
https:
git clone https://github.com/trilinos/Trilinos.git
Note that if you choose to use https you will need to make sure your proxies are set correctly on every machine that you intend to do work from. You can find the information for Sandia's proxies here: https://sems.sandia.gov/sa/how-do-i-configure-sandia-proxy-settings
ssh:
git@github.com:trilinos/Trilinos.git
Note that if you choose to use ssh you will need to upload your public key to github for each machine that you intend to do work from. Github has a convenient way to add keys to your profile through the website. You can find instructions on how to add keys at: https://help.github.com/articles/generating-ssh-keys/#step-4-add-your-ssh-key-to-your-account

If you ever forget this information github provides the various URLs on the page for each repo on the right hand menu bar.

One of the most useful features we can leverage with the move to Github is forking. Forking can be used to avoid pushing branches to the main repository in most cases, and also can be useful for facilitating interactions with people who don't have push access to the repository. Here is an introduction to the concept: https://help.github.com/articles/fork-a-repo/

What to do if you didn't get all your changes in before the move:
If you didn't get everything pushed before the move fret not as it is still possible to get that work onto a clone from github. You will need to have committed everything that you want to move over. These commits don't have to be clean and ready to push, but you should get them as close as you can otherwise you will have

EuroTUG as external collaboration diagnostic

EuroTUG meeting series has been around since 2012:

- 2012 in Lausanne, Switzerland
- 2013 in Munich, Germany
- 2014 in Lugano, Switzerland
- 2015 in Paris, France
- 2016 in Garching, Germany
- 2019 in Zurich, Switzerland
- 2022 virtually in Munich, Germany

Recent challenges (starting in 2015 or so):

- Dev team focused on GPUs
  - Heavy technical co-design work
  - Disruptive usage model
- Many users not ready for GPU investment
  - Ubiquitous, disruptive code changes
  - GPU benefits for sparse codes only modest

Presently:

- Trilinos more ready for broad user group
- Users must transition to GPUs for performance

Time to renew outreach:

- Virtual and on-demand
- In-person as circumstances permit

June 5, 2012 EuroTUG, EPFL, Lausanne, Switzerland

Observations from Trilinos 2010 - 2016

Focus on expanding communities:

- Developers outside of Sandia
- Users outside of Sandia

Mature software products:

- Good documentation
- Lots of examples
- Very powerful compositional capabilities for multi-physics
- Rich capabilities for circuits
- MPI-only

Transition to new tools:

- CMake (via TriBITS)
- Git and GitHub
- External web presence

This version of Trilinos is still widely used today

New Package: Kokkos

- Very new project.
- Goal:
  - Isolate key non-BLAS kernels for the purposes of optimization.
- Kernels:
  - Dense vector/multivector updates and collective ops (not in BLAS).
  - Sparse MV, MM, SV, SM.
- Serial-only for now.
- Reference implementation provided.
- Mechanism for improving performance:
  - Default is aggressive compilation of reference source.
  - BeBOP: Jim Demmel, Kathy Yelick, Rich Vuduc, UC Berkeley.
  - Vector version: Cray.

Sandia National Laboratories

Example Kernels: axpy() and dot()

```
template <class WDP>
void
Node::parallel_for(int beg, int end,
                   WDP workdata    );
```

```
template <class WDP>
WDP::ReductionType
Node::parallel_reduce(int beg, int end,
                      WDP workdata    );
```

```
template <class T>
struct AxpyOp {
  const T * x;
  T * y;
  T alpha, beta;
  void execute(int i)
  { y[i] = alpha*x[i] + beta*y[i]; }
};
```

```
template <class T>
struct DotOp {
  typedef T ReductionType;
  const T * x, * y;
  T identity()       { return (T)0;      }
  T generate(int i)  { return x[i]*y[i]; }
  T reduce(T x, T y) { return x + y;     }
};
```

```
AxpyOp<double> op;
op.x = ...;  op.alpha = ...;
op.y = ...;  op.beta  = ...;
node.parallel_for< AxpyOp<double> >
                 (0, length, op);
```

```
DotOp<float> op;
op.x = ...;  op.y = ...;
float dot;
dot = node.parallel_reduce< DotOp<float> >
                          (0, length, op);
```

SIAM PP10
Baker & Heroux

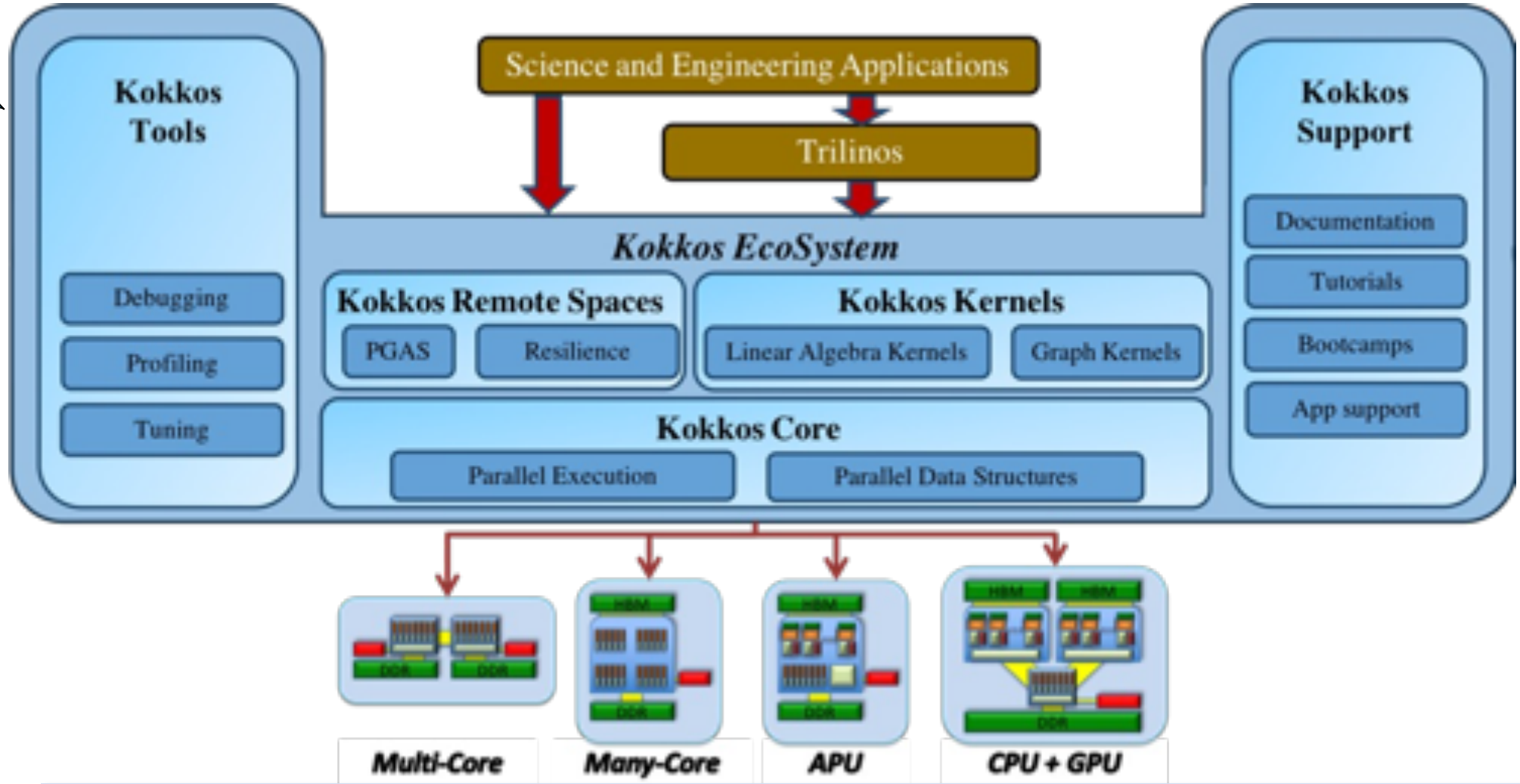
Hybrid Timings (Tpetra)

- Tests of a simple iterations:
  - power method: one sparse mat-vec, two vector operations
  - conjugate gradient: one sparse mat-vec, five vector operations
- DNVS/x104 from UF Sparse Matrix Collection (100K rows, 9M entries)
- NCCS/ORNL Lens node includes:
  - one NVIDIA Tesla C1060
  - one NVIDIA 8800 GTX
  - Four AMD quad-core CPUs
- Results are very tentative!
  - suboptimal GPU traffic
  - bad format/kernel for GPU
  - bad data placement for threads

Node	PM (mflop/s)	CG (mflop/s)
Single thread	140	614
8800 GPU	1,172	1,222
Tesla GPU	1,475	1,531
Tesla + 8800	981	1,025
16 threads	816	1,376
1 node 15 threads + Tesla	867	1,731
2 nodes 15 threads + Tesla	1,677	2,102

Kokkos Ecosystem for Performance Portability

2021 Alphabet Talk
S. Rajamanickam



Kokkos Ecosystem addresses complexity of supporting numerous many/multi-core architectures that are central to DOE HPC enterprise

Observations from Trilinos 2017 - now

The move to accelerator platforms has been incredibly disruptive for everyone:

- Change in execution model (scale inward, discrete memory, new ISAs, new programming models, etc)
- New algorithms, aggregated applications
- New vendor hardware and software products
- Ubiquitous change to application source code

Demands a vertical co-design/development from vendor to libraries to applications

Result is an inward focus:

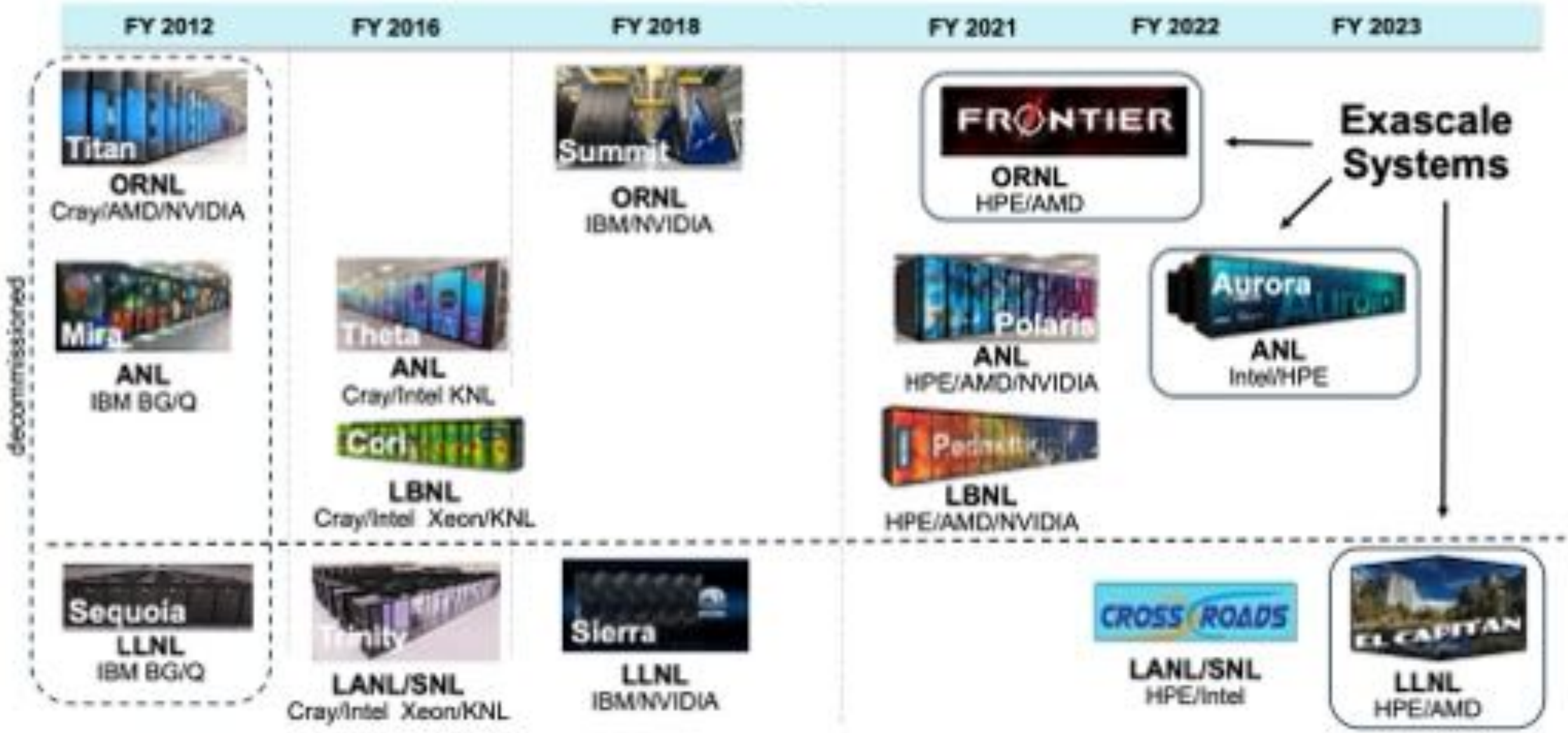
- Work with teams who are funded to work together and paid to embrace disruption
- Others must wait for new functionality and documentation until intensive design and development efforts stabilize

Still in this phase, but approaching its end

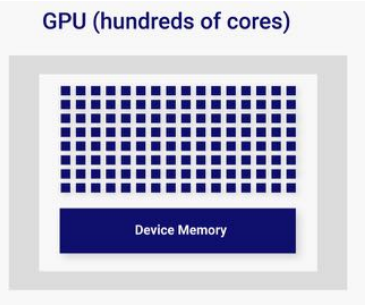
- EuroTUG 2022 is evidence we are emerging from an inward focus
- Lots of work to assist users in migrating to GPUs

Expanding the DOE Open-Source Software Ecosystem: ECP and E4S

DOE HPC Roadmap to Exascale Systems



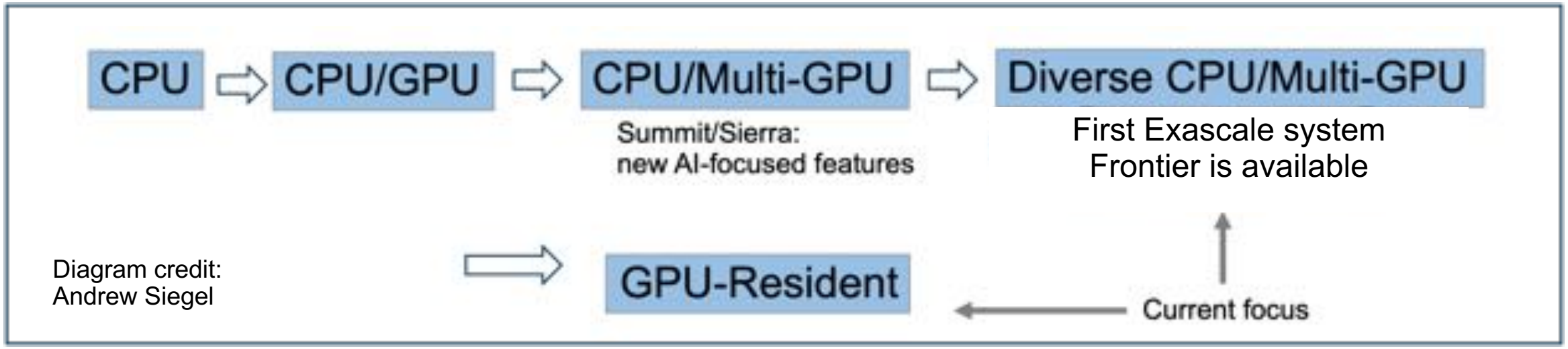
Heterogeneous accelerated-node computing



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.

Text credit: Doug Kothe



Ref: A Gentle Introduction to GPU Programming, Michele Rosso and Andrew Myers, May 2021

ST L4 Teams

- WBS
- Name
- PIs
- PCs - Project Coordinators

ECP ST Stats

- 35 L4 subprojects
- ~27% ECP budget

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	Bosilica, 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	Lang, Michael	Vigil, Brittney
2.3.1.17	OMPI-X	Bernholdt, David	Grundhoffer, Alicia
2.3.1.18	RAJA/Kokkos	Trott, Christian Robert	Prince, Kellsie
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	Vetter, 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	Glassbrook, Dick
2.3.2.11	SOLLVE: Scaling OpenMP with LLVm for Exascale	Chapman, Barbara	Kale, Vivek
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	Heroux, Jim	Grundhoffer, Alicia
2.3.3.15	Sake: Scalable Algorithms and Kernels for Exascale	Rajamanickam, Siva	Prince, Kellsie
2.3.4	**Data and Visualization**	**Ahrens, James**	
2.3.4.01	Data and Visualization Software Development Kit	Ahrens, James	Bagha, Neelam
2.3.4.09	ADIOS Framework for Scientific Data on Exascale Systems	Klasky, Scott	Grundhoffer, Alicia
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	Cappello, Franck	Ebling, 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	Prince, Kellsie

- ~250 staff
- ~70 products
- 35 teams
- ~30 universities
- ~9 DOE labs
- 6 technical areas
- 1 focus area of 3 in ECP

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

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

Software categories:
- **Next generation established products:** Widely used HPC products (e.g., MPICH, OpenMPI, Trilinos)
- **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

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)

Trilinos has been several of these types of platforms over time, but none is a perfect fit

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 uses platforms to foster collaboration and cooperation as we head into the frontier
- The ECP has two primary software platforms:
  - 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

Levels of Integration → Product → Source and Delivery

ECP ST Open Product Integration Architecture

- Build all SDKs
- Build complete stack
- Containerize binaries

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

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

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

- Standard workflow
- Existed before ECP

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

ECP ST Individual Products

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

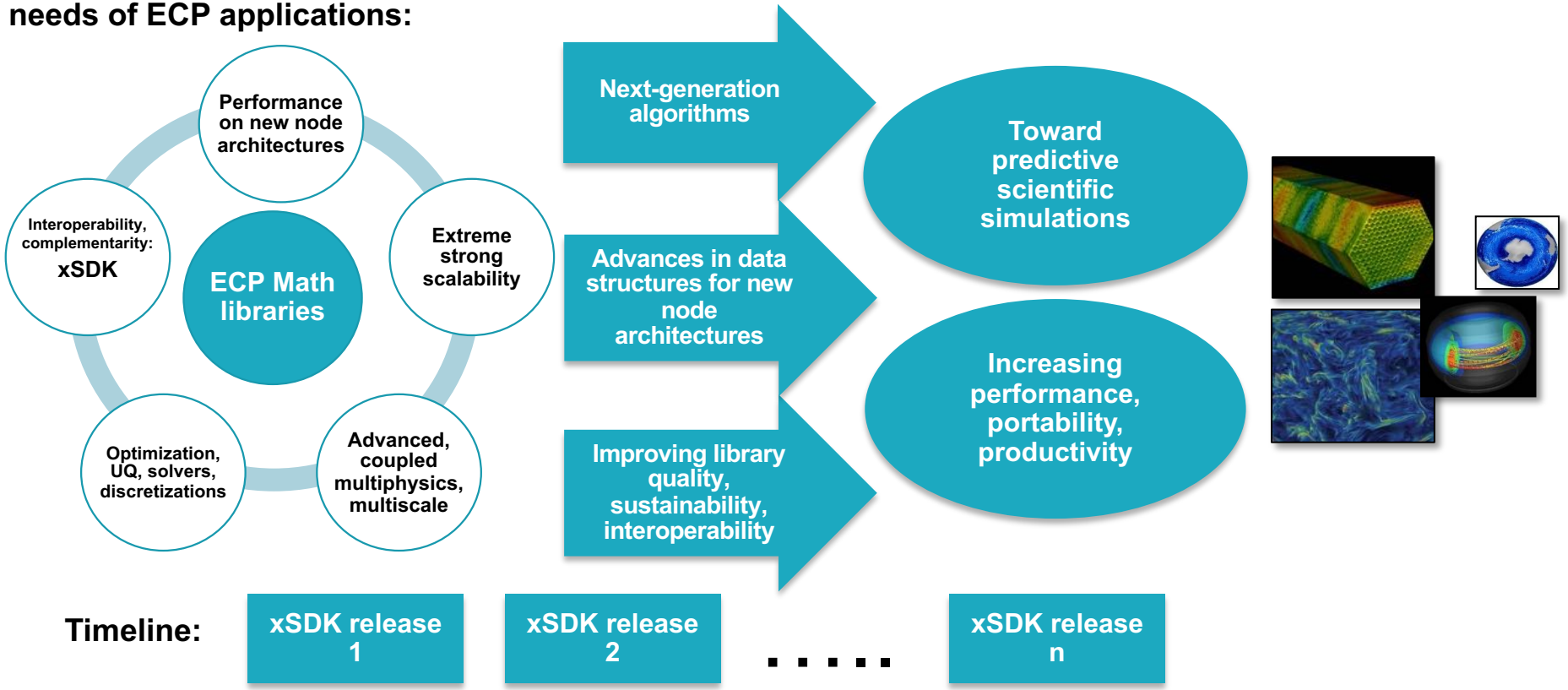

xSDK lead: Ulrike Meier Yang (LLNL)
xSDK release lead: Satish Balay (ANL)

xSDK release 0.7.0 (Nov 2021)

- hypre
- PETSc/TAO
- SuperLU
- Trilinos
- AMReX
- ArborX
- ButterflyPACK
- DTK
- Ginkgo
- heFFTe
- libEnsemble
- MAGMA
- MFEM
- Omega_h
- PLASMA
- PUMI
- SLATE
- Tasmanian
- SUNDIALS
- Strumpack
- Alquimia
- PFLOTRAN

} from the broader community
- deal.II
- preCICE
- PHIST
- SLEPc

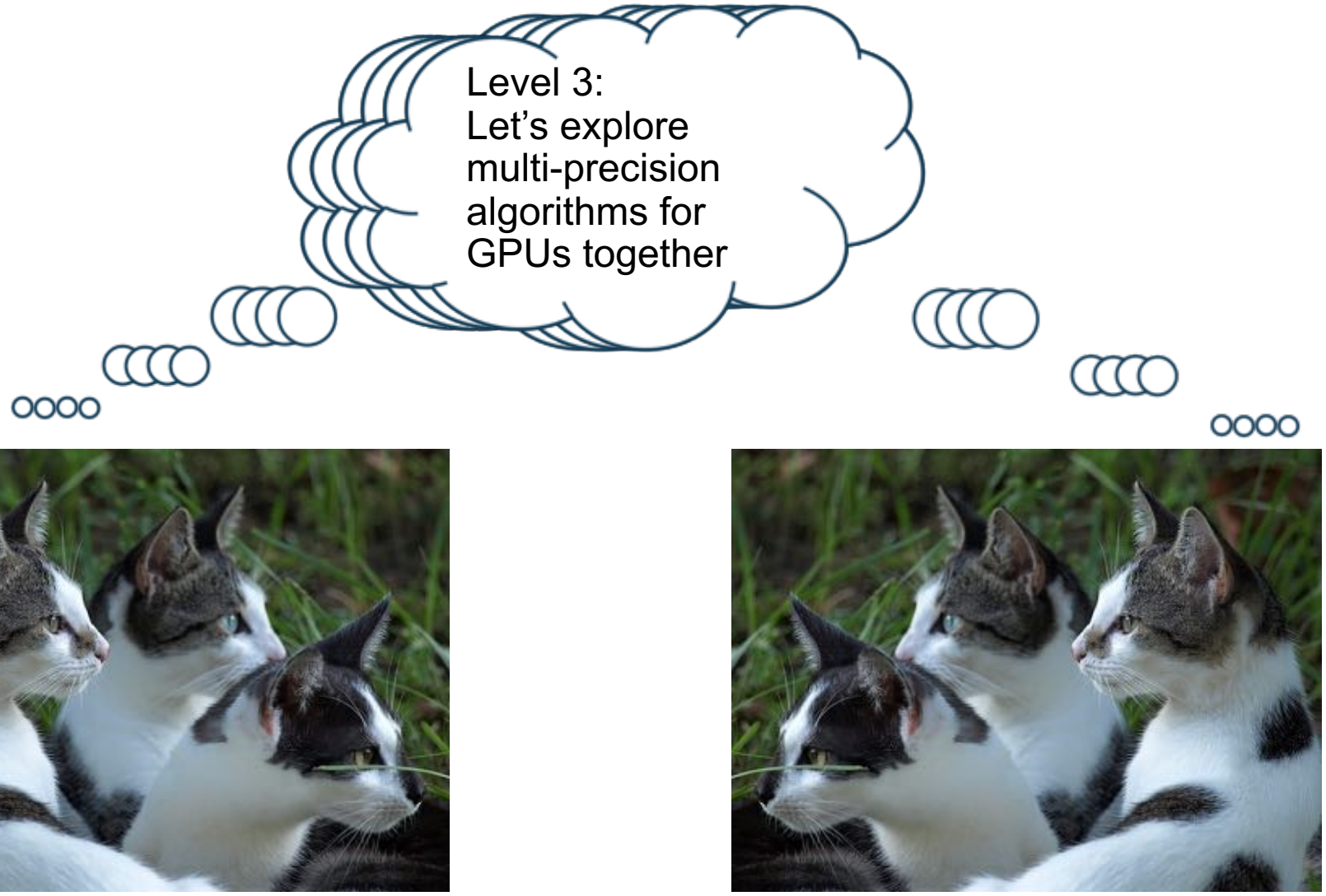
As motivated and validated by the needs of ECP applications:



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

An SDK Maturity Model or, The Benefits of Coop-etition

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



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);*

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

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

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

Step 2: Incorporate lessons learned into library ecosystem

For library interoperability and mixed precision usage:

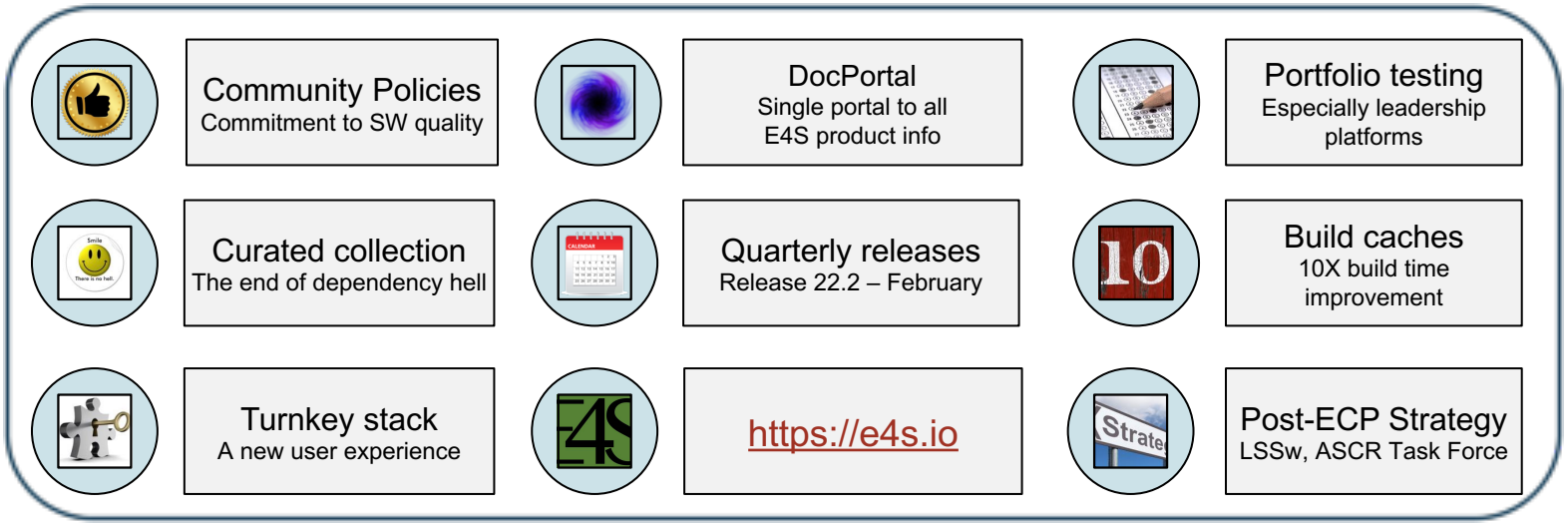
- **PETSc** develops an abstraction layer to device solvers (vendor libraries, Kokkos Kernels, etc.) that allows flexible composition of Krylov solves in mixed-precision;
- **hypre** already supports the compilation in different precisions and work now focuses on compiling multiple precisions at a time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

Takeaways from SDKs

- 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, some developers don’t see the net benefit
  - 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

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: Aug 2022: E4S 22.08 – 100+ full release products



Spack

https://spack.io

Spack lead: Todd Gamblin (LLNL)

E4S

https://e4s.io

E4S lead: Sameer Shende (U Oregon)

E4S 22.08 Highlights

- ARM64 systems with NVIDIA GPUs
- Base and full featured container images targeting three GPU architectures (Intel, AMD, NVIDIA)
- Base images may be used to build custom containers and support GPUs
- New versions of AI/ML frameworks TensorFlow and PyTorch optimized for GPUs on all three architectures: x86_64, ppc64le, and aarch64.

Download E4S 22.05 GPU Container Images: NVIDIA, AMD, Intel

- Separate full featured Singularity images for 3 GPU architectures
- GPU base images for
  - x86_64 (Intel, AMD, NVIDIA)
  - ppc64le
  - aarch64

https://e4s.io

E4S Community Policies: *A commitment to quality improvement*

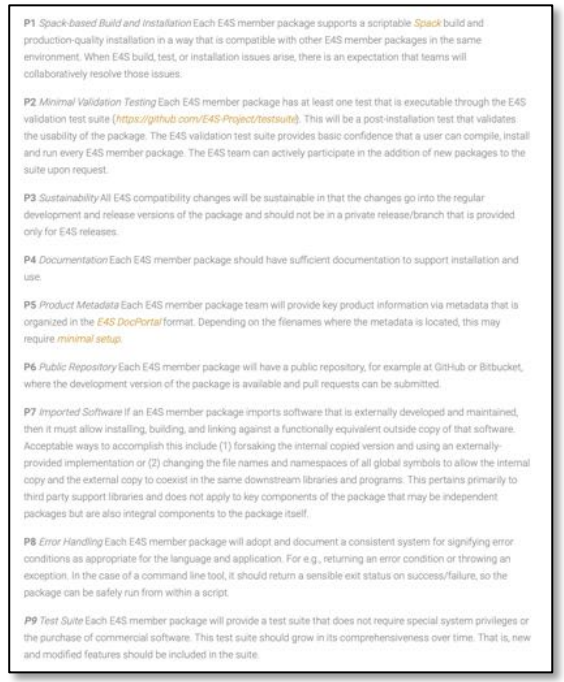
- Purpose: Enhance sustainability and interoperability
- Will serve as membership criteria for E4S
  - Membership is not required for *inclusion* in E4S
  - Also includes forward-looking draft policies
- Modeled after xSDK community policies
- Multi-year effort led by SDK team
  - Included representation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

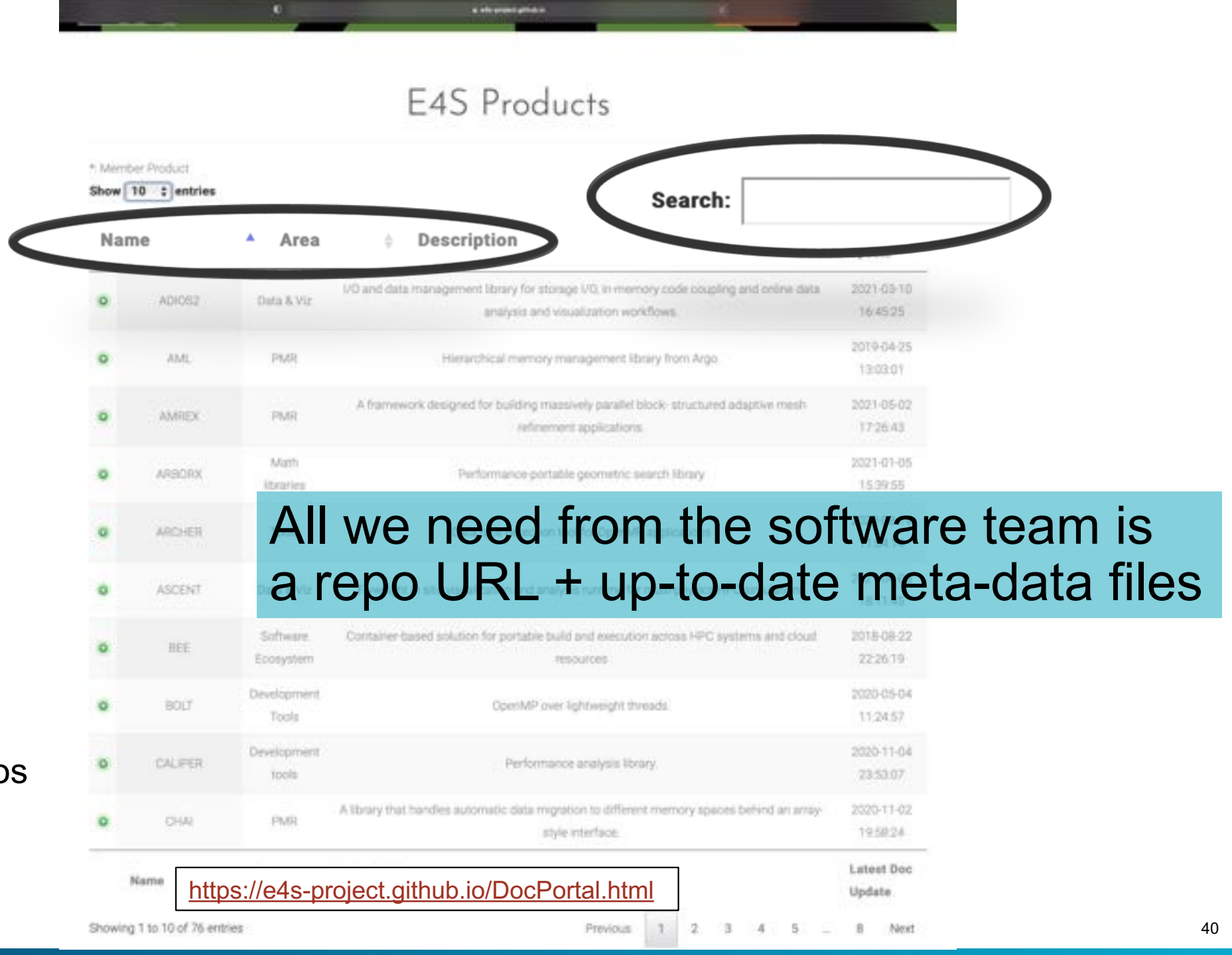
- **P1: *Spack-based Build and Installation***
- **P2: *Minimal Validation Testing***
- **P3: *Sustainability***
- **P4: *Documentation***
- **P5: *Product Metadata***
- **P6: *Public Repository***
- **P7: *Imported Software***
- **P8: *Error Handling***
- **P9: *Test Suite***



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

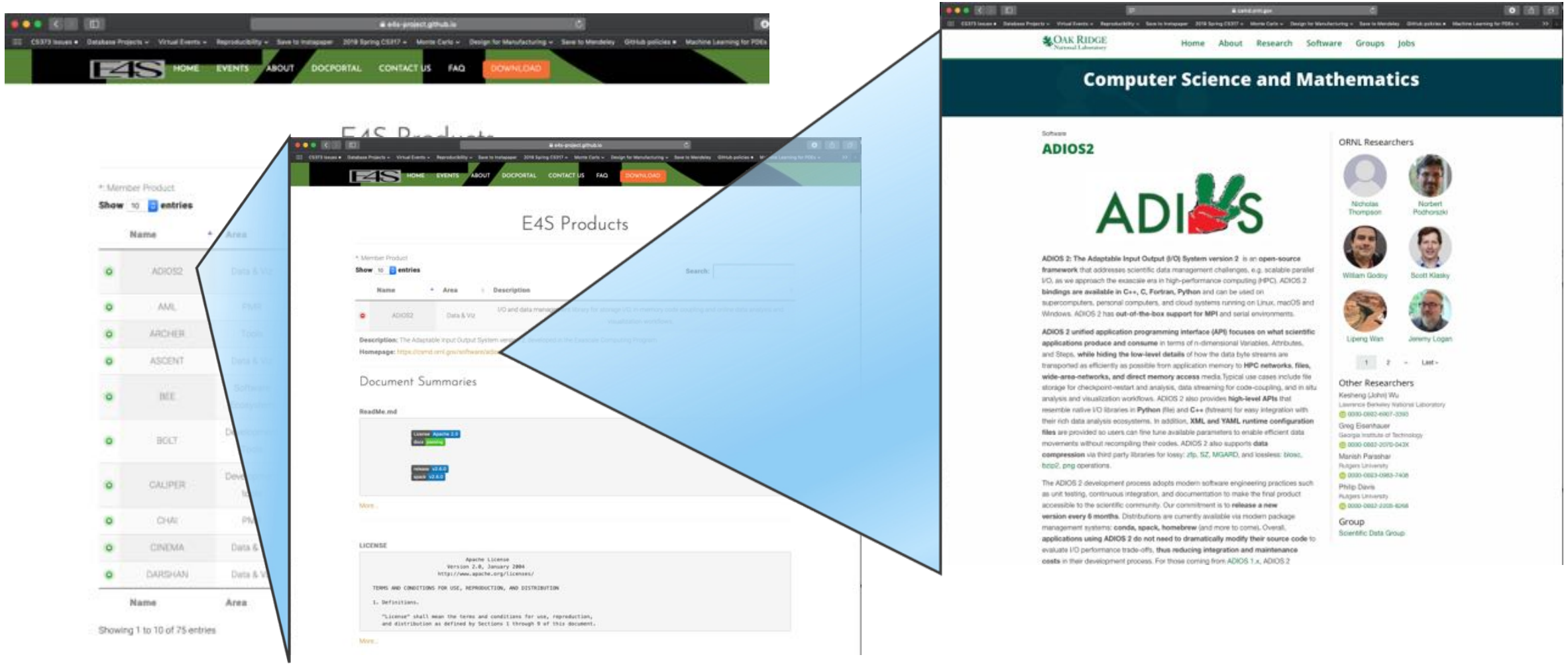
E4S DocPortal

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



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

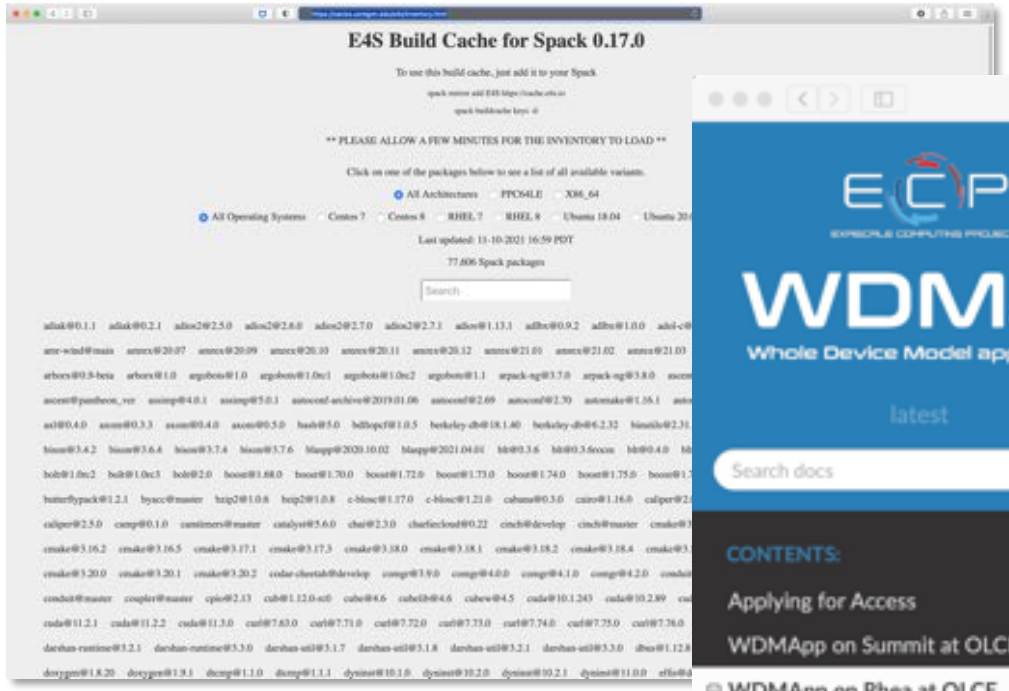
Goal: All E4S product documentation accessible from single portal on E4S.io (working mock webpage below)

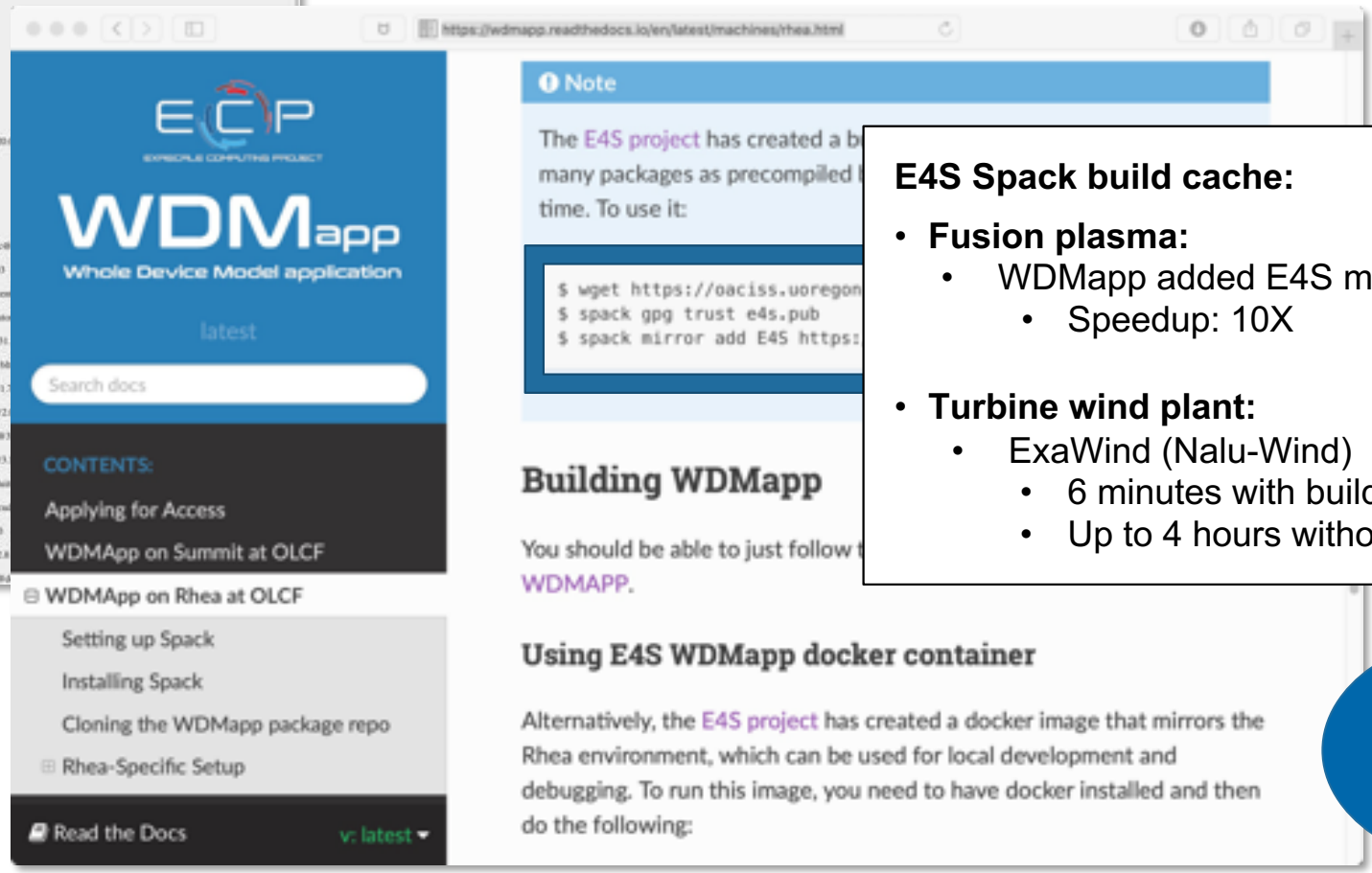


https://e4s-project.github.io/DocPortal.html

Speeding up bare-metal installs using the E4S build cache

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

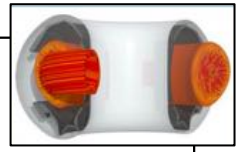


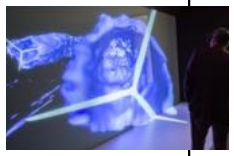


- 88,000+ binaries
- S3 mirror
- No need to build from source code!

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





Special thanks to Sameer Shende, WDMapp and ExaWind teams

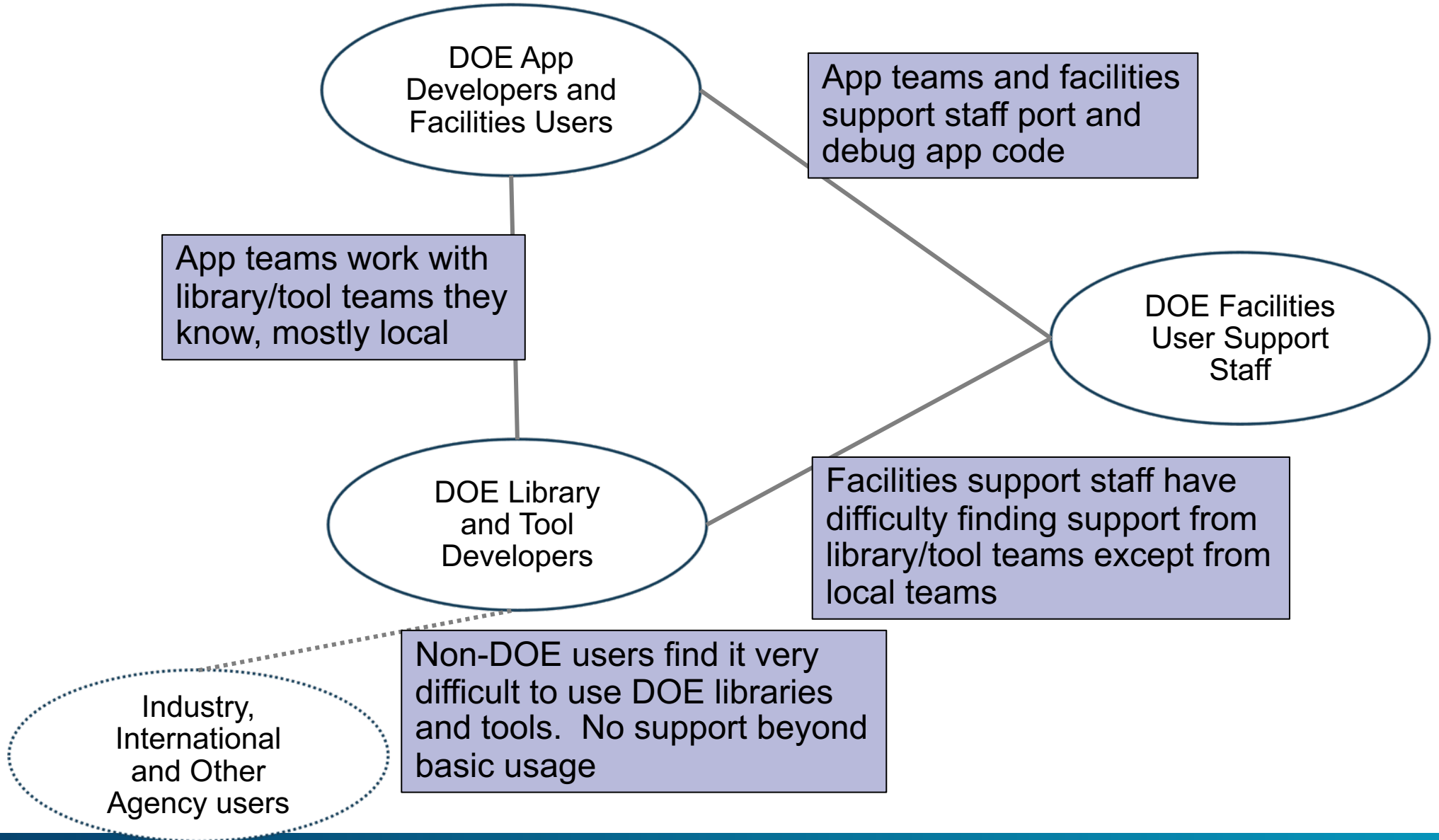
https://wdmapp.readthedocs.io/en/latest/machines/rhea.html

Summary: E4S and SDKs as Platforms

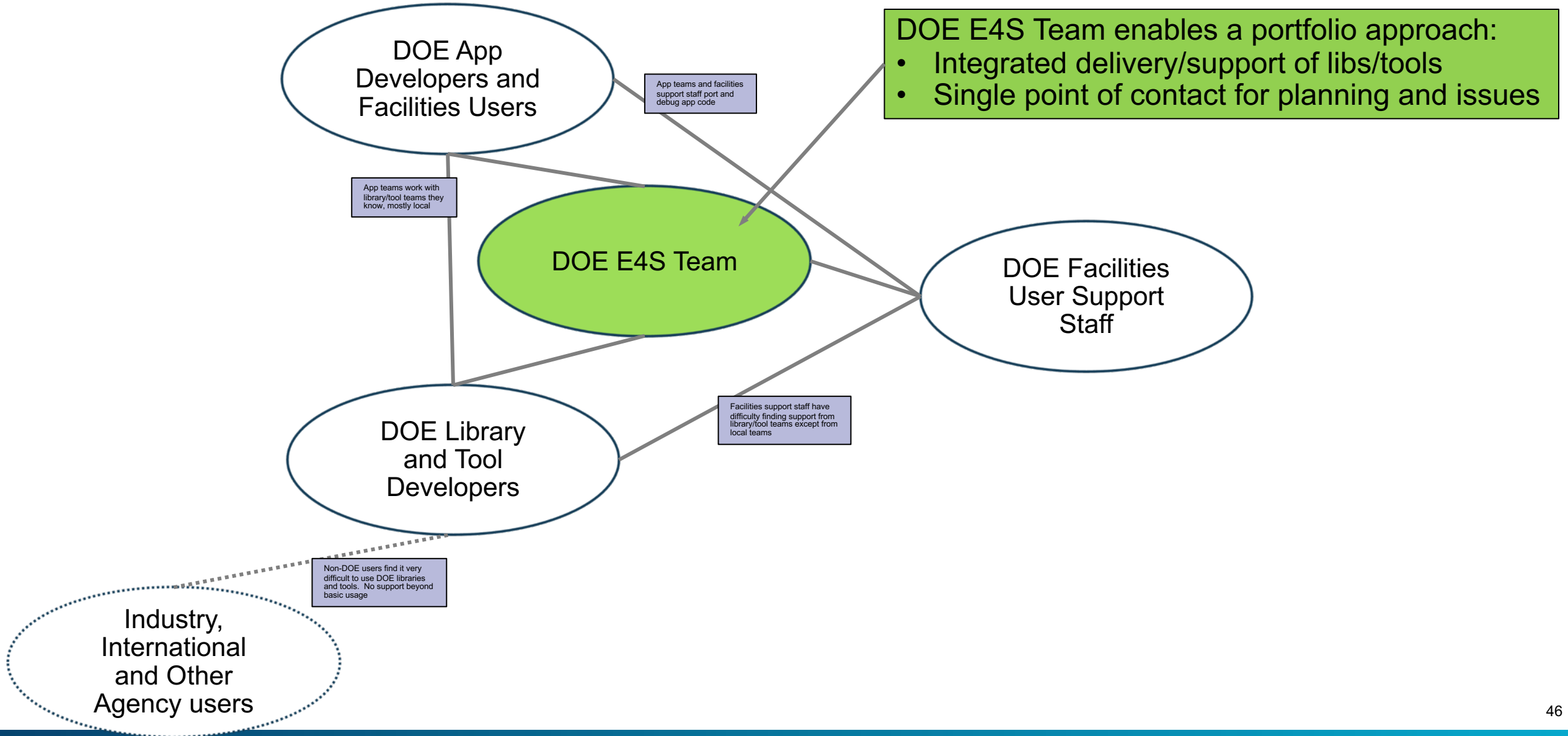
Activity	SDKs	E4S
Planning	Transparent and collaborative requirements, analysis and design, delivery	Campaign-based portfolio planning coordinated with Facilities, vendors, community ecosystem, non-DOE partners
Implementation	Leverage shared knowledge, infrastructure, best practices	ID and assist product 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
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	Portfolio-wide quality policies, documentation portal, portfolio testing on many platforms not available to developers
Path-finding	Exploration and development of leading-edge computational tools that provide capabilities and guidance for others	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	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	Programmatic cultivation of community through outreach and funded opportunities that expand the membership possibilities

Expanding the Value and Impact of Software Ecosystems Going Forward

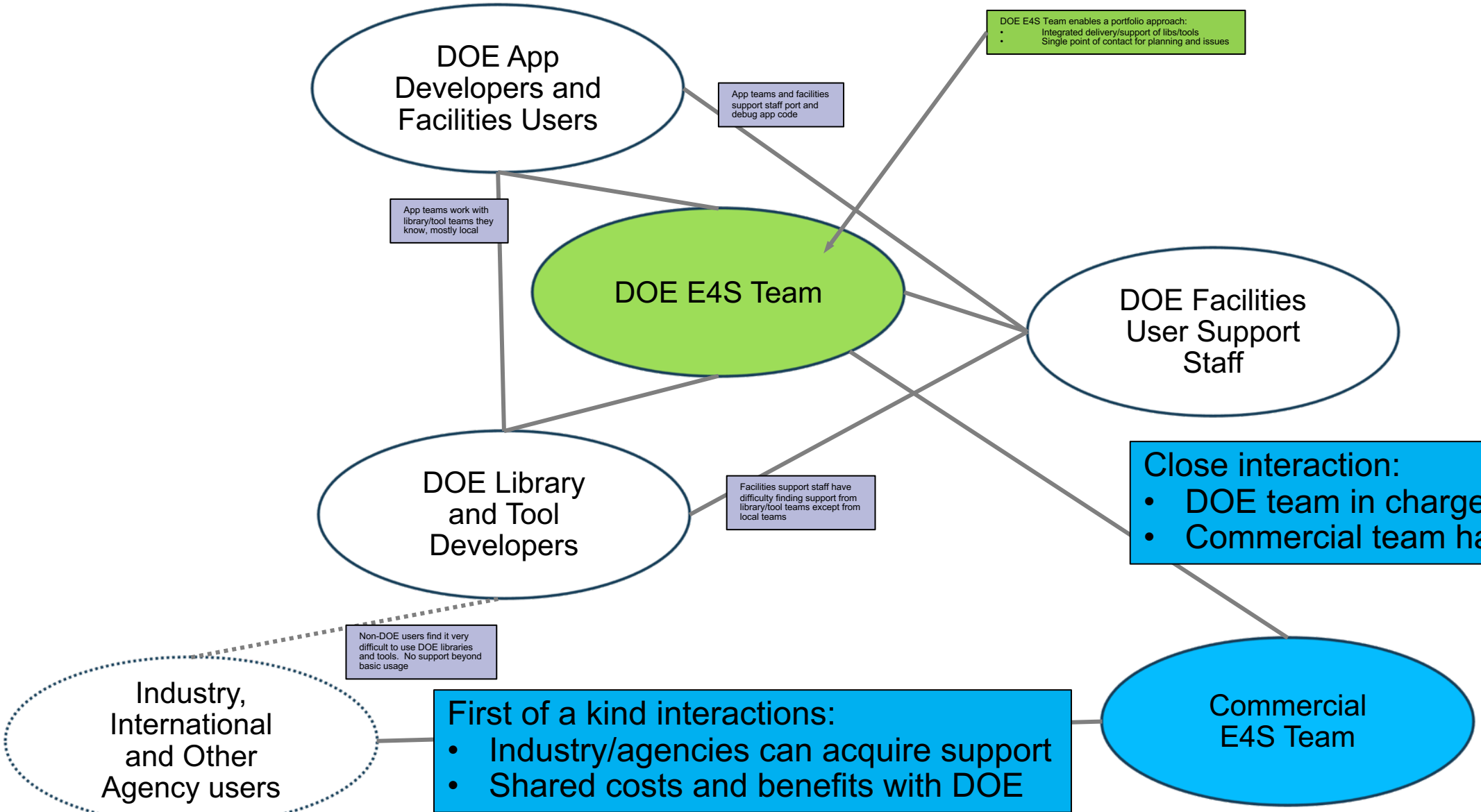
Pre-E4S User Support Model



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.

Observations for Trilinos and ECP E4S and the SDKs

ECP is large, structured, and spanning enough time to establish new software approaches
- Creation of a 3-tier software org and corresponding levels of software aggregation (product, SDK, E4S)
- Time enough to change culture and demonstrate value to stakeholders

Trilinos efforts are both part of E4S and the xSDK and outside of them
- Majority of Trilinos funding is not ECP-related
- Benefits to being part of E4S and xSDK include
  - Being part of a larger community
  - Increased mindshare, recruiting new staff,
  - Shared exploration of new topics (e.g., mixed/multi-precision)
  - Better ecosystem interoperability
- Costs of being part of E4S and xSDK include
  - Overheads of synchronizing, coordinating
  - Complications from need for collaborative open-source development and mission security needs

Introducing a commercial partner facilitates:
- Universal cost and benefit sharing
- Off-loading of open testing to a non-Sandia partner
- Easier partnering with non-Sandia developers

Basic Strategies for GPUs

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.

Research Software Science

Expanding the skillset for producing & using scientific software

What is Research Software Science?

- Definition: *Applying the scientific method to understanding and improving how software is developed and used for research*

  - **Scientific Method**
    - Use formal observation and experimentation to obtain & disseminate knowledge
    - Current approach is ad hoc, engineered: See a problem, explore options to improve, pick one, move on
    - Yes, there is software engineering research, so let's call it science too
  - **Understanding and Improving**
    - Obtain data to detect correlation, design experiments to identify cause and effect
  - **Developed and Used**
    - Developer/User, User-only, individuals, teams, communities
    - Leverage cognitive and social sciences
  - **Research**
    - Focus on software used in service of scientific advances

RSS Components

- Technical component
  - Research software addresses highly technical domains
    - Participation requires advanced degrees, on-going participation in domain community – significant time investment
    - Reason why “off-the-shelf” software tools & processes often need adaptation, or may not address high-priority needs
- Social component
  - Scientific software development and use are increasingly a team (and team of teams) activity
  - Teams often composed of members who are unaware (and uninterested?) in exploring human factors
  - Community engagement is increasingly important
- Cognitive component
  - Research software community members are problem solvers, love new and challenging problems
  - Are also sometimes described as “herds of cats”, resistant to prescriptive approaches

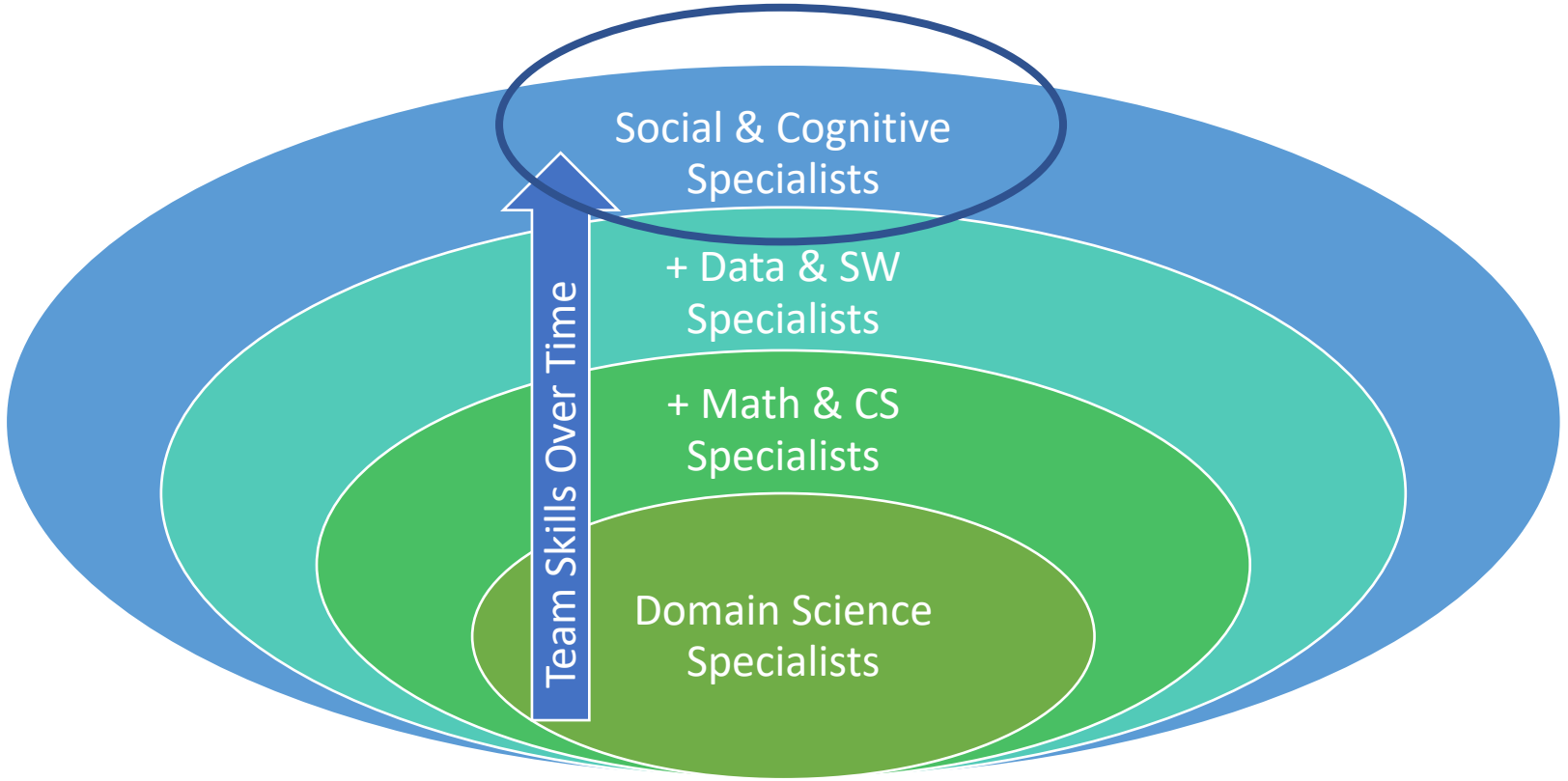
Expanding Software Team Skills: Research Software Science (RSS)

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**

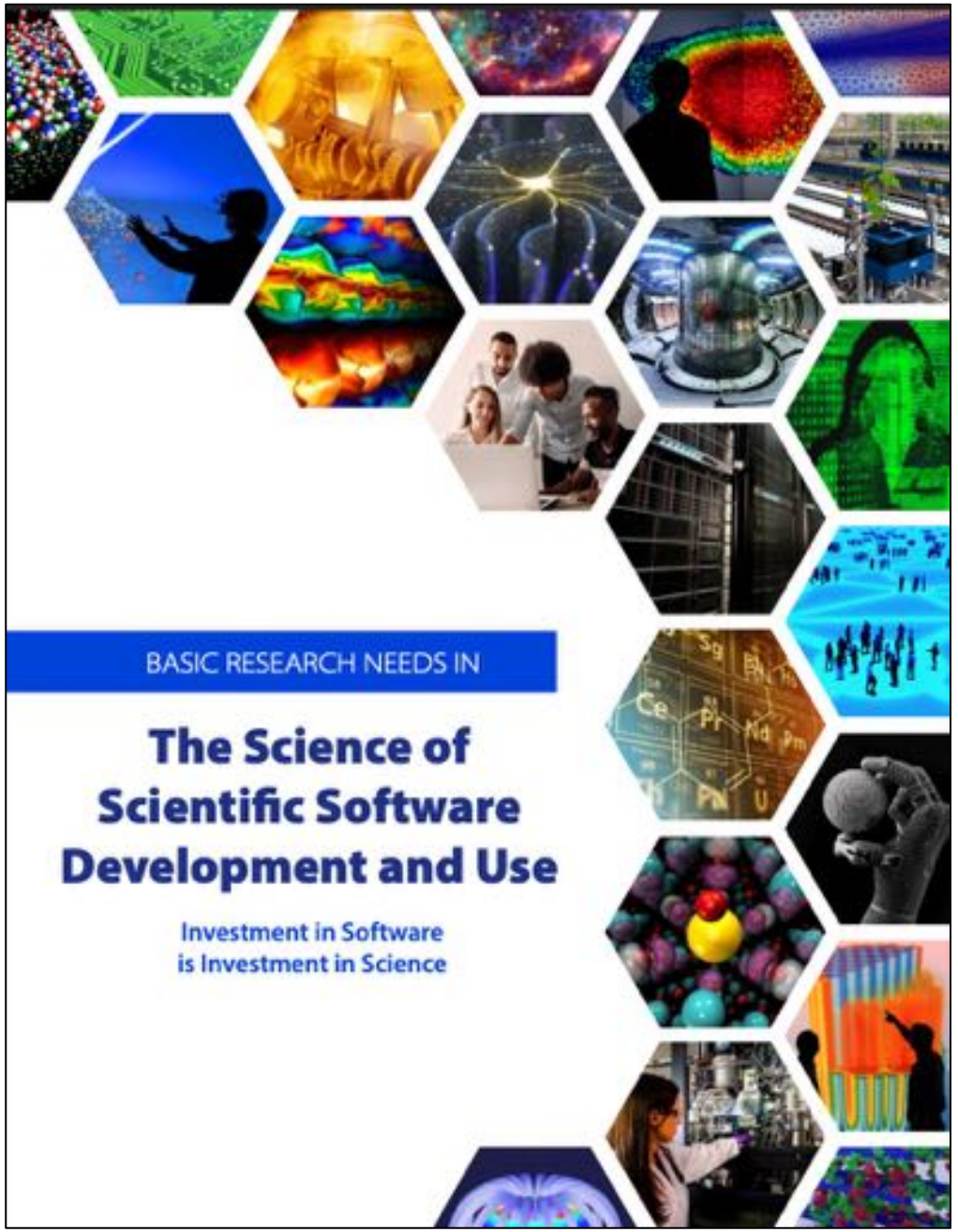
- Use scientific method to understand, improve development & use of software for research.
- **Incorporate cognitive & social sciences.**



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 Cross-cutting Themes



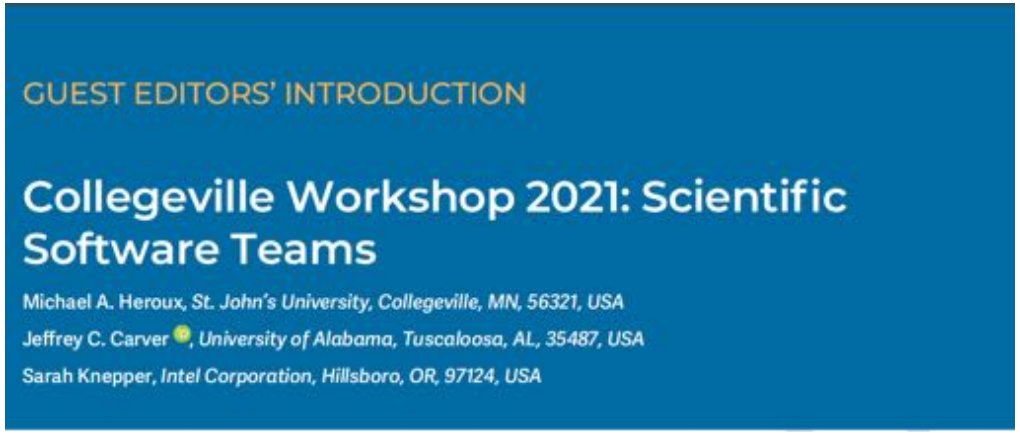
SSSDU Priority Research Directions

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

SSSDU Cross-cutting 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.

Special Issue: IEEE Computing in Science and Engineering, May/June 2022



- ***The PETSc Community as Infrastructure***
  Mark Adams, et. al.
- ***Challenges of and Opportunities for a Large Diverse Software Team***
  Cody J. Balos, et. al.
- ***Structured and Unstructured Teams for Research Software Development at the Netherlands eScience Center***
  Carlos Martinez-Ortiz, et. al.
- ***Experiences Integrating Interns into Research Software Teams***
  Jay Lofstead
- ***In Their Shoes: Persona-Based Approaches to Software Quality Practice Incentivization***
  M. R. Mundt, R. M. Milewicz, E. M. Raybourn

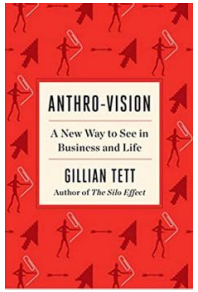
Collegeville Workshops on Scientific Software

- *Three Days:*
  - *Experiences and Challenges*
  - *Technical Approaches for Improvement*
  - *Cultural Approaches for Improvement*
- *Themes:*
  - *2019: Sustainability*
  - *2020: Productivity*
  - *2021: Teams*
  - *2022: Skip due pandemic workforce challenges*
  - *2023: Design*

Trends (I see) in Scientific Software that increase value of RSS

- AI-assisted development
  - Elevated thinking – intent to C++
  - Not unlike C++ to machine code
  - Fewer programmers? Maybe
  - Opportunity: More emphasis on purpose & design

- Deeper awareness of technology and society
  - Software systems adapted to fit scientists
  - Broaden usability, accessibility, impact



- UX applied to scientific software products
  - Personas & journey stories – not new
  - Applied to scientific software teams of developer-users – less common?
  - Just getting started

Opportunities for Trilinos 2022 - 2030

The ECP has demonstrated the potential of a sustained open-source software organization to:

- Deliver DOE ASCR R&D to users, facilities, vendors and the open-source community via a curated software portfolio
- Grow the next generation workforce
- Address growing reliance on software as first-class entity
- Raise the quality of the software we provide

Software platforms like GitHub, Spack, containers provide unprecedented opportunities to accelerate scientific progress:

- Tools and workflows enable rich collaboration
- E4S, the SDKs, and Spack binary caches will transform our ability to use rich software ecosystems
- Cloud environments will be major HPC resource

Next generation software teams need to include skills in cognitive and social sciences

- Many future challenges and opportunities for scientific progress are about people and technology
- As computational scientists we can appreciate the role of science to inform and improve how we develop and use software to do research

The path to HPC success is through execution on heterogeneous devices

- Solving the problem of utilizing multiple homogeneous GPU devices is just the first step
- ECP helps toward portability across multiple vendor GPU offerings, but there is so much more to come

The Trilinos team can be a leader among peers in establishing this organization

- Trilinos on top of Kokkos is well positioned to rapidly adapt to future emerging devices
- Trilinos team has deep knowledge and experience in key areas needed for organization success
- Trilinos team in a privileged position to explore the critical need for software quality assurance while “Working in Public”

Opportunities for Trilinos 2031 – 2040

Programming by intent
- GitHub CoPilot and Amazon CodeWhisperer
- Programmer coaches the AI to produce code
- Programmer focus is on requirements, design, test definitions

ML replacements for physics source code
- Lines of C++ replaced by Neural Network graph weights
- Key kernel: Matrix multiplication

Software design starts with thorough user-developer requirements gathering and analysis
- Leverage knowledge and skills from social and cognitive sciences
- Focus on making people most productive

Overall theme:
- Domain knowledge becomes increasingly important
- Mechanics of producing code become less important

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