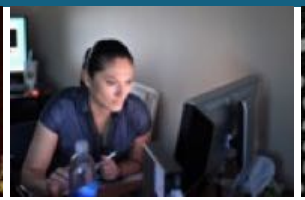
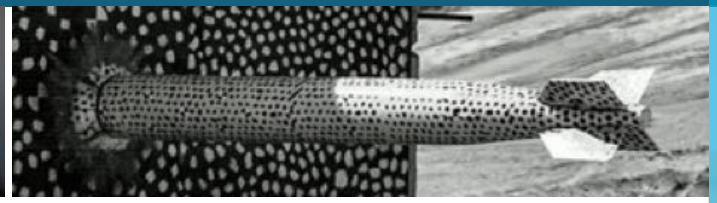
HPC Software Platform Trends:
The Evolution of Trilinos from 2001 to 2026





PRESENTED BY

Michael Heroux

SAND2021-15046 PE

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.

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.

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

Maturation Jumpstart: NewPackage

- NewPackage provides jump start to develop/integrate a new package
- NewPackage is a "Hello World" program and website:
  - Simple but it does work with autotools.
  - Compiles and builds.
- NewPackage directory contains:
  - Commonly used directory structure: src, test, doc, example, config.
  - Working Autoconf/Automake files.
  - Documentation templates (doxygen).
  - Working regression test setup.
  - Working Python and Thyra adaptors.
- Substantially cuts down on:
  - Time to integrate new package.
  - Variation in package integration details.
  - Development of website.

NOTE: NewPackage can be use independent from Trilinos

Sandia National Laboratories

Developer-Package Edges

A(i,j) = 1 if developer i contributes to package j

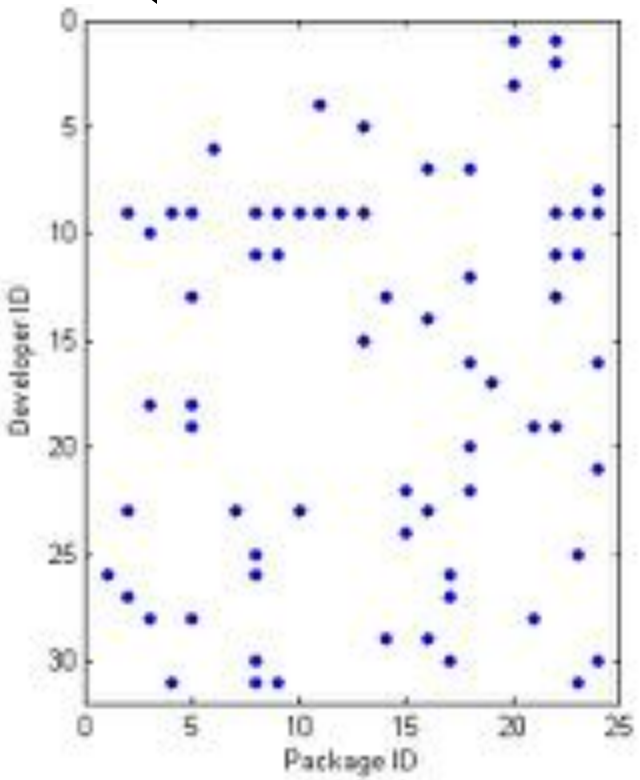
```
A = sparse(31,24);

A(RossBartlett,rythmos) = 1;
A(RossBartlett,thyra) = 1;
A(PaulBoggs,thyra) = 1;
A(ToddCoffey,rythmos) = 1;
A(JasonCross,jpetra) = 1;
A(DavidDay,komplex) = 1;
A(ClarkDohrmann,claps) = 1;
A(MichaelGee,ml) = 1;
A(MichaelGee,nox) = 1;
A(BobHeaphy,trilinosframework) = 1;
A(MikeHeroux,trilinosframework) = 1;
A(MikeHeroux,epetra) = 1;
A(MikeHeroux,aztecoo) = 1;
A(MikeHeroux,kokkos) = 1;
A(MikeHeroux,komplex) = 1;
A(MikeHeroux,ifpack) = 1;
A(MikeHeroux,thyra) = 1;
A(MikeHeroux,tpetra) = 1;
A(MikeHeroux,amesos) = 1;
A(MikeHeroux,belos) = 1;
A(MikeHeroux,epetraext) = 1;
A(MikeHeroux,jpetra) = 1;
A(UlrichHetmaniuk,anasazi) = 1;
A(RobHoekstra,epetra) = 1;
A(RobHoekstra,thyra) = 1;
A(RobHoekstra,tpetra) = 1;
A(RobHoekstra,epetraext) = 1;
A(RussellHooper,nox) = 1;
A(VickiHowle,meros) = 1;
A(VickiHowle,belos) = 1;
A(VickiHowle,thyra) = 1;
A(JonathanHu,ml) = 1;
A(SarahKnepper,komplex) = 1;
A(TammyKolda,nox) = 1;
A(TammyKolda,trilinosframework) = 1;
A(JoeKotulski,pliris) = 1;
A(RichLehoucq,anasazi) = 1;
A(RichLehoucq,belos) = 1;
A(KevinLong,thyra) = 1;
A(KevinLong,belos) = 1;
A(KevinLong,teuchos) = 1;
A(RogerPawlowski,nox) = 1;
A(MichaelPhenow,trilinosframework) = 1;
A(MichaelPhenow,trilinosframework) = 1;
A(EricPhipps,loca) = 1;
A(EricPhipps,nox) = 1;
A(MarzioSala,didasko) = 1;
A(MarzioSala,ifpack) = 1;
A(MarzioSala,ml) = 1;
A(MarzioSala,amesos) = 1;
A(AndrewSalinger,loca) = 1;
A(PaulSexton,epetra) = 1;
A(PaulSexton,tpetra) = 1;
A(BillSpotz,PyTrilinos) = 1;
A(BillSpotz,epetra) = 1;
A(BillSpotz,new_package) = 1;
A(KenStanley,amesos) = 1;
A(KenStanley,new_package) = 1;
A(HeidiThornquist,anasazi) = 1;
A(HeidiThornquist,belos) = 1;
A(HeidiThornquist,teuchos) = 1;
A(RayTuminaro,ml) = 1;
A(RayTuminaro,meros) = 1;
A(JimWillenbring,epetra) = 1;
A(JimWillenbring,new_package) = 1;
A(JimWillenbring,trilinosframework) = 1;
A(AlanWilliams,epetra) = 1;
A(AlanWilliams,epetraext) = 1;
A(AlanWilliams,aztecoo) = 1;
A(AlanWilliams,tpetra) = 1;
```

ACTS Tutorial 2005

spy(A);

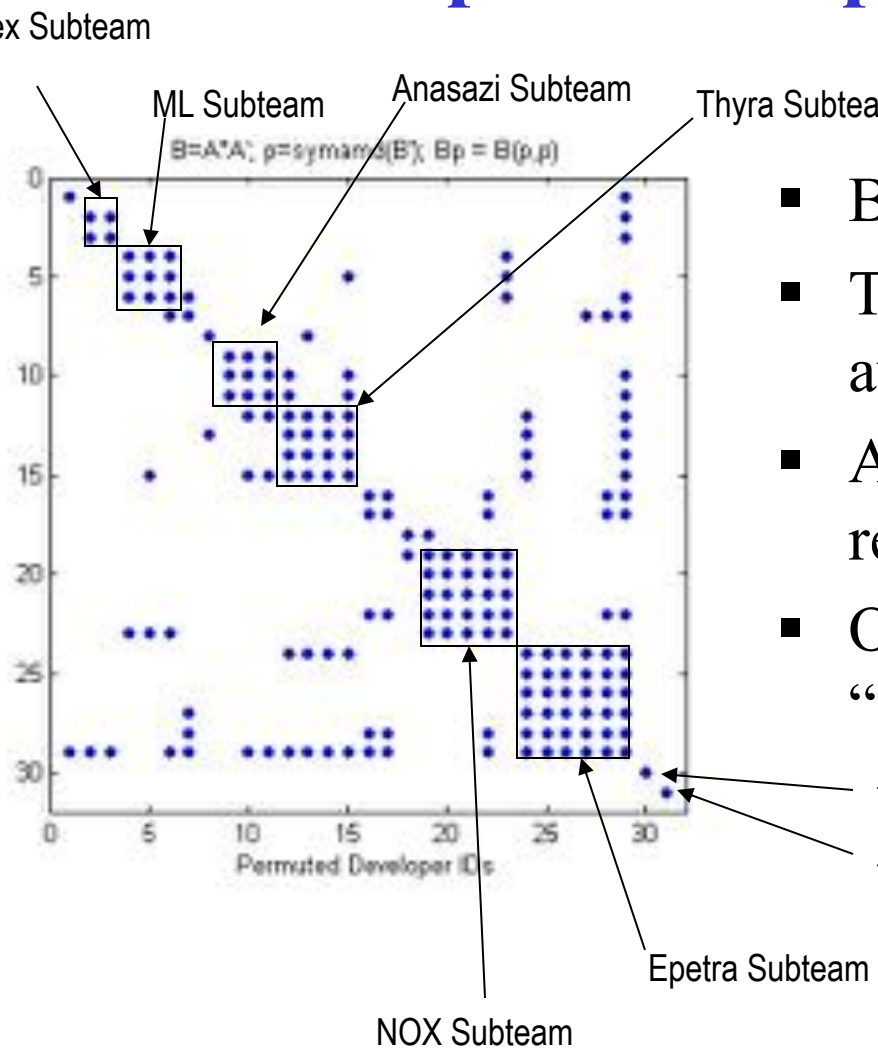
Developer-Package Matrix



- Number of developers per package:
  - Maximum: max(sum(A)) = 6
  - Average: sum(sum(A))/24 = 2.875
- Number of package affiliations per developer:
  - Maximum: max(sum(A')) = 12.
    - Minus outlier: = 4.
  - Average: sum(sum(A'))/31 = 2.26
- Observations:
  - Several developers per package.
  - Several packages per developer.

Sandia National Laboratories

Developer-Developer Matrix (A*A')



- B = A*A'
- Two developers connected if co-authors of a package.
- Apply Symmetric AMD reordering.
- Only two developers "disconnected":
  - Clark Dohrmann: CLAPS.
  - Joe Kotulski: Pliris.

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)

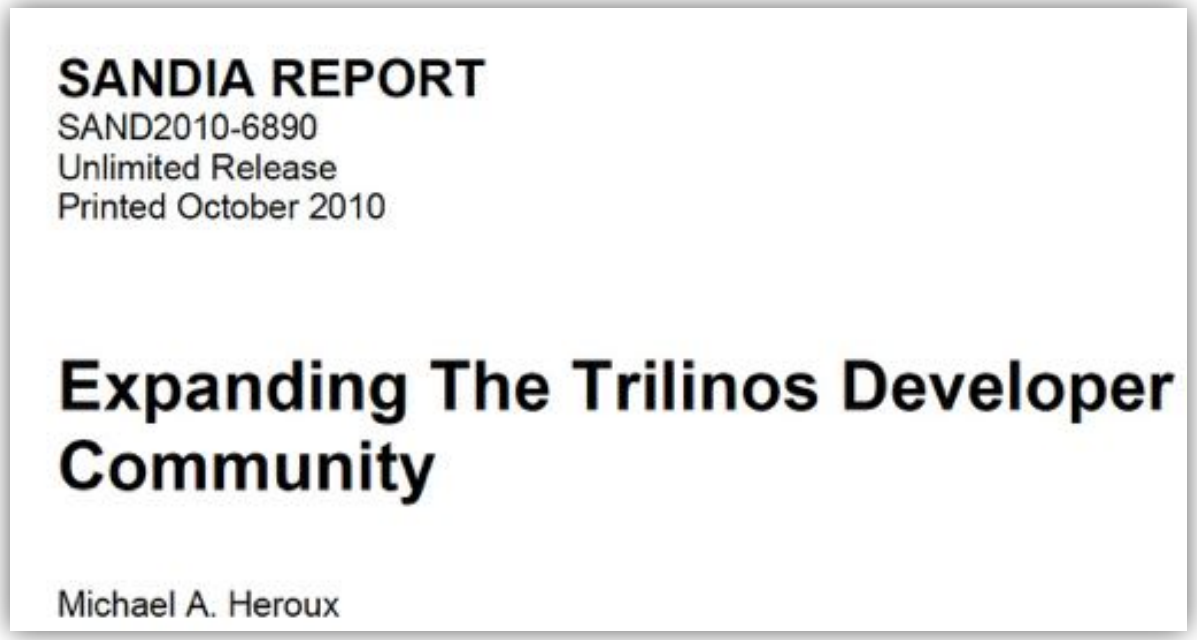
Many of these attributes have modern replacements:

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

Some have dropped (not always for the best)

- Use of APIs for inter-package interactions
- Kickstart for new package

Expanding the Trilinos Developer Community



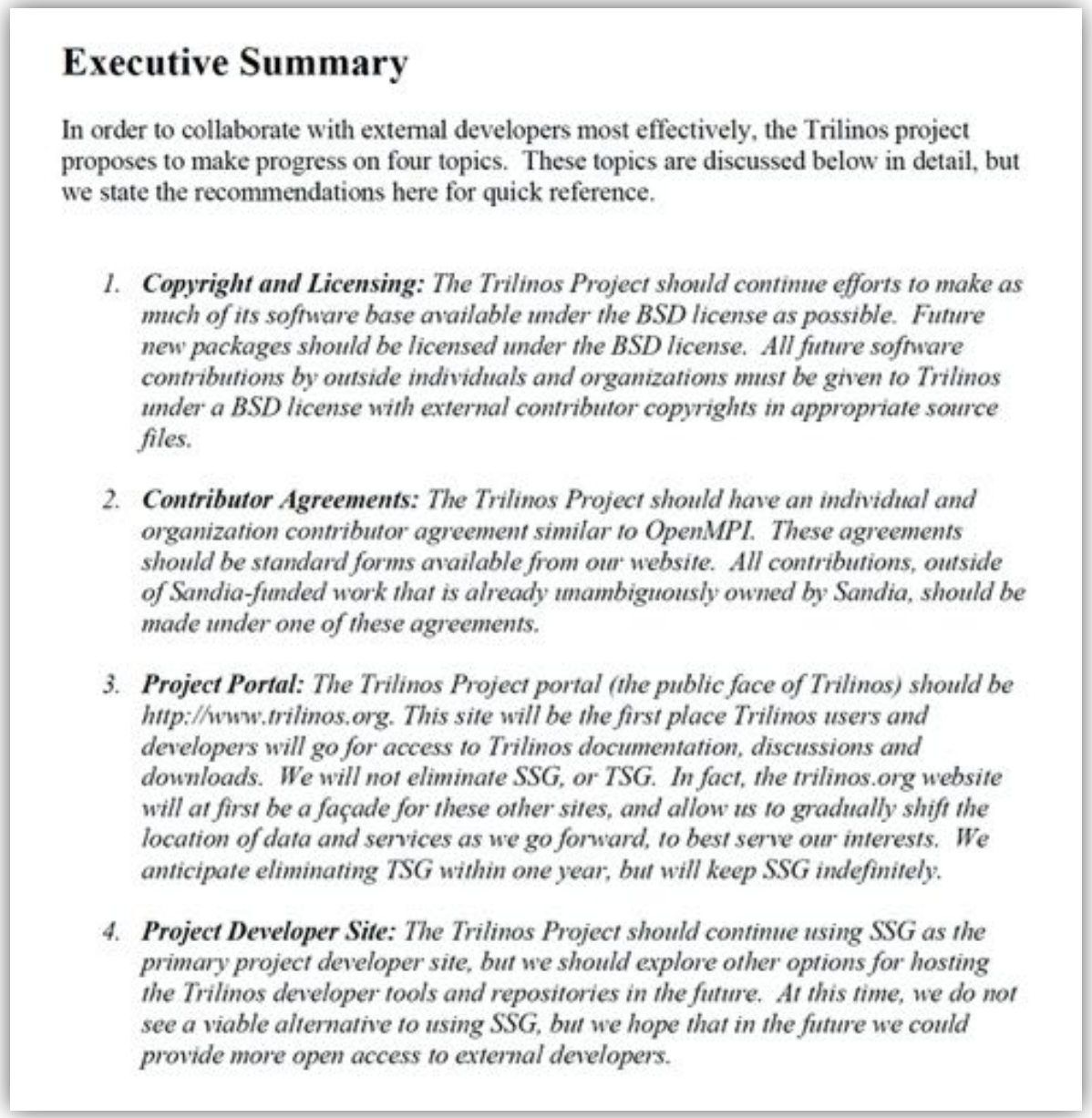
SANDIA REPORT
SAND2010-6890
Unlimited Release
Printed October 2010

Expanding The Trilinos Developer Community

Michael A. Heroux

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



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.*

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

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

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

Changing HPC Landscape and Need for Performance Portability

2021 Alphabet Talk S. Rajamanickam

Several many/multi-core architecture central to DOE/NNSA HPC

Intel Multicore


NVIDIA GPU

IBM Power

Intel Manycore

AMD Multicore/APU

ARM

2012	2016	2018	2021
IBM BGQ (Sequoia, Mira)	Intel KNL (Trinity, Cori)	NVIDIA Volta (Summit, Sierra)	Intel A21
NVIDIA Kepler (Titan)		ARM (Astra)	AMD GPU
			NVIDIA GPU

Decade of DOE HPC will have seen 4-5 "new" paradigms!

- **Several architectures, many with different programming models**
- **Applications struggle to obtain good performance on all of these**

Approaches to Programming GPUs

Native Programming Models

- CUDA (NVIDIA), HIP (AMD), SYCL (Intel)
- Pros: Customized for each architecture, so low level control
- Cons: Rewrite code every time you buy a hardware from a new vendor

Directive Based Approach

- OpenMP, OpenACC
- Pros: Standards based, General
- Cons: Long lag time between what is needed and when they are needed, Might have to resort to #ifdef after all, Different level of support from vendors

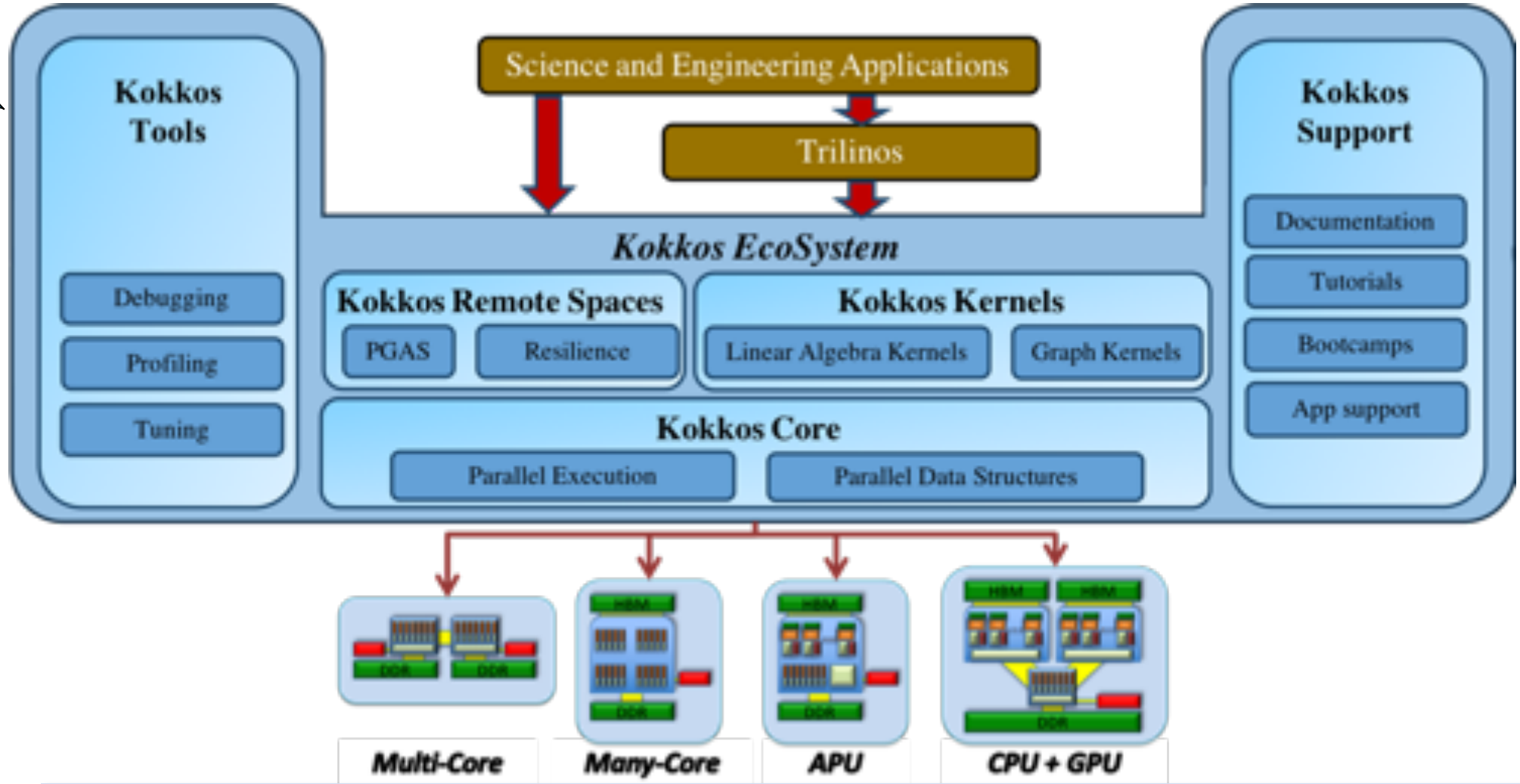
Library Based Approach

- Kokkos, RAJA
- Pros: Portable, Clean abstractions, Quicker turnaround, Reference implementations of standards
- Cons: Dependency on libraries

Library based performance portability allows for writing applications to several architectures with limited dependencies

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

New Features in Kokkos Kernels 3.X

2021 Alphabet Talk S. Rajamanickam

Sparse Linear Algebra
- ✓Cluster Gauss-Seidel
- ✓ Sparse ILU factorization
- ✓ Sparse triangular solves for sparse L and U
- ✓ Sparse triangular solves for supernodal L and U
- ✓ Structured sparse matrix vector multiply
- ✓ Cluster Gauss Seidel

Dense Linear Algebra
- ✓ Faster kernels for orthogonalization
- ✓ Complex support for dense LU factorization
- ✓Interfaces to vendor libraries
- ✓More BLAS and LAPACK support with Kokkos views

Graph Algorithms
- ✓Distance-2 graph coloring
- ✓Faster distance-1 graph coloring
- ✓Balanced distance-1 coloring
- ✓Balanced “well shaped” graph clustering
- ✓RCM ordering for preconditioners
- ✓ MIS-2 and Coarsening

Portable Vectorization
- ✓ Support ARM platforms
- ✓ Improved application performance on CPU, KNL, GPU and ARM
- ✓ Portable SIMD primitive

Team Level Kernels
- ✓ Team level sorting utilities
- ✓ Team level DFS
- ✓ More team level BLAS and LAPACK support

Software
- ✓CMake support
- ✓ETI changes to allow ETI file generation at compile time

Kokkos Kernels is rapidly growing to support the needs of computational science applications.

Trilinos: Open-Source Toolkit of Mathematical Algorithms for HPC

Trilinos Software

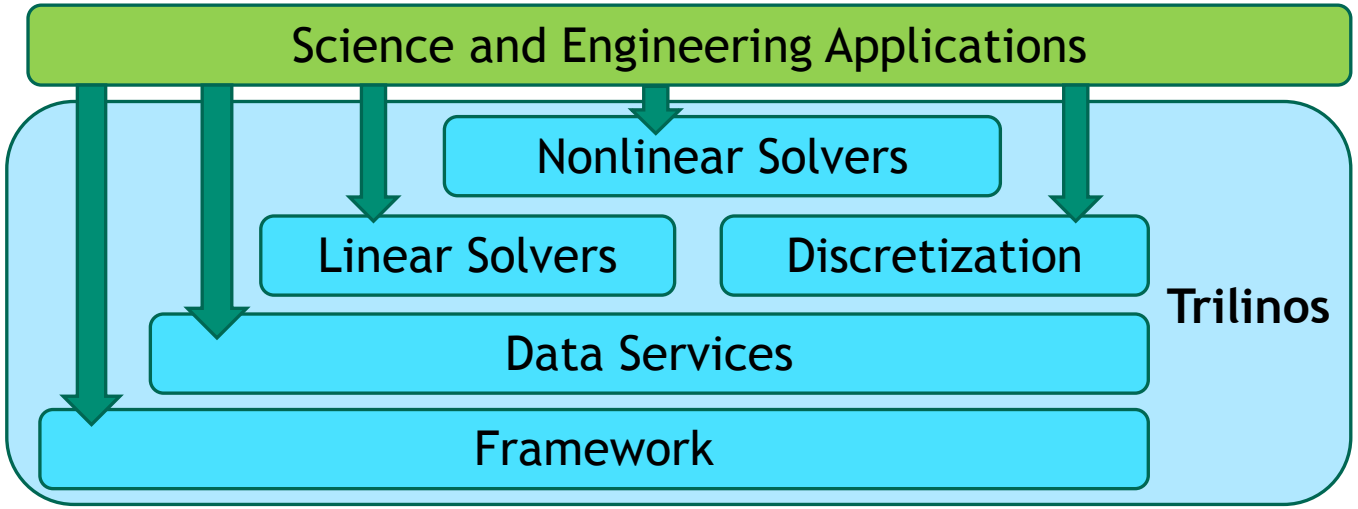
55 packages in five areas

~100 contributors in total

~50+ active contributors

30-140 commits per week

400 forks



Application Impact

- Solid mechanics, fluid dynamics, electrical circuits, etc.
- SIERRA, Empire, SPARC, Xyce, Drekar, Charon, etc.

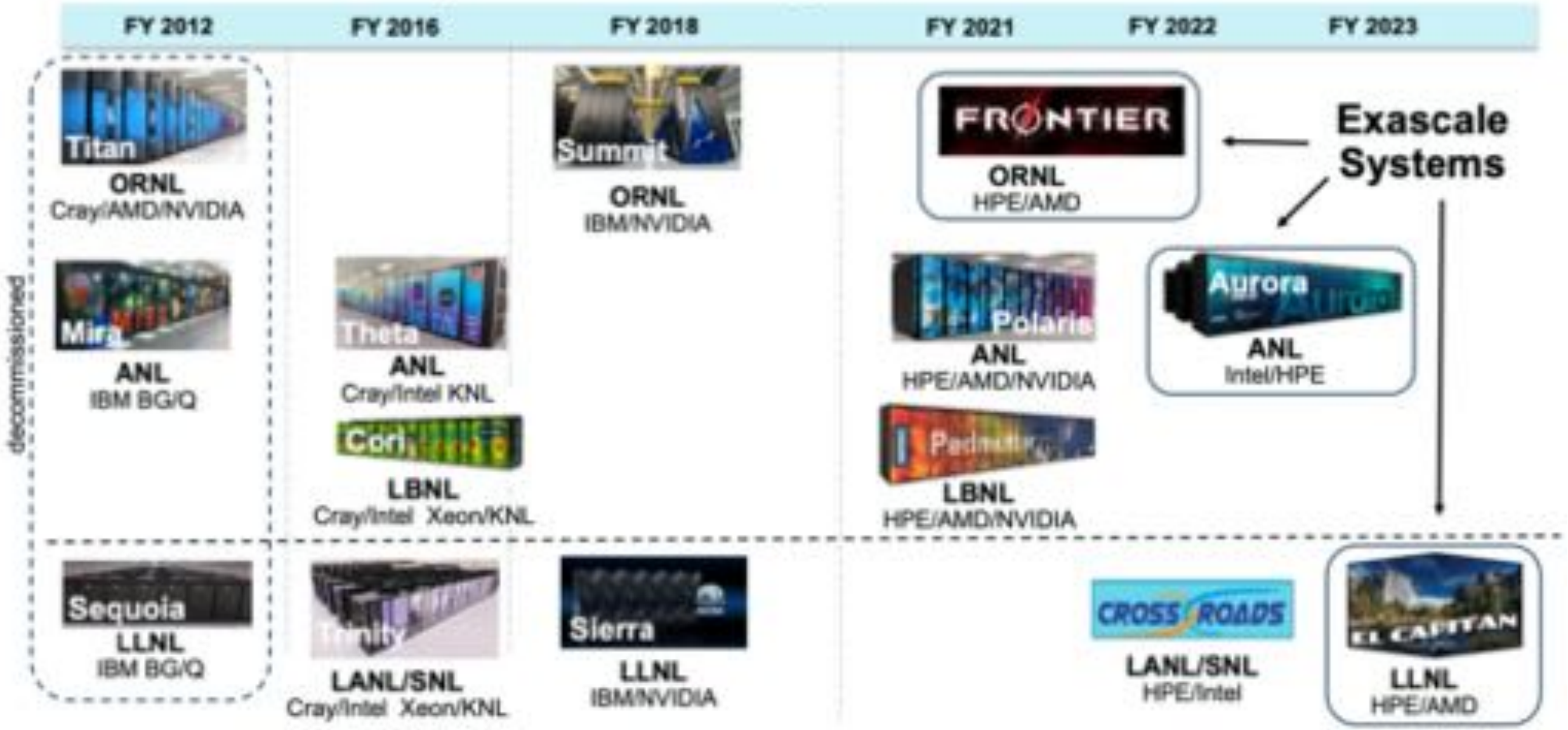
Two main codes paths:
- 32-bit stack (maintenance)
- Templated C++ stack (active)

Trilinos product areas (Lead : Heroux)

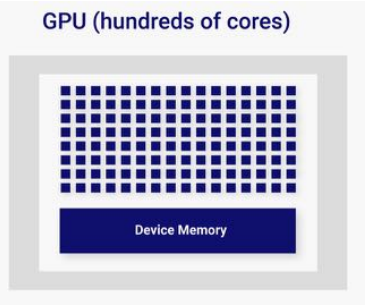
- **Framework** – Build, install, and test infrastructure; application integration (Product Lead: Willenbring)
- **Data Services** – Linear algebra, **Kokkos** performance-portability, **load balancing**, mesh services (Product Lead: Devine)
- **Linear Solvers** – Iterative/direct solvers, preconditioners (domain-decomposition, multigrid, block) (Product Lead: Rajamanickam)
- **Nonlinear Solvers** – Time-stepping methods, non-linear solvers (Product Lead: Pawlowski)
- **Discretization** – Matrix assembly, discretization support (Product Lead: Perego)

Trilinos provides scalable algorithms to ASC-IC/ATDM applications, enabling high performance on current and next generation HPC platforms

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

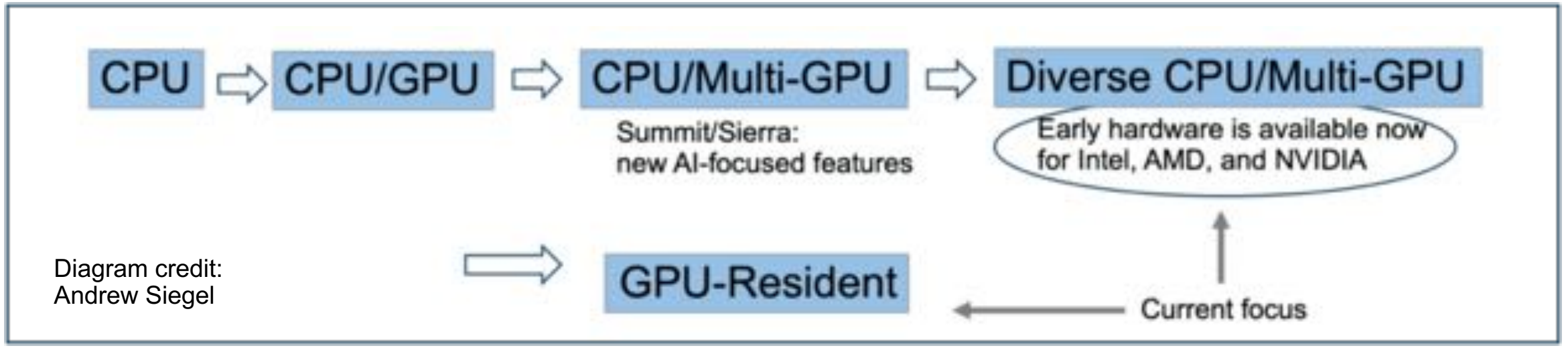


Diagram credit:
Andrew Siegel

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

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?

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

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)	Balaji, Pavan	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	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	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, Mike	Grundhoffer, Alicia
2.3.3.15	Sake: Scalable Algorithms 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	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	Mohror, Kathryn	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	Willenbring, James
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

- ~250 staff
- ~70 products
- 34 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

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

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 a task: Provide capabilities for next-generation leadership platforms
  - Going together into the frontier is better than going alone

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

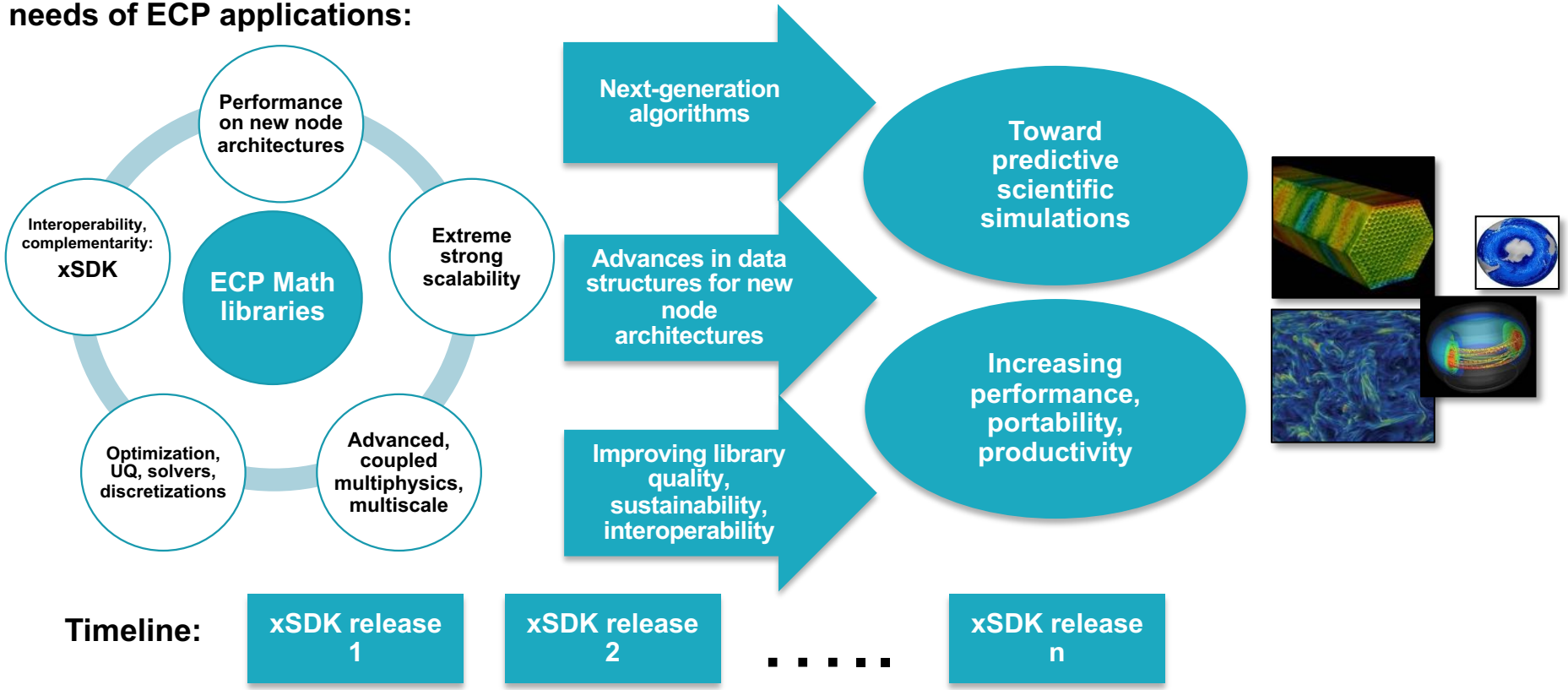
xSDK: Primary delivery mechanism for ECP math libraries' continual advancements


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

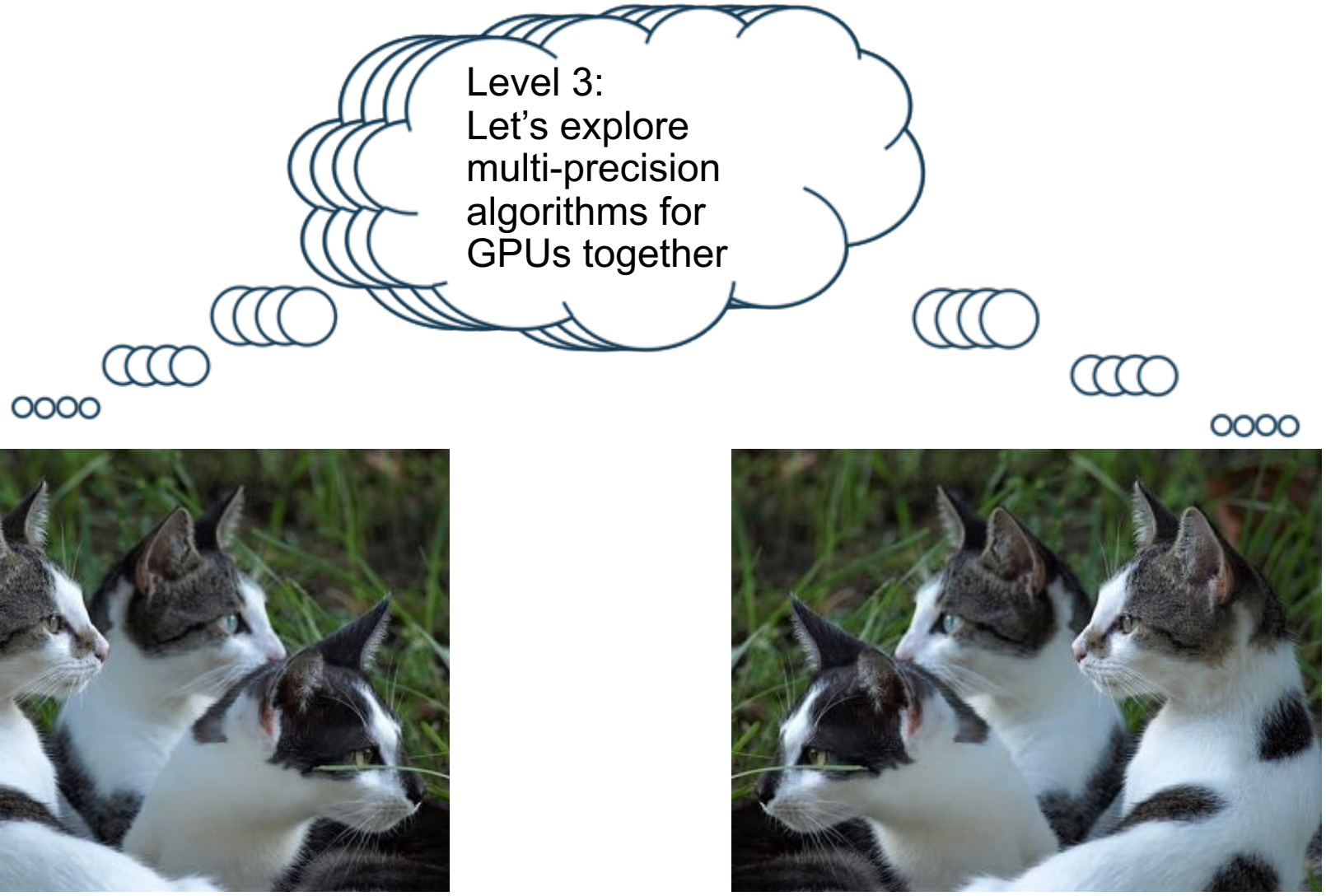
As motivated and validated by the needs of ECP applications:



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)



ECP xSDK Mixed/Multi-precision Initiative:
Pitch talks Nov 2021 – May 2020

- Thomas Grützmacher (KIT) Are Posits an option for reducing the memory access volume?
- Stan Tomov (UTK) Low Precision BLAS on AMD GPUs
- Natalie Beams (UTK) A mixed-precision future in libCEED
- Thomas Grützmacher (KIT) Status update on the memory accessor design and interface
- Toshiyuki Imamura (RIKEN) Mixed Precision Numerics and HPL-AI on the Fugaku Supercomputer
- Nico Trost (AMD) ROCm™ Low Precision Capabilities
- James Diffenderfer (LLNL) QDOT: Quantized Dot Product Kernel for Approximate High-Performance Computing
- Mantas Mikaitis (University of Manchester) Numerical Behavior of NVIDIA Tensor Cores
- Azzam Haider (Nvidia) Tensor Core Accelerated Iterative Refinement Solvers and its impact on scientific computing
- Sebastien Cayrols (UTK) Design and optimization of mpi_alltoall for mixed-precision FFT algorithms
- **Ichitaro Yamazaki (SNL) Mixed precision s-step Lanczos and CG**
- Hartwig Anzt (KIT) Pushing the memory roofline with the accessor
- Daniel Osei-Kuffuor (LLNL) Towards a Multi-Precision Linear Solver Library in hypre
- Tim Kelley (NCSU) Newton's Method in Mixed Precision
- Theo Mary (Sorbonne University) Mixed Precision Low Rank Compression and its Application to BLR Matrix Factorization
- Erin Carson (Charles University) Using Mixed Precision in s-step Krylov Subspace Methods
- Sherry Li (LBNL) Performance and accuracy of sparse direct solver with mixed precision arithmetic on GPU
- Stephen Thomas (NREL) The Mathematics of Arrays (MoA) for Fast Matrix Algorithms on Exascale Architectures
- Tobias Ribizel (KIT) Implementing Mixed Precision Operations in Ginkgo
- Hartwig Anzt (KIT,UTK) Preparing for the Multiprecision ECP Review
- **Jennifer Loe (Sandia National Laboratories) Multiprecision Krylov Solvers in Trilinos**
- Mike Tsai (University of Tennessee) Mixed-precision algorithm for finding selected eigenvalues and eigenvectors of symmetric and Hermitian matrices
- Erin Carson (Charles University) Mixed Precision Lanczos-CG
- Andres Tomas (University of Jaume I) Balanced and Compressed Coordinate Layout for the Sparse Matrix-Vector Product on GPUs
- Yu Pei, George Bosilca (Tennessee) Accelerating Geostatistical Modeling and Prediction With Mixed-Precision Computations: A High-Productivity Approach with Parsec
- Thomas Grützmacher (KIT) Memory Accessor Design
- Srikara Pranesh (University of Manchester) Answering two numerical linear algebra questions with the help of fp16
- Sebastien Cayrols (UTK) Design and optimization of MPI_Alltoall for Mixed-precision algorithms
- Piotr Luszczek, Mike Tsai, Jack Dongarra (UTK) Towards LU Factorization Based on Integer Arithmetic with Floating-Point Accuracy
- Thomas Grützmacher (KIT) Compressed Basis GMRES on High Performance GPUs
- Steve Thomas and Kasia Swirydowicz (NREL) Mixed Precision FGMRES Iterative Refinement for Large Sparse Indefinite A=LDL^T
- Rasmus Tamstorf (Disney Research) Mixed Precision Multigrid
- Fritz Göbel (KIT) Multiprecision block-Jacobi for Iterative Triangular Solves

2021 Mixed/Multi-precision Progress Report

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

Preface

Over the last year, the ECP xSDK-multiprecision effort has made tremendous progress in developing and deploying new mixed precision technology and customizing the algorithms for the hardware deployed in the ECP flagship supercomputers. The effort also has succeeded in creating a cross-laboratory community of scientists interested in mixed precision technology and now working together in deploying this technology for ECP applications. In this report, we highlight some of the most promising and impactful achievements of the last year. Among the highlights we present are

- Mixed precision IR using a dense LU factorization and achieving a 1.8× speedup on Spock;
- Results and strategies for mixed precision IR using a sparse LU factorization;
- A mixed precision eigenvalue solver;
- Mixed Precision GMRES-IR being deployed in Trilinos, and achieving a speedup of 1.4× over standard GMRES;
- Compressed Basis (CB) GMRES being deployed in Ginkgo and achieving an average 1.4× speedup over standard GMRES;
- Preparing hypre for mixed precision execution;
- Mixed precision sparse approximate inverse preconditioners achieving an average speedup of 1.2×;
- Detailed description of the memory accessor separating the arithmetic precision from the memory precision, and enabling memory-bound low precision BLAS 1/2 operations to increase the accuracy by using high precision in the computations without degrading the performance;

We emphasize that many of the highlights presented here have also been submitted to peer-reviewed journals or established conferences, and are under peer-review or have already been published.

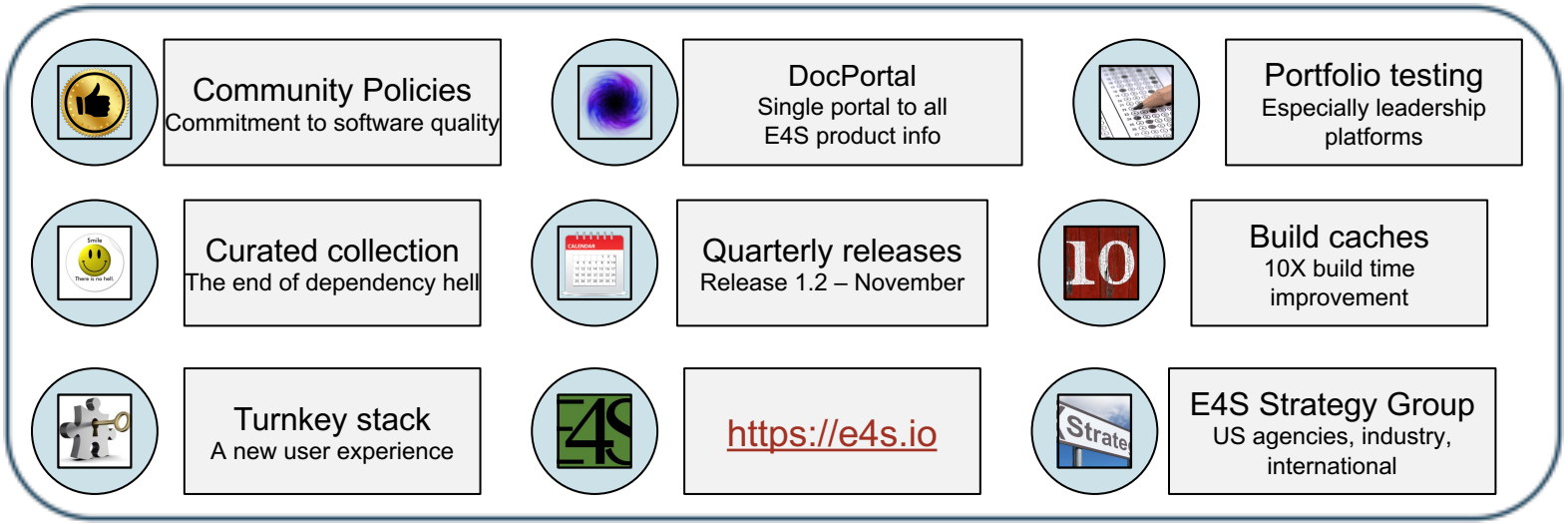
- Cross-team design space exploration
- Rapid info creation and dissemination
- Integration of ideas into all products
- Example of coop-etition

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: Nov 2021: E4S 21.11 – 91 full release products



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 Community Policies: *A commitment to quality improvement*

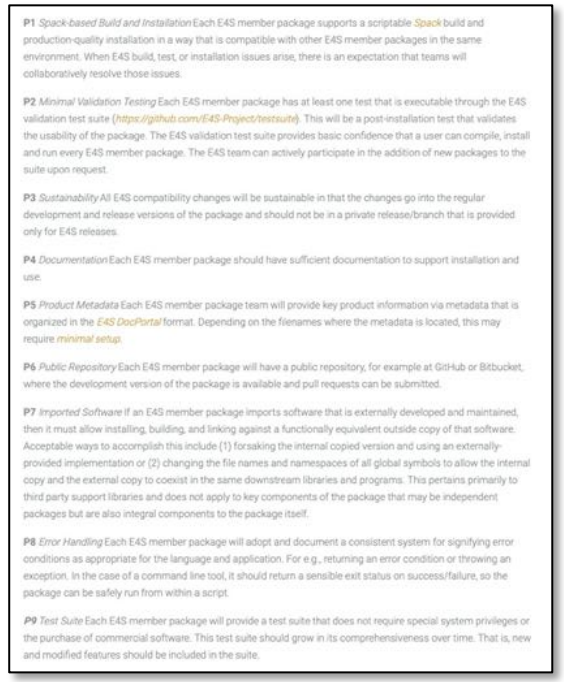
- Purpose: Enhance sustainability and interoperability
- Will serve as membership criteria for E4S
  - Membership is not required for *inclusion* in E4S
  - Also includes forward-looking draft policies
- Modeled after xSDK community policies
- Multi-year effort led by SDK team
  - Included representation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

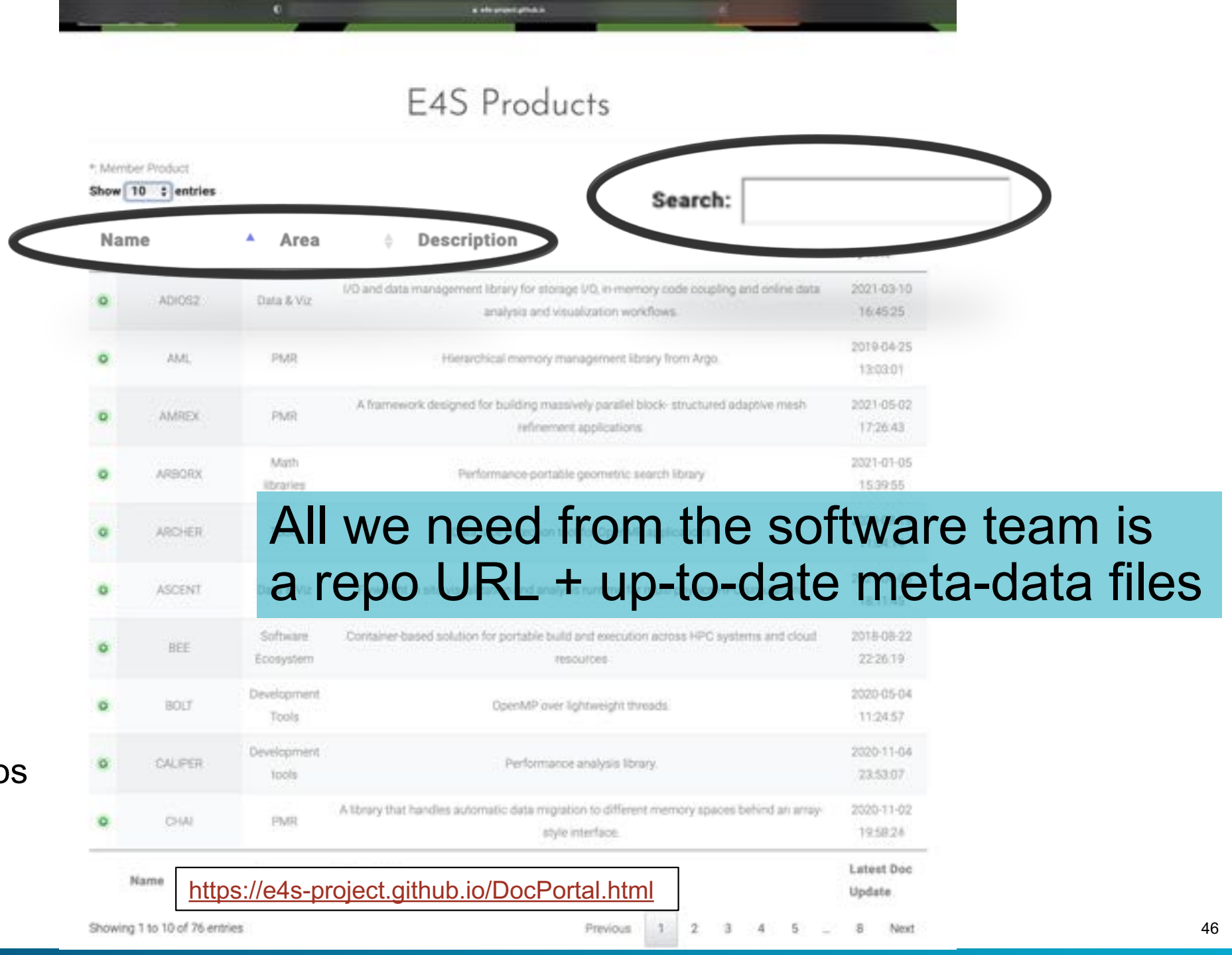
- **P1: *Spack-based Build and Installation***
- **P2: *Minimal Validation Testing***
- **P3: *Sustainability***
- **P4: *Documentation***
- **P5: *Product Metadata***
- **P6: *Public Repository***
- **P7: *Imported Software***
- **P8: *Error Handling***
- **P9: *Test Suite***



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

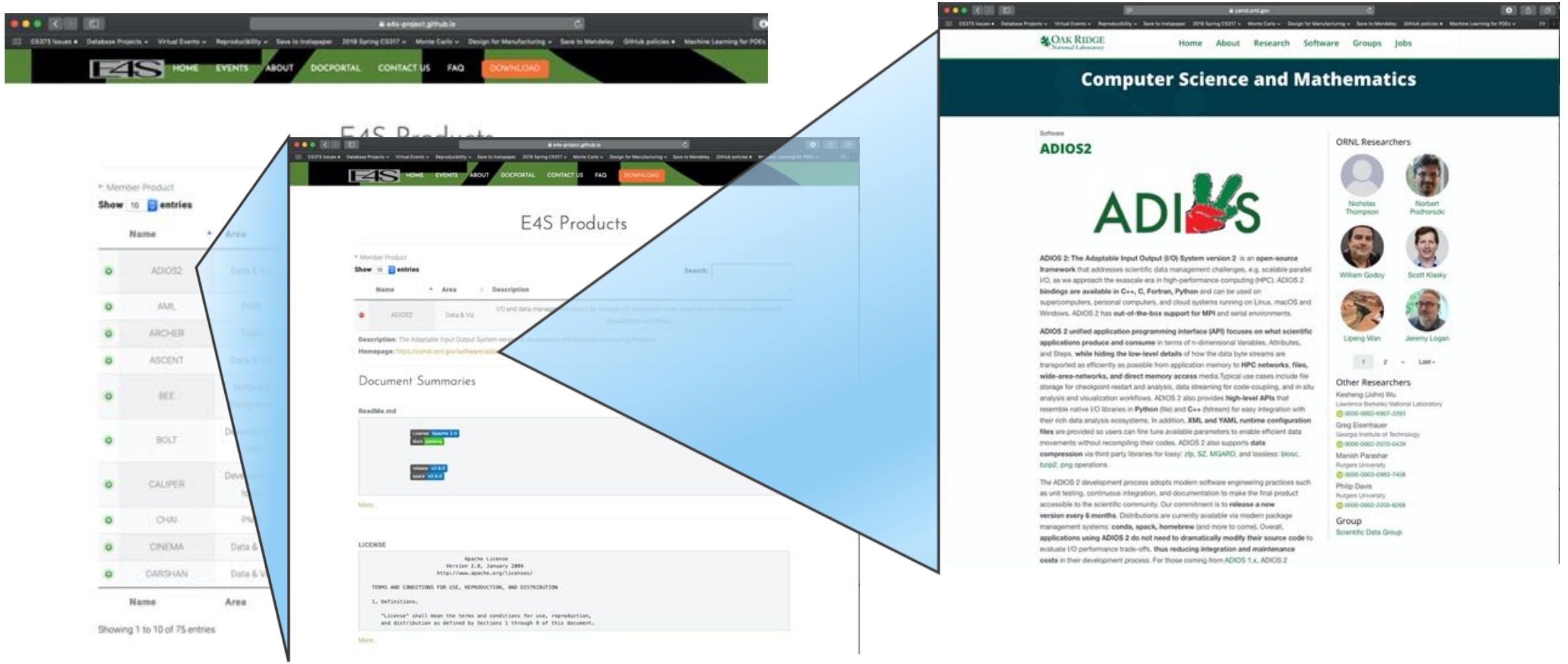
E4S DocPortal

- 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

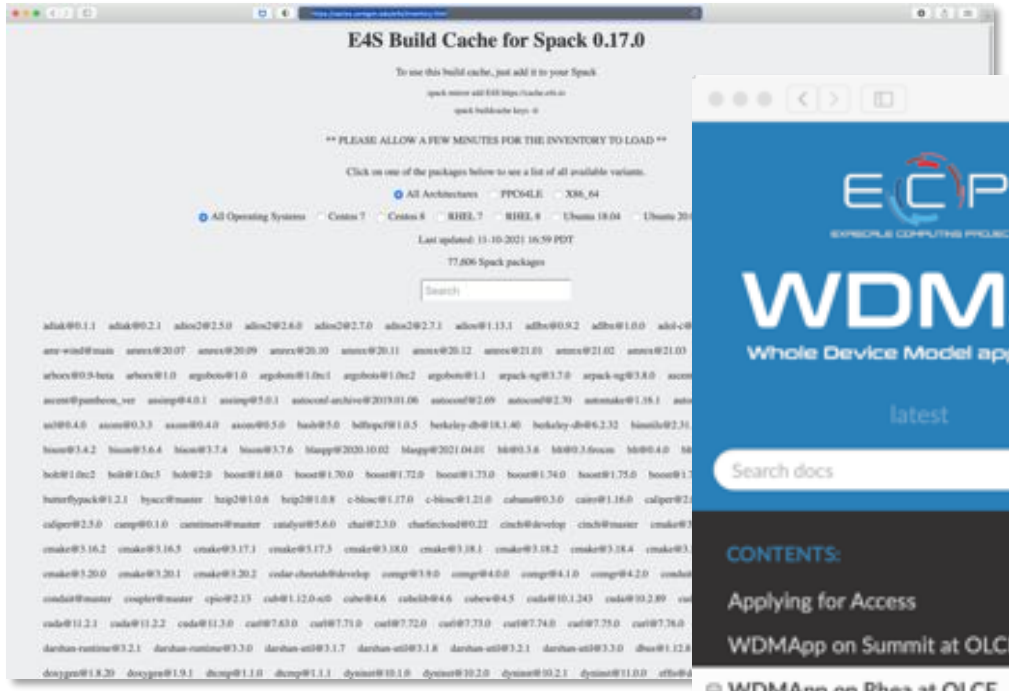
Goal: All E4S product documentation accessible from single portal on E4S.io (working mock webpage below)



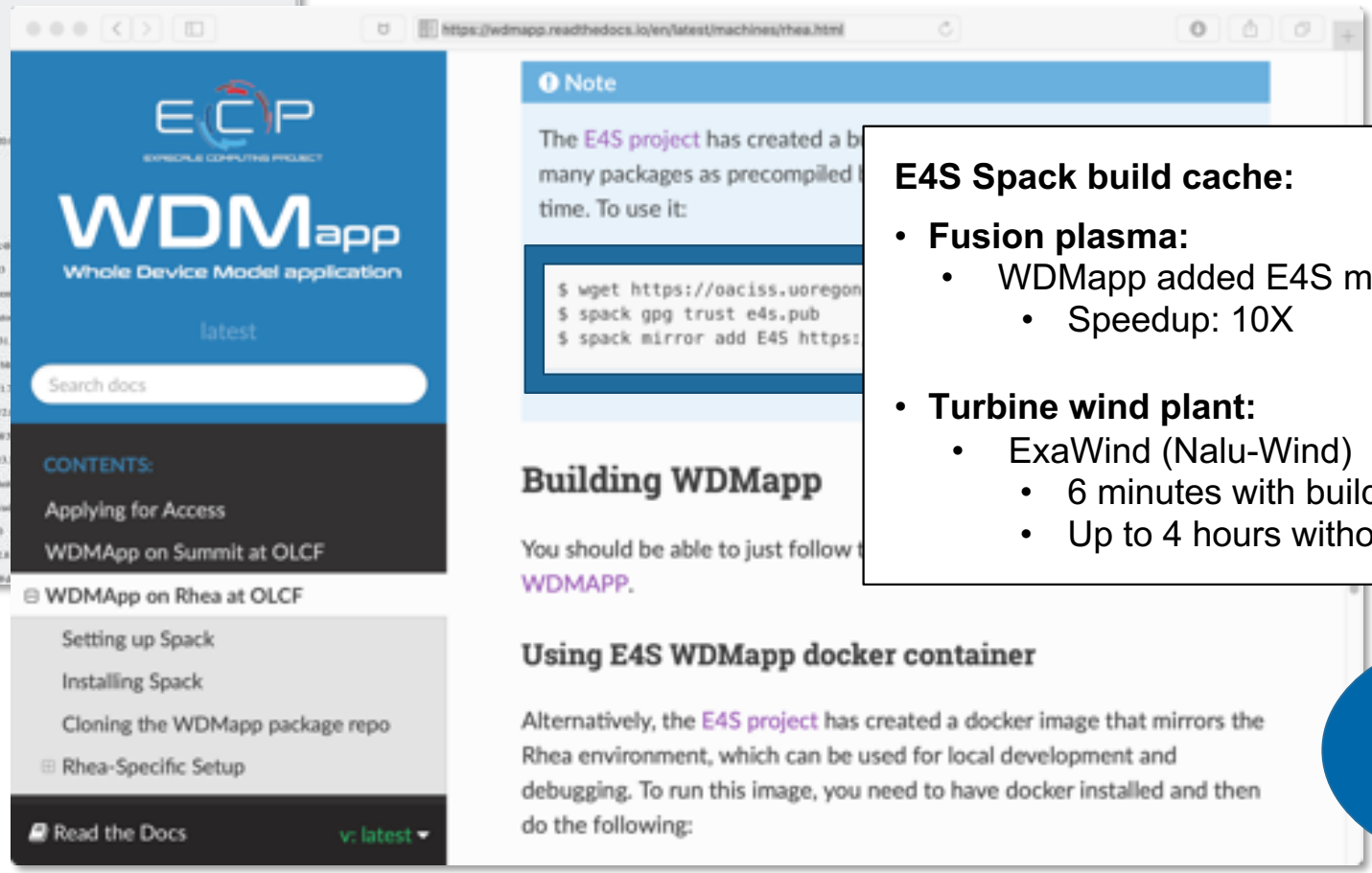
https://e4s-project.github.io/DocPortal.html

Speeding up bare-metal installs using the E4S build cache

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

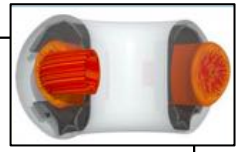


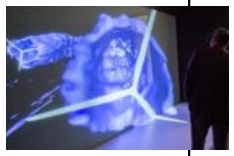
- 75,000+ binaries
- S3 mirror
- No need to build from source code!



E4S Spack build cache:

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





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

IDEAS productivity

https://ideas-productivity.org

Advancing scientific productivity through better scientific software

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

1 Customize and curate methodologies

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

2 Incrementally and iteratively improve software practices

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

3 Establish software communities

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

4 Engage in community outreach

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

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 funding is not ECP-related
- Benefits 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 include
  - Overheads of synchronizing, coordinating
  - Complications from need for collaborative open-source development and mission security needs

Toward a DOE ASCR Leadership Software Center (LSC)

Transforming ASCR Science R&D into World-class Leadership Software

Background

- The US DOE Exascale Computing Project (ECP) initiated the Extreme-scale Scientific Software Stack (E4S)
- E4S development will continue under ECP for two more years
- To better ensure continued growth and sustainability beyond ECP, we are exploring ideas now to better orient E4S efforts toward the post-ECP era
- Engaging key US agencies and international institutions is essential to the longevity of E4S
- We propose a plan for
  - A DOE ASCR Leadership* Software Center (LSC)
  - A leadership and stewardship role in sustaining and growing E4S through LSC

*We intend leadership in our setting to mean emerging and leading-edge software for emerging and leading-edge scientific computing environments, including HPC, AI/ML for science, large-scale edge computing for science, quantum, and other scientific computing software products that complement industry efforts and facilitate scientific progress.

LSC Motivation

- ECP makes a compelling case for coordinated development and delivery of DOE software products
  - **Planning:** Portfolio of inter-related capabilities in collaboration with application teams, facilities, vendors, open-source communities
  - **Execution:** Development and dissemination of best practices; use of shared platforms (e.g., Atlassian tools), testing infrastructure, effective and efficient processes
  - **Tracking:** Coordinated and transparent progress tracking, adaptation to evolving requirements
  - **Assessment:** Regular assessment and reporting of progress to stakeholders and community
- The ECP ST Portfolio approach promises improved effectiveness and efficiency of DOE software efforts vs independent software teams working alone
- The E4S/SDK open software architecture provides a framework for successful software development and delivery
- ECP has fostered a holistic approach to scientific software workforce development
- **A Leadership Software Center (LSC) provides a compelling approach as an enabler to coordinate the development and delivery of DOE software products after the end of ECP**

LSC Sketch

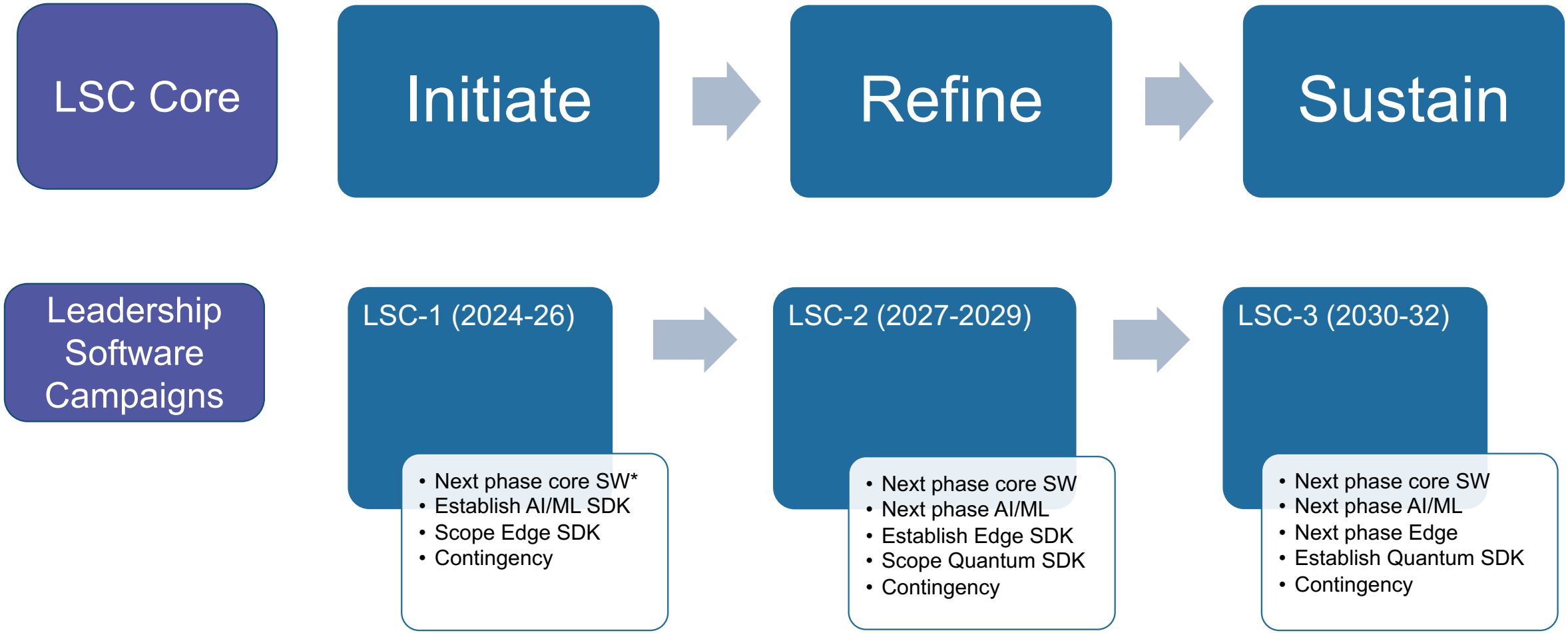
ECP Sustainability: The Leadership Software Center (LSC) will enable the sustainability of ECP contributions, and development and delivery of future capabilities, including new domains like AI/ML, Edge and Quantum

Tailored Agile: The LSC will use tailored project management practices, processes, tools, and a distributed multi-institutional organization to enable effective and efficient delivery of ASCR software investments

New Ecosystem Entity: The LSC will establish an essential and new ecosystem entity to complement Facilities, ASCR Research, vendors, industry and other entities.

Workforce Development: Establishing the LSC assures the creation of a scientific software workforce for sustainable leadership scientific software development and delivery

Leadership Software Center Cadence
Ongoing + Campaigns



*Next phase core SW: Scope necessary to address emerging needs in programming models, runtimes, tools, math libs, data, visualization, workflows and other established software technologies. Often this scope will be new features in existing LSC products, such as AI linear algebra features being added, or support for the latest AI devices, or both, to our existing Math Libs SDK.

Portfolio Project Management

Plan, Execute, Track, Assess Lifecycle

- All activities governed by phased development process
- Executed as “campaigns”
- Tailored agile approach
- Collective coordination, first-class HPC ecosystem entity
- Hierarchical approach:
  - Multi-year baseline as campaign
  - Refine annually
  - Add milestone fidelity at “last responsible moment”

Change Management Process:

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

Capability Integration Strategy

DOE software products have four primary integration targets:

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

Project goals:

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

Current Activities & Next Steps

- Software Sustainability Strategic Plan:
  - ECP ST & Co-design leadership draft document
  - Part of response to Feb 2021 IPR recommendation
  - Target 20 – 30 pages
  - Current draft provided much of the content for these slides
  - Developing concurrently with community outreach
- Leadership Scientific Software Sustainability Town Halls:
  - Themed discussions led by ECP ST leads, E4S leads
  - 3rd Thursday of the month, 3 – 4:30 pm ET
  - Broadest possible public engagement
  - https://lssw.io
- DOE ASCR Request for Information (RFI) on Software Sustainability
  - https://www.federalregister.gov/documents/2021/10/29/2021-23582/stewardship-of-software-for-scientific-and-high-performance-computing
  - RFI responses due Dec 13, 2021

Opportunities for Trilinos 2022 - 2026

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
- Example: Richard McElreath "Science as Amateur Software Development" https://youtu.be/zwRdO9_GGhY

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"
- Challenge: We must learn how to be fully part of future DOE open science software efforts while also addressing DOE mission needs

Working toward software sustainability: Join the conversation

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

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

- LSSw Town Hall Meetings (ongoing)
  - 3rd Thursday each month, 3 – 4:30 pm Eastern US time
  - Next meeting Dec 16, topic: Leadership SW beyond HPC
- Slack: Share your ideas interactively
- Whitepapers: Written content for LSSw conversations
  - We need your ideas (2-4 page whitepapers)
  - Submit via GitHub PR or attachment to contribute@lssw.io
- References
  - Help us build a reading list
  - Submit via GitHub PR or email to contribute@lssw.io

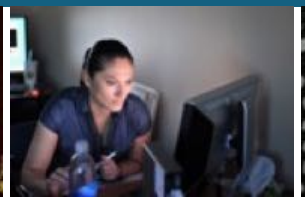
Workshop on Research Software Science

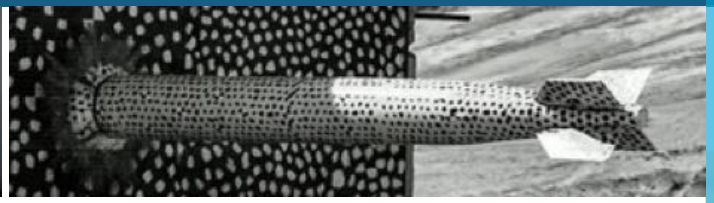
Software is an increasingly important component in the pursuit of scientific discovery. Both its development and use are essential activities for many scientific teams. At the same time, very little scientific study has been conducted to understand, characterize, and improve the development and use of software for science.

- Info and registration at: https://www.orau.gov/SSSDU2021
- Whitepapers: 120+ submissions

RFI: Stewardship of Software for Scientific and High-Performance Computing, deadline Dec 13
https://www.federalregister.gov/documents/2021/10/29/2021-23582/stewardship-of-software-for-scientific-and-high-performance-computing

Trilinos 20th Anniversary Celebration





PRESENTED BY

The Trilinos Community

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.

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

Ray Tuminaro, John Shadid, and Scott Hutchinson, co-inventors of Aztec (the precursor to AztecOO) circa 1997.

Best commit message ever…

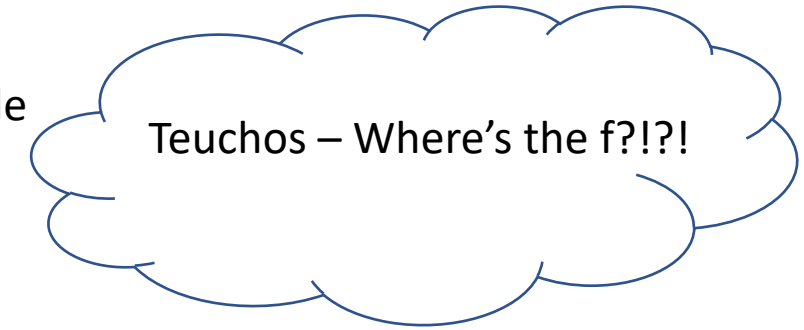
"hope this works ... but vacation will be nice either way."

Ray Tuminaro, Commit message for a9065d33268142873f5d575164b5fd2f07d335c4, 06/23/2011

What's in a name?

Trilinos is a Greek word for a (three-stranded) string of pearls. From the beginning it was intended to evoke the idea that Trilinos is a collection of packages (pearls) whose whole is more than just the sum of its parts.

Costas Bekas heard a talk on Trilinos at the U of MN while a grad student of Youcef Saad. He corrected (as best he could) the English pronunciation of Teuchos.



The acronym FROSch actually is the German word for "frog". Hence, the logo shows two overlapping frogs and is strongly inspired by the domain decomposition methods logo (see http://www.ddm.org) showing a computational domain composed of a circle overlapping with a rectangle.

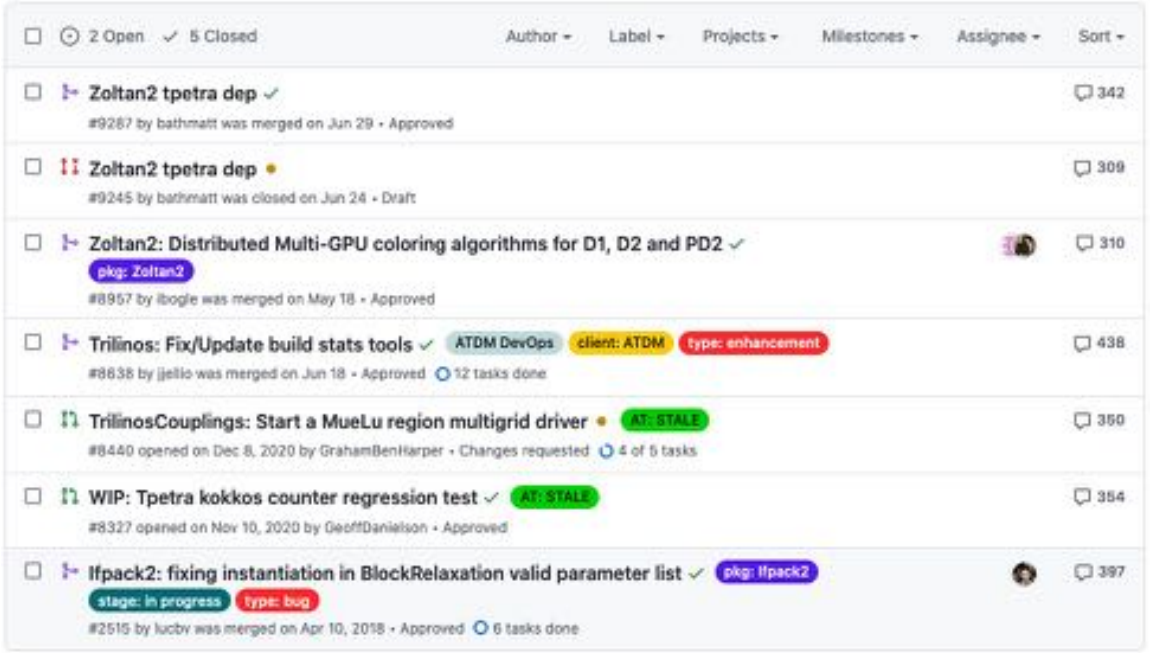
Git and GitHub stats

1. Contributors before=12/14/2001 – 16 contributors
   589 Karen D. Devine
   462 Raymond Tuminaro
   387 Erik G. Boman
   231 Roscoe A. Bartlett
   179 Jonathan Hu
    95 chtong
    59 Tamara G. Kolda
    58 Bruce A. Hendrickson
    56 wfmitch
    44 Courtenay T. Vaughan
    39 mmstjohn
    21 Roger P. Pawlowski
    14 gerval
    13 acbauer
     5 Robert Heaphy
     4 Michael A. Heroux

1. Contributors at 20 years – 540 contributors
   7818 Roscoe A. Bartlett
   6158 Mark Hoemmen
   3969 Karen D. Devine
   3492 Jonathan Hu
   3208 Christopher Siefert
   2846 trilinos-autotester ← ????
   2517 Eric T. Phipps
   2456 Marzio Sala
   2334 William F. Spotz
   2177 James M. Willenbring
   2165 Greg Sjaardema
   2053 Roger P. Pawlowski
   1900 Tobias Wiesner
   1760 Lee Ann Riesen
   1702 Eric C. Cyr
   1657 Andrey Prokopenko
   …

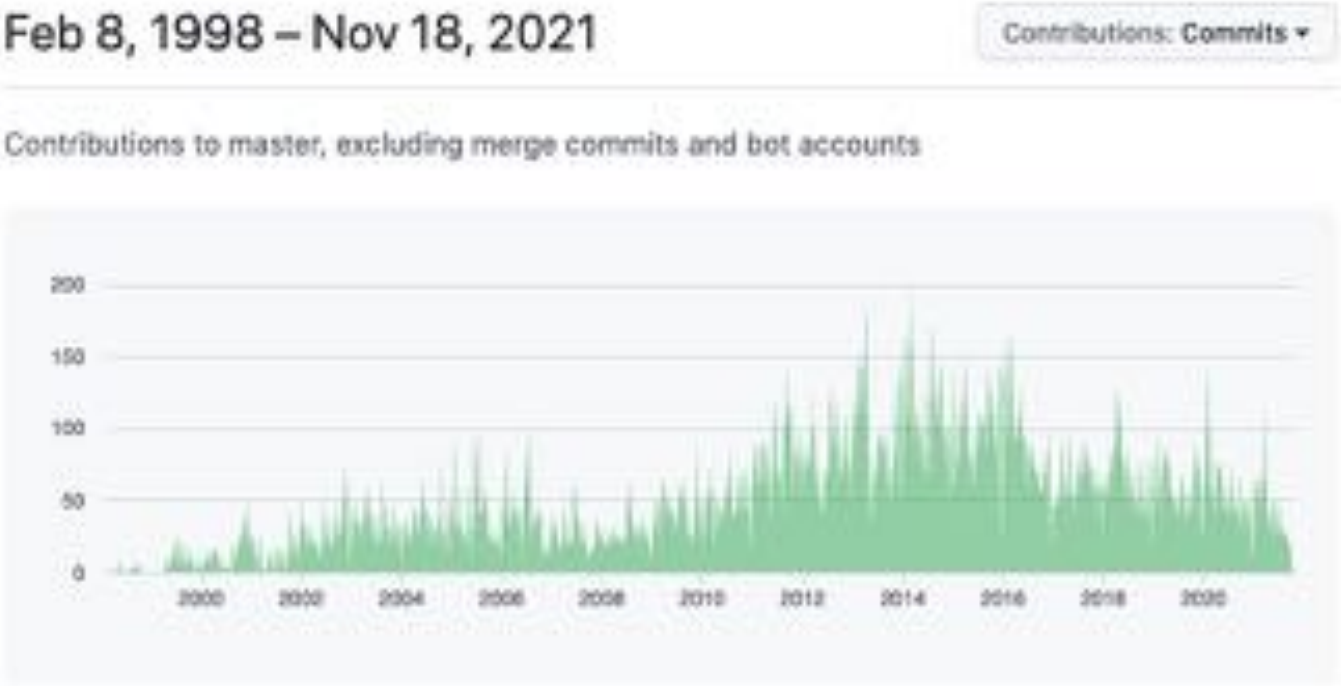
Git and GitHub stats

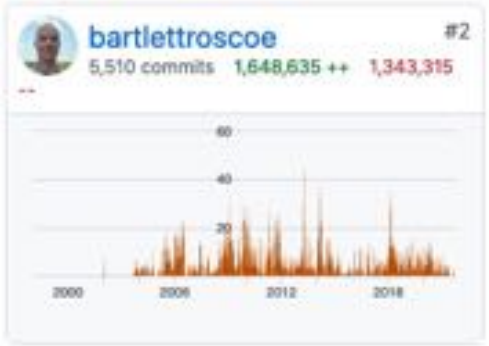
1. PR with the most comments
    1. 438 - Trilinos: Fix/Update build stats tools #8638
    2. only four participants: James Elliot, Ross Bartlett, Christian Glusa, and Trilinos Autotester.
    3. Here are the top 7.
    4. Two are still open!

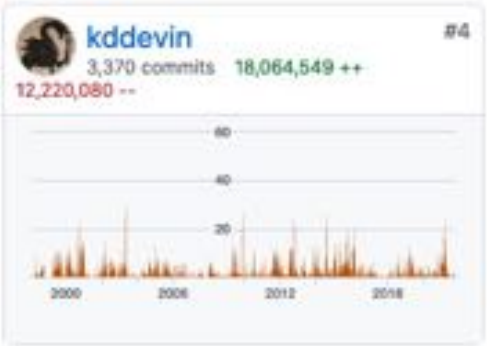


Git and GitHub stats

1. Most commits
2. jhux2 and csiefer2? Are these their clones, e.g., <id>2?

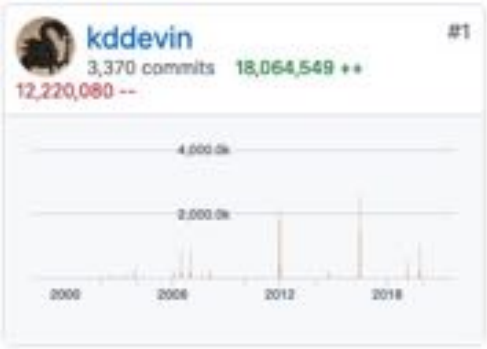






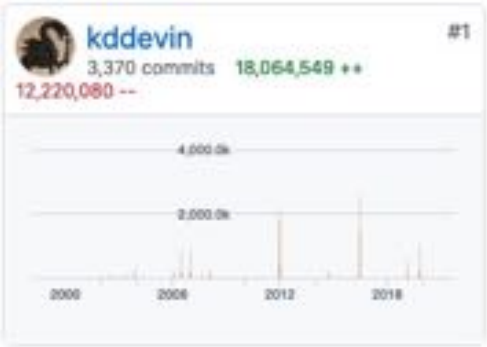
Git and GitHub stats

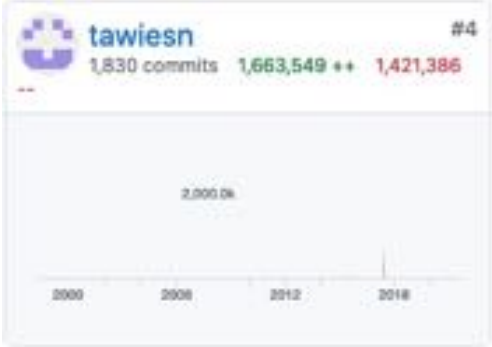
1. Most lines added
2. Karen by a lot!



Git and GitHub stats

1. Most lines deleted.
2. Karen by a lot!





Trilinos is an effective recruiting tool!

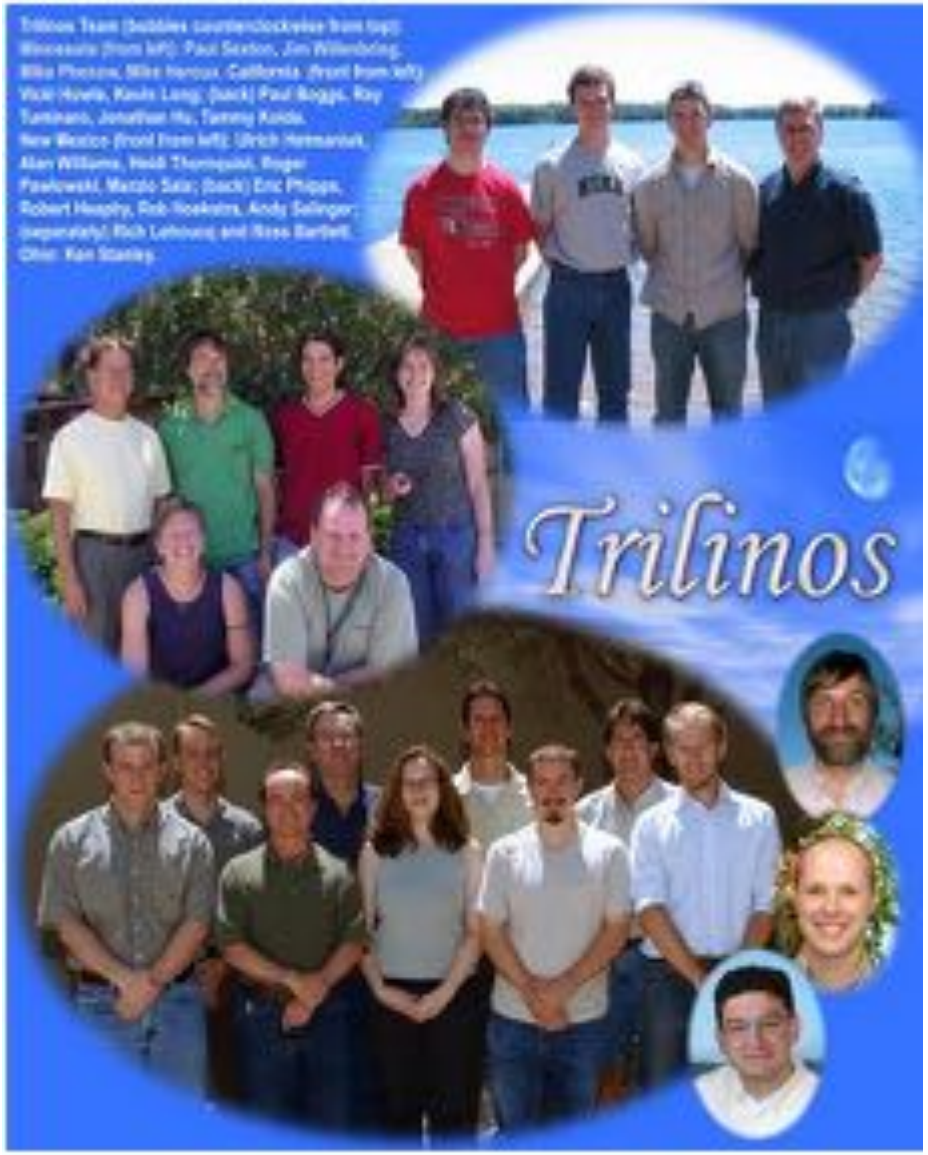
Working with and then on Trilinos starting in 2013 was my first exposure to SNL (in fact attending a TUG either in 2013 or 2014 was my first trip visit to SNL). Working on Trilinos and with the Trilinos team made me consider SNL as a future career opportunity and led to me joining the labs in 2017 as a staff member in Center 1600, where I'm now managing a department.

Kris Beckwith, 1684 Manager

Awards

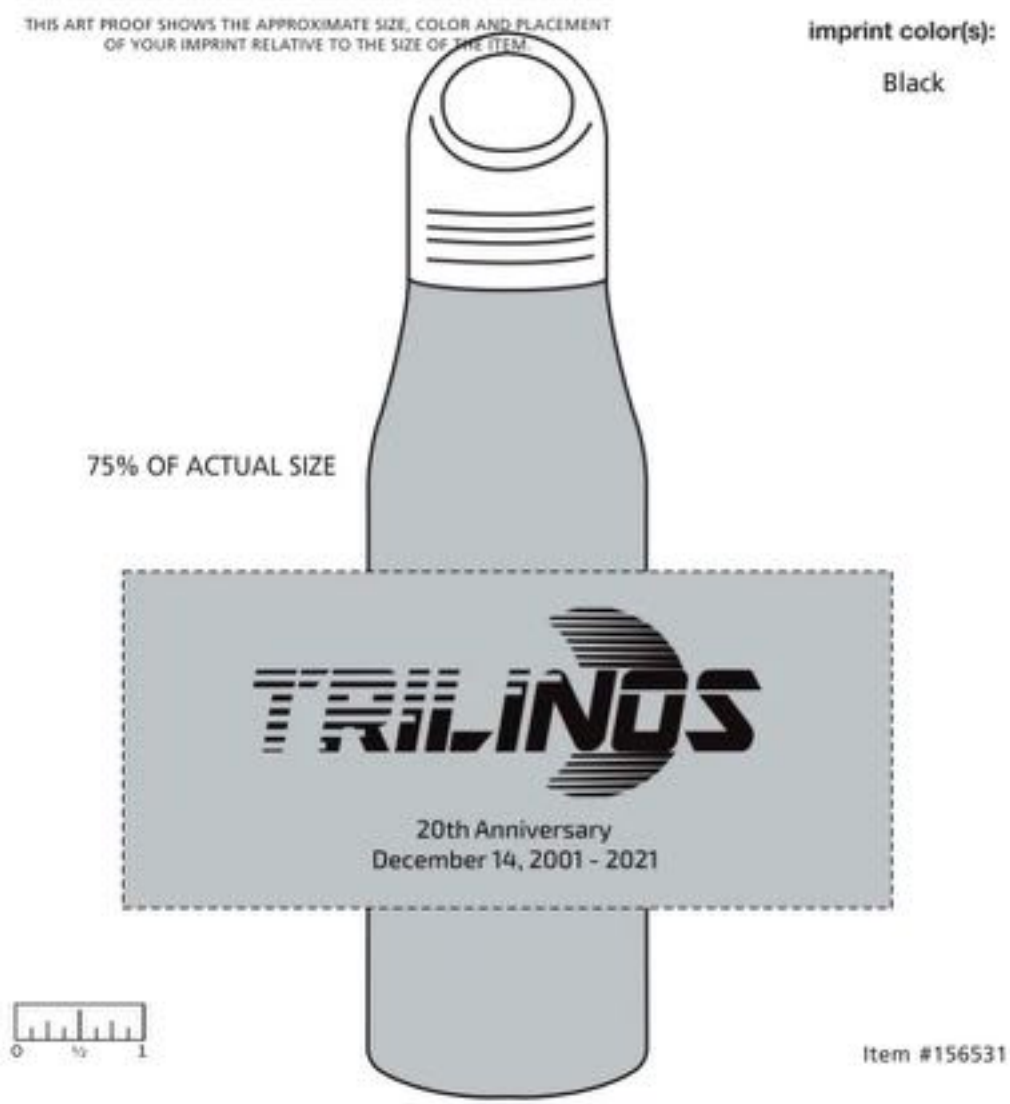
Trilinos received a 2004 R&D 100 Award, given out yearly by R&D Magazine to recognize the “100 most technologically significant products introduced in the past year.”

The formal announcement was made in the September 2004 issue of R&D Magazine.



Water bottles

- We are getting the silver water bottle
- With
  - Trilinos 2nd generation logo
  - Black on silver lettering
- Phyllis Rutka is coordinating



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