LEADERSHIP SCIENTIFIC SOFTWARE

Kick off Town Hall Meeting

September 16, 2021

Agenda

- Introduction to LSSw
- Overview of ECP efforts
- Next steps
- Q&A

Background

- US Department of Energy (DOE) Exascale Computing Project (ECP)
  - Developing enabling technologies for upcoming exascale computers
    - ECP Software Technology (ST) focus area:
      - Uses a macro-engineering software lifecycle to
        - Plan, execute, track, and assess product development toward the
          - Delivery of a curated portfolio of reusable, open-source software products called
            - The Extreme-scale Scientific Software Stack or E4S (https://e4s.io)

- During the final years of ECP, one key objective is to:
  - Transition our efforts to a sustainable organization and model for
    - Continued development and delivery of future capabilities, including
      - Incorporation of new scientific software domains, and
        - Expansion of the contributor and user communities
- LSSw is key component toward sustainability

LSSw Mission

- LSSw is dedicated to
  - Building community and understanding around the
    - Development and sustainable delivery of
      - Leadership scientific software
- Development
  - Portfolio-driven approach
  - Co-design with hardware, system software, applications
- Sustainable
  - Organizational stability
  - Emphasis on quality
  - Broad accessibility

Leadership Scientific Software (defn)

- Libraries, tools and environments that
  - Contribute to scientific discovery and insight in
    - New and emerging computing environments
- Are end-user applications within scope?
  - Yes, as stakeholders in the effort
  - Goal is to provide
    - High-priority functionality not available elsewhere
    - Portable performance on leading edge and emerging platforms
    - A sustainable turnkey software ecosystem

Leadership Scientific Software (defn)

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

ECP Efforts

- ECP is an notable project:
  - Stable, sustained funding of a national project with clear goals
  - Infrastructure to innovate and establish new collaborative work
- ECP enables tremendous opportunities to:
  - Create a new generation of scientific software
  - Provide a curated portfolio of reusable software products for apps
  - Qualitatively change how we plan, develop and deliver leadership SW

Sustainability of the Exascale Computing Project Software Stack

EXASCALE COMPUTING PROJECT

Michael Heroux, Director of Software Technology
Lois Curfman McInnes, Deputy Director
Rajeev Thakur, Programming Models & Runtimes
Jeff Vetter, Development Tools
Sherry Li, Math Libraries
Jim Ahrens, Data & Viz
Todd Munson, SW Ecosystem
Kathryn Mohror, NNSA ST

National Nuclear Security Administration

U.S. DEPARTMENT OF ENERGY | Office of Science

ECP Organizational Sketch

ECP Software Technology (ST) is one of three focus areas

Performant mission and science applications @ scale

Aggressive RD&D Project	Mission apps & integrated S/W stack	Deployment to DOE HPC Facilities	Hardware tech advances

Application Development (AD)

Develop and enhance the predictive capability of applications critical to the DOE

24 applications including national security, to energy, earth systems, economic security, materials, and data

Software Technology (ST)

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

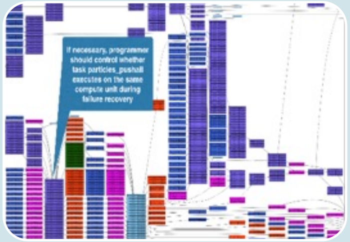
70 unique software products spanning programming models and run times, math libraries, data and visualization

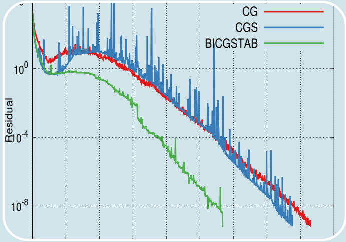
Hardware and Integration (HI)

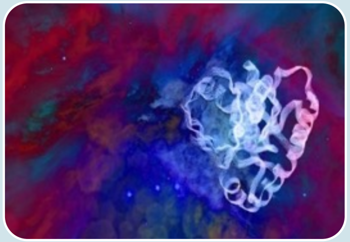
Integrated delivery of ECP products on targeted systems at leading DOE HPC facilities

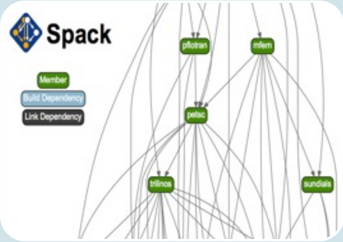
6 US HPC vendors focused on exascale node and system design; application integration and software deployment to facilities

ECP ST has six technical areas









National Nuclear Security Administration

Programming Models & Runtimes

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

Development Tools

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

Math Libraries

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

Data and Visualization

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

Software Ecosystem

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

NNSA ST

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

ECP Software Technology Leadership Team

Mike Heroux, Software Technology Director
Mike has been involved in scientific software R&D for 30 years. His first 10 were at Cray in the LIBSCI and scalable apps groups. At Sandia he started the Trilinos and Mantevo projects, is author of the HPCG benchmark for TOP500, and leads productivity and sustainability efforts for DOE.

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

Rajeev Thakur, Programming Models and Runtimes
Rajeev is a senior computer scientist at ANL and most recently led the ECP Software Technology focus area. His research interests are in parallel programming models, runtime systems, communication libraries, and scalable parallel I/O. He has been involved in the development of open source software for large-scale HPC systems for over 20 years.

Jeff Vetter, Development Tools
Jeff is a computer scientist at ORNL, where he leads the Future Technologies Group. He has been involved in research and development of architectures and software for emerging technologies, such as heterogeneous computing and nonvolatile memory, for HPC for over 15 years.

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

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

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

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

ST L4 Teams

- WBS
- Name
- PIs
- PCs - Project Coordinators

WBS	WBS Name	CAM/PI	PC
2.3	**Software Technology**	**Heroux, Mike, McInnes, Lois**	-
2.3.1	**Programming Models & Runtimes**	**Thakur, Rajeev**	-
2.3.1.01	PMR SDK	Shende, Sameer	Shende, Sameer
2.3.1.07	Exascale MPI (MPICH)	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	Miller, Barton	Tim Haines
2.3.2.06	Exa-PAPI++: The Exascale Performance Application Programming Interface with Modern C++	Dongarra, Jack	Jagode, Heike
2.3.2.08	Extending HPCToolkit to Measure and Analyze Code Performance on Exascale Platforms	Mellor-Crummey, John	Meng, Xiaozhu
2.3.2.10	PROTEAS-TUNE	Vetter, Jeff	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	Turner, John	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	Atkins, Chuck	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/Sz	Cappello, Franck	Ehling, Scott
2.3.4.15	ExaIO - Delivering Efficient Parallel I/O on Exascale Computing Systems with HDF5 and Unify	Byna, Suren	Bagha, Neelam
2.3.4.16	ALPINE: Algorithms and Infrastructure for In Situ Visualization and Analysis/ZFP	Ahrens, James	Turton, Terry
2.3.5	**Software Ecosystem and Delivery**	**Munson, Todd**	
2.3.5.01	Software Ecosystem and Delivery Software Development Kit	Willenbring, James M	Willenbring, James M
2.3.5.09	SW Packaging Technologies	Gamblin, Todd	Gamblin, Todd
2.3.5.10	ExaWorks	Laney, Dan	Laney, Dan
2.3.6	**NNSA ST**	**Mohror, Kathryn**	-
2.3.6.01	LANL ATDM	Mike Lang	Vandenbusch, Tanya Marie
2.3.6.02	LLNL ATDM	Becky Springmeyer	Gamblin, Todd
2.3.6.03	SNL ATDM	Jim Stewart	Trujillo, Gabrielle

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

ECP ST Stats

- 35 L4 subprojects
- 11 PI/PC same
- 24 PI/PC different
- ~27% ECP budget

EXASCALE COMPUTING PROJECT

We work on products applications need now and into the future

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

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

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

One example: SLATE port to AMD and Intel platforms

Scope and objectives

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

Port to AMD and Intel

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

Impact

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

Accomplishment

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

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

Key ECP Software Stack Legacy:
- Portable execution on:
  - CPUs
  - 3 different GPUs
- A bridge from CPUs to GPUs

Thanks to the ECP community

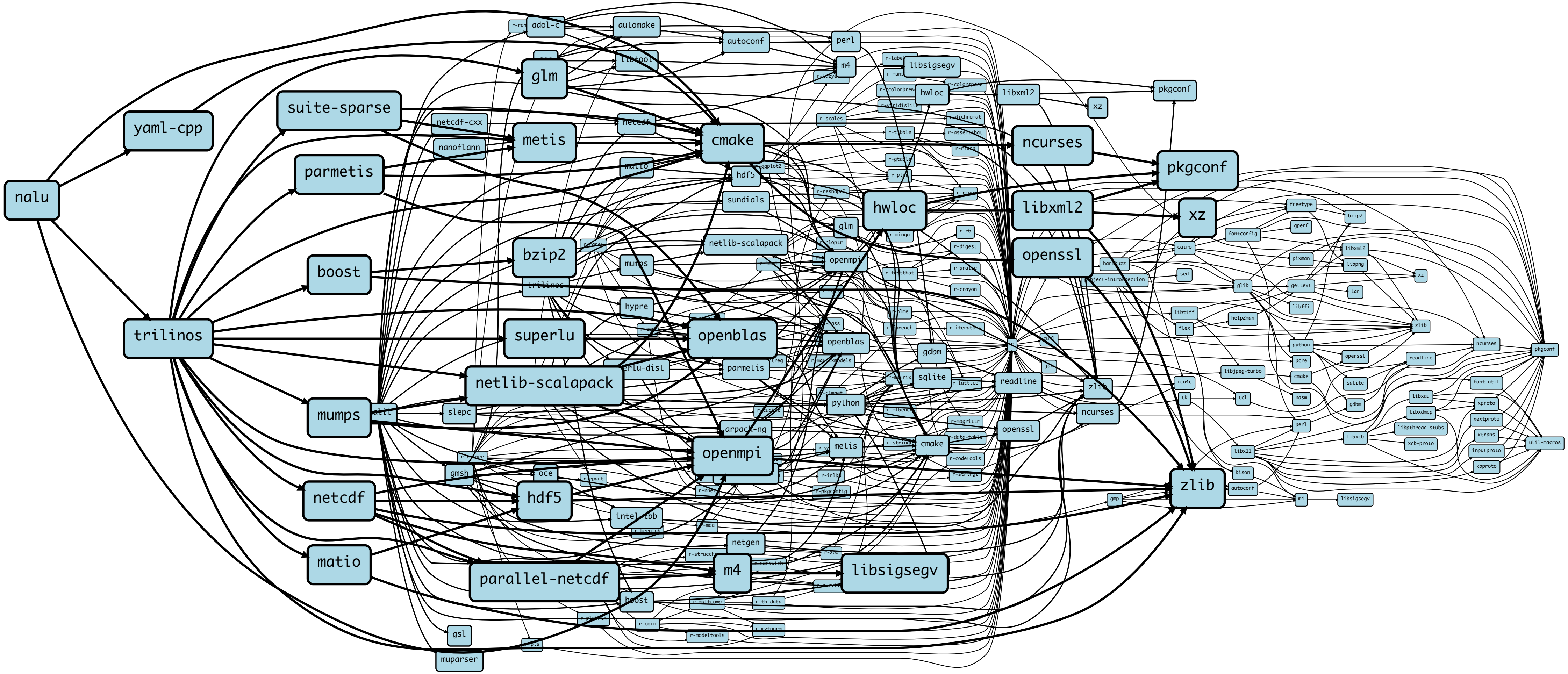
- The demands of a formal project like ECP are significant
- ECP staff have adapted to the new environment with innovative solutions
- The progress we have made in ECP has been a collective effort of hundreds of committed people
- Thank you

The Growing Complexity of Scientific Application Software Stacks

Challenges

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

Scientific software is becoming extremely complex



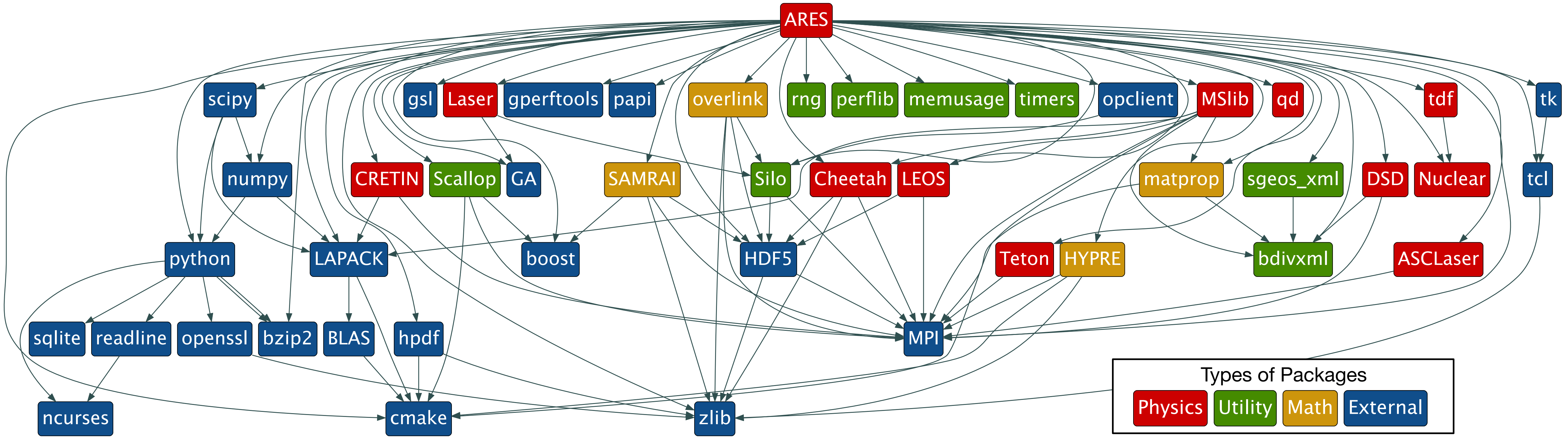
Nalu: Generalized Unstructured Massively Parallel Low Mach Flow

dealii: C++ Finite Element Library

R Miner: R Data Mining Library

EXASCALE COMPUTING PROJECT

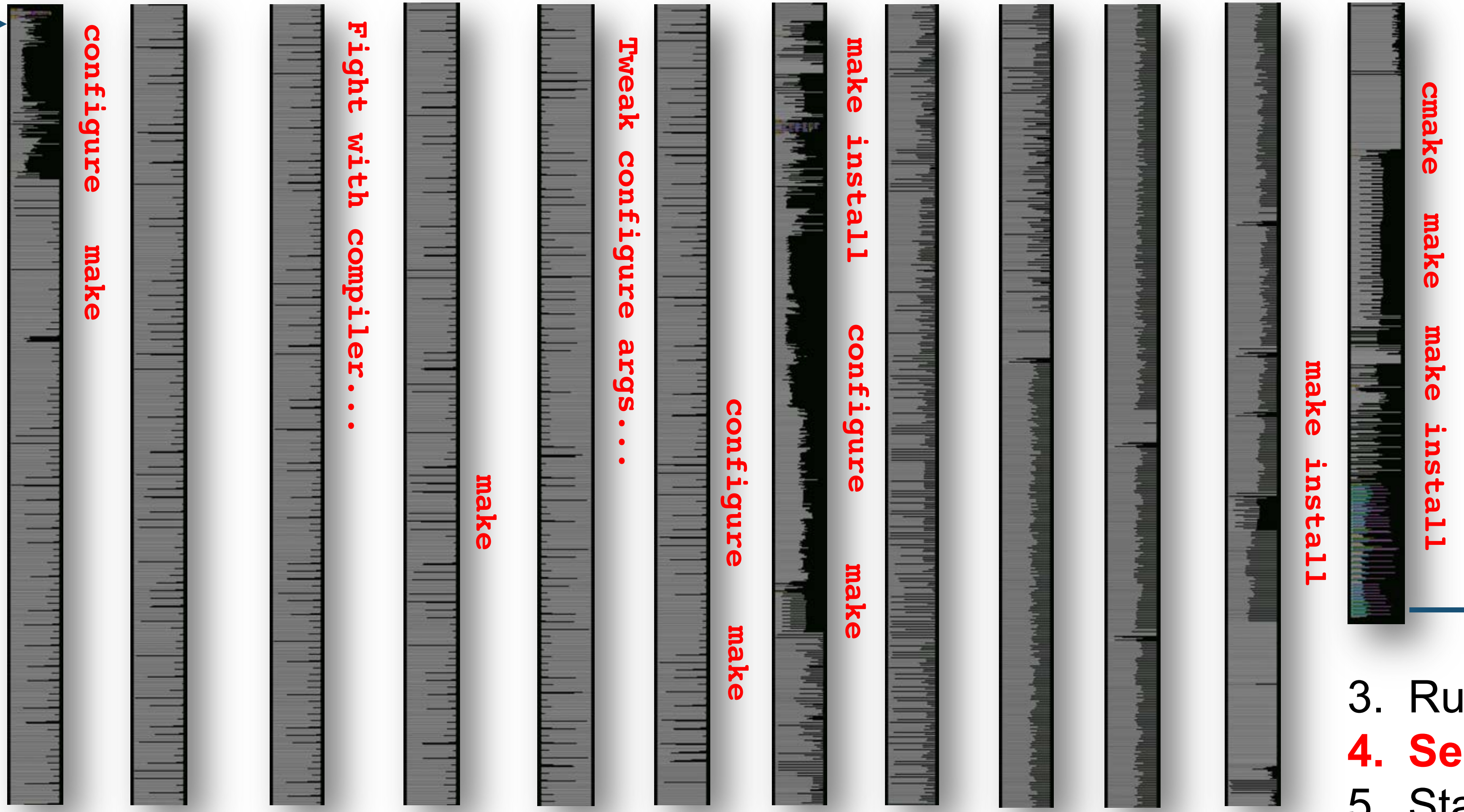
Even proprietary codes are based on many open source libraries



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

How to install software on a supercomputer

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



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

The Exascale Computing Project is building an entire *ecosystem*

25+ applications x **80+ software packages** x **6+ target architectures/platforms**
Xeon Power Intel GPUs
NVIDIA AMD GPUs ARM

x **Up to 7 compilers**
Intel GCC Clang XL
PGI Cray NAG

x **10+ Programming Models**
OpenMPI MPICH MVAPICH OpenMP CUDA
OpenACC Dharma Legion RAJA Kokkos

x **2-3 versions of each package + external dependencies**

> **1,000,000** combinations!

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

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

EXASCALE COMPUTING PROJECT

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

Extreme-scale Scientific Software Stack (E4S)

- E4S: HPC Software Ecosystem – a curated software portfolio
- A **Spack-based** distribution of software tested for interoperability and portability to multiple architectures
- Available from **source**, **containers**, **cloud, binary caches**
- Leverages and enhances SDK interoperability thrust
- Not a commercial product – an open resource for all
- Oct 2018: E4S 0.1 - 24 full, 24 partial release products
- Jan 2019: E4S 0.2 - 37 full, 10 partial release products
- Nov 2019: E4S 1.0 - 50 full, 5 partial release products
- Feb 2020: E4S 1.1 - 61 full release products
- Nov 2020: E4S 1.2 (aka, 20.10) - 67 full release products
- Feb 2021: E4S 21.02 - 67 full release, 4 partial release
- May 2021: E4S 21.05 - 76 full release products
- August 2021: E4S 21.08 - 88 full release products

https://e4s.io

Lead: Sameer Shende (U Oregon)

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

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

xSDK release 0.6.0 (Nov 2020)

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

(deal.II, preCICE, PHIST, SLEPc) from the broader community

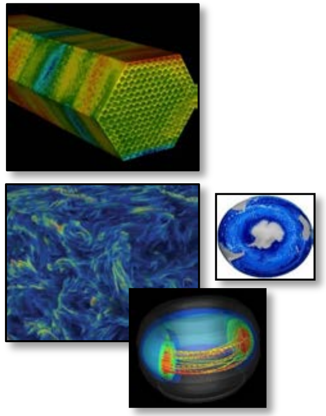
As motivated and validated by the needs of ECP applications:

ECP Math libraries

- Performance on new node architectures
- Extreme strong scalability
- Advanced, coupled multiphysics, multiscale
- Optimization, UQ, solvers, discretizations
- Interoperability, complementarity: xSDK

- Next-generation algorithms
- Advances in data structures for new node architectures
- Improving library quality, sustainability, interoperability

- Toward predictive scientific simulations
- Increasing performance, portability, productivity



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

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

Delivering an open, hierarchical software ecosystem

More than a collection of individual products

Levels of Integration | Product | Source and Delivery

ECP ST Open Product Integration Architecture

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

E4S

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

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

SDKs

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

- Standard workflow
- Existed before ECP

ST Products

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

ECP ST Individual Products

ECP EXASCALE COMPUTING PROJECT

E4S Community Policies & DocPortal

E4S Community Policies V1.0 Released

HOME EVENTS ABOUT DOCPO POLICIES DOWNLOAD

E4S Project

The Extreme-scale Scientific Software Stack

What is E4S?

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

Purpose

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

Approach

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

E4S Community Policies Version 1

A Commitment to Quality Improvement

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

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

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

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

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

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

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

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

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

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

EXASCALE COMPUTING PROJECT

E4S DocPortal

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

E4S Products

*: Member Product

Show 10 entries

Search:

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

Showing 1 to 10 of 76 entries

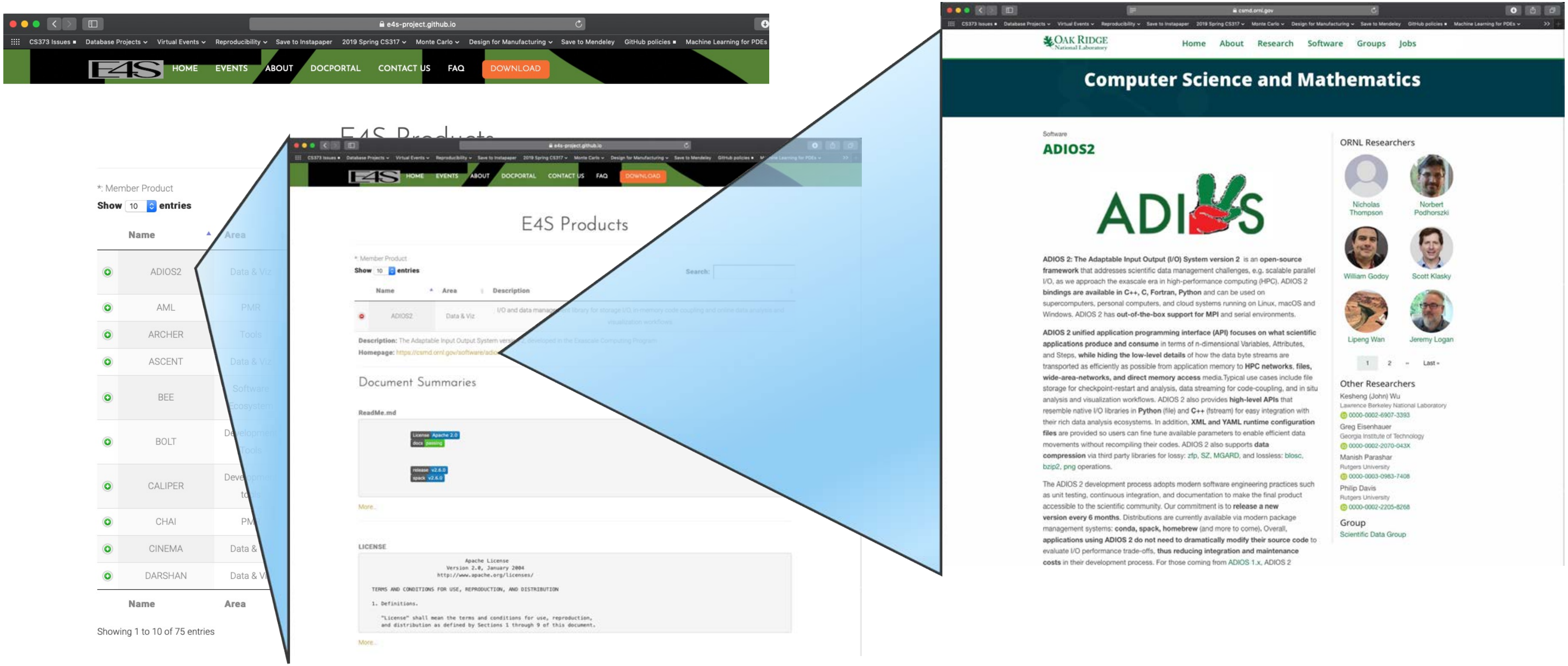
Previous 1 2 3 4 5 … 8 Next

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

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

EXASCALE COMPUTING PROJECT

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

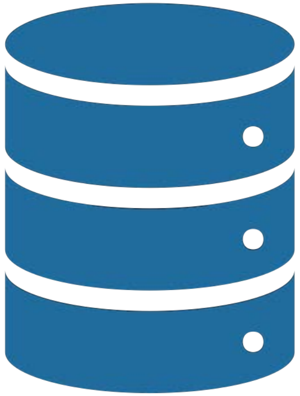


https://e4s-project.github.io/DocPortal.html

E4S Planning, Executing, Delivering

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

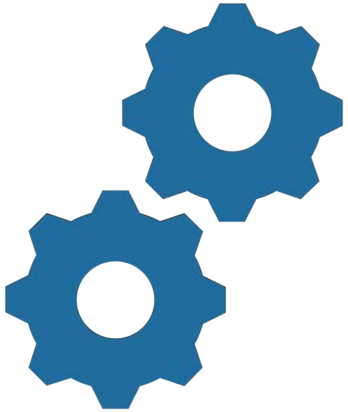
Baseline



FY20–23 Baseline Plan
High level Definitions

- Q2 FY19 start
- FY20 Base plan
- FY21–23 planning packages

Annual Refinement



FY Refine Baseline Plan As Needed
Basic activity definitions

- 6 months prior to FY
- 4–6 P6 Activities/year
- Each activity:
  - % annual budget
  - Baseline start/end
  - High level description

Per Activity

Detailed Plan
Complete activity definitions

- 8 weeks prior to start
- High-fidelity description
- Execution strategy
- Completion criteria
- Personnel details

Two-level Review Process

Changes to Cost, Scope, and Schedule

Minor	Major
Lightweight Review in Jira, L3 and L2 leads	Change Control Board Review, ECP leadership

Variance Recorded in Jira

Proceed with Execution

EXASCALE COMPUTING PROJECT

KPP-3: Focus on capability integration

- **Capability:** Any significant product functionality, including existing features adapted to the pre-exascale and exascale environments, that can be integrated into a client environment.
- **Capability Integration:** Complete, sustainable integration of a significant product capability into a client environment in a pre-exascale environment (tentative score) and in an exascale environment (confirmed score).

ECP ST Lifecycle summary

Managed by P6 Activity Process

Create Annual Planning Package
- Each product has its own planning packages
- Defined for all FYs

Refine upcoming FY plan
- Complete 6 months prior to FY
- 4 or more P6 activities per product

Refine upcoming P6 activity
- Complete 8 weeks prior to activity start
- Include all details

Develop capabilities and track progress via tailored EVM

Measured by KPP-3 Process

Integrate into product
- Full testing, documentation, etc.
- Direct access for some users

Integrate into SDK
- Satisfy SDK community policies
- Direct access for some users

Integrate into E4S
- Satisfy E4S community policies
- Full ecosystem with high value

Deliver to users
- From source (spack)
- Containers, cloud

EXASCALE COMPUTING PROJECT

Using E4S

Spack

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

E4S Download from https://e4s.io

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

E4S

HOME EVENTS ABOUT E4S PRODUCT DOCPORTAL E4S COMMUNITY POLICIES CONTACT US JOIN E4S FAQ DOWNLOAD

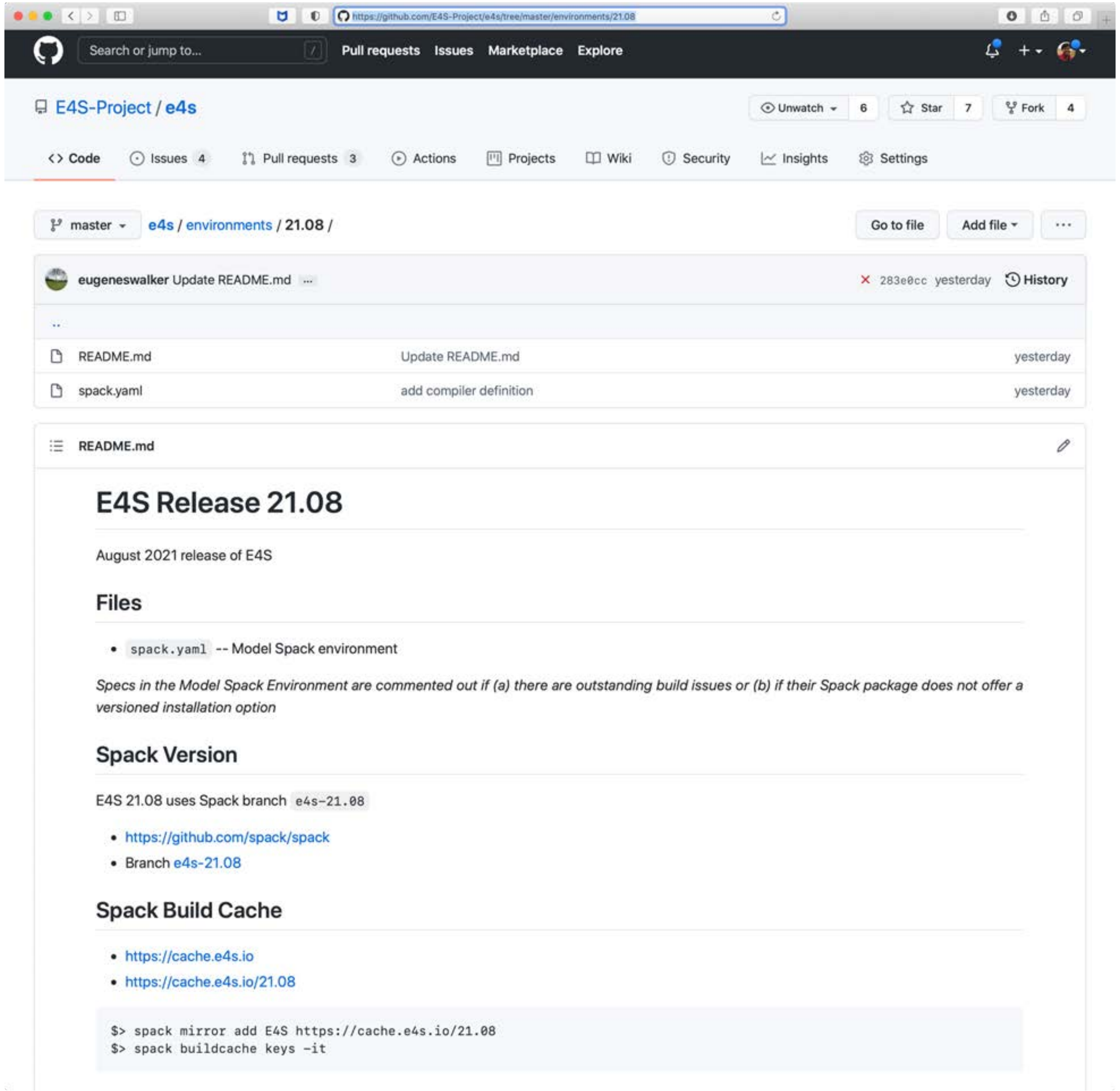
Extreme-Scale Scientific Software Stack (E4S) version 21.08

Exascale Computing Project (ECP) Software Technologies (ST) software, Extreme-Scale Scientific Software Stack (E4S) v21.08, includes a subset of ECP ST software products, and demonstrates the target approach for future delivery of the full ECP ST software stack. Also available are a number of ECP ST software products that support a Spack package, but are not yet fully interoperable. As the primary purpose of the v21.08 is demonstrating the ST software stack release approach, not all ECP ST software products were targeted for this release. Software products were targeted primarily based on existing Spack package maturity, location within the scientific software stack, and ECP SDK developer experience with the software. Each release will include additional software products, with the ultimate goal of including all ECP ST software products.

E4S v21.08 Notes.

E4S Container Installation Instructions.

E4S for bare-metal installation



E4S-Project / e4s

master e4s / environments / 21.08 /

eugeneswalker Update README.md — 283e0cc yesterday History

README.md	Update README.md	yesterday
spack.yaml	add compiler definition	yesterday

README.md

E4S Release 21.08

August 2021 release of E4S

Files

- `spack.yaml` -- Model Spack environment

Specs in the Model Spack Environment are commented out if (a) there are outstanding build issues or (b) if their Spack package does not offer a versioned installation option

Spack Version

E4S 21.08 uses Spack branch `e4s-21.08`

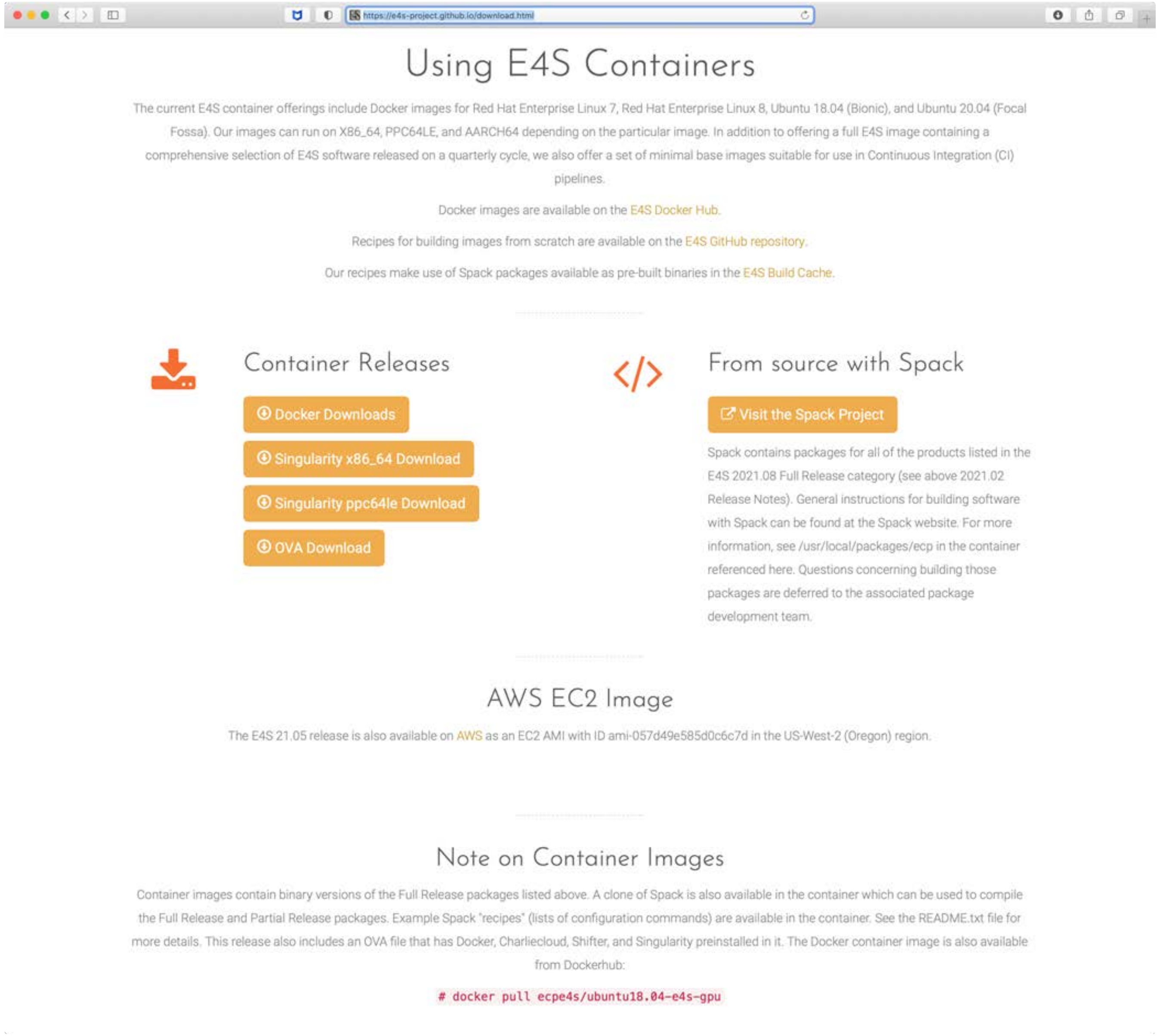
- https://github.com/spack/spack
- Branch e4s-21.08

Spack Build Cache

- https://cache.e4s.io
- https://cache.e4s.io/21.08

```
$> spack mirror add E4S https://cache.e4s.io/21.08
$> spack buildcache keys -it
```

E4S Docker and Singularity Containers



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

Using E4S Containers

The current E4S container offerings include Docker images for Red Hat Enterprise Linux 7, Red Hat Enterprise Linux 8, Ubuntu 18.04 (Bionic), and Ubuntu 20.04 (Focal Fossa). Our images can run on X86_64, PPC64LE, and AARCH64 depending on the particular image. In addition to offering a full E4S image containing a comprehensive selection of E4S software released on a quarterly cycle, we also offer a set of minimal base images suitable for use in Continuous Integration (CI) pipelines.

Docker images are available on the E4S Docker Hub.

Recipes for building images from scratch are available on the E4S GitHub repository.

Our recipes make use of Spack packages available as pre-built binaries in the E4S Build Cache.

Container Releases

Docker Downloads

Singularity x86_64 Download

Singularity ppc64le Download

OVA Download

From source with Spack

Visit the Spack Project

Spack contains packages for all of the products listed in the E4S 2021.08 Full Release category (see above 2021.02 Release Notes). General instructions for building software with Spack can be found at the Spack website. For more information, see /usr/local/packages/ecp in the container referenced here. Questions concerning building those packages are deferred to the associated package development team.

AWS EC2 Image

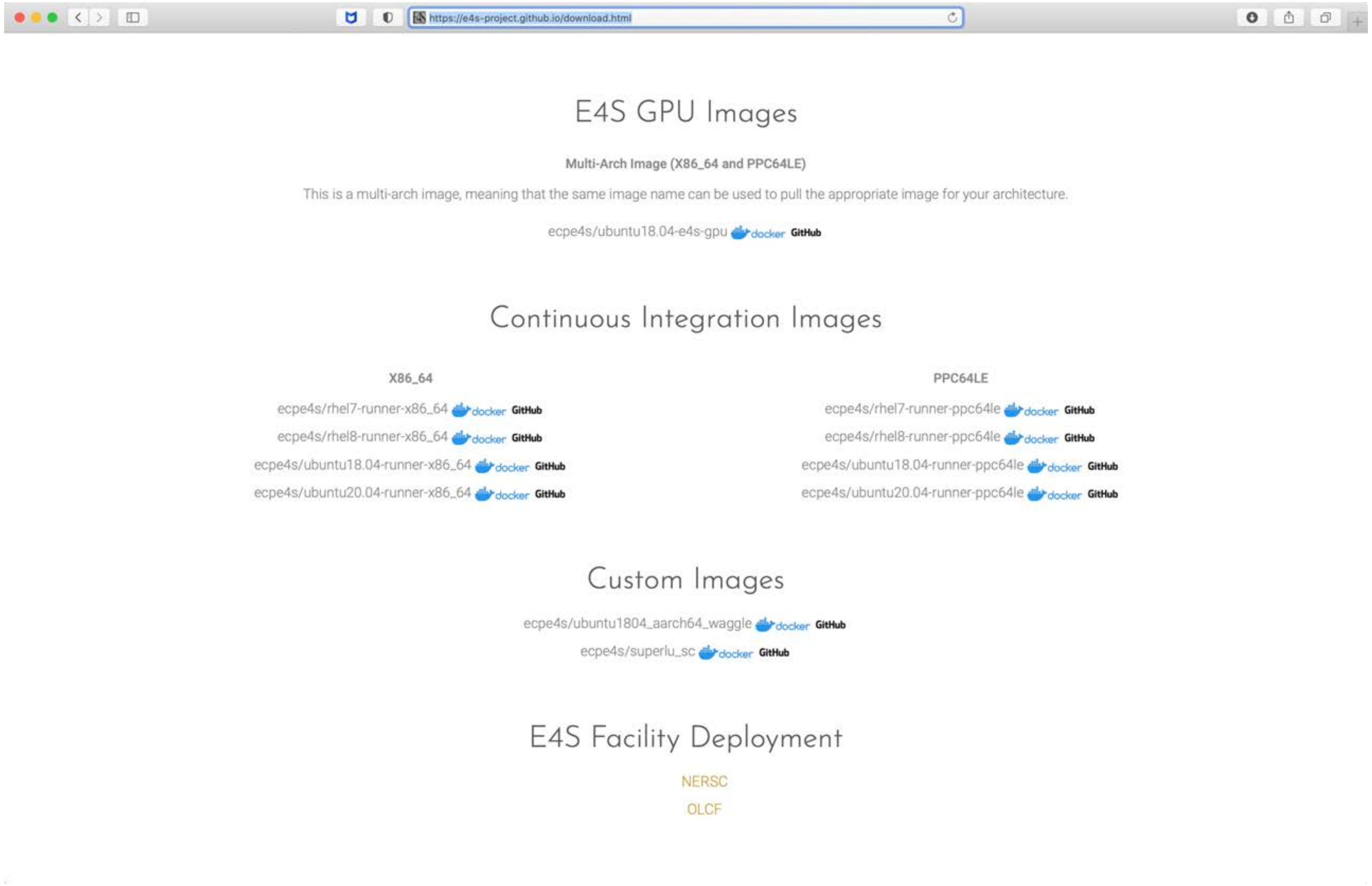
The E4S 21.05 release is also available on AWS as an EC2 AMI with ID ami-057d49e585d0c6c7d in the US-West-2 (Oregon) region.

Note on Container Images

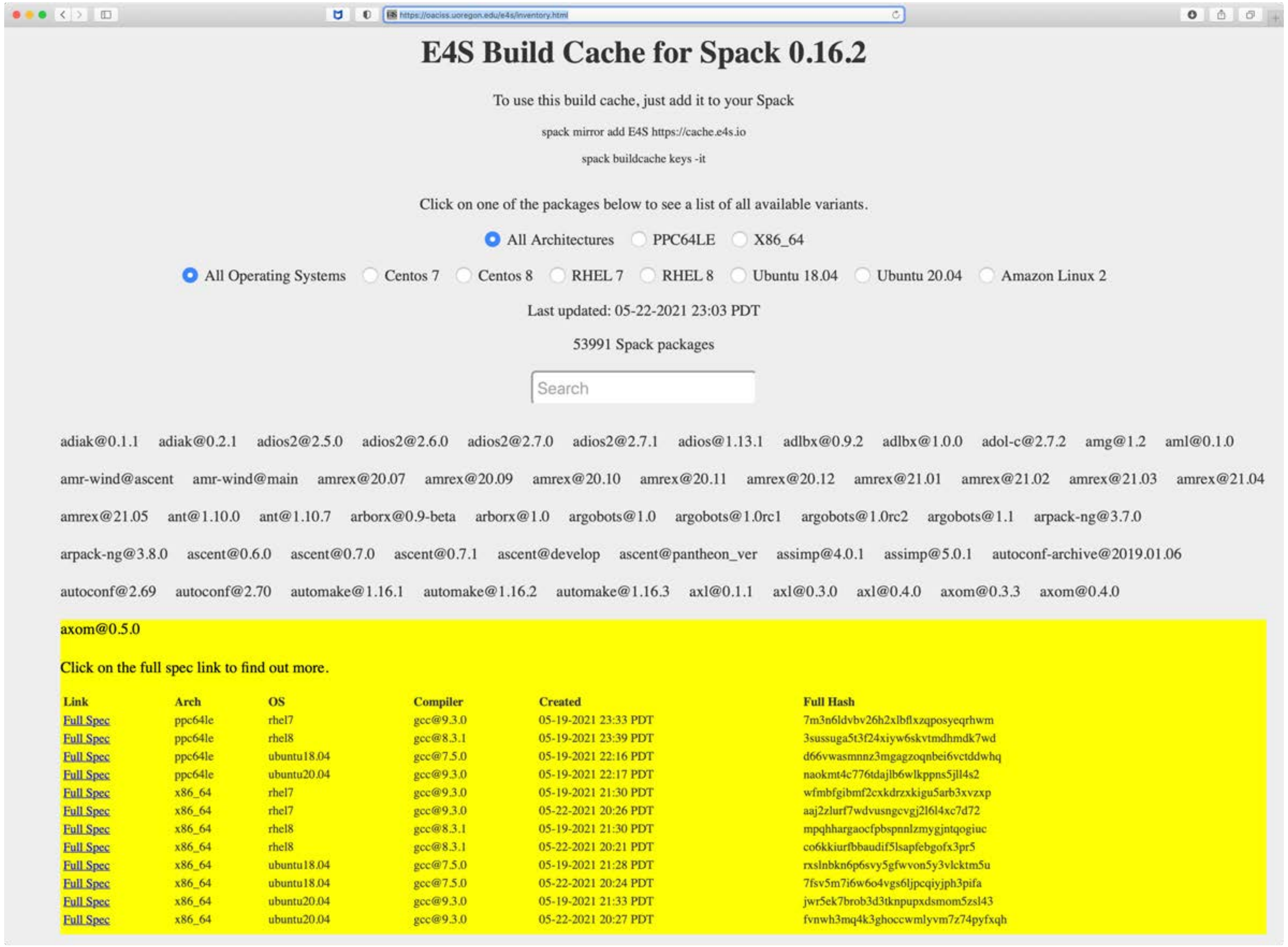
Container images contain binary versions of the Full Release packages listed above. A clone of Spack is also available in the container which can be used to compile the Full Release and Partial Release packages. Example Spack "recipes" (lists of configuration commands) are available in the container. See the README.txt file for more details. This release also includes an OVA file that has Docker, Charliecloud, Shifter, and Singularity preinstalled in it. The Docker container image is also available from Dockerhub:

```
# docker pull ecpe4s/ubuntu18.04-e4s-gpu
```

E4S base images for custom container deployment and CI images



E4S: Spack Build Cache at U. Oregon and AWS



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

E4S Build Cache for Spack 0.16.2

To use this build cache, just add it to your Spack

spack mirror add E4S https://cache.e4s.io

spack buildcache keys -it

Click on one of the packages below to see a list of all available variants.

All Architectures | PPC64LE | X86_64

All Operating Systems | Centos 7 | Centos 8 | RHEL 7 | RHEL 8 | Ubuntu 18.04 | Ubuntu 20.04 | Amazon Linux 2

Last updated: 05-22-2021 23:03 PDT

53991 Spack packages

Search

adiak@0.1.1 adiak@0.2.1 adios2@2.5.0 adios2@2.6.0 adios2@2.7.0 adios2@2.7.1 adios@1.13.1 adlbx@0.9.2 adlbx@1.0.0 adol-c@2.7.2 amg@1.2 aml@0.1.0

amr-wind@ascent amr-wind@main amrex@20.07 amrex@20.09 amrex@20.10 amrex@20.11 amrex@20.12 amrex@21.01 amrex@21.02 amrex@21.03 amrex@21.04

amrex@21.05 ant@1.10.0 ant@1.10.7 arborx@0.9-beta arborx@1.0 argobots@1.0 argobots@1.0rc1 argobots@1.0rc2 argobots@1.1 arpack-ng@3.7.0

arpack-ng@3.8.0 ascent@0.6.0 ascent@0.7.0 ascent@0.7.1 ascent@develop ascent@pantheon_ver assimp@4.0.1 assimp@5.0.1 autoconf-archive@2019.01.06

autoconf@2.69 autoconf@2.70 automake@1.16.1 automake@1.16.2 automake@1.16.3 axl@0.1.1 axl@0.3.0 axl@0.4.0 axom@0.3.3 axom@0.4.0

axom@0.5.0

Click on the full spec link to find out more.

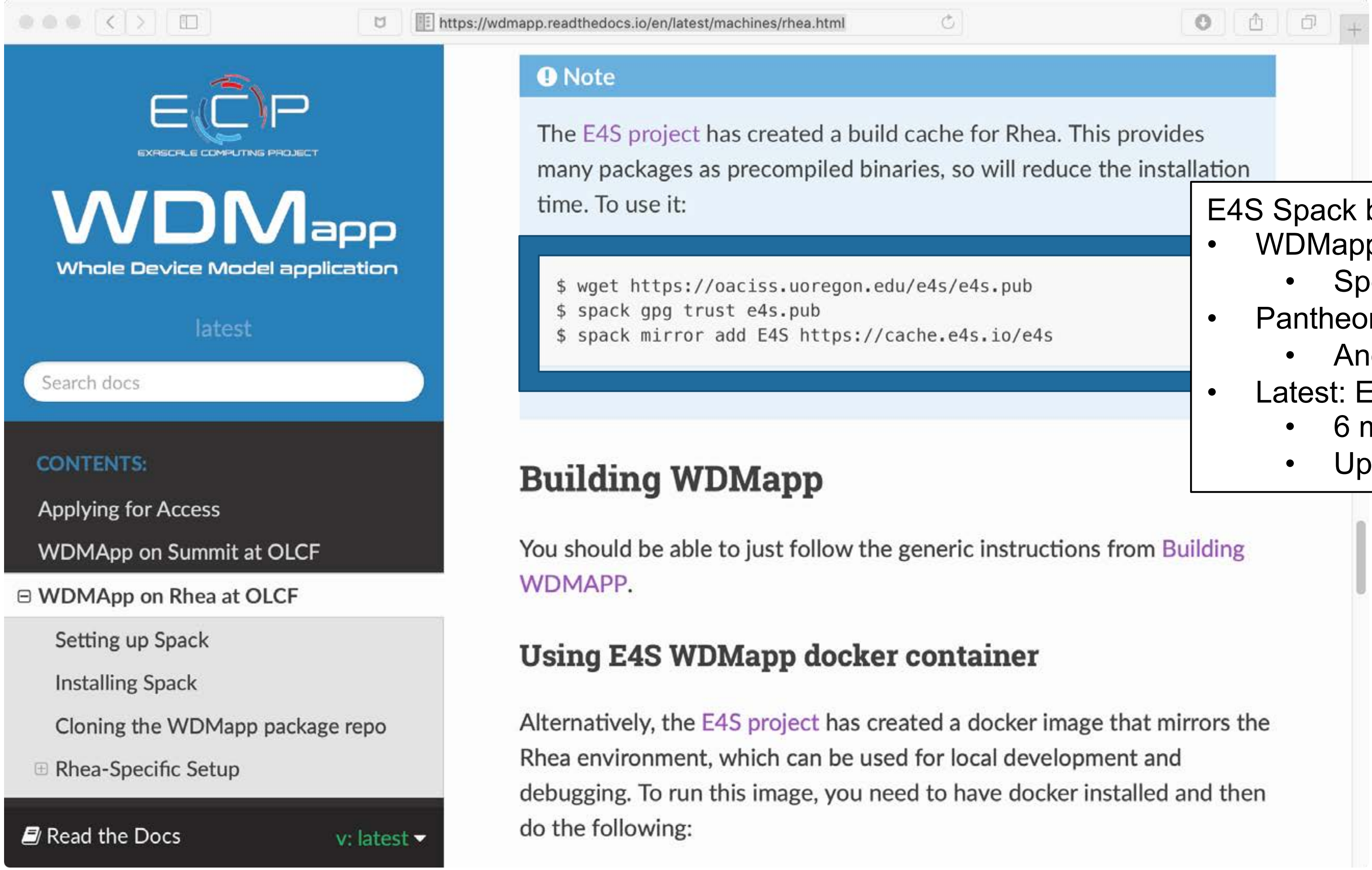
Link	Arch	OS	Compiler	Created	Full Hash
Full Spec	ppc64le	rhel7	gcc@9.3.0	05-19-2021 23:33 PDT	7m3n6ldvbv26h2xlbflxzqposyeqrhwm
Full Spec	ppc64le	rhel8	gcc@8.3.1	05-19-2021 23:39 PDT	3sussuga5t3f24xiyw6skvtmdhmdk7wd
Full Spec	ppc64le	ubuntu18.04	gcc@7.5.0	05-19-2021 22:16 PDT	d66vwasmnnz3mgagzoqnbei6vctddwhq
Full Spec	ppc64le	ubuntu20.04	gcc@9.3.0	05-19-2021 22:17 PDT	naokmt4c776tdajlb6wlkppns5jll4s2
Full Spec	x86_64	rhel7	gcc@9.3.0	05-19-2021 21:30 PDT	wfmbfgibmf2cxkdrzxkigu5arb3xvzxp
Full Spec	x86_64	rhel7	gcc@9.3.0	05-22-2021 20:26 PDT	aaj2zlurf7wdvusngcvgj2l6l4xc7d72
Full Spec	x86_64	rhel8	gcc@8.3.1	05-19-2021 21:30 PDT	mpqhhargaocfpbspnnlzmygjntqogiuc
Full Spec	x86_64	rhel8	gcc@8.3.1	05-22-2021 20:21 PDT	co6kkiurfbbaudif5lsapfebgofx3pr5
Full Spec	x86_64	ubuntu18.04	gcc@7.5.0	05-19-2021 21:28 PDT	rxslnbkn6p6svy5gfwvon5y3vlcktm5u
Full Spec	x86_64	ubuntu18.04	gcc@7.5.0	05-22-2021 20:24 PDT	7fsv5m7i6w6o4vgs6ljpcqiyjph3pifa
Full Spec	x86_64	ubuntu20.04	gcc@9.3.0	05-19-2021 21:33 PDT	jwr5ek7brob3d3tknpupxdsmom5zsl43
Full Spec	x86_64	ubuntu20.04	gcc@9.3.0	05-22-2021 20:27 PDT	fvnwh3mq4k3ghoccwmlyvm7z74pyfxqh

- 50,000+ binaries
- S3 mirror
- No need to build from source code!

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

EXASCALE COMPUTING PROJECT

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


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

ECP
EXASCALE COMPUTING PROJECT

WDMapp
Whole Device Model application

latest

Search docs

CONTENTS:

Applying for Access

WDMApp on Summit at OLCF

⊟ WDMApp on Rhea at OLCF

Setting up Spack

Installing Spack

Cloning the WDMapp package repo

⊞ Rhea-Specific Setup

Read the Docs v: latest

Note

The E4S project has created a build cache for Rhea. This provides many packages as precompiled binaries, so will reduce the installation time. To use it:

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

Building WDMapp

You should be able to just follow the generic instructions from Building WDMAPP.

Using E4S WDMapp docker container

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

E4S Spack build cache:

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

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

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

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

Community Policies
Commitment to software quality

DocPortal
Single portal to all E4S product info

Portfolio testing
Especially leadership platforms

Curated collection
The end of dependency hell

Quarterly releases
Release 1.2 – November

Build caches
10X build time improvement

Turnkey stack
A new user experience

https://e4s.io

LSSw
Community Engagement

Summary

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

Growing and Sustaining the Software Community

https://ideas-productivity.org

IDEAS-ECP team works with the ECP community to improve developer productivity and software sustainability as key aspects of increasing overall scientific productivity.

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

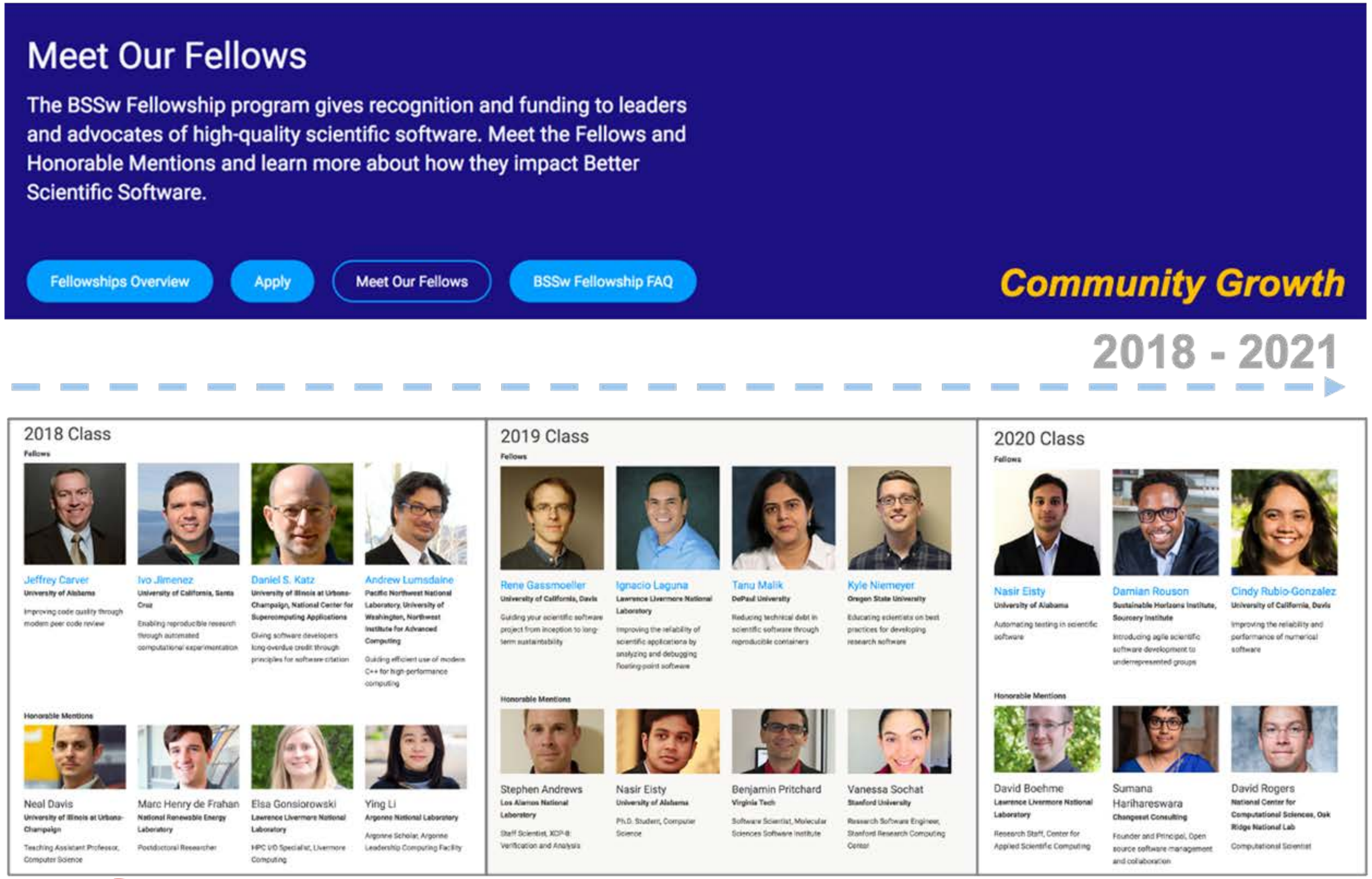
- Determine community policies to improve software quality and compatibility
- Create Software Development Kits (SDKs) to facilitate the combined use of complementary libraries and tools

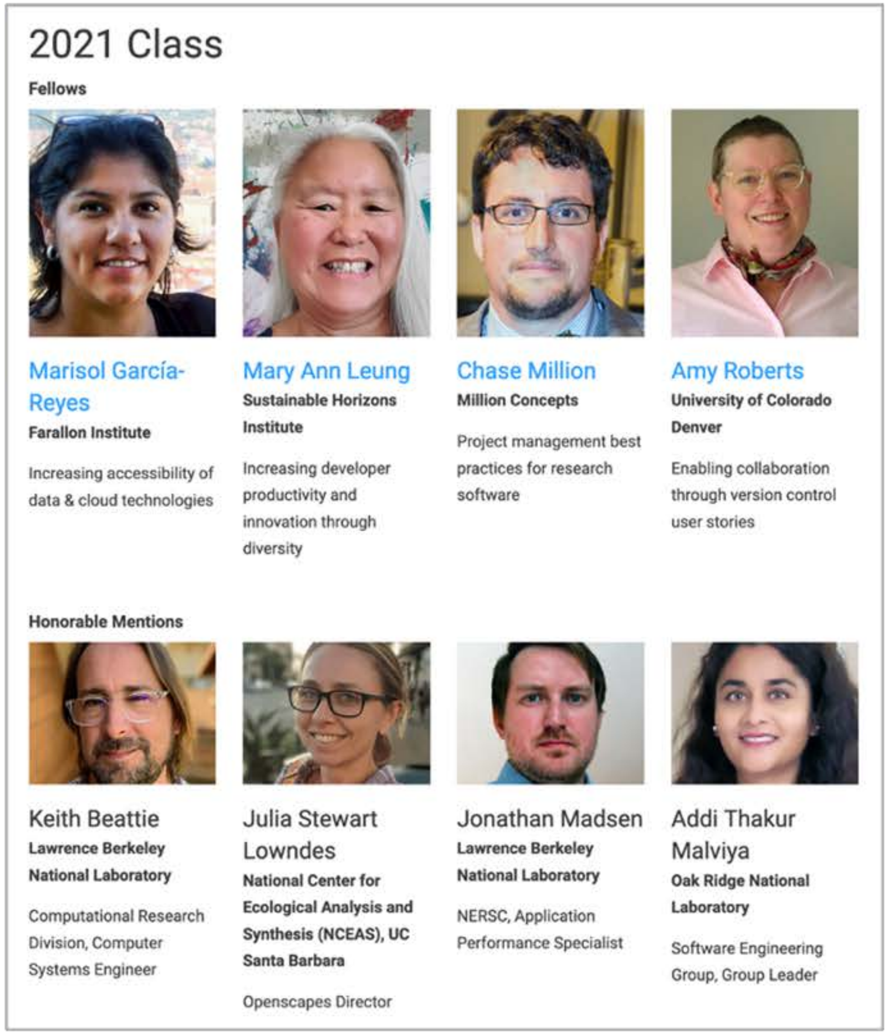
4 Engage in community outreach

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

BSSw Fellowship: Meet the Fellows

https://bssw.io/fellowship





2022 Fellows applications open: https://bssw.io/blog_posts/applications-open-for-the-2022-bssw-fellowship-program

Advancing Scientific Productivity through Better Scientific Software:

Developer Productivity & Software Sustainability Report

Disruptive changes in computer architectures and the complexities of tackling new frontiers in extreme-scale modeling, simulation, and analysis present daunting challenges to software productivity and sustainability.

This report explains the IDEAS approach, outcomes, and impact of work (in partnership with the ECP and broader computational science community).

Target readers are all those who care about the quality and integrity of scientific discoveries based on simulation and analysis. While the difficulties of extreme-scale computing intensify software challenges, issues are relevant across all computing scales, given universal increases in complexity and the need to ensure the trustworthiness of computational results.

https://exascaleproject.org/better-scientific-productivity-through-better-scientific-software-the-ideas-report

Summary & Next Steps

- Scientific software capabilities and complexity are increasing
- Computing systems are becoming more diverse
- A portfolio approach to planning and delivering is attractive
- ECP provides a working example to address complexity:
  - ECP ST lifecycle enables coordinated planning, executing, tracking and assessing
  - E4S and SDKs provide a scalable software architecture and portfolio for “turnkey” software stack
  - The IDEAS project and BSSw provide community building for scientific software developers
  - Goal: Better, faster and cheaper
- We believe the next steps require broad community engagement:
  - What are other fundamental requirements for improving leadership scientific software?
  - How can we collaborate as a broad community in development and use?
  - Are there other working software ecosystems we should learn from?
  - What topics are missing from the conversation?
- We need your engagement in this effort!

Join the conversation

- https://lssw.io: Main portal for the LSSw community
- LSSw Town Hall Meetings:
  - 3rd Thursday each month, 3 – 4:30 pm Eastern US time
- Slack: Share your ideas interactively
- White Papers: Written content for LSSw conversations
  - We need your ideas
  - 2 – 4 page white paper
  - 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

Q&A

- Put questions and comments into Zoom chat
- We will give you the opportunity to unmute to ask in person

Thank you

https://www.exascaleproject.org

This research was supported by the Exascale Computing Project (17-SC-20-SC), a joint project of the U.S. Department of Energy's Office of Science and National Nuclear Security Administration, responsible for delivering a capable exascale ecosystem, including software, applications, and hardware technology, to support the nation's exascale computing imperative.

EXASCALE COMPUTING PROJECT

Thank you to all collaborators in the ECP and broader computational science communities. The work discussed in this presentation represents creative contributions of many people who are passionately working toward next-generation computational science.