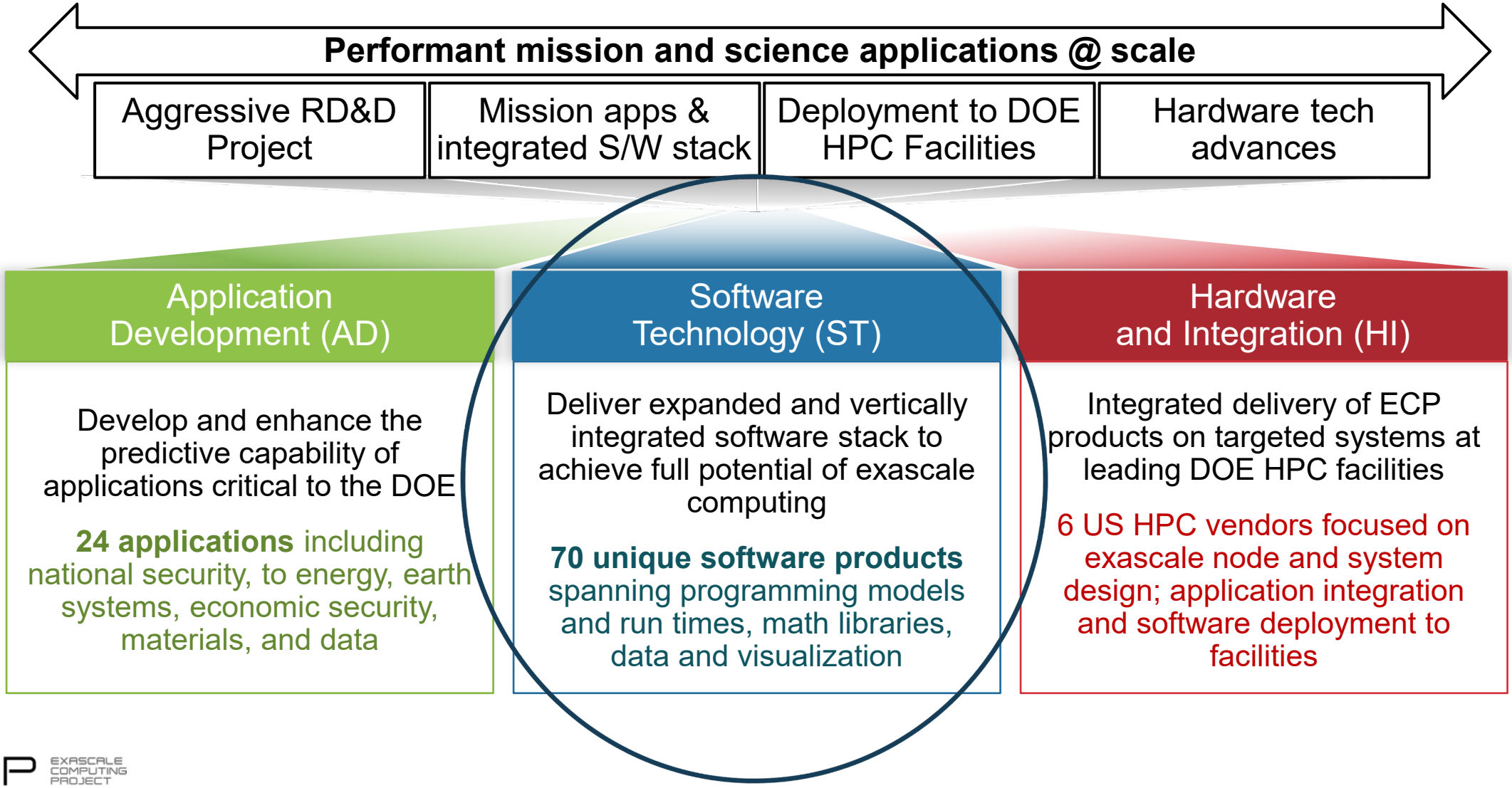
Macro-Engineering Scientific Software

Michael A. Heroux, Sandia National Laboratories
Director of Software Technology, US Exascale Computing Project

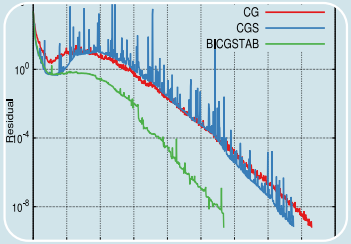
SC20 SWE-CSE BOF
November 17 – 18, 2020

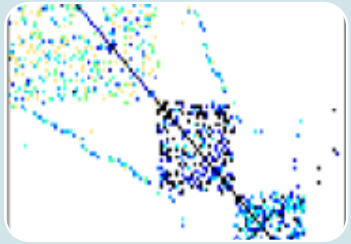
NNSA National Nuclear Security Administration
U.S. DEPARTMENT OF ENERGY | Office of Science

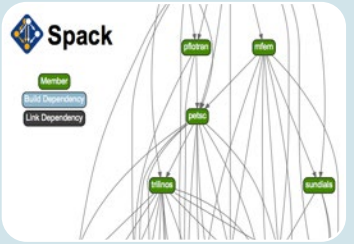
ECP Software Technology (ST) is one of three focus areas



ECP ST has six technical areas







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

ST L4 Teams

- WBS
- Name
- PIs
- PCs - Project Coordinators

ECP ST Stats

- 34 L4 subprojects
- 11 PI/PC same
- 23 PI/PC different
- ~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**	**Hollingsworth, 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	Vuduc, Rich	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	Meza, 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	Bernholdt, David	Carr, Earl
2.3.3.14	ALExa: Accelerated Libraries for Exascale/ForTrilinos	Turner, John	Grundhoffer, Alicia
2.3.4	**Data and Visualization**	**Ahrens, James**	
2.3.4.01	Data and Visualization Software Development Kit	Ahrens, 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 Beyond	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

Extreme-scale Scientific Software Stack (E4S)

- E4S: HPC Linux Ecosystem – a Software Portfolio
- A **Spack-based** distribution of software tested for interoperability and portability to multiple architectures
- Available from **source**, **containers**, **binary caches**
- Leverages and enhances SDK interoperability thrust
- Not a commercial product – an open resource for all

e4s.io

- 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
- Nov 2020: E4S 2.0 - Look for SC20 announcement

Lead: Sameer Shende (U Oregon)

We work on products applications need now and into the future

Key themes:
- Exploration/development of new algorithms/software for emerging HPC capabilities:
- High-concurrency node architectures and advanced memory & storage technologies.
- Enabling access and use via standard APIs.

Software categories:
- The next generation of well-known and widely used HPC products (e.g., MPICH, OpenMPI, PETSc)
- Some lesser used but known products that address key new requirements (e.g., Kokkos, RAJA, Spack)
- New products that 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.

Key quality metric for ECP ST: Capability integration (KPP-3)

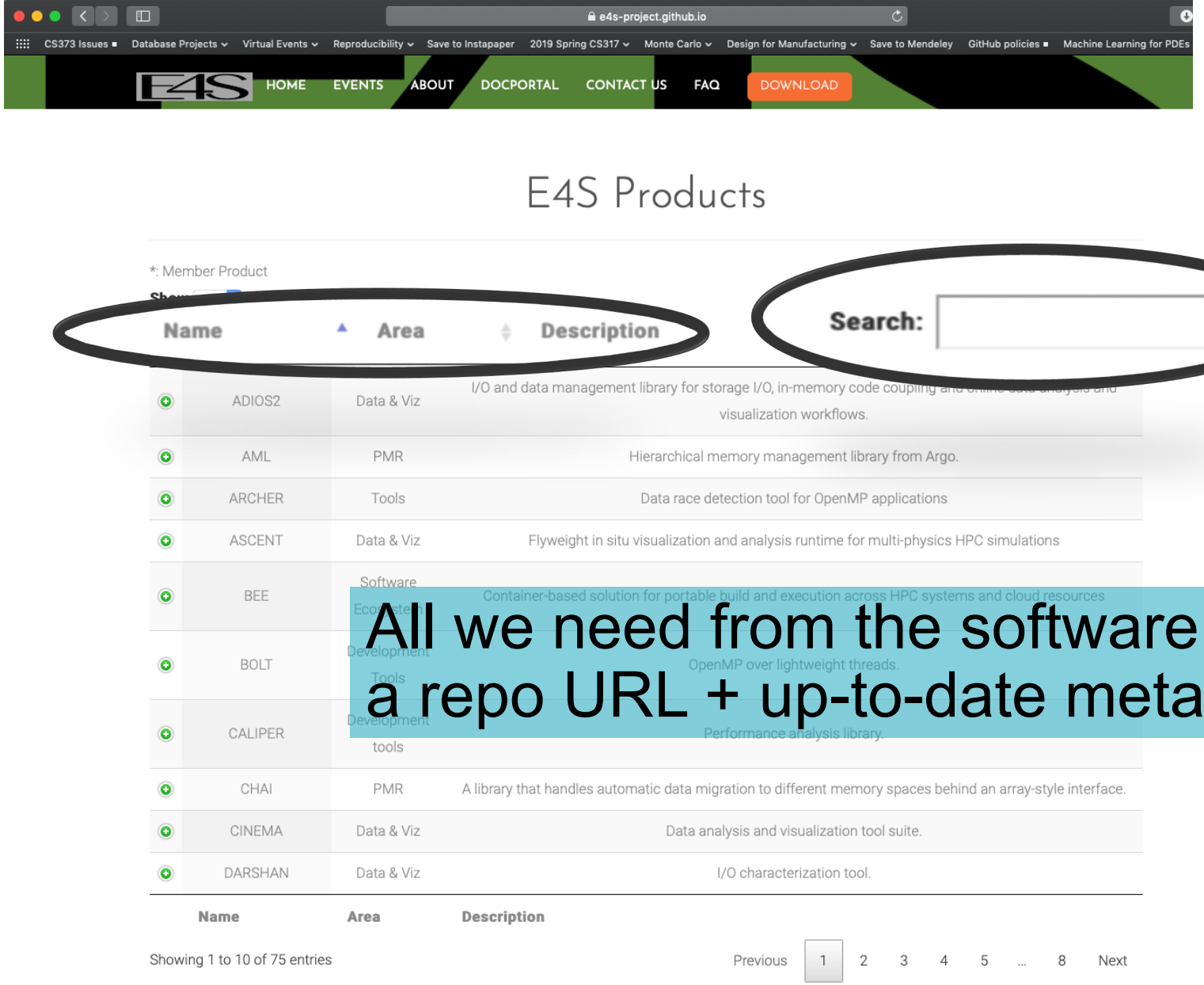
- **Capability:** Any significant product functionality, including existing features adapted to the pre-exascale and exascale environments, that can be integrated into a client environment
- **Capability Integration:** Complete, sustainable integration of a significant product capability into a client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

- **Success:** Achieve 4 – 8 sustainable integrations of capabilities in exascale environments
  - **Not an arms race:** Success comes from achieving modest integration
  - **Integration signals value in the ecosystem:** Focus on collaboration once a part of the ecosystem

Key Performance Parameter (KPP) – Used to manage large projects

E4S DocPortal provide single access point to independent product docs

- Summary Info
  - Name
  - Functional Area
  - Description
  - License
- Searchable
- Sortable



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

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

E4S Community Policies V 1.0 – A community commitment to improving quality

- **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.

- **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.

- **Documentation**

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

- **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.

- **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.

- **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.

- **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.

- **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.

- **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.

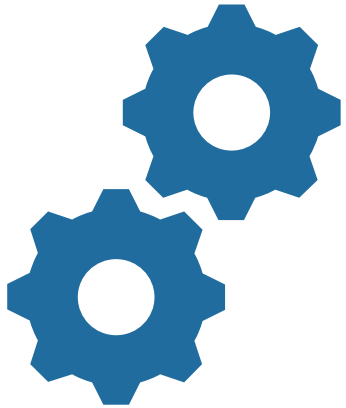
ECP ST Planning Process: Hierarchical, three-phase, cyclical

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 Change Control

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

IDEAS productivity

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

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

2 Incrementally and iteratively improve software practices

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

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)

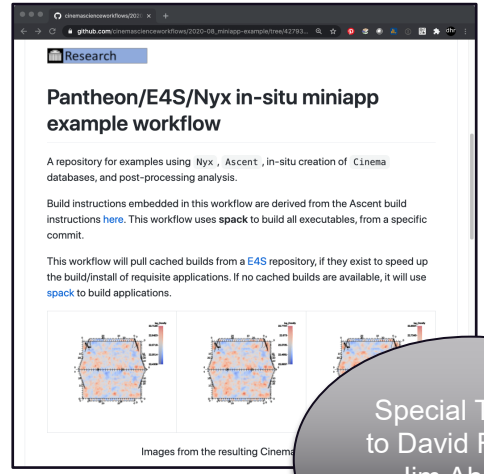
Pantheon and E4S support end-to-end ECP examples

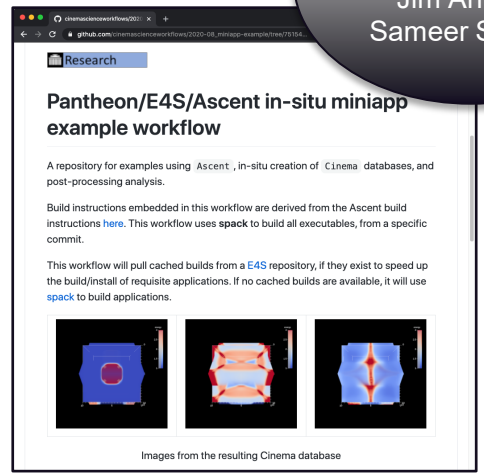
Overview: The Exascale Computing Project (ECP) is a complex undertaking, involving a myriad of technologies working together. An outstanding need is a way to capture, curate, communicate and validate workflows that cross all of these boundaries.

The **Pantheon** and **E4S** projects are collaborating to advance the integration and testing of capabilities, and to promote understanding of the complex workflows required by the ECP project. Utilizing a host of ECP technologies (spack, Ascent, Cinema, among others), this collaboration brings curated workflows to the fingertips of ECP researchers.

Contributions

- Curated end-to-end application/in-situ analysis examples can be run quickly by anyone on Summit. (https://github.com/pantheonscience/ECP-E4S-Examples)
- Pantheon/E4S integration speeds up build/setup times over source builds due to cached binaries (approx.10x speed up).





Special Thanks to David Rogers, Jim Ahrens, Sameer Shende

Instructions page for (top) Nyx, Ascent and Cinema workflow repository, and (bottom) Cloverleaf3d, Ascent, Cinema workflow. These curated workflows use Pantheon, E4S and spack to provide curated workflows for ECP.

Final thoughts

- Curated, turn-key software ecosystems represent important advancement for scientific software
  - Example: Jupyter notebooks & Python ecosystem
  - Goal: E4S becomes one of these ecosystems for HPC (including AI/ML)
- Macro-engineering of advanced HPC capabilities made possible by
  - low-overhead, high value software architectures
    - E4S, SDKs, DocPortal, community policies
  - Quality metrics
    - Capability integration
  - Tailored agile processes
    - Long-term coarse grain refined to near-term fine grain
  - Use of scalable distributed tools and workflows
    - Modern platforms (Jira, Confluence, GitHub, Zoom, etc) essential
  - Research software science
    - Using tools of cognitive and social sciences to improve how we develop and use software for science
- Collaborate with us!