

The Exascale Computing Project's Software Strategy



Approved for public release

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

HPC User Forum
Lugano, Switzerland
October 7, 2019

The Exascale Computing Project (ECP) enables US revolutions in technology development; scientific discovery; healthcare; energy, economic, and national security

ECP mission

Develop exascale-ready applications and solutions that address currently intractable problems of strategic importance and national interest.

Create and deploy an expanded, sustainable and vertically integrated software stack on DOE HPC exascale and pre-exascale systems, defining the enduring US exascale ecosystem.

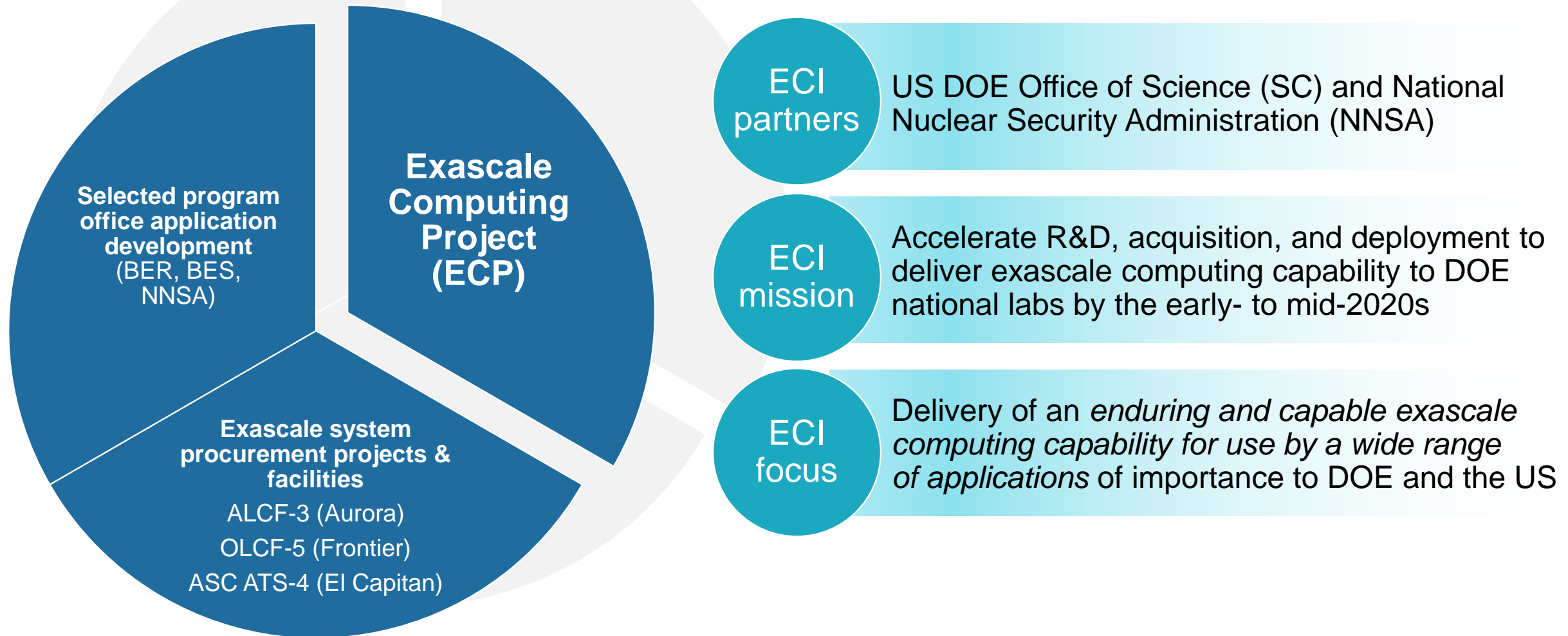
Deliver **US HPC vendor technology advances and deploy ECP products** to DOE HPC pre-exascale and exascale systems.

ECP vision

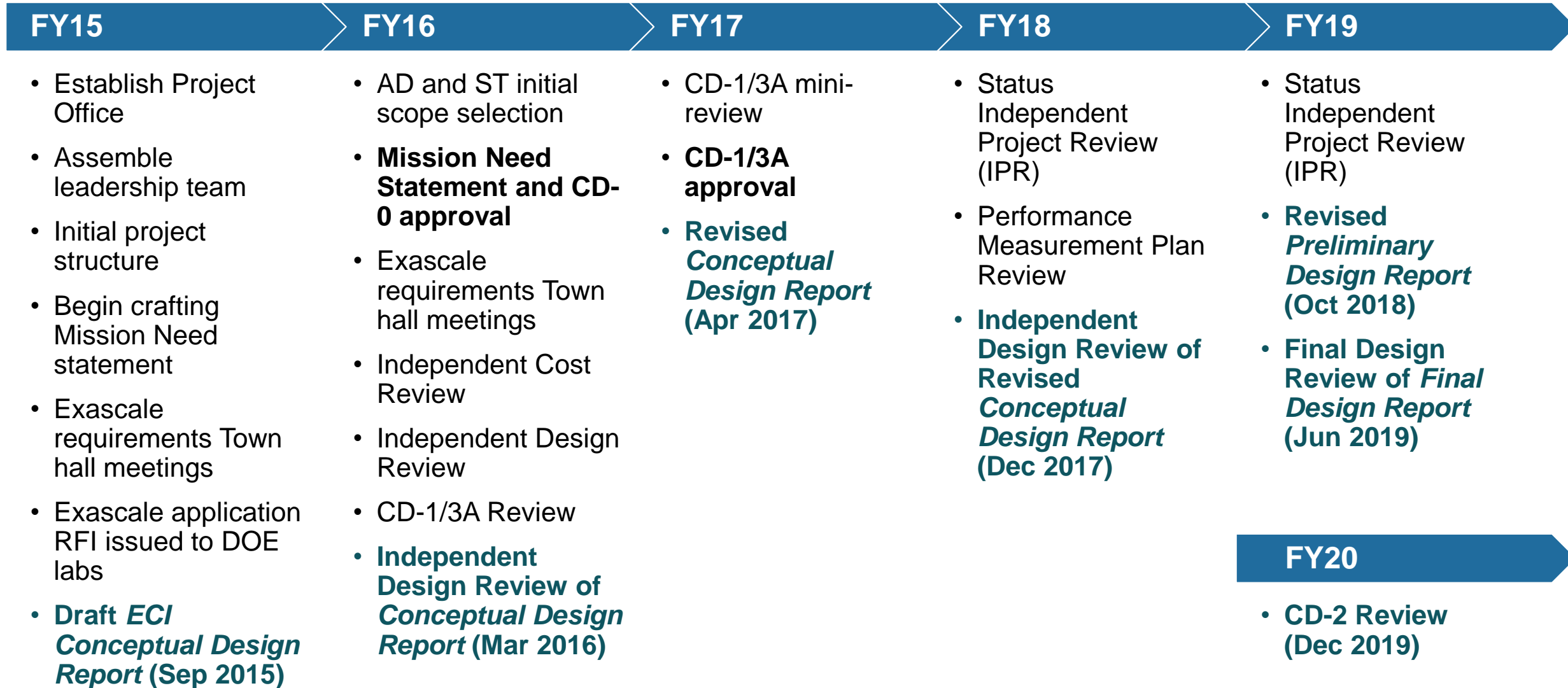
Deliver **exascale simulation and data science innovations and solutions to national problems** that enhance US economic competitiveness, change our quality of life, and strengthen our national security.

DOE Exascale Program: The Exascale Computing Initiative (ECI)

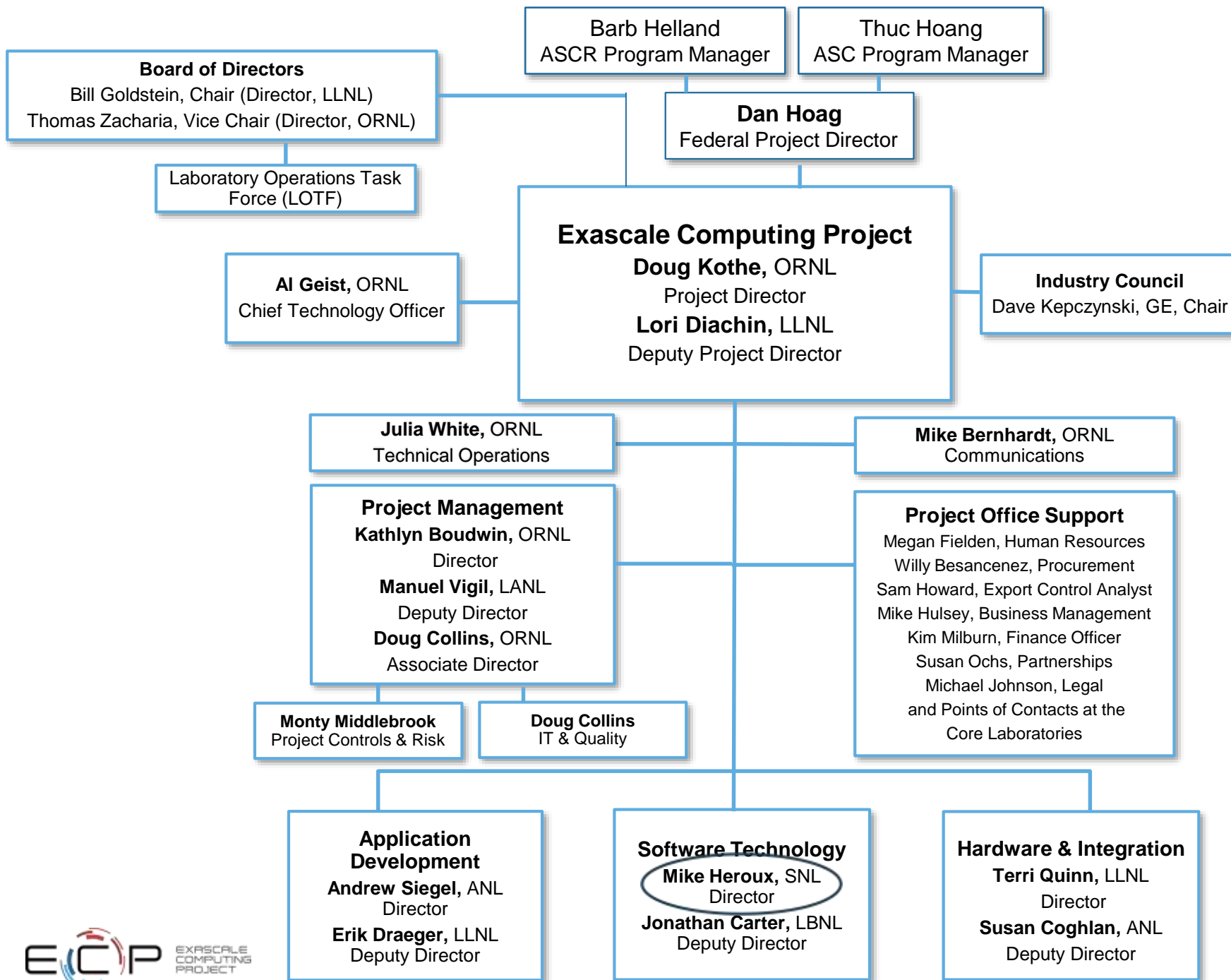
Three Major Components of the ECI



ECP: From inception to present



ECP Organization

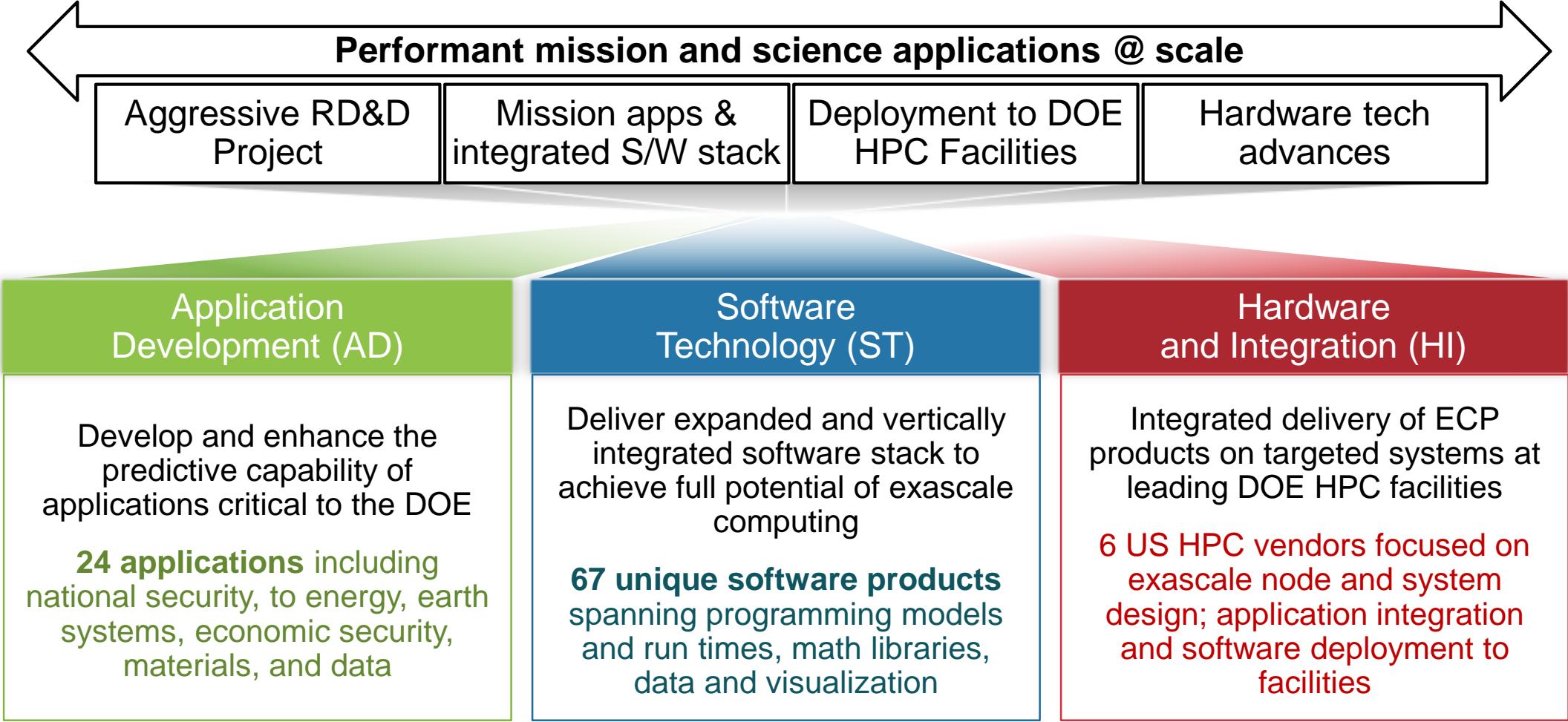


DOE HPC Facilities

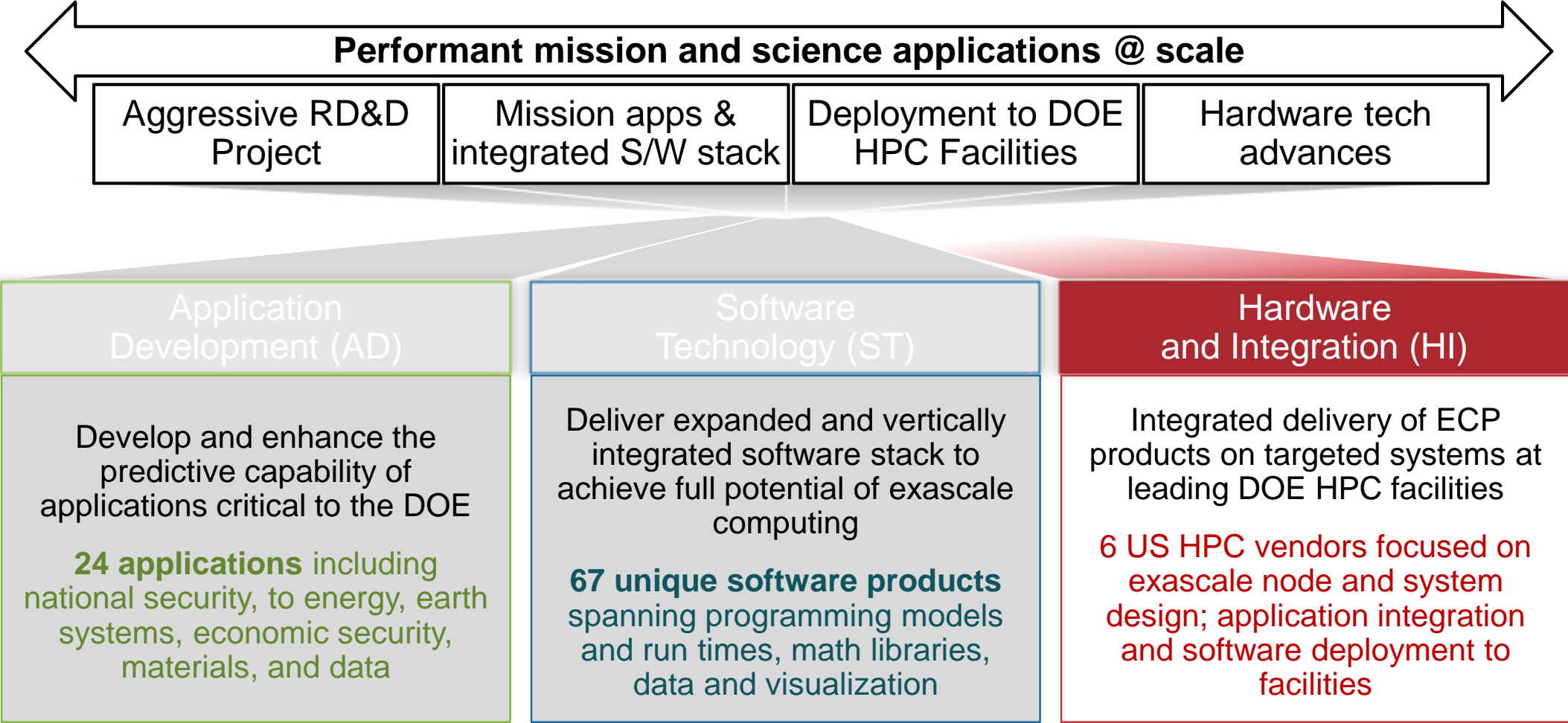
Core Laboratories



ECP's three technical areas have the necessary components to meet national goals



ECP's three technical areas have the necessary components to meet national goals

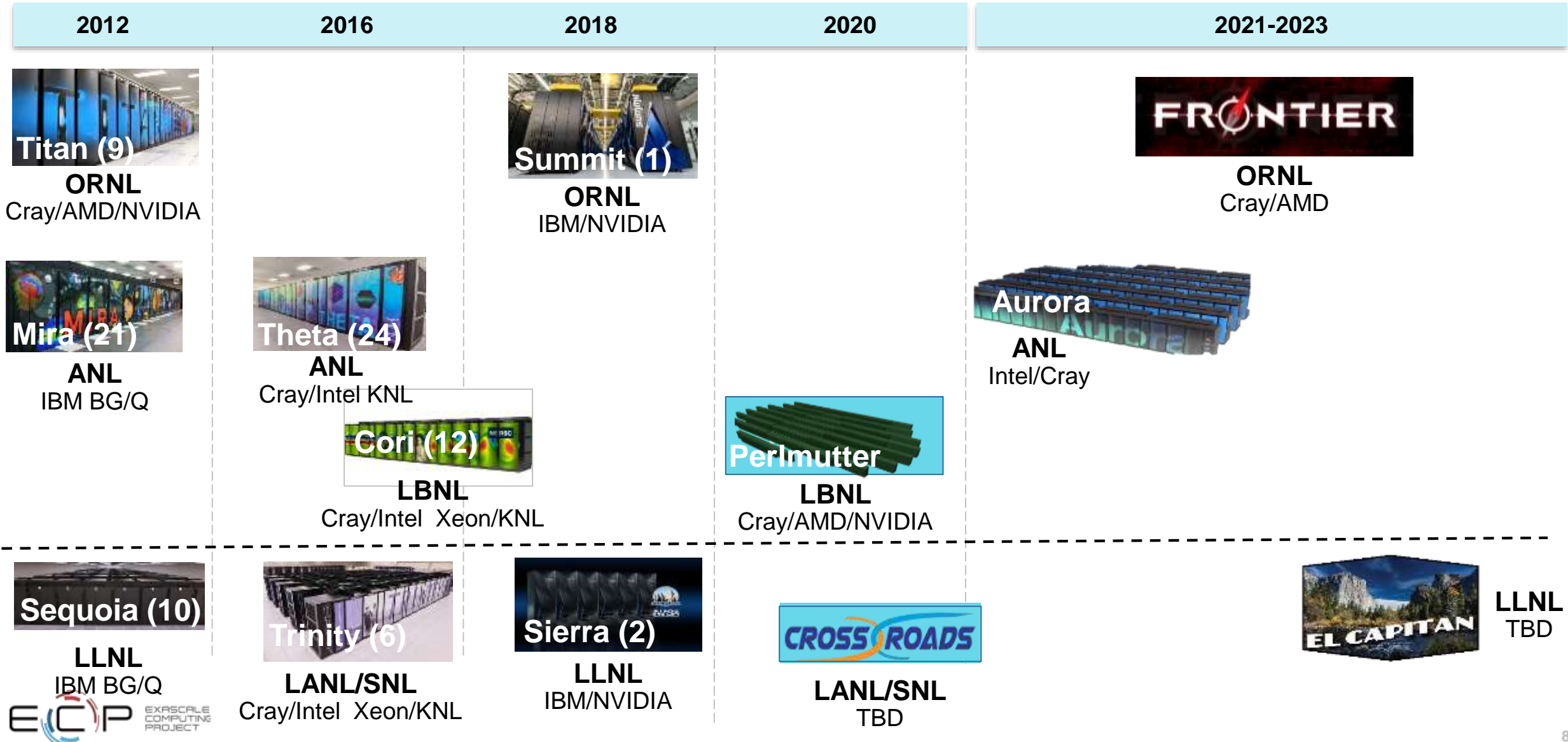


Department of Energy (DOE) Roadmap to Exascale Systems

An impressive, productive lineup of *accelerated node* systems supporting DOE's mission

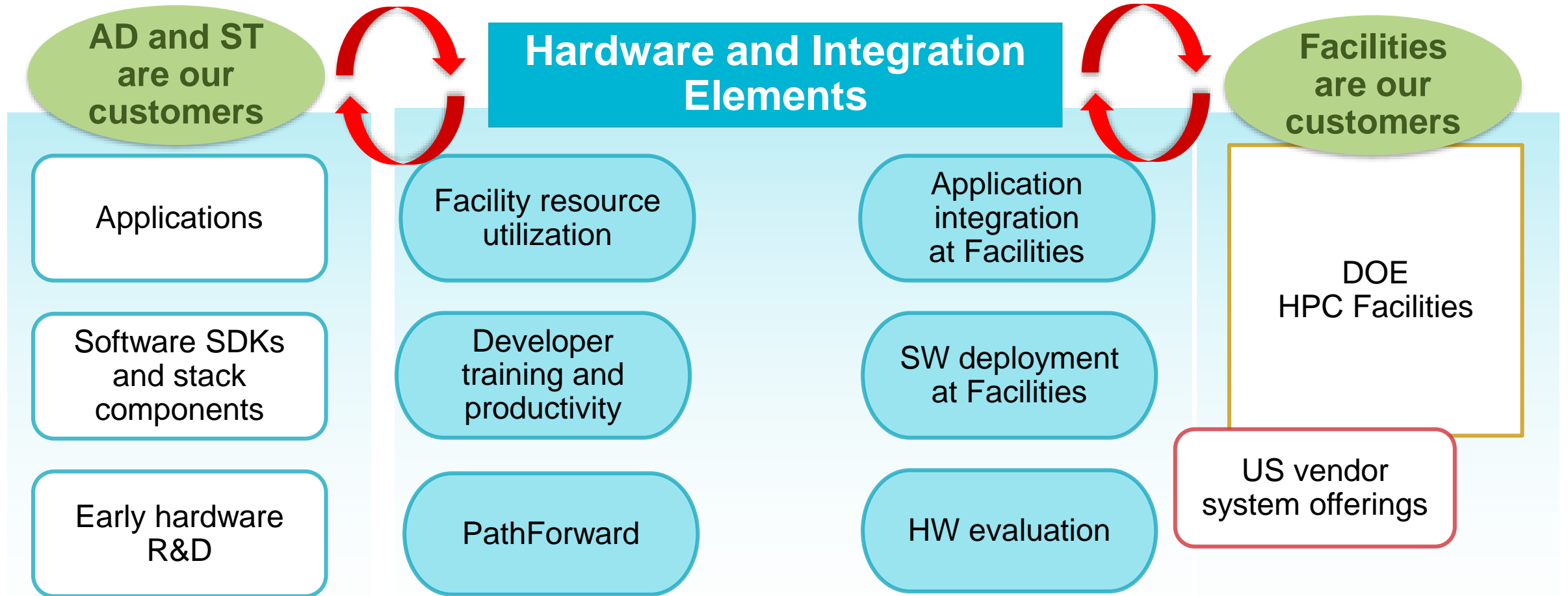
Pre-Exascale Systems [Aggregate Linpack (Rmax) = 323 PF]

First U.S. Exascale Systems

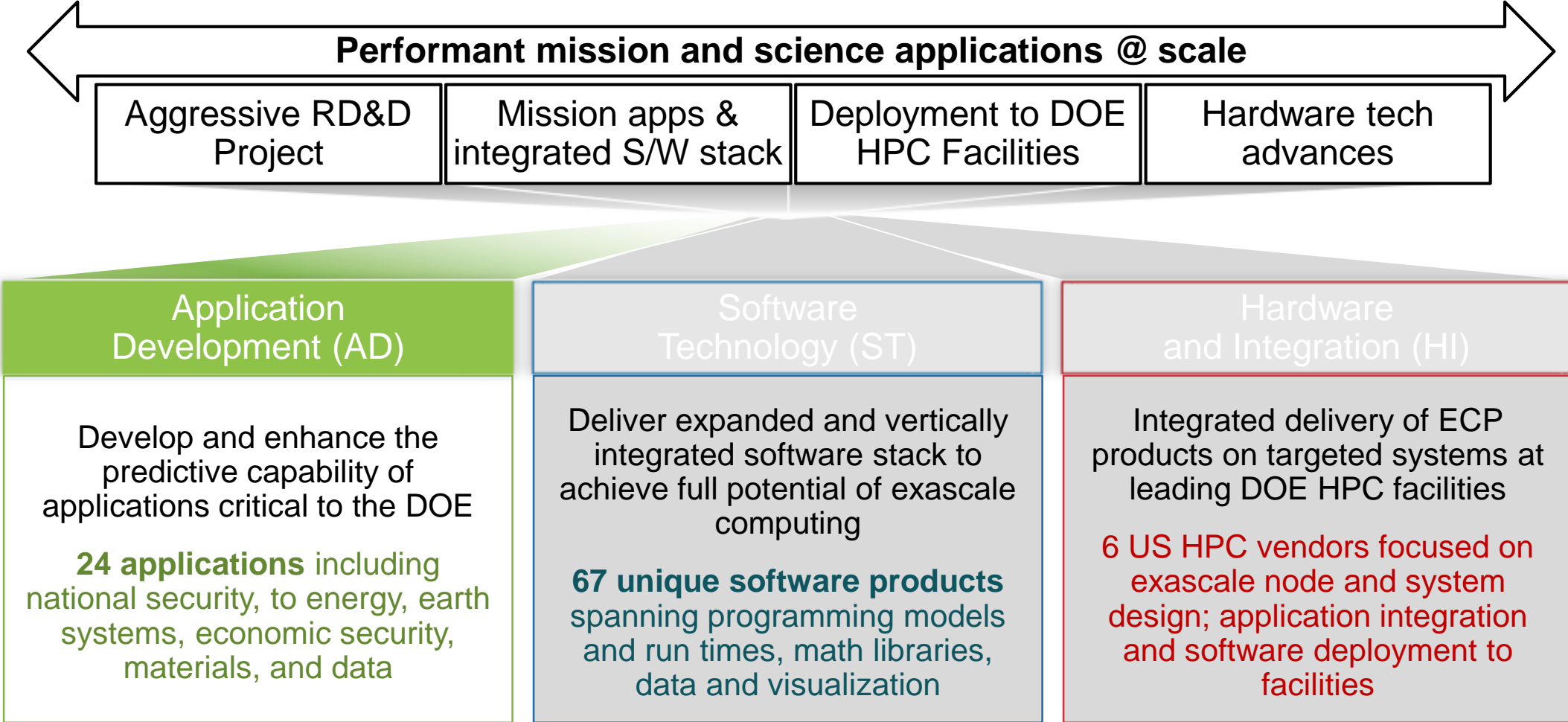


Hardware and Integration is designed to enable integration of ECP's products into HPC environments at the Facilities

ECP will meet its objectives on Facility resources



ECP's three technical areas have the necessary components to meet national goals



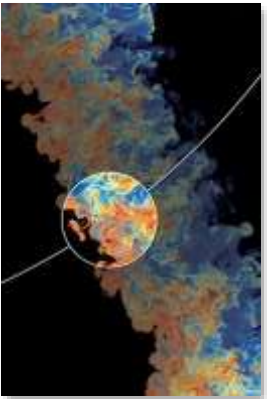
ECP applications target national problems in DOE mission areas

National security

Next-generation, **stockpile stewardship** codes

Reentry-vehicle-environment simulation

Multi-physics science simulations of **high-energy density physics** conditions



Energy security

Turbine **wind plant** efficiency

Design and commercialization of **SMRs**

Nuclear fission and fusion reactor **materials design**

Subsurface use for **carbon capture**, petroleum extraction, waste disposal

High-efficiency, low-emission **combustion engine** and gas turbine design

Scale up of **clean fossil fuel** combustion

Biofuel catalyst design

Economic security

Additive manufacturing of qualifiable metal parts

Reliable and efficient planning of the **power grid**

Seismic hazard risk assessment



Scientific discovery

Cosmological probe of the standard model of particle physics

Validate fundamental laws of nature

Plasma wakefield accelerator design

Light source-enabled **analysis of protein and molecular structure** and design

Find, predict, and control materials and properties

Predict and control **magnetically confined fusion plasmas**

Demystify **origin of chemical elements**

Earth system

Accurate regional impact assessments in **Earth system models**

Stress-resistant crop analysis and catalytic conversion of **biomass-derived alcohols**

Metagenomics for analysis of biogeochemical cycles, climate change, environmental remediation

Health care

Accelerate and translate **cancer research** (partnership with NIH)



ECP Apps: Delivering on Challenge Problems

Requires Overcoming Computational Hurdles

Domain	Challenge Problem	Computational Hurdles
Wind Energy	Optimize 50-100 turbine wind farms	Linear solvers; structured / unstructured overset meshes
Nuclear Energy	Virtualize small & micro reactors	Coupled CFD + Monte Carlo neutronics; MC on GPUs
Fossil Energy	Burn fossil fuels cleanly with CLR	AMR + EB + DEM + multiphase incompressible CFD
Combustion	Reactivity controlled compression ignition	AMR + EB + CFD + LES/DNS + reactive chemistry
Accelerator Design	TeV-class 100-1000X cheaper & smaller	AMR on Maxwell's equations + FFT linear solvers + PIC
Magnetic Fusion	Coupled gyrokinetics for ITER in H-mode	Coupled continuum delta-F + stochastic full-F gyrokinetics
Nuclear Physics: Lattice QCD	Use correct light quark masses for first principle light nuclei properties	Critical slowing down; strong scaling performance of MG-preconditioned Krylov solvers
Chemistry	Heterogeneous catalysis: MSN reactions	HF + DFT + coupled cluster (CC) + fragmentation methods
Chemistry	Catalytic conversion of biomass	Hybrid DFT + CC; CC energy gradients
Extreme Materials	Microstructure evolution in nuclear mats	AMD via replica dynamics; OTF quantum-based potentials
Additive Manufacturing	Born-qualified 3D printed metal alloys	Coupled micro + meso + continuum; linear solvers
Quantum Materials	Predict & control mats @ quantum level	Parallel on-node performance of Markov-chain Monte Carlo
Astrophysics	Supernovae explosions & neutron star mergers	AMR + nucleosynthesis + GR + neutrino transport

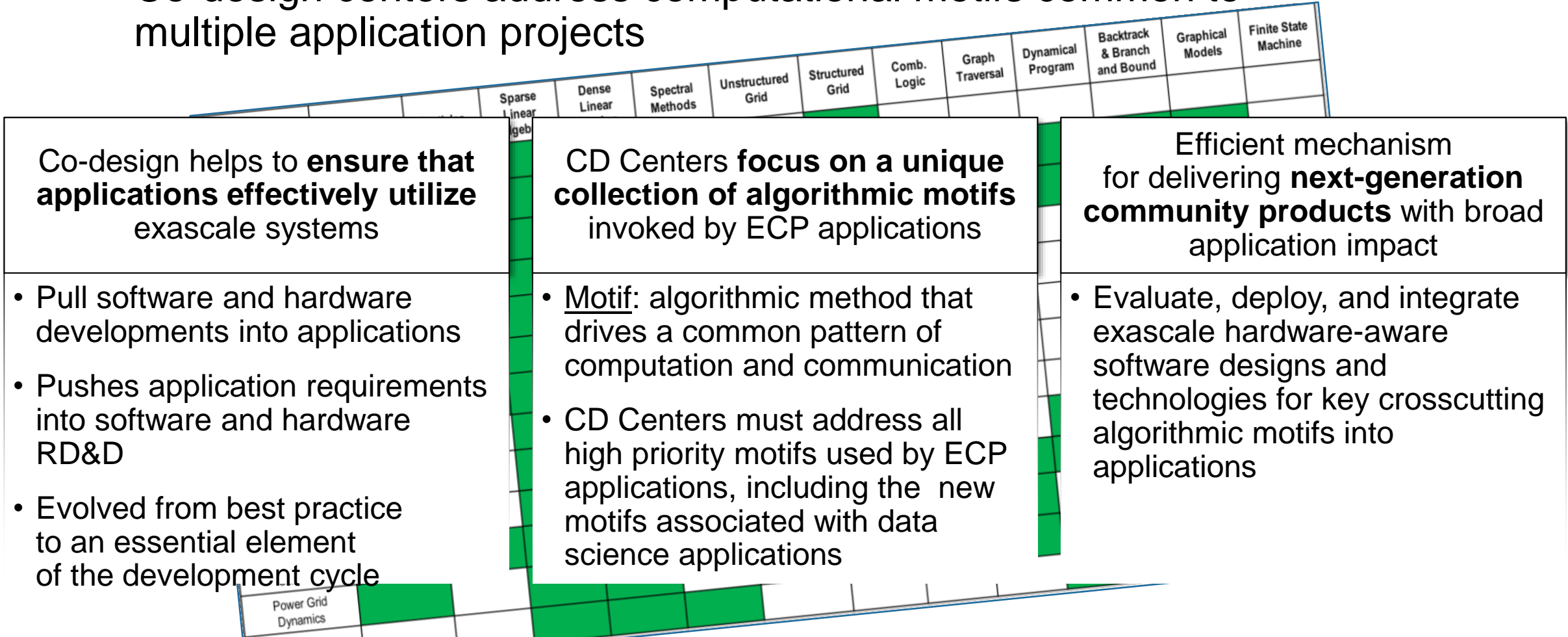
ECP Apps: Delivering on Challenge Problems

Requires Overcoming Computational Hurdles

Domain	Challenge Problem	Computational Hurdles
Cosmology	Extract “dark sector” physics from upcoming cosmological surveys	AMR or particles (PIC & SPH); subgrid model accuracy; insitu data analytics
Earthquakes	Regional hazard and risk assessment	Seismic wave propagation coupled to structural mechanics
Geoscience	Geomechanical and geochemical evolution of a wellbore system at near-reservoir scale	Coupled AMR flow + transport + reactions to Lagrangian mechanics and fracture
Earth System	Assess regional impacts of climate change on the water cycle @ 5 SYPD	Viability of Multiscale Modeling Framework (MMF) approach for cloud-resolving model; GPU port of radiation and ocean
Power Grid	Efficient planning; underfrequency response	Parallel performance of nonlinear optimization based on discrete algebraic equations and MIP
Cancer Research	Predictive preclinical models and accelerate diagnostic and targeted therapy	Increasing accelerator utilization for model search; exploiting reduced/mixed precision; preparing for any data management or communication bottlenecks
Metagenomics	Discover, understand (find genes) and control species in microbial communities	Efficient and performant implementation of UPC, UPC++, GASNet; graph algorithms; SpGEMM performance
FEL Light Source	Light source-enabled analysis of protein and molecular structure and design	Strong scaling (one event processed over many cores) of compute-intensive algorithms (ray tracing, M-TIP) on accelerators

Co-design Projects

- Co-design centers address computational motifs common to multiple application projects



<p>CODAR <i>Data and workflows</i></p>	<p>COPA <i>Particles/mesh methods</i></p>	<p>AMReX <i>Block structured AMR</i></p>	<p>CEED <i>Finite element discretization</i></p>	<p>ExaGraph <i>Graph-based algorithms</i></p>	<p>ExaLearn <i>Machine Learning</i></p>
---	--	---	---	--	--

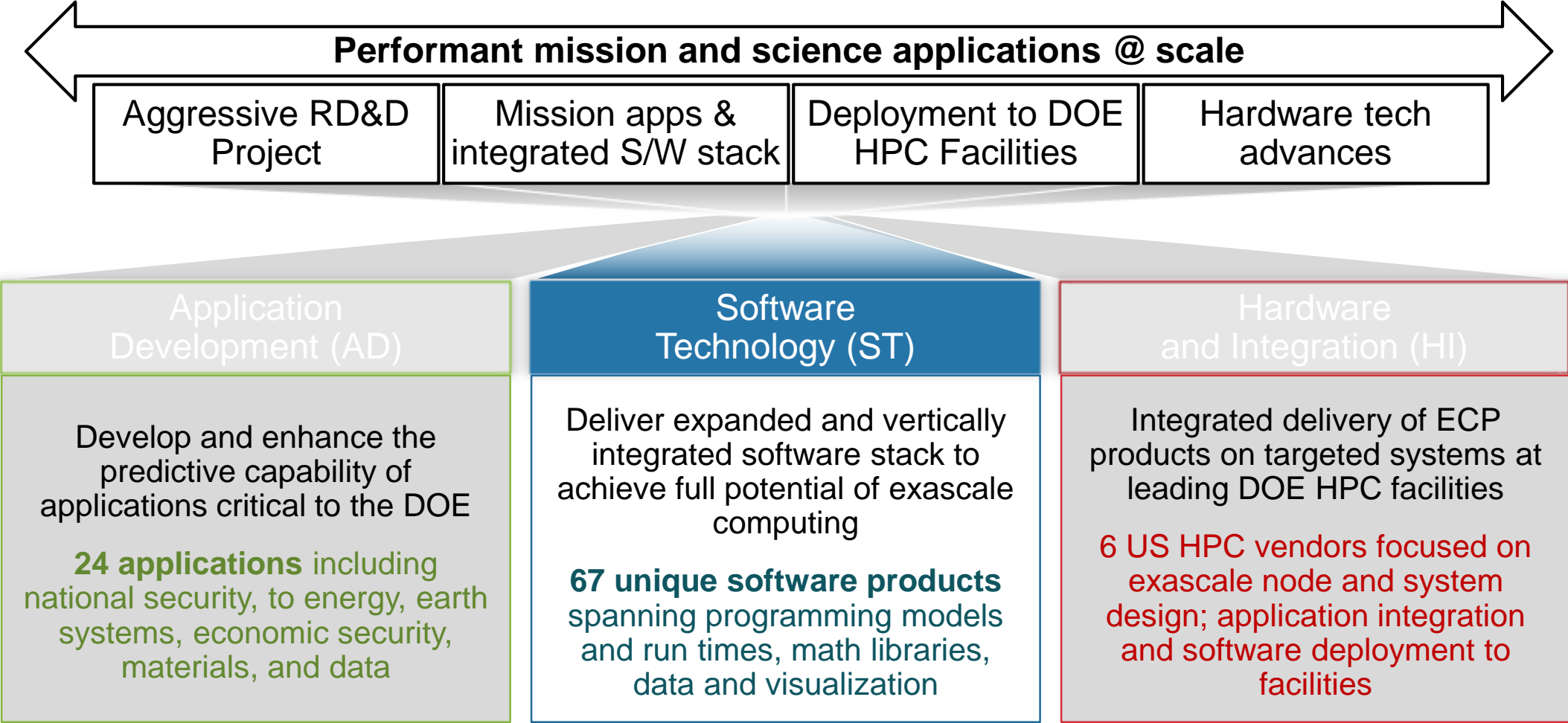
Co-design centers each impact multiple applications

Co-design	Application
CEED	ExaSMR, LLNL NNSA App, <i>ExaAM</i> , <i>ExaWind</i> , <i>Combustion-PELE</i> , <i>Subsurface</i> , <i>E3SM</i> , <i>SNL NNSA App</i>
AMReX	ExaAM, <i>Combustion-PELE</i> , <i>MFIX-Exa</i> , <i>WarpX</i> , <i>ExaStar</i> , <i>ExaSky</i>
CoPA	EXAALT, ExaAM, <i>WDMApp</i> , <i>MFIX-Exa</i> , <i>WarpX</i> , <i>ExaSky</i> , <i>AMReX</i>
CODAR	<i>WDMApp</i> , <i>CANDLE</i> , <i>NWChemEx</i> , <i>EXAALT</i> , <i>Combustion-PELE</i> , <i>ExaSky</i>
ExaGraph	<i>ExaWind</i> , <i>ExaBiome</i> , <i>ExaSGD</i> , <i>SNL NNSA App</i>
ExaLearn	<i>ExaSky</i> , <i>CANDLE</i> , <i>NWChemEx</i> , <i>ExaAM</i>

Common R&D activities/challenges that applications face

- 1) Porting to accelerator-based architectures**
- 2) Exposing additional parallelism**
- 3) Coupling codes to create new multiphysics capability**
- 4) Adopting new mathematical approaches**
- 5) Algorithmic or model improvements**
- 6) Leveraging optimized libraries**

ECP's three technical areas have the necessary components to meet national goals

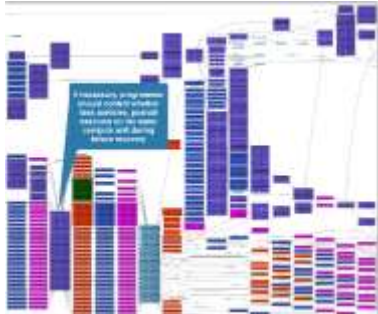


ECP Software Technology (ST)

Goal

Build a comprehensive, coherent software stack that enables application developers to productively write highly parallel applications that effectively target diverse exascale architectures

- Extend current technologies to exascale where possible
- Perform R&D required for new approaches when necessary
- Guide, and complement, and integrate with vendor efforts
- Develop and deploy high-quality and robust software products



The Bottom Line for ECP Software Technology

- Next-generation **HPC technologies contributing to 67 open source scientific software products**
- The performance potential of leadership computers in preparation for exascale
- **Software development kits (SDKs)** with turnkey installation and interoperability
- The **Extreme-scale Scientific Software Stack (E4S)**:
 - Target: Comprehensive software environment for HPC scientific applications
 - Tested on growing collection of HPC platforms in preparation for Exascale systems
 - Managed complexity using SDKs as components
 - From-source builds for leadership environments
 - Pre-built containers for development, debugging and portability
- A commitment to software quality leveraging industry best practices
- A legacy to build upon for US security, science, industry and technology leadership

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.



Jonathan Carter, Software Technology Deputy Director

Jonathan has been involved in the support and development of HPC applications for chemistry, the procurement of HPC systems, and the evaluation of novel computing hardware for over 25 years. He currently a senior manager in Computing Sciences at Berkeley Lab.



Rajeev Thakur, Programming Models and Runtimes (2.3.1)

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



Jeff Vetter, Development Tools (2.3.2)

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



Lois Curfman McInnes, Math Libraries (2.3.3)

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



Jim Ahrens, Data and Visualization (2.3.4)

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



Todd Munson, Software Ecosystem and Delivery (2.3.5)

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



Rob Neely, NNSA ST (2.3.6)

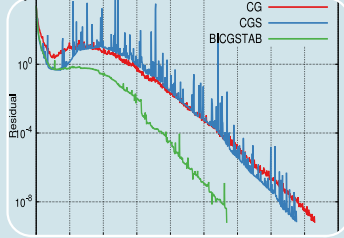
Rob is an Associate Division Leader in the Center for Applied Scientific Computing (CASC) at LLNL, chair of the Weapons Simulation & Computing Research Council, and lead for the Sierra Center of Excellence. His efforts span applications, CS research, platforms, and vendor interactions.

ECP software technologies are a fundamental underpinning in delivering on DOE's exascale mission



Programming Models & Runtimes

- Enhance & prepare OpenMP and MPI programming models (hybrid programming models, deep memory copies) for exascale
- 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 & 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



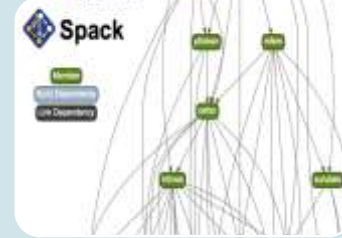
Math Libraries

- 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

- 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
- Open source NNSA Software projects
- Subject to the same planning, reporting and review processes

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. Flang front-end.
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.

ECP ST Subprojects

- WBS
- Name
- PIs
- Project Managers (PMs)

ECP ST Stats

- 33 L4 subprojects
- 10 PI/PC same
- 23 PI/PC different

WBS	WBS Name	CAM/PI	PC
2.3	Software Technology	Heroux, Mike, Carter, J.	-
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	Bayyapu, Neelima
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	Baden, Scott	Hargrove, Paul (and PI)
2.3.1.16	SICM	Lang, Michael	Vigil, Brittney
2.3.1.17	OMPI-X	Bernholdt, David	TBD PMA Through ORNL PMO
2.3.1.18	RAJA/Kokkos	Trott, Christian Robert	Trott, Christian
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	Mellor-Crummey, John
2.3.2.10	PROTEAS-TUNE	Vetter, Jeff	Glassbrook, Dick
2.3.2.11	SOLLVE: Scaling OpenMP with LLVM for Exascale	Chapman, Barbara	Kong, Martin
2.3.2.12	FLANG	McCormick, Pat	Perry-Holby, Alexis
2.3.3	Mathematical Libraries	McInnes, Lois	-
2.3.3.01	Extreme-scale Scientific xSDK for ECP	Yang, Ulrike	Yang, Ulrike
2.3.3.06	Preparing PETSc/TAO for Exascale	Smith, Barry	Munson, Todd
2.3.3.07	STRUMPACK/SuperLU/FFTX: sparse direct solvers, preconditioners, and FFT libraries	Li, Xiaoye	Li, Xiaoye
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	TBD PMA Through ORNL PMO
2.3.4	Data and Visualization	Ahrens, James	-
2.3.4.01	Data and Visualization Software Development Kit	Atkins, Chuck	Atkins, Chuck
2.3.4.09	ADIOS Framework for Scientific Data on Exascale Systems	Klasky, Scott	TBD PMA Through ORNL PMO
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
2.3.6	NNSA ST	Neely, Rob	-
2.3.6.01	LANL ATDM	Mike Lang	Vandenbusch, Tanya Marie
2.3.6.02	LLNL ATDM	Becky Springmeyer	Gamblin, Todd
2.3.6.03	SNL ATDM	Jim Stewart	Trujillo, Gabrielle



ECP ST Product Dictionary List

Widely-recognized product names. Enables mapping between AD & Facilities dependencies and ST development efforts.

- MPI – MPICH, OpenMP
- C++/C/Fortran – LLVM
- Fortran – Flang
- hypre – hypre

ADIOS

AML

Ascent

BLAS

C

C++

Caliper

Catalyst

CHAI

Cinema

CUDA

Darshan

DTK

Dyninst

E4S

FFT

FleCSI

Flux

Fortran

GASNet

Ginkgo

HDF5

HPCToolkit

hypre

Kokkos

KokkosKernels

LAPACK

Legion

libEnsemble

MarFS

MFEM

MPI

OpenACC

OpenCL

OpenMP

PAPI

Papyrus

Paraview

PaRSEC

PETSc/TAO

PnetCDF

PowerStack

RAJA

MPI-IO

ScaLAPACK

SCR

SICM

Spack

SPOT

STRUMPACK

SUNDIALS

SuperLU

SYCL

SZ

TASMANIAN

TAU

Trilinos

UMap

Umpire

Unify

UPC++

VeloC

Visit

VTK-m

xSDK

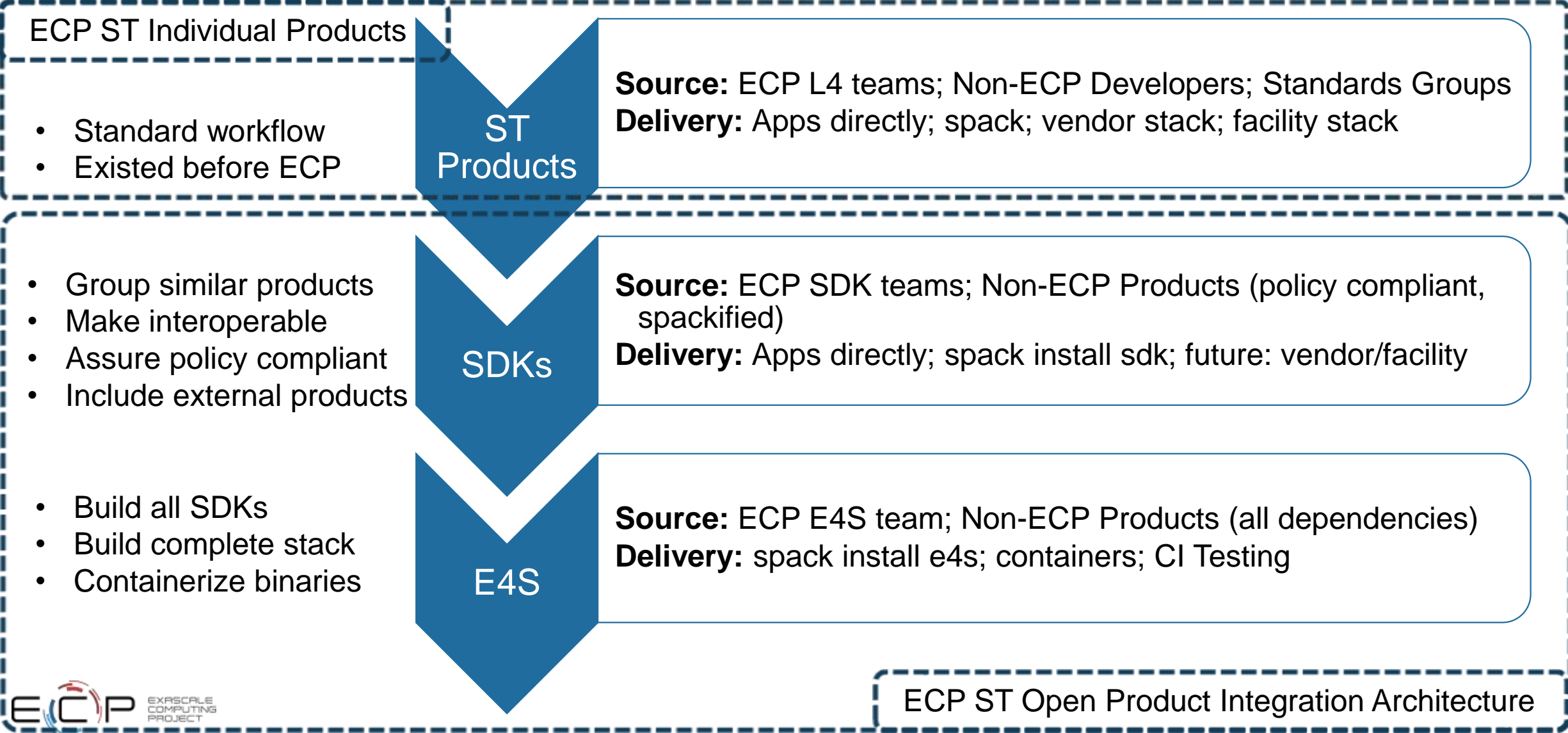
ZFP

ECP SW Technology Software Architecture



Software Technology Ecosystem

Levels of Integration Product Source and Delivery



ECP SW Technology Software Architecture – Software Development Kits



Software Development Kits (SDKs): Key delivery vehicle for ECP

A collection of related software products (packages) where coordination across package teams improves usability and practices, and foster community growth among teams that develop similar and complementary capabilities

- **Domain scope**
Collection makes functional sense
- **Interaction model**
How packages interact; compatible, complementary, interoperable
- **Community policies**
Value statements; serve as criteria for membership
- **Meta-infrastructure**
Invokes build of all packages (Spack), shared test suites
- **Coordinated plans**
Inter-package planning. Augments autonomous package planning
- **Community outreach**
Coordinated, combined tutorials, documentation, best practices

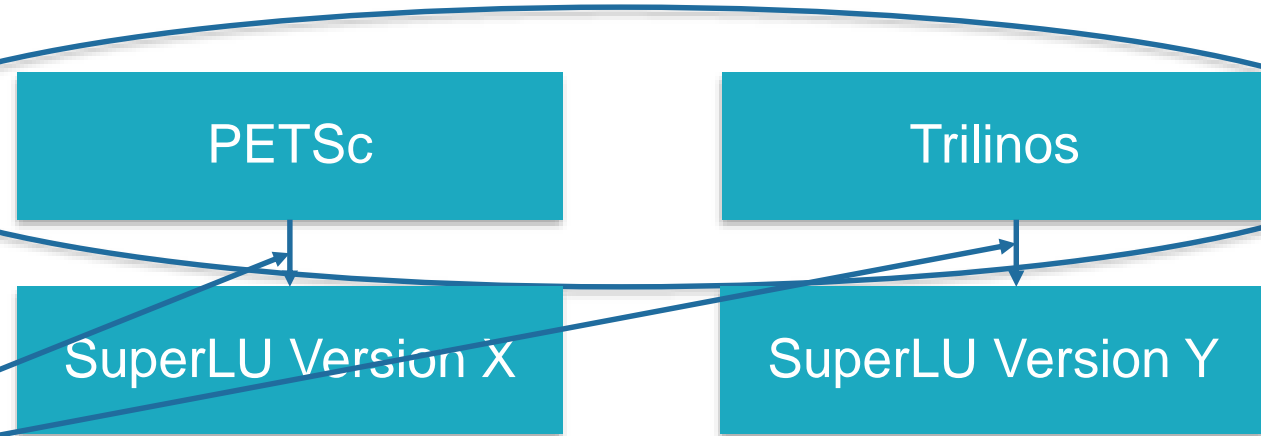
ECP ST SDKs: Grouping similar products for collaboration & usability

Programming Models &
Runtimes Core
Tools & Technologies
Compilers & Support
Math Libraries (xSDK)
Viz Analysis and Reduction
Data mgmt., I/O Services & Checkpoint/
Restart



“Unity in essentials, otherwise diversity”

SDK “Horizontal” Grouping: Key Quality Improvement Driver



Horizontal (vs Vertical) Coupling

- Common substrate
- Similar function and purpose
 - e.g., compiler frameworks, math libraries
- Potential benefit from common Community Policies
 - Best practices in software design and development and customer support
- Used together, but not in the long vertical dependency chain sense
- Support for (and design of) common interfaces
 - Commonly an aspiration, not yet reality

Horizontal grouping:

- Assures $X=Y$.
- Protects against regressions.
- Transforms code coupling from heroic effort to turnkey.

ECP ST SDK community policies: Important team building, quality improvement, membership criteria.

SDK Community Policy Strategy

- Review and revise xSDK community policies and categorize
 - Generally applicable
 - In what context the policy is applicable
- Allow each SDK latitude in customizing appropriate community policies
- Establish baseline policies in FY19 Q2, continually refine

xSDK compatible package: Must satisfy mandatory xSDK policies:

- M1.** Support xSDK community GNU Autoconf or CMake options.
- M2.** Provide a comprehensive test suite.
- M3.** Employ user-provided MPI communicator.
- M4.** Give best effort at portability to key architectures.
- M5.** Provide a documented, reliable way to contact the development team.
- ...

Prior to defining and complying with these policies, a user could not correctly, much less easily, build hypre, PETSc, SuperLU and Trilinos in a single executable: a basic requirement for some ECP app multi-scale/multi-physics efforts.

Recommended policies: encouraged, not required:

- R1.** Have a public repository.
- R2.** Possible to run test suite under valgrind in order to test for memory corruption issues.
- R3.** Adopt and document consistent system for error conditions/exceptions.
- R4.** Free all system resources it has acquired as soon as they are no longer needed.
- R5.** Provide a mechanism to export ordered list of library dependencies.

xSDK member package: An xSDK-compatible package, *that* uses or can be used by another package in the xSDK, and the connecting interface is regularly tested for regressions.

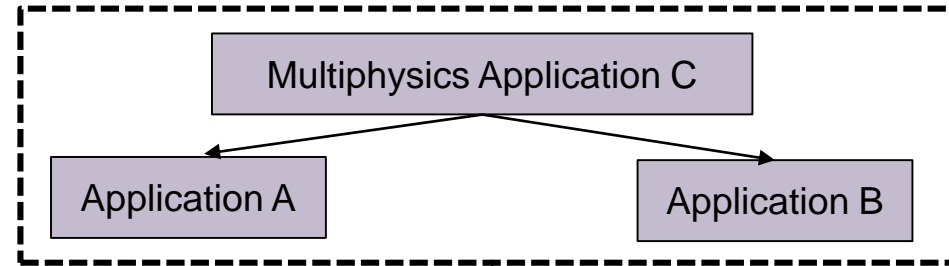
<https://xsdk.info/policies>

Initially the xSDK team did not have sufficient common understanding to jointly define community policies.

xSDK-0.3.0: Dec 2017... (that was then..)

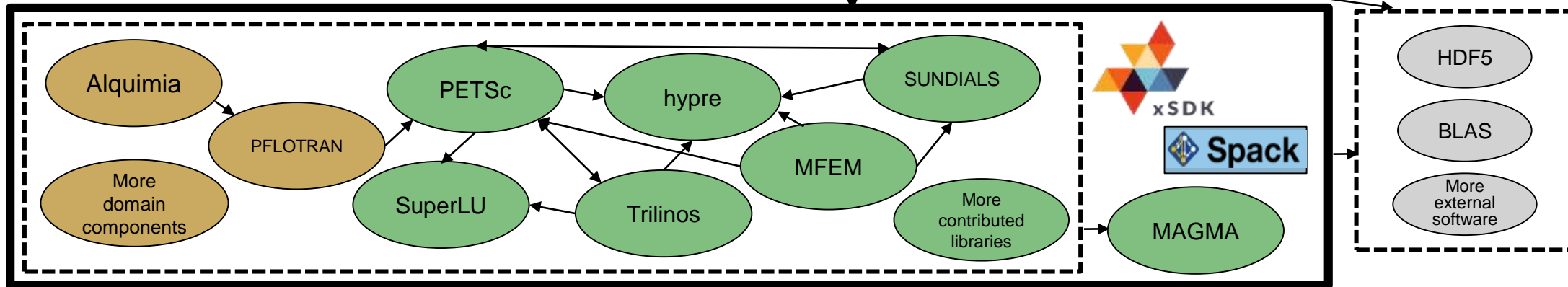
<https://xsdk.info>

Notation: A → B:
A can use B to provide functionality on behalf of A



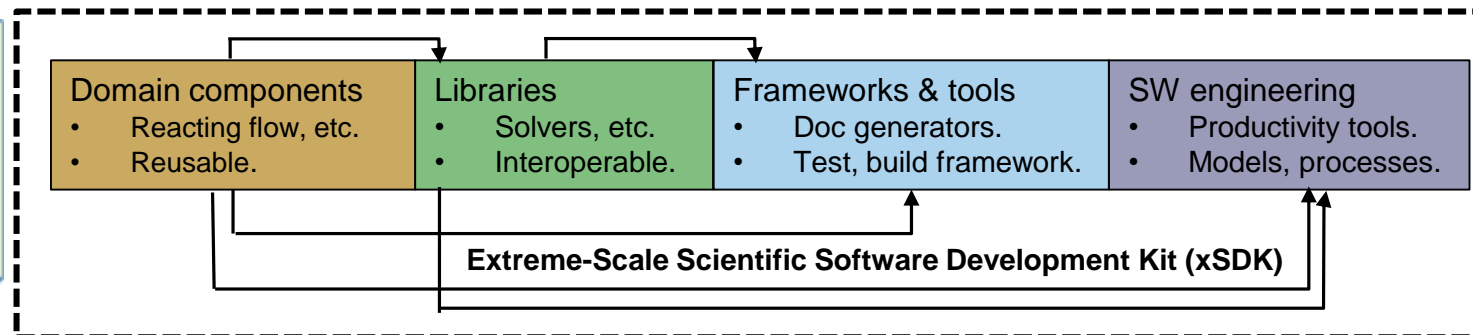
xSDK functionality, Dec 2017

Tested on key machines at ALCF, NERSC, OLCF, also Linux, Mac OS X



**July 2018:
Revisions of xSDK
Community Policies**

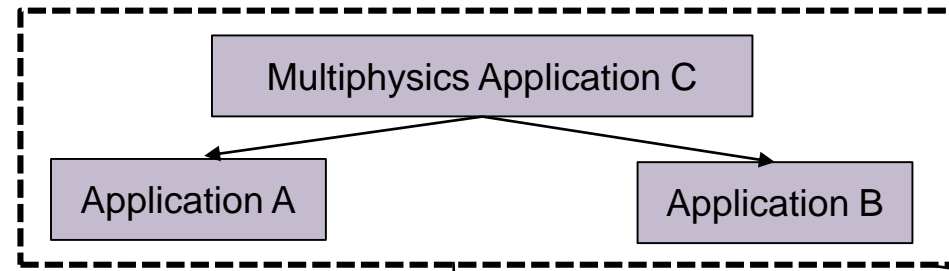
<https://xsdk.info/policies>



xSDK Version 0.4.0: December 2018 (this is now)

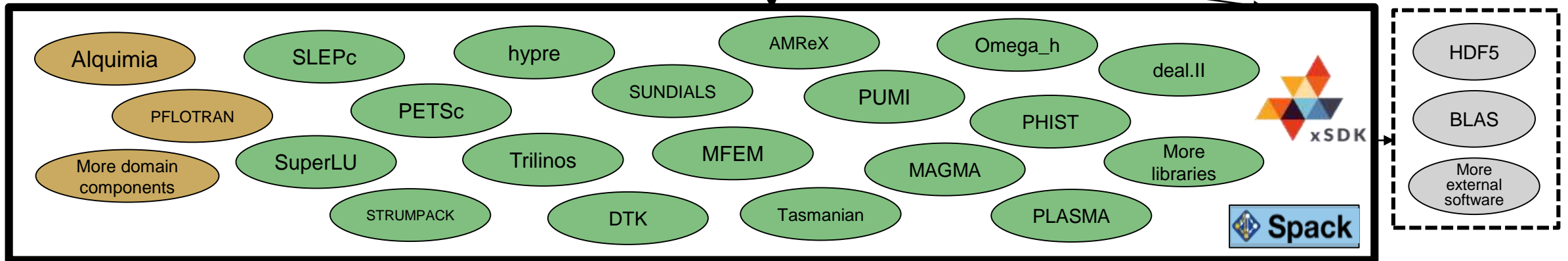
<https://xsdk.info>

Each xSDK member package uses or can be used with one or more xSDK packages, and the connecting interface is regularly tested for regressions.



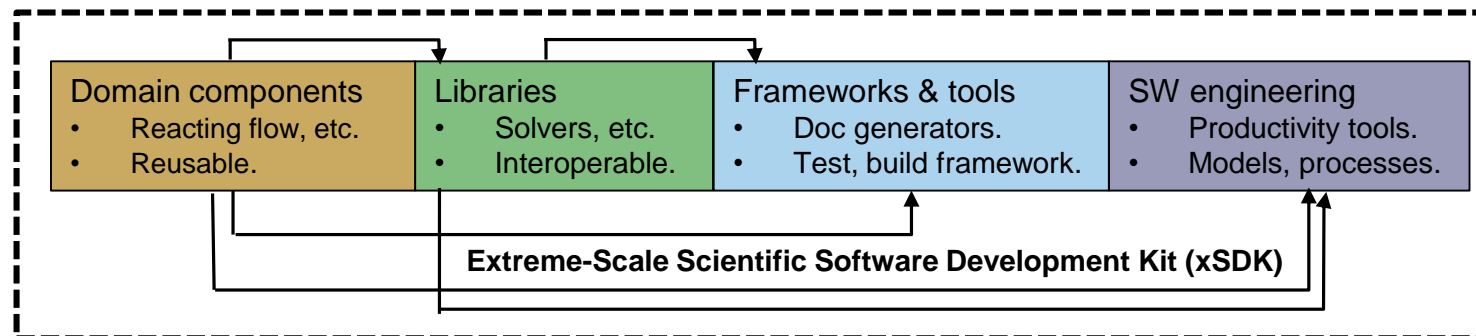
xSDK functionality, Dec 2018

Tested on key machines at ALCF, NERSC, OLCF, also Linux, Mac OS X



December 2018

- 17 math libraries
- 2 domain components
- 16 mandatory xSDK community policies
- Spack xSDK installer

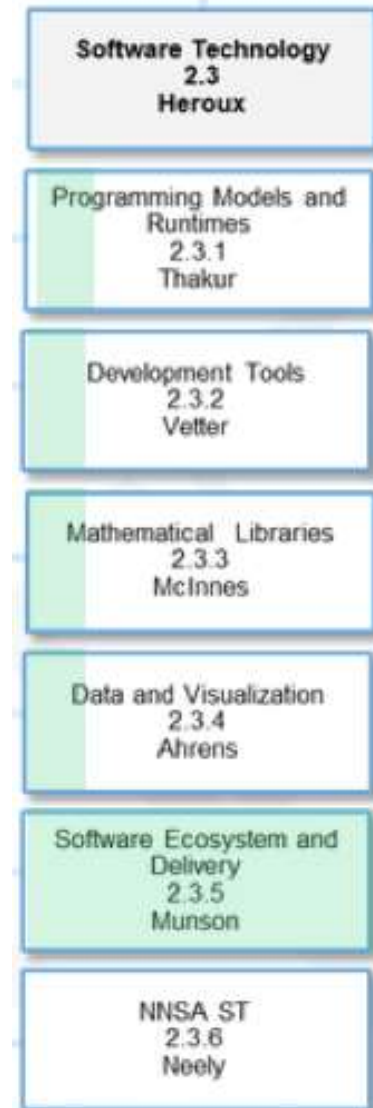


Impact: Improved code quality, usability, access, sustainability

Foundation for work on performance portability, deeper levels of package interoperability

There are 5 L4 projects to define and/or enhance SDKs

- Each L3 area has an L4 project devoted to SDK definition and coordination across the portfolio
- Software ecosystem L4 project focuses on packaging
 - Spack
 - Containers
- Strong coordination with HI Software Deployment projects
- Drives milestone-based planning



ECP ST SDKs will span all technology areas

Motivation: Properly chosen cross-team interactions will build relationships that support interoperability, usability, sustainability, quality, and productivity within ECP ST.

Action Plan: Identify product groupings where coordination across development teams will improve usability and practices, and foster community growth among teams that develop similar and complementary capabilities.

PMR Core (17)	Compilers and Support (7)	Tools and Technology (11)	xSDK (16)	Visualization Analysis and Reduction (9)	Data mgmt, I/O Services, Checkpoint restart (12)	Ecosystem/E4S at-large (12)
QUO	openarc	TAU	hypr	ParaView	SCR	mpiFileUtils
Papyrus	Kitsune	HPCToolkit	FleSCI	Catalyst	FAODEL	TriBITS
SICM	LLVM	Dyninst Binary Tools	MFEM	VTK-m	ROMIO	MarFS
Legion	CHiLL autotuning comp	Gotcha	Kokkoskernels	SZ	Mercury (Mochi suite)	GUFI
Kokkos (support)	LLVM openMP comp	Caliper	Trilinos	zfp	HDF5	Intel GEOPM
RAJA	OpenMP V & V	PAPI	SUNDIALS	VisIt	Parallel netCDF	BEE
CHAI	Flang/LLVM Fortran comp	Program Database Toolkit	PETSc/TAO	ASCENT	ADIOS	FSEFI
PaRSEC*		Search (random forests)	libEnsemble	Cinema	Darshan	Kitten Lightweight Kernel
DARMA		Siboka	STRUMPACK	ROVER	UnifyCR	COOLR
GASNet-EX		C2C	SuperLU		VeloC	NRM
Qthreads		Sonar	ForTrilinos		IOSS	ArgoContainers
BOLT			SLATE		HXHIM	Spack
UPC++			MAGMA			
MPICH			DTK			
Open MPI			Tasmanian			
Umpire			TuckerMPI			
AML						

Legend

- PMR
- Tools
- Math Libraries
- Data and Vis
- Ecosystems and delivery

SDK Summary

- SDKs will help reduce complexity of delivery:
 - Hierarchical build targets.
 - Distribution of software integration responsibilities.
- New Effort: Started in April 2018, fully established in August 2018.
- Extending the SDK approach to all ECP ST domains.
 - SDKs create a horizontal coupling of software products, teams.
 - Create opportunities for better, faster, cheaper – pick all three.
- First concrete effort: Spack target to build all packages in an SDK.
 - Decide on good groupings.
 - Not necessarily trivial: Version compatibility issues, Coordination of common dependencies.
- Longer term:
 - Establish community policies, enhance best practices sharing.
 - Provide a mechanism for shared infrastructure, testing, training, etc.
 - Enable community expansion beyond ECP.

ECP SW Technology Software Architecture – Extreme-scale Scientific Software Stack (E4S)



Extreme-Scale Scientific Software Stack – E4S

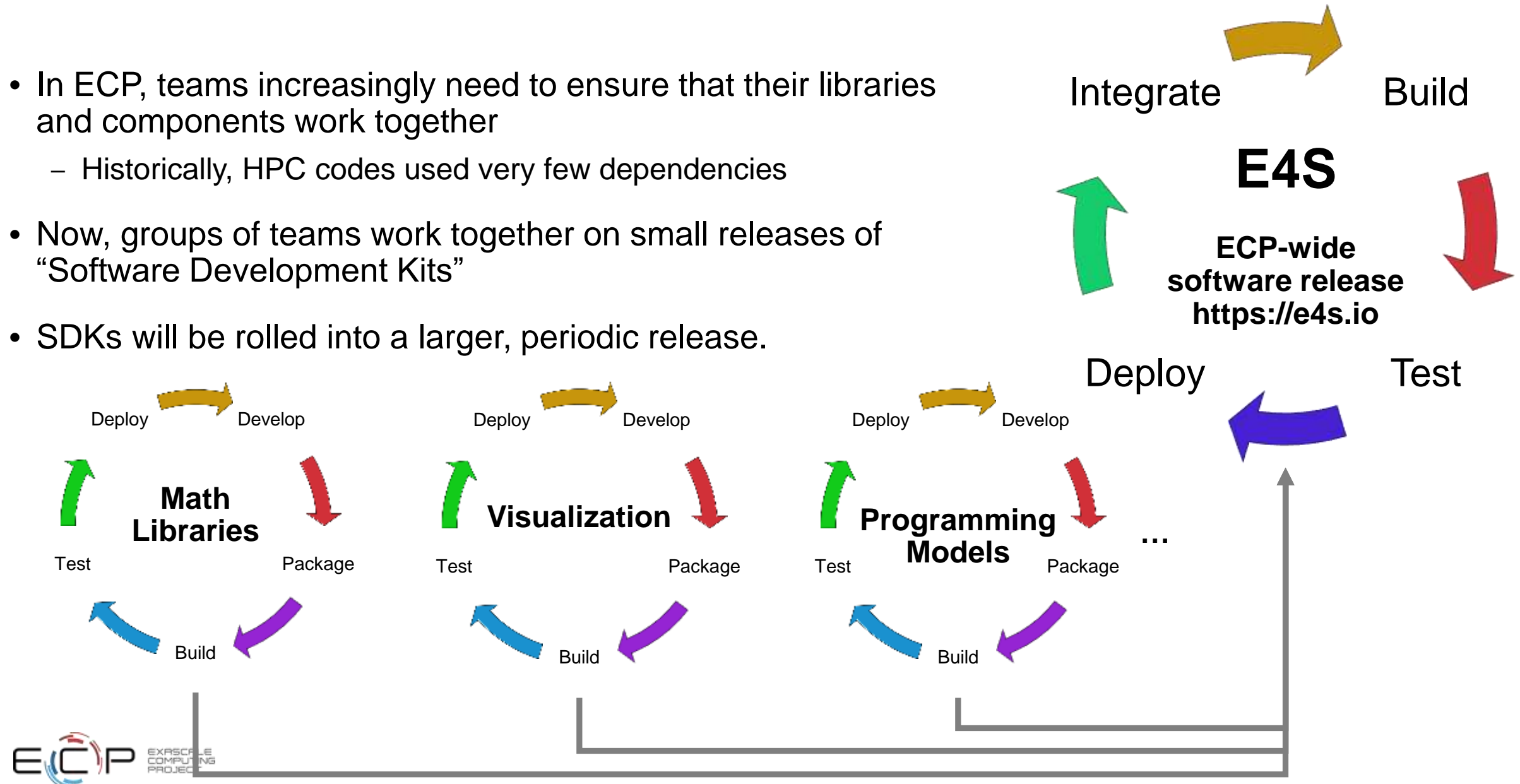
- E4S: A Spack-based distribution of ECP ST and related and dependent software tested for interoperability and portability to multiple architectures
- Provides distinction between SDK usability / general quality / community and deployment / testing goals
- Will leverage and enhance SDK interoperability thrust
- Oct '18: E4S 0.1 - 24 full, 24 partial release products
- Jan '19 : E4S 0.2 - 40 full, 10 partial release products
- Nov '19: Next release, BOF.



<https://e4s.io>

ECP is working towards a periodic, hierarchical release process

- In ECP, teams increasingly need to ensure that their libraries and components work together
 - Historically, HPC codes used very few dependencies
- Now, groups of teams work together on small releases of “Software Development Kits”
- SDKs will be rolled into a larger, periodic release.



Spack enables Software distribution for HPC

- Spack automates the build and installation of scientific software
- Packages are *templated*, so that users can easily tune for the host environment

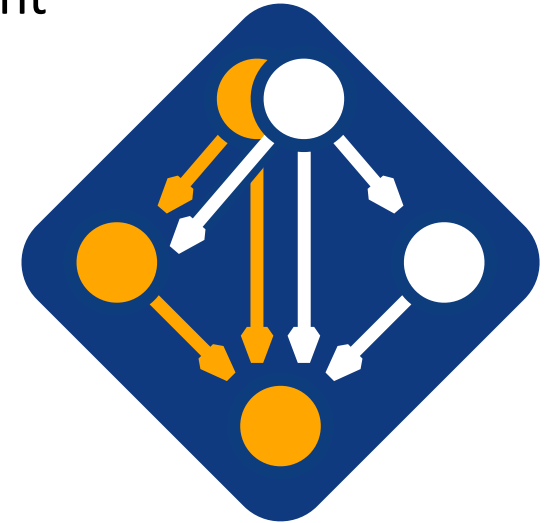
No installation required: clone and go

```
$ git clone https://github.com/spack/spack
$ spack install hdf5
```

Simple syntax enables complex installs

```
$ spack install hdf5@1.10.5
$ spack install hdf5@1.10.5 %clang@6.0
$ spack install hdf5@1.10.5 +threadssafe
$ spack install hdf5@1.10.5 cppflags="-O3 -g3"
$ spack install hdf5@1.10.5 target=haswell
$ spack install hdf5@1.10.5 +mpi ^mpich@3.2
```

- Ease of use of mainstream tools, with flexibility needed for HPC tuning
- Major victories:
 - ARES porting time on a new platform was reduced from **2 weeks to 3 hours**
 - Deployment time for 1,300-package stack on Summit supercomputer reduced from **2 weeks to a 12-hour overnight build**
 - Used by teams across ECP to **accelerate development**



github.com/spack/spack

Spack is being used on many of the top HPC systems

- Official deployment tool for the U.S. Exascale Computing Project
- 7 of the top 10 supercomputers
- High Energy Physics community
 - Fermilab, CERN, collaborators
- Astra (Sandia)
- Fugaku (Japanese National Supercomputer Project)



Fugaku coming to RIKEN in 2021
DOE/MEXT collaboration



Summit (ORNL), Sierra (LLNL)



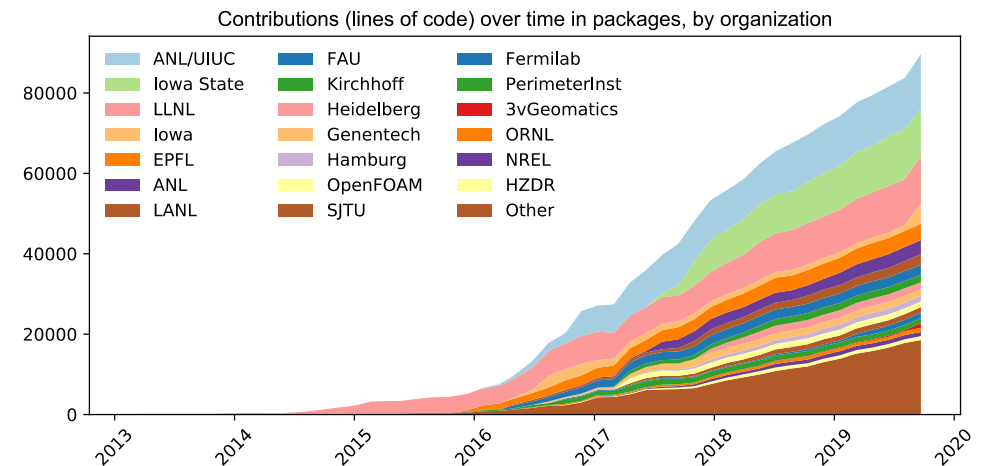
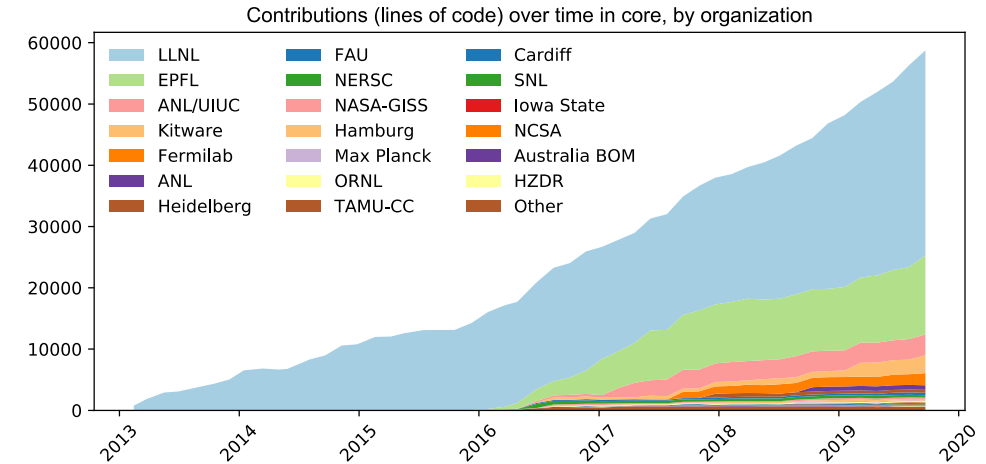
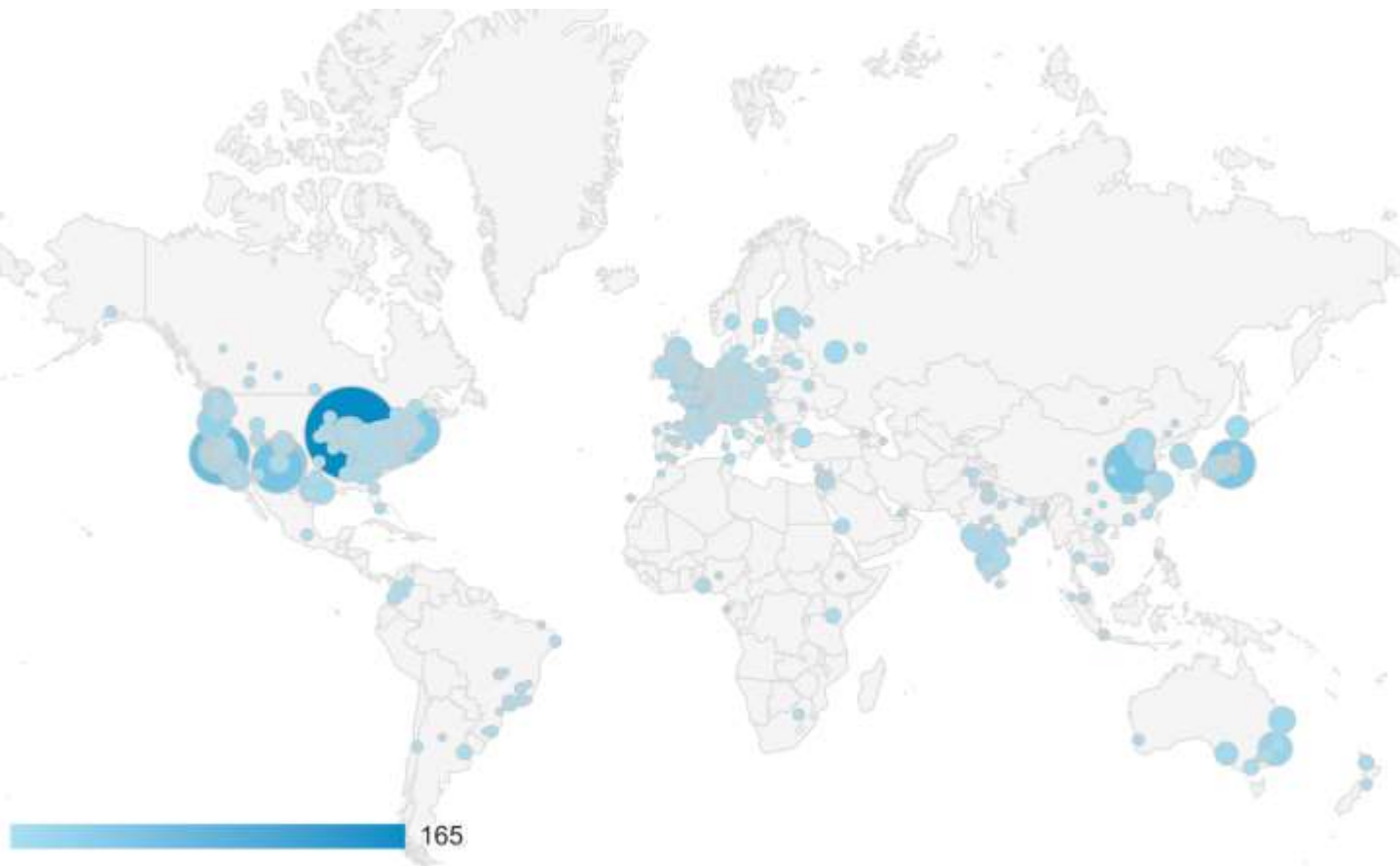
SuperMUC-NG (LRZ,
Germany)



Edison, Cori, Perlmutter (NERSC)

Spack is used worldwide!

Over **3,400** software packages
Over **2,000** monthly active users
Over **400** contributors (and growing)



Active users of Spack documentation site for one month

<https://spack.readthedocs.io>

Spack strategy is to enable exascale software distribution on *both* bare metal and containers

1. New capabilities to make HPC packaging easy and automated

- Optimized builds and package binaries that exploit the hardware
- Workflow automation for facilities, developers, and users
- Strong integration with containers as well as facility deployments

2. Develop exascale enhancements to container runtimes

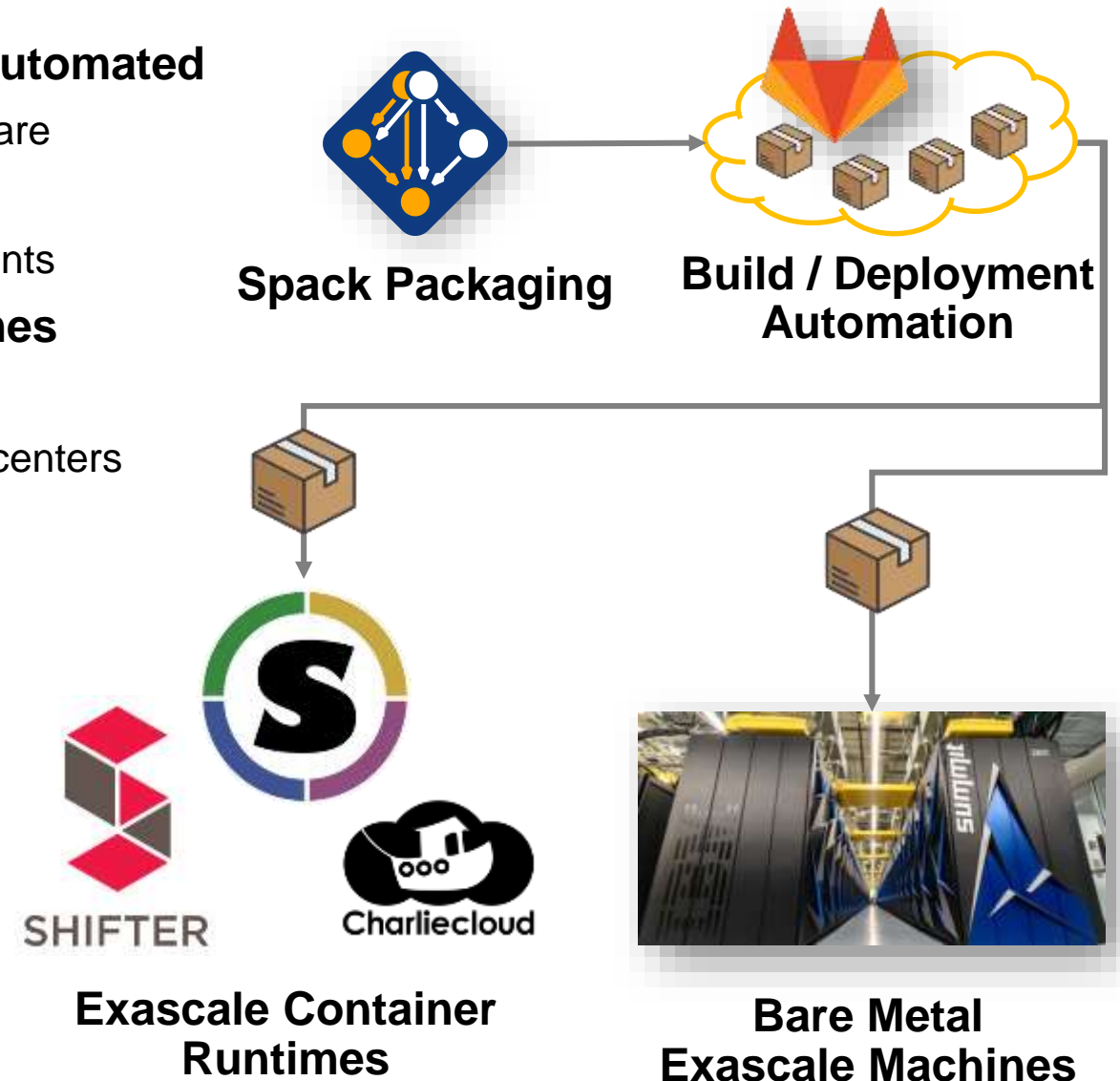
- Understand and automate performance/portability tradeoffs
- Optimized containers & build automation for exascale HPC centers
- Enable portability from laptops to exascale

3. Outreach to users

- Ongoing working groups, Best practice guides
- Tutorials, workshops, BOFs for Spack and Containers

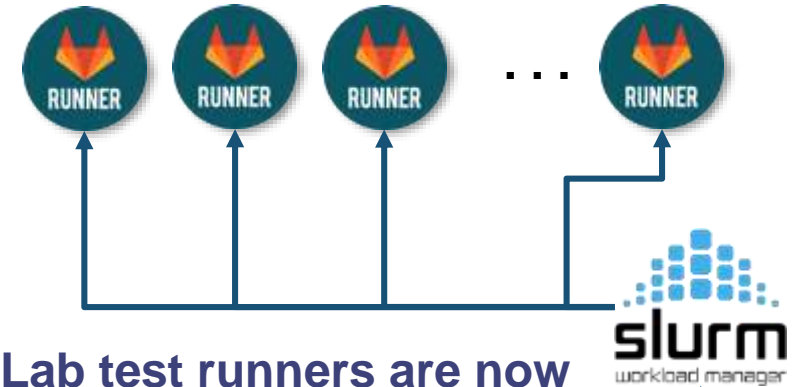
4. Collaboration across ECP

- Work with HI and other areas to build service infrastructure
- Facilitate curation of packages through E4S and facilities
- Ongoing ECP user support

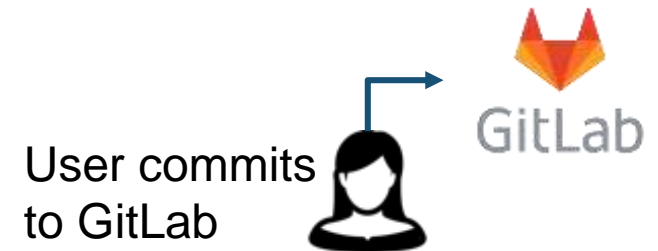


Spack heavily involved in the ECP CI project.

- We have added security features to the open source GitLab product
 - Integration with center identity management
 - Integration with schedulers like SLURM, LSF
- We are democratizing testing at Livermore Computing
 - Users can run tests across 30+ machines by editing a file
 - Previously, each team had to administer own servers
- ECP sites are deploying GitLab CI for users
 - All HPC centers can leverage these improvements
 - NNSA labs plan to deploy common high-side CI infrastructure
 - We are developing new security policies to allow external open source code to be tested safely on key machines

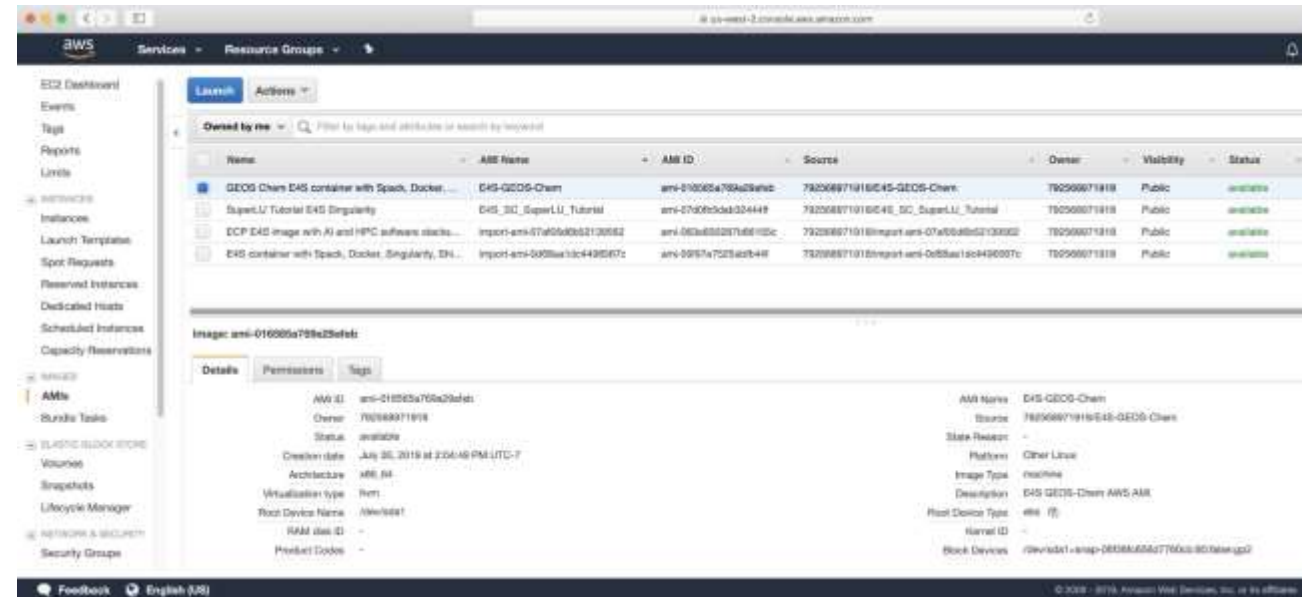


GitLab test runners are now integrated with HPC machines



Integration and Interoperability: E4S on AWS

- E4S AWS public image ami-063e830287b86155c (US-West-2 Oregon) has following container runtimes:
 - Docker
 - Shifter
 - Singularity
 - Charliecloud
- Spack with base PMR components
- E4S full featured Singularity image
 - (exascaleproject/sdk:AHM19)
- Used in ISC-HPC 2019 tutorials
- **Used as base image for NASA GEOS-Chem E4S public image**
- Resources provided by AWS AI/ML team



Reproducible, Customizable Container Builds & Spack Mirrors

- E4S provides base images and recipes for building Docker containers based on SDKs
 - Git: <https://github.com/UO-OACISS/e4s>
 - Base images released (September 2019):
 - UBI 7.6 (RHEL Universal Binary Image for container builds) for x86_64
 - Centos 7.6 for x86_64
 - Ubuntu 18.04 for x86_64
 - UBI 7.6 (RHEL) for ppc64le
- E4S provides **build caches for Spack for native bare-metal as well as container builds based installation** of ST products
 - Build caches: <https://oaciss.uoregon.edu/e4s>
 - **The build cache model can be extended to target platforms**, and can be managed by facilities staff when appropriate.

E4S Build Cache Binaries

Spack E4S Build Cache

Last updated: 20-Sep-2019 11:07 PST

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

1148 Spack binaries in the build cache

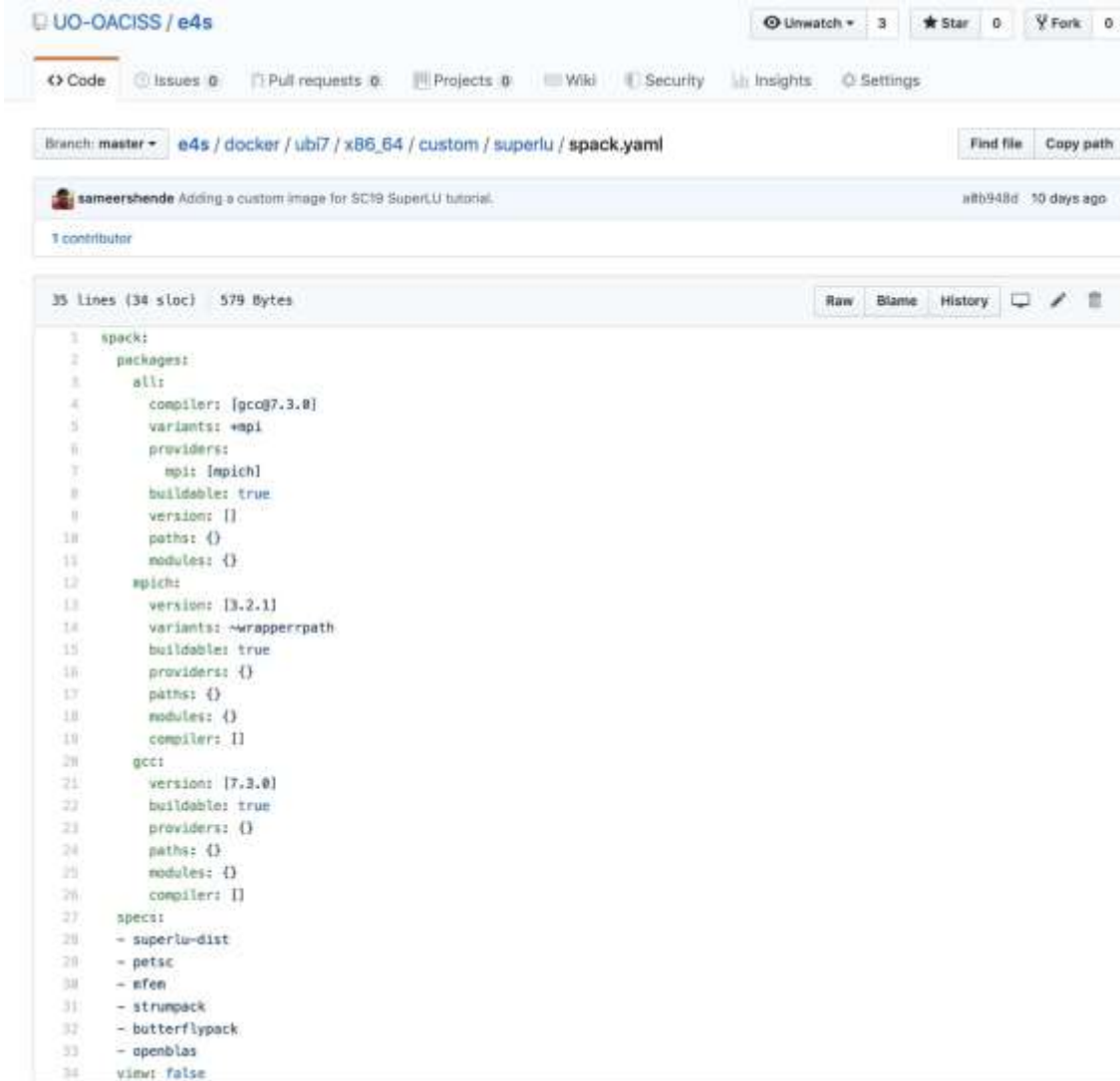
adios2@2.4.0 aml@0.1.0 argobots@1.0rc1 arpack-ng@3.7.0 autoconf@2.69 automake@1.16.1 axl@0.1.1 bdfpc@1.0.5 binutils@2.32 bison@3.0.5 bmi@develop
bolt@1.0rc1 boost@1.70.0 butterflypack@1.0.1 bzip2@1.0.8 c-blosc@1.17.0 cairo@1.16.0 caliper@2.0.1 cinch@develop cmake@3.15.1 cmake@3.15.3
cuda@10.0.130 cuda@10.1.243 curl@7.63.0 darshan-runtime@3.1.7 darshan-util@3.1.7 diffutils@3.7 double-conversion@2.0.1 dtemp@1.1.0
dyninst@10.1.0

Click on the full spec link to find out more.

Link	Arch	OS	Compiler	Created	Full Hash
Full Spec	x86_64	centos7	gcc@7.3.0	18-Sep-2019 19:07	m46bcmvfkvly5iz5iygg4mmta7myiers
Full Spec	x86_64	centos7	gcc@7.3.0	18-Sep-2019 19:11	v2wfu3g3n7x4gndevks2vblmgc53qs7n
Full Spec	x86_64	rhel7	gcc@7.3.0	15-Sep-2019 22:08	cwadhzs6dnelh5dw2kvihtgi5477uxjv
Full Spec	x86_64	rhel7	gcc@7.3.0	15-Sep-2019 22:13	nokyntwy4pocce4ips3pqlam2stf5s7sk
Full Spec	x86_64	ubuntu18.04	gcc@7.3.0	18-Sep-2019 19:16	lke6kplc5tpwxnwlthstek3wpelydy
Full Spec	x86_64	ubuntu18.04	gcc@7.3.0	18-Sep-2019 19:21	e25zc3763g75iaas5wwbsfmuxjybujfz

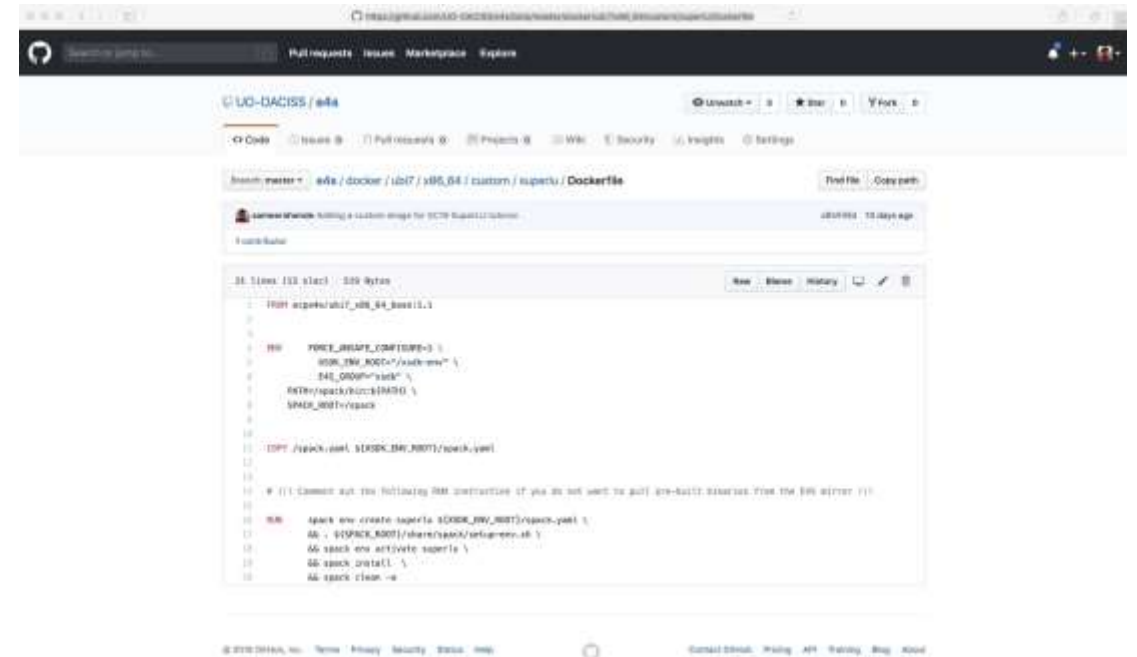
elfutils@0.176 environment-modules@4.3.0 er@0.0.3 expat@2.2.5 faodel@1.1906.1 findutils@4.6.0 flatcc@0.5.3 flecsi@develop flex@2.6.4 font-util@1.3.2
fontconfig@2.12.3 fontproto@2.1.3 freetype@2.9.1 gasnet@2019.3.0 gcc@7.3.0 gdbm@1.18.1 gettext@0.19.8.1 git@2.21.0 glib@2.56.3 glm@0.9.7.1
glproto@1.4.17 gmp@6.1.2 googletest@1.8.1 gotcha@0.0.2 gotcha@1.0.2 gperf@3.0.4 harfbuzz@2.3.1 hdf5@1.10.5 help2man@1.47.8 hpctoolkit@2019.08.14
hwloc@1.11.11 hypre@2.16.0 hypre@2.17.0 icu4c@64.1 inputproto@2.3.2 intel-mkl@2019.3.199 intel-tbb@2019.4 intel-xed@2019.03.01 isl@0.18 jsoncpp@1.9.1

Reproducible Container Builds using E4S Base Images



The screenshot shows a GitHub repository for 'UO-OACISS / e4s'. The file path is 'e4s / docker / ubi7 / x86_64 / custom / superlu / spack.yaml'. A commit by 'sameershende' is visible, titled 'Adding a custom image for SC19 SuperLU tutorial'. The code content is as follows:

```
1 spack:
2   packages:
3     all:
4       compiler: [gcc@7.3.0]
5       variants: +mpi
6       providers:
7         mpi: [mpich]
8       buildable: true
9       version: []
10      paths: {}
11      modules: {}
12    mpich:
13      version: [3.2.1]
14      variants: ~wrapperrpath
15      buildable: true
16      providers: {}
17      paths: {}
18      modules: {}
19      compiler: []
20    gcc:
21      version: [7.3.0]
22      buildable: true
23      providers: {}
24      paths: {}
25      modules: {}
26      compiler: []
27    specs:
28      - superlu-dist
29      - petsc
30      - mfen
31      - strumpack
32      - butterflypack
33      - openblas
34    vim: false
```



The screenshot shows a GitHub repository for 'UO-OACISS / e4s'. The file path is 'e4s / docker / ubi7 / x86_64 / custom / superlu / Dockerfile'. A commit by 'sameershende' is visible, titled 'Adding a custom image for SC19 SuperLU tutorial'. The code content is as follows:

```
1 FROM superlu/ubi7_x86_64_base:1.1
2
3
4 RUN FORCE_REBASE_CONFIGURE=1
5     USE_HW_ROOT="/x86_64/" \
6     S4I_GROUP="s4i" \
7     RKT="/spack/kc:/RKT/" \
8     SPACK_ROOT="/spack"
9
10
11 COPY /spack.yaml $SPACK_ROOT/spack.yaml
12
13 # !! Comment out the following RUN instructions if you do not want to pull pre-built binaries from the E4S mirror !!
14
15 RUN spack env create superlu $SPACK_ROOT/spack.yaml \
16     --with-spack-root=/share/spack/superlu \
17     --spack-etc-activate=superlu \
18     --spack-install="" \
19     --spack-clean=""
```

- PMR SDK base image (UBI 7.6) has Spack build cache mirror and GPG key installed.
- Base image has GCC and MPICH configured for MPICH ABI level replacement (with system MPI).
- Customized container build using binaries from E4S Spack build cache for fast deployment.
- No need to rebuild packages from the source code.
- Same recipe for container and native bare-metal builds with Spack!

Spack Build Caches from E4S Base Images

Index of /e4s/x86_64/build_cache/linux-rhel7-x86_64/gcc-7.3.0

Name	Last modified	Size	Description
Parent Directory			
adobe2-2.4.0/	19-Sep-2019 06:25	-	
arpack-ng-3.7.0/	12-Sep-2019 12:25	-	
autocore-2.69/	17-Sep-2019 17:24	-	
automake-1.16.1/	17-Sep-2019 17:31	-	
aws-9.1.1/	18-Sep-2019 06:29	-	
awscli-2.12/	18-Sep-2019 21:57	-	
bleve-1.9.0/	26-Aug-2019 09:17	-	
boost-1.78.0/	19-Sep-2019 06:16	-	
butterflypack-1.0.1/	12-Sep-2019 12:33	-	
bz2-1.1.0/	17-Sep-2019 17:47	-	
boost-1.11.0/	27-Aug-2019 09:08	-	
caliper-2.0.1/	04-Sep-2019 08:41	-	
catch-devel/	26-Aug-2019 18:51	-	
cmake-1.15.1/	17-Sep-2019 17:53	-	
cmake-3.15.1/	18-Sep-2019 13:17	-	
cuda-10.0.130/	05-Sep-2019 12:59	-	
cuda-10.1.243/	11-Sep-2019 13:42	-	
curl-7.61.0/	17-Sep-2019 18:01	-	
darshan-runtime-3.1.1/	18-Sep-2019 06:22	-	
darshan-util-3.1.1/	27-Aug-2019 09:02	-	
diffutils-3.7/	18-Sep-2019 16:22	-	
dlpack-1.1.0/	19-Sep-2019 06:28	-	
erlang-19.1.2/	15-Sep-2019 22:13	-	
erlang-19.1.3/	15-Sep-2019 21:06	-	
erlang-19.1.4/	17-Sep-2019 17:58	-	
erlang-19.1.5/	19-Sep-2019 06:29	-	
erlang-19.1.6/	17-Sep-2019 17:24	-	
erlang-19.1.7/	19-Sep-2019 06:27	-	
erlang-19.1.8/	17-Sep-2019 17:24	-	
erlang-19.1.9/	16-Sep-2019 14:44	-	
erlang-19.1.10/	28-Aug-2019 08:17	-	
erlang-19.1.11/	27-Aug-2019 11:48	-	
erlang-19.1.12/	27-Aug-2019 10:18	-	
erlang-19.1.13/	17-Sep-2019 17:47	-	
erlang-19.1.14/	17-Sep-2019 17:24	-	
erlang-19.1.15/	17-Sep-2019 18:03	-	
erlang-19.1.16/	29-Aug-2019 12:41	-	
erlang-19.1.17/	27-Aug-2019 11:42	-	
erlang-19.1.18/	27-Aug-2019 09:09	-	
erlang-19.1.19/	10-Sep-2019 13:01	-	
erlang-19.1.20/	15-Sep-2019 06:21	-	
erlang-19.1.21/	28-Aug-2019 08:13	-	
erlang-19.1.22/	15-Sep-2019 22:17	-	
erlang-19.1.23/	15-Sep-2019 18:27	-	
erlang-19.1.24/	17-Aug-2019 09:08	-	
erlang-19.1.25/	29-Aug-2019 12:55	-	
erlang-19.1.26/	27-Aug-2019 21:48	-	
erlang-19.1.27/	05-Sep-2019 13:08	-	
erlang-19.1.28/	10-Sep-2019 12:49	-	
erlang-19.1.29/	10-Sep-2019 12:57	-	
erlang-19.1.30/	28-Aug-2019 08:27	-	
erlang-19.1.31/	10-Aug-2019 07:35	-	
erlang-19.1.32/	27-Aug-2019 10:14	-	
erlang-19.1.33/	18-Sep-2019 06:25	-	
erlang-19.1.34/	27-Aug-2019 10:19	-	
erlang-19.1.35/	17-Sep-2019 17:30	-	
erlang-19.1.36/	10-Sep-2019 12:57	-	
erlang-19.1.37/	27-Aug-2019 11:48	-	
erlang-19.1.38/	10-Sep-2019 12:48	-	
erlang-19.1.39/	27-Aug-2019 09:04	-	
erlang-19.1.40/	05-Sep-2019 13:13	-	
erlang-19.1.41/	10-Sep-2019 12:52	-	
erlang-19.1.42/	27-Aug-2019 11:48	-	
erlang-19.1.43/	17-Sep-2019 17:48	-	
erlang-19.1.44/	28-Aug-2019 08:28	-	
erlang-19.1.45/	10-Sep-2019 12:48	-	
erlang-19.1.46/	17-Sep-2019 17:30	-	
erlang-19.1.47/	10-Sep-2019 13:02	-	
erlang-19.1.48/	27-Aug-2019 09:03	-	
erlang-19.1.49/	27-Aug-2019 09:03	-	
erlang-19.1.50/	27-Aug-2019 10:18	-	
erlang-19.1.51/	17-Sep-2019 17:54	-	
erlang-19.1.52/	27-Aug-2019 11:43	-	
erlang-19.1.53/	27-Aug-2019 09:02	-	

Index of /e4s/ppc64le/build_cache/linux-centos7-ppc64le/gcc-7.3.0

Name	Last modified	Size	Description
Parent Directory			
adobe2-2.4.0/	20-Sep-2019 09:38	-	
aml-2.1.0/	20-Sep-2019 11:03	-	
argobots-1.0rc1/	20-Sep-2019 09:44	-	
autocore-2.69/	18-Sep-2019 12:23	-	
automake-1.16.1/	20-Sep-2019 09:38	-	
aws-9.1.1/	20-Sep-2019 09:37	-	
awscli-2.12/	20-Sep-2019 09:43	-	
bleve-1.9.0/	18-Sep-2019 12:31	-	
boost-1.78.0/	20-Sep-2019 09:44	-	
butterflypack-1.0.1/	18-Sep-2019 12:25	-	
curl-7.61.0/	18-Sep-2019 12:31	-	
darshan-runtime-3.1.1/	20-Sep-2019 09:37	-	
darshan-util-3.1.1/	20-Sep-2019 09:39	-	
diffutils-3.7/	18-Sep-2019 12:31	-	
dtcm-1.1.0/	20-Sep-2019 09:37	-	
environment-modules-4.3.0/	18-Sep-2019 12:30	-	
er-9.0.3/	20-Sep-2019 09:37	-	
erlang-19.1.2/	18-Sep-2019 12:24	-	
erlang-19.1.3/	18-Sep-2019 12:23	-	
erlang-19.1.4/	20-Sep-2019 09:37	-	
erlang-19.1.5/	20-Sep-2019 11:05	-	
erlang-19.1.6/	18-Sep-2019 12:35	-	
erlang-19.1.7/	18-Sep-2019 12:35	-	
erlang-19.1.8/	18-Sep-2019 12:33	-	
erlang-19.1.9/	20-Sep-2019 09:44	-	
erlang-19.1.10/	20-Sep-2019 11:03	-	
erlang-19.1.11/	20-Sep-2019 11:02	-	
erlang-19.1.12/	20-Sep-2019 11:02	-	
erlang-19.1.13/	20-Sep-2019 09:38	-	
erlang-19.1.14/	20-Sep-2019 11:03	-	
erlang-19.1.15/	20-Sep-2019 09:37	-	
erlang-19.1.16/	18-Sep-2019 12:36	-	
erlang-19.1.17/	20-Sep-2019 09:37	-	
erlang-19.1.18/	18-Sep-2019 12:31	-	
erlang-19.1.19/	18-Sep-2019 12:31	-	
erlang-19.1.20/	18-Sep-2019 09:44	-	
erlang-19.1.21/	20-Sep-2019 09:37	-	
erlang-19.1.22/	18-Sep-2019 12:22	-	
erlang-19.1.23/	18-Sep-2019 12:30	-	

- x86_64 build cache
 - 914 binaries
 - 40 GB on disk

- IBM Power 9 (ppc64le) build cache
 - 101 binaries
 - 2.6 GB on disk
 - early stages of effort
 - Initial ARM 64 build cache is underway

ECP SW Technology Other Notable Activities



ECP ST staff contribute to ISO and *de facto* standards groups: Assuring sustainability through standards

ST contributes requirements, analysis, design, prototype and reference implementations to vendor & community products that address ECP mission needs.

- **ECP ST Product contributions:** ST efforts provide fundamental software contributions to all of these projects.
 - Examples: MPICH, OpenMPI, SOLLVE (OpenMP, LLVM)
 - Others: Kokkos/RAJA (C++)
- **MPI/OpenMP:** Several key leadership positions.
- Heavy involvement in all aspects.
- **C++:** Getting HPC requirements considered, contributing working code.
- **Fortran:** Flang front end for LLVM.
- **De facto:** Specific HPC efforts.

Standards Effort	ECP ST Committee Participants
MPI Forum	15
OpenMP	15
BLAS	6
C++	4
Fortran	4
OpenACC	3
LLVM	2
PowerAPI	1

Successes discussed in L3 Technical Break Out

Making Transition to Accelerators: xSDK ECP Math Libraries Porting Efforts



Package	NVIDIA GPU	AMD GPU	Intel Xe
DTK	support (Kokkos)	in progress (Kokkos)	in progress (Kokkos)
Ginkgo	support (CUDA)	in progress (HIP)	no support
hypre	support (CUDA/OMP4.5/RAJA/Kokkos)	no support	no support
MAGMA	support (CUDA)	support (OpenCL)	in progress (OpenCL/SYCL)
MFEM	support (CUDA/OCCA/RAJA)	support (HIP/OCCA)	in progress (OCCA/OpenCL)
PETSc	support (CUDA)	support (ViennaCL/OpenCL)	no support
STRUMPACK	in progress (CUDA)	no support	no support
SUNDIALS	support (CUDA/RAJA/OMP4.5)	no support	in progress (OMP4.5)
SuperLU	support (CUDA)	no support	no support
Tasmanian	support (CUDA)	no support	no support
Trilinos	support (Kokkos)	in progress (Kokkos)	in progress (Kokkos)

ECP Focus on Improved Developer Productivity & SW Sustainability

- **Productivity and Sustainability Improvement Planning (PSIP)** advances software practices
 - <https://github.com/betterscientificsoftware/PSIP-Tools>
 - EXAALT, MPICH, ExaStar, Tasmanian, mpifuleutils, ...
- **Resources for Software Development Kits (SDKs)** facilitate work toward a sustainable ECP software ecosystem
 - <https://github.com/betterscientificsoftware/SDK-Tools>
- **IDEAS-ECP outreach** communicates best practices; partners with synergistic community groups
 - Webinar series on *Best Practices for HPC Software Developers*
 - *Tools and Techniques for Floating-Point Analysis*
 - Date and Time: October 16, 2019, 01:00 pm EDT
 - Presenter: Ignacio Laguna (Lawrence Livermore National Laboratory)
 - Tutorials on *Better Scientific Software*
 - Code coverage and continuous integration
 - Improving reproducibility through better scientific software
 - Verification & refactoring
 - Git Workflows.
 - Introduction to software licensing
 - Better (small) scientific software teams
 - <https://ideas-productivity.org/events/hpc-best-practices-webinars/>

Advancing ECP software practices & scientific productivity through partnerships: Posters at 3rd ECP Annual Meeting, Jan 2019



- **Better Scientific Software portal** (<https://bssw.io>)
 - Provides a community-based resource for sharing information on practices, techniques, and tools to improve developer productivity & software sustainability for computational science and engineering
- **BSSw Fellowship program**
 - Provides recognition & funding to leaders & advocates of high-quality scientific software; provides SWP methodologies for ECP community

Detailed Information about the software technology projects is available in the ECP ST Capability Assessment Report

- Products discussed here are presented with more detail and further citations.
- We classify ECP ST Products deployment as Broad, Moderate or Experimental.
 - Broad and Moderate Deployment is typical suitable for collaboration.
 - Web links are available for almost all products.
 - About 1/3 of ECP ST Products are available as part of the Extreme-scale Scientific Software Stack (E4S) <http://e4s.io>.



<https://www.exascaleproject.org/ecp-software-technology-capability-assessment-report-second-release/>

Some ECP-European Collaboration Models

Approach	Comments/Potential
Read ECP-related papers	Traditional Approach. Works well for small scope: algorithmic advances.
Participate in ECP-related tutorials and webinars	Many ST technologies offer tutorial/webex forums to learn more; range from introductory to advanced
Develop <i>de facto</i> and ISO standards	MPI, OpenMP, C++, Fortran, PAPI, BLAS: Happening, more is better.
Evaluate/prototype new capabilities using ECP software products	Accelerator-enabled software stack (compilers, programming environments, tools, math libraries, in situ), next-generation IO (HDF5, ADIOS, PNetCDF)
Adopt and rely upon ECP software (as an option)	A goal for us: Want to explore how to make this possible. Collaboration can help us improve our product development and delivery.
Contributions to E4S, SDKs	E4S and SDKs are open architectures enabling light-weight product coupling, improved user experience.
Overall	Two way interactions benefit everyone.

ECP Software Technology Outcomes

Programming environments and tools for scalable accelerator-based system

- Advancements in MPI **and** “+” **and** X of MPI+X: MPICH/OpenMPI, Kokkos/RAJA, OpenMP, LLVM, Flang.
- Compiler contributions: HPC features in C++, Flang front end, delivered via LLVM.
- Accelerator-aware tools: HPCToolkit, TAU, PAPI.

Math, Data and Viz libraries

- Accelerator enabled algorithms and implementations (hypre, PETSc, SuperLU, VTK-m).
- Latency-hiding, scalable I/O (HDF5, ADIOS, PnetCDF, MPI-IO, VeloC).
- In situ data: compression, analysis (ZFP, SZ, Ascent).

Software Stack

- 67 software technology products being actively developed for next generation architectures.
- Focused on portable performance on three accelerated platforms: Nvidia, AMD, Intel. Tracking Arm.
- From-source and containerized delivery of products via SDKs and the E4S.

For more information...

<https://www.exascaleproject.org>

or reach out to the leadership team in the areas that interest you..



Doug Kothe, ECP Director
kothe@ornl.gov



Mike Heroux, **Software Technology** Director
maherou@sandia.gov



Lori Diachin, ECP Deputy Director
diachin2@llnl.gov



Jonathan Carter, **Software Technology** Deputy Director
jtcarter@lbl.gov



Andrew Siegel, **Application Development** Director
seigela@uchicago.edu.



Terri Quinn, **Hardware and Integration** Director
quinn1@llnl.gov



Erik Draeger, **Application Development** Deputy Director
draeger1@llnl.gov



Susan Coghlan, **Hardware and Integration** Deputy Director
smc@alcf.anl.gov

The Exascale Computing Project is accelerating delivery of a capable exascale computing ecosystem for breakthroughs in scientific discovery, energy assurance, economic competitiveness, and national security.

WHAT'S NEW

Project Highlights

Comprehensive Molecular Dynamics Capability

March 14, 2019

Training Events



Testing Fortran Software with pFUnit

Event Date: April 10, 2019

Podcast



The EZ Project Focuses on Providing Fast, Effective Exascale Lossy Compression for Scientific Data

February 26, 2019

News



First Exascale Computer by 2021

March 20, 2019

Where exascale will make a difference >

STAY INFORMED

Sign up with your email to receive the latest ECP updates.

email address

SUBSCRIBE

Stay Informed of
ECP Activities

Go to the ECP
homepage

Exascaleproject.org

Simply enter your email
and you will be
subscribed to ECP
newsletters and
frequent updates