The ECP Software Stack

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

ASCAC
September 17, 2018

Scope

Deliver a software stack that enables sustainable exascale capabilities

Mission need

Capabilities across the entire HPC software stack that complement and coordinate with facilities, vendors and other software providers to enable effective execution of ECP apps, and **deliver a capable, sustainable exascale ecosystem**.

Objective

Provide the next generation of DOE software capabilities targeted toward exascale applications and platforms. Provide these capabilities for the specific exascale systems as a high quality, sustainable product suite.

ECP Software: Productive, sustainable ecosystem

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

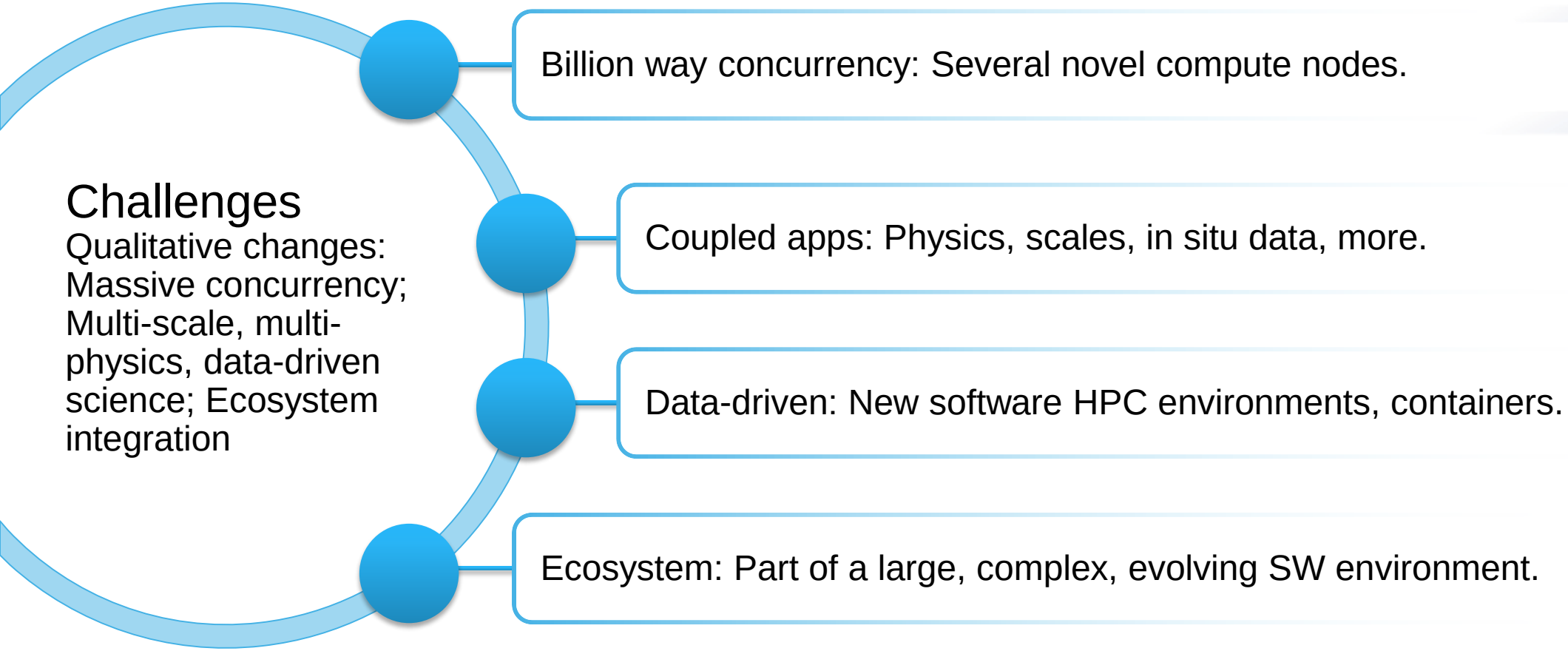
- Perform R&D required for new approaches when necessary

- Coordinate with and complement vendor efforts

- Develop and deploy high-quality and robust software products

55 WBS L4 subprojects executing RD&D
185 L4 subproject (P6) milestones delivered in FY17
67 delivered so far in FY18, 77 in progress right now
564 L4 subproject (P6) milestones planned in FY18-19

ECP software: Challenges



Exascale Computing Project
2.0

Project Management
2.1

- Project Planning and Management 2.1.1
- Project Controls and Risk Management 2.1.2
- Business Management 2.1.3
- Procurement Management 2.1.4
- Information Technology and Quality Management 2.1.5
- Communications and Outreach 2.1.6

Application Development
2.2

- Chemistry and Materials Applications 2.2.1
- Energy Applications 2.2.2
- Earth and Space Science Applications 2.2.3
- Data Analytics and Optimization Applications 2.2.4
- National Security Applications 2.2.5
- Co-Design 2.2.6

Software Technology
2.3

- Programming Models and Runtimes 2.3.1
- Development Tools 2.3.2
- Mathematical Libraries 2.3.3
- Data and Visualization 2.3.4
- Software Ecosystem and Delivery 2.3.5

Hardware and Integration
2.4

- PathForward 2.4.1
- Hardware Evaluation 2.4.2
- Application Integration at Facilities 2.4.3
- Software Deployment at Facilities 2.4.4
- Facility Resource Utilization 2.4.5
- Training and Productivity 2.4.6

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.

Rob Neely, Software Ecosystem and Delivery (2.3.5)
Rob has several leadership roles at LLNL spanning applications, CS research, platforms, and vendor interactions. He is an Associate Division Leader in the Center for Applied Scientific Computing (CASC), chair of the Weapons Simulation and Computing Research Council, and the lead for the Sierra Center of Excellence.

Phase 1:
A loose collection of 66 R&D projects

Phase 2:
Toward a coordinated, planned, hierarchical Project of 54 RD&D projects

AD hits the ground running. 9 months ahead of ST.

ST projects selected: Majority of proposals, some with funding adjustments

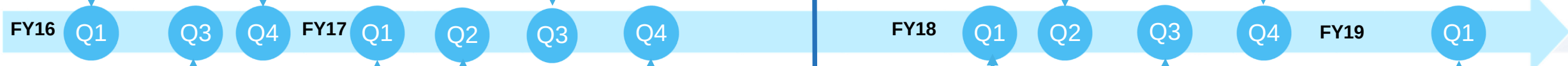
Lab teams fully engaged, U partners ramp up, basic ECP ST software processes emerge. Many quarterly milestones and capabilities delivered by individual projects, independent product releases.

SDK efforts start: Building communities. Shared planning, resources, aggregated software suites.

PPEP: Impact goals, metrics, progress measurement milestone. Gap strategy.

Gap management complete.

ST Project Review Prep with SMEs.



ST: Call for proposals: Based on existing ASCR portfolio, distributed lab representation.

Projects ramp up at labs (Q1), sub-contracts go to universities (Q2).

Even more milestones, capabilities, products. Delivery of gap analysis.

Planning: ST-wide reviews, planning. Communication:

Team-wide regular meetings, clearly defined and documented processes.

Capabilities assessment report. Integrated ST Plan.

Complete impact goals/metrics definition and process.

Project Reviews & Impact Assessments

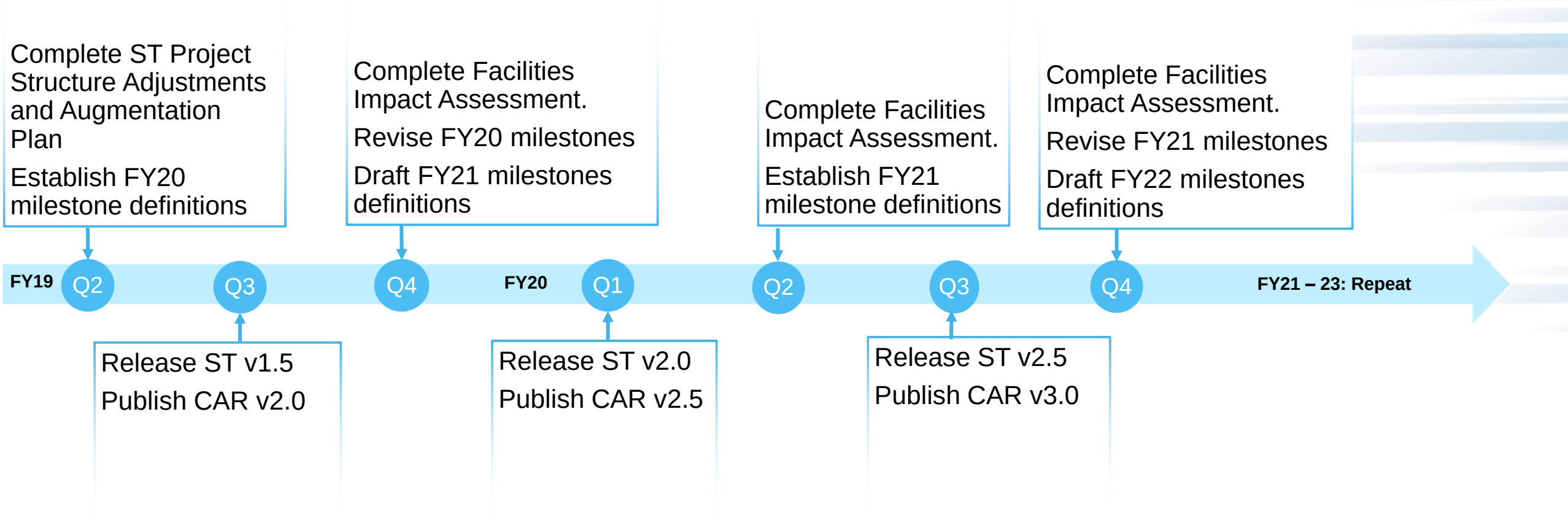
First annual coordinated release of ST products.

Publish CAR v1.5

ECP ST Timeline (1)

EXASCALE COMPUTING PROJECT

Phase 3: Plan, Deliver, Report, Assess, Repeat
Adjustments & Augmentation for CD2
Semi-annual delivery: Milestone Refinement, ST Release, ST CAR, ST Impact Assessment



ECP ST Timeline (2)

2018 ECP ST Highlights

- Completed 84 Release Milestones: product getting to users.
- Developed comprehensive work management systems in Jira: Milestones, KPP-3.
- Established regular 6-month milestone refinement, release, report, assess cycle.
- Created new SDK projects: distributed across ST, seasoned team.
- Addressed gaps/risks: FFTs, OpenACC option, in situ data.
- ID’ed next round of gaps/risks: Future tech directions, project assessment.
- Released first Capability Assessment Report: Comprehensive communication tool.
- Established streamlined communication & reporting system in Confluence.
- Initiated ECP ST adjustments and augmentations in preparation for CD2.

2.3.1 Programming Models and Runtimes

Goals and Objectives

- A cross-platform, production-ready programming environment that enables and accelerates the development of mission-critical software at both the node and full-system levels
- Enable application developers to productively write highly parallel applications that can portably target diverse exascale architectures

Impact Goals and Metrics

- Impact goal: Deliver scalable parallel programming models and runtimes to ECP applications, codesign centers, and other software technology projects
- Impact metrics: Number of ECP application, codesign, and software technology projects using the programming models and features developed in this area
- KPPs: Direct impact on application performance, exascale science capability, and productive and sustainable software ecosystem KPPs

Scope (e.g., projects or areas)

- Enhancement of the MPI and OpenMP standards to meet the needs of exascale
- Production-ready implementation of exascale MPI features in MPICH and Open MPI implementations
- OpenMP standards enhancement and robust OpenMP implementation in the LLVM compiler framework
- Improved MPI and OpenMP interoperability
- Exascale-ready PGAS programming models and runtimes (UPC++/GASNet, Global Arrays)
- Performance-portability solutions for complex, heterogeneous node architectures (Kokkos, RAJA)
- New capabilities in task-based models (Legion, PaRSEC, DARMA) to support exascale applications
- API and library for accessing complex memory hierarchy
- Runtime library for application-level power management
- Lightweight user-level threads (Qthreads)

New Release of UPC++ Library

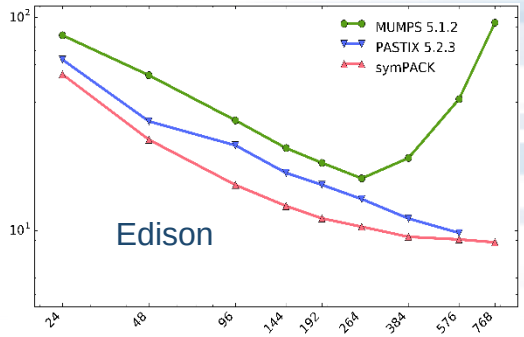
ECP WBS	2.3.1.14 - UPC++ & GASNet
PI	Scott B. Baden, LBNL
Members	LBNL

The Pagoda Project (scope and objectives)

- Developing the UPC++ PGAS programming interface for irregular applications and libraries and GASNet-EX, a portable, high-performance one-sided communication layer
- UPC++: A C++ PGAS library for Exascale)
- GASNet-EX: Communications middleware to support multiple Exascale clients (users: ST, AD and non-ECP)

UPC++ Improves Sparse Solver Performance

- UPC++ enables an efficient pull communication strategy and event-driven scheduling for the symPACK sparse solver, which delivers a ✕2.65 speedup over best state-of-the-art sparse symmetric solver



UPC++ Impact

- ECP client projects will use UPC++ to meet performance goals
  - **AMREx.** By proxy, five AD collaborators.
  - **ExaGraph**
  - **Sparse Solvers**
- In use by non-ECP partners in academia
- Community outreach: Talks at U. Tokyo, Information Tech. Center and RIKEN Advanced Institute for Computational Science

Milestone accomplishment

- UPC++ 2018.3.0 was released March 22, 2018 (There was also a Jan 2018 Beta release)
- Implements 3 new features
  - Non-contiguous transfers
  - Remote Atomics (uses network offload when available)
  - View-Based Serialization
- Also includes Generalized Completion, which simplifies and generalizes the interface for data movement operations

Deliverables

STPM17-7 Milestone Memo: https://confluence.exascaleproject.org/download/attachments/29000066/ECP-Milestone-Memo-STPM17-7.pdf
UPC++ Software and Documentation: http://upcxx.lbl.gov
Development of this software release used ECP's ALCC allocations at NERSC, ALCF and OLCF.

Application Requirements for OpenMP: Deep copy and memory management APIs

ECP WBS	2.3.1.13 SOLLVE
PI	Barbara Chapman, BNL
Members	ORNL, ANL, LLNL, GA Tech

Overall Scope and objectives

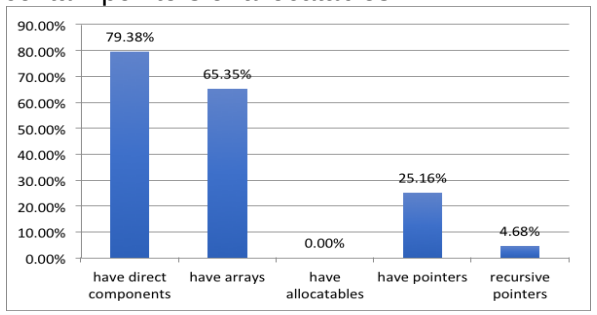
- Gather application requirements to prepare OpenMP for Exascale
- Identify gaps and areas that OpenMP needs to address or improvement in OpenMP 5.0 or beyond
- Work with the application teams to develop use-cases to drive OpenMP extensions.
- Provide feedback to the OpenMP validation and verification team to test extensions and their implementations.

Impact

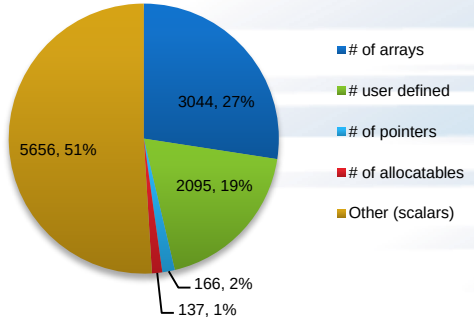
- The HPC community will benefit from the new functionality available in OpenMP
- We are addressing ECP application needs to program the different memories that will be available on exascale nodes and to optimize the mapping of complex data structures to/from memories (deep copy).
- Application developers will benefit from this new functionality that is scheduled to be available in OpenMP 5.0

Deep Copy Application Requirements

Need to move data to/from different memories and user-defined structures that contain pointers or allocatables.



Survey of Application Data Structures (E3SM - CAM/SE)



Milestone accomplishment

- Identified that one of the most important gaps in OpenMP is how to program the different memories available in the node and how to map to/from complex data structures between them that require deep copy
- Gathered application use cases from E3SM, QMCPack, XGC, LSMS3, LAMMPS, GENESIS, etc. Used this information to drive the memory management API, declare mappers and manual deep copy functionality in OpenMP 5.0.
- Developed test cases to implement manual deep copy, declare mappers and the new memory management API in OpenMP 5.0.

Deliverables Milestone report: https://confluence.exascaleproject.org/display/STPM15/Application+Requirements+Milestones

2.3.2 Development Tools

Goals and Objectives

- Development tools including compilers, tools for correctness and debugging, performance analysis, and programming technologies.
- Focus on features for emerging architectures: heterogeneous computing, deep memory hierarchies, etc.
- Development of capabilities for current architectures including Summit, Cori, and Theta
- Low-level software like code generation and performance counters are critical to all applications and other software projects AND intimately tied to the architecture

Dashboard

- **Impact Goal**: Improve and deliver ECP ST products and make broadly available to key stakeholders. Focus on integration with facility and vendor solutions.
- **Impact Metrics**: Number of ECP ST products deployed and supported at facilities, adopted by vendors, and used by applications and other software products
- **Connection to KPPs**: Direct impact on Productive Software Ecosystem KPP

Scope (e.g., projects or areas)

- Performance portability metrics, tools, and strategies
- Expertise and software systems for
  - heterogeneous computing (GPUs, FPGAs, Manycore)
  - deep memory hierarchies including nonvolatile memory.
- LLVM including open-source improvements including parallel IR, Flang, interoperability
- OpenACC, CUDA, OpenCL
- Performance tools with TAU, PAPI, HPCToolkit
- Infrastructure including autotuning, instrumentation, and system software for other ST areas.
- Focus on emerging ECP application trends: Kokkos, Legion, etc.
- Focus on emerging architectures: A21, SoCs, etc

STTO12-26 / 13717 / Develop prototype implementation of partial OpenACC support (CLACC) in open-source LLVM framework **(2.3.2.09)**

- Goal: Implement a prototype OpenACC compiler in Clang/LLVM for evaluation on selection ECP applications.

Approach / Milestones	Impact
Evaluate design alternatives for OpenACC implementation	Provide requested programming model to ECP users using an source strategy, and request feedback from LLVM community.
Implement prototype OpenACC design	Determine practical implementation strategy for deployment.
Test and demonstrate OpenACC on real applications to understand limits and priorities	Test and validate integrated implementation on real software and existing systems.

Milestone to be completed in late September.

Fulfillment of IBM Power9 Hardware Counter Support (2.3.2.06)

- Goal: Finalize development and implementation of PAPI hardware performance counter support for IBM Power9 core and uncore architecture. This includes validation and testing of the implemented core and uncore events against the event tables released for IBM Power9.

Approach / Milestones	Impact
Solicit input from users and performance tool developers from testing the early IBM Power9 core and uncore support developed in	Provide requested features to ECP users using an open source strategy, and prioritize feedback from the user community.
Implement the final version of PAPI components enabling IBM Power9 core and uncore support.	Determine practical implementation strategy for deployment.
Release of PAPI components for IBM Power9 core and uncore support that are fully integrated into the PAPI library.	Test and validate integrated implementation on real software and existing systems.

Milestone completed and software released in FY18.

2.3.3 Math Libraries

Goals and Objectives

Objective: Provide scalable, robust, resilient numerical algorithms, encapsulated in reusable libraries that facilitate efficient next-generation scientific simulations on exascale computers.

Drivers: Broad range of ECP applications rely on DOE math libraries for the most advanced technologies available in math and computer science R&D.

Impact Goals and Metrics

- **Impact Goal:** Mathematical libraries that (i) interoperate with the ECP software stack; (ii) are incorporated into the ECP applications; and (iii) provide scalable, resilient algorithms that facilitate efficient exascale simulations.
- **Impact Metrics:** Number of ECP projects using math libraries developed by this L3 area, richness of interoperability among math libraries and ECP SW stack.
- **Connection to KPPs**: Contribute toward performance, portability, and productivity KPPs.

Scope

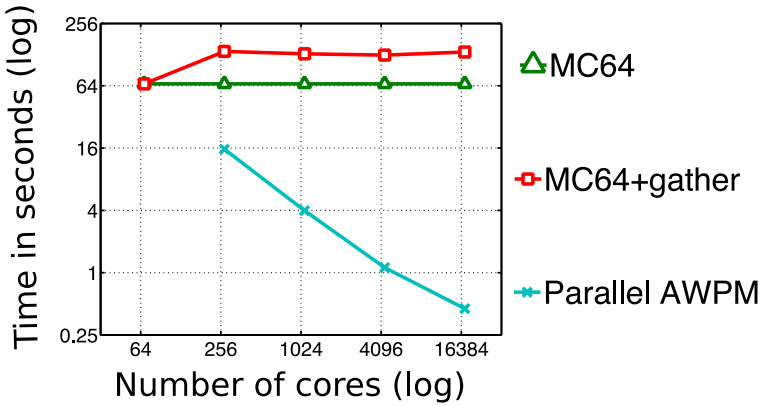
- Advanced, coupled multiphysics and multiscale algorithms
  - Discretizations
  - Preconditioners and Krylov solvers
  - Nonlinear and timestepping solvers
  - Coupling technologies
- Toward predictive simulations and analysis
  - Optimization, sensitivities, UQ, ensembles
- Performance on new node architectures
- Extreme strong scalability
- Math library interoperability and complementarity through the xSDK (Extreme-scale Scientific Software Development Kit)
  - Improving package usability, quality, sustainability
  - Community coordination and collaboration while retaining package autonomy

Recent Math Libraries Highlights

Factorization-based Sparse Solvers for Exascale

- **STRUMPACK/SuperLU** team collaborating with the **ExaGraph** team
- Develop distributed-memory maximum-weight perfect matching algorithm for stable pivot selection.
- The parallel AWPM code scales to 256 nodes (17K cores) on the Cori/KNL supercomputer and can run up to 2500x faster than the sequential algorithm on 256 nodes
- More info:
  - https://www.exascaleproject.org/exagraph-with-strumpack-superlu/
  - A. Azad, A. Buluc, X.S. Li, X. Wang, J. Langguth, arXiv:1801.09809v1, 30 Jan 2018

The parallel algorithm runs 300x faster than the sequential algorithm on 16K cores of NERSC/Cori



Accelerated Matrix Norms

- **SLATE** provides accelerated implementation of norms, a capability that is currently not available in ScaLAPACK or vendor libraries.

- Implementation of norms: max, one, infinity, Frobenius
  - Covering the standard precisions (S, C, D, Z)
  - Covering the GE, TR, SY, HE matrix types
  - Covering all routines with testers
  - Producing a performance report
  - Maps well to modern supercomputers, with multiple hardware accelerators per node
- More info:
  - A. Kurzak, M. Gates, A. Yarkhan, I. Yamazaki, P. Luszczek, J. Finney, J. Dongarra, http://www.icl.utk.edu/publications/swan-006,
  - https://bitbucket.org/icl/slate/

2.3.4 Data and Visualization

Goals and Objectives

Objective: A production-quality storage infrastructure necessary to checkpoint, manage, share, and facilitate analysis of data in support of mission critical codes. Data analytics and visualization software that effectively supports scientific discovery and understanding of data produced by exascale platforms.
Drivers: ECP applications rely on DOE-supported data and visualization approaches to checkpoint and analyze their simulation results.

Dashboard

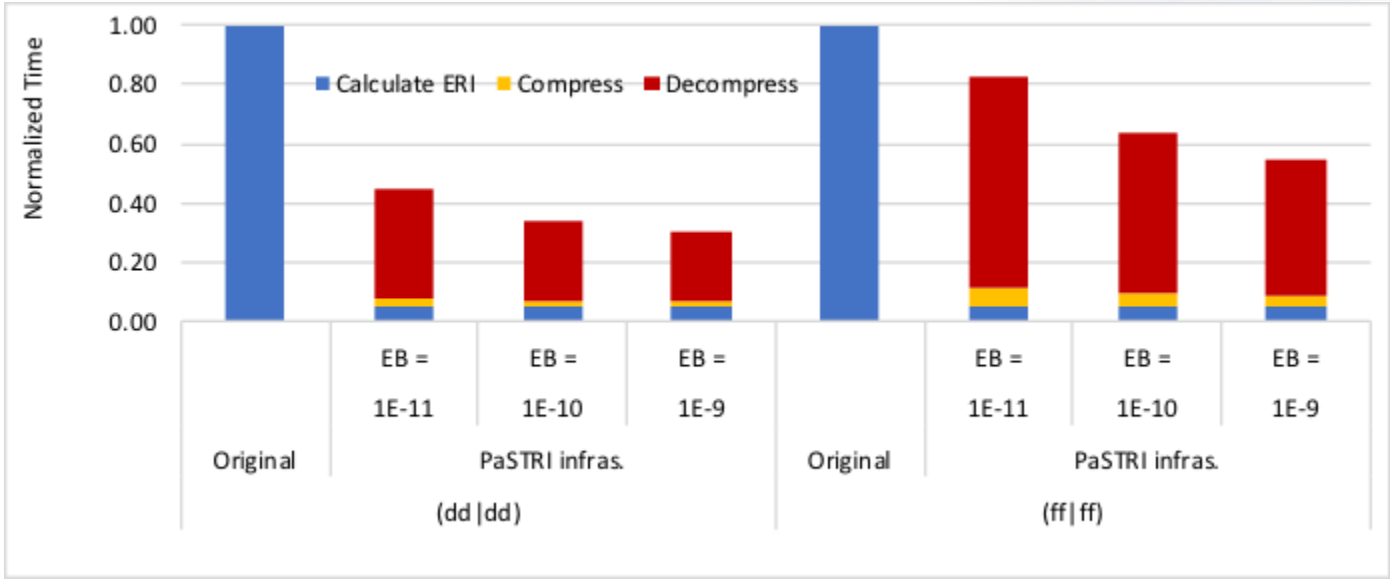
- **Impact goal:** Data and visualization libraries and tools that (i) interoperate with the ECP software stack; (ii) are incorporated into the ECP applications; and (iii) provide checkpoint, data reduction and visual analysis that facilitate simulation science on exascale computers.
- **Impact metrics:** Number of ECP projects using data and visualization libraries developed by this L3 area, amount of data saved and reduced
- **KPPs:** Contribute toward performance, portability, and productivity KPPs

Scope (e.g., projects or areas)

2.3.4.01 Data and Visualization Software Development Kit
2.3.4.02-04 LANL/LLNL/SNL ATDM Data and Visualization Projects
2.3.4.05 VeloC: Very Low Overhead Transparent Multilevel Checkpoint/Restart
2.3.4.06 Ez: Fast, Effective, Parallel Error-bounded Exascale Lossy Compression for Scientific Data
2.3.4.07 UnifyCR: A Checkpoint/Restart File System for Distributed Burst Buffers
2.3.4.08 ExaHDF5: Delivering Efficient Parallel I/O on Exacale Computing Systems
2.3.4.09 ADIOS Framework for Scientific Data on Exascale Systems
2.3.4.10 DataLib: Data Libraries and Services Enabling Exascale Science
2.3.4.11 ZFP: Compressed Floating-Point Arrays
2.3.4.12 ALPINE: Algorithms and Infrastructure for In Situ Visualization and Analysis
2.3.4.13 ECP VTK-m

Highlight - EZ

- Performance of two electron integral simulations is limited by the re-computation of large number of integrals at each iteration because of lack of memory space
- -The EZ team developed a new algorithm for compression of two electron integral simulation with very high compression ratio and fidelity
- -Integration in SZ and test in GAMESS
- -16.8X compression ratio, 661MB/s compression rate, 1116MB/s decompression rate
- -Performance improvement of GAMESS (on the tested cases) of 1.25 to 2.5 depending on the simulations and error bound (these accelerations are highly dependent on the simulation)
- -Overall best paper award at IEEE Cluster 2018



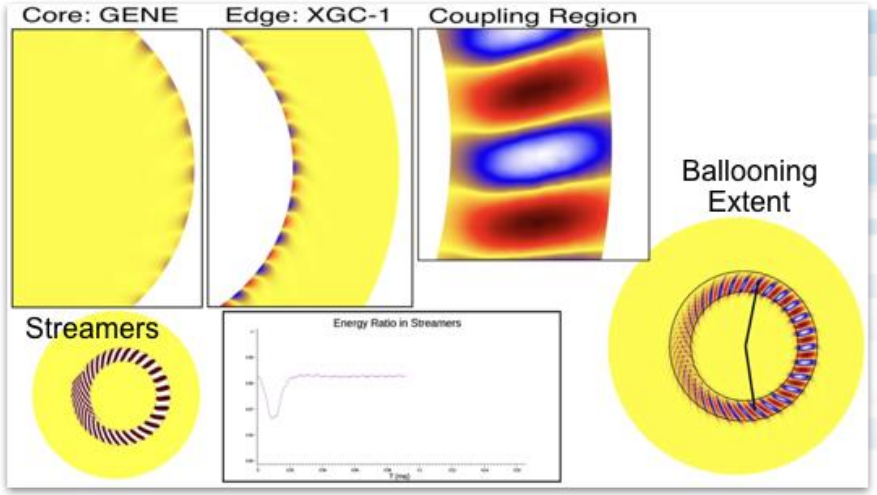
Ref: Ali Murat Gok, Sheng Di, Yuri Alexeev, and Dingwen Tao, Vladimir Mironov, Xin Liang, Franck Cappello, PaSTRI: Error-Bounded Lossy Compression for Two-Electron Integrals in Quantum Chemistry, IEEE Cluster 2018, Belfast, IEEE press

Figure: acceleration on 1 process, serial execution, compressed integrals stored in memory.

We also have the parallel performance with the compressed integrals stored on disks.

VTK-m: In situ visualization of WDMApp code coupling

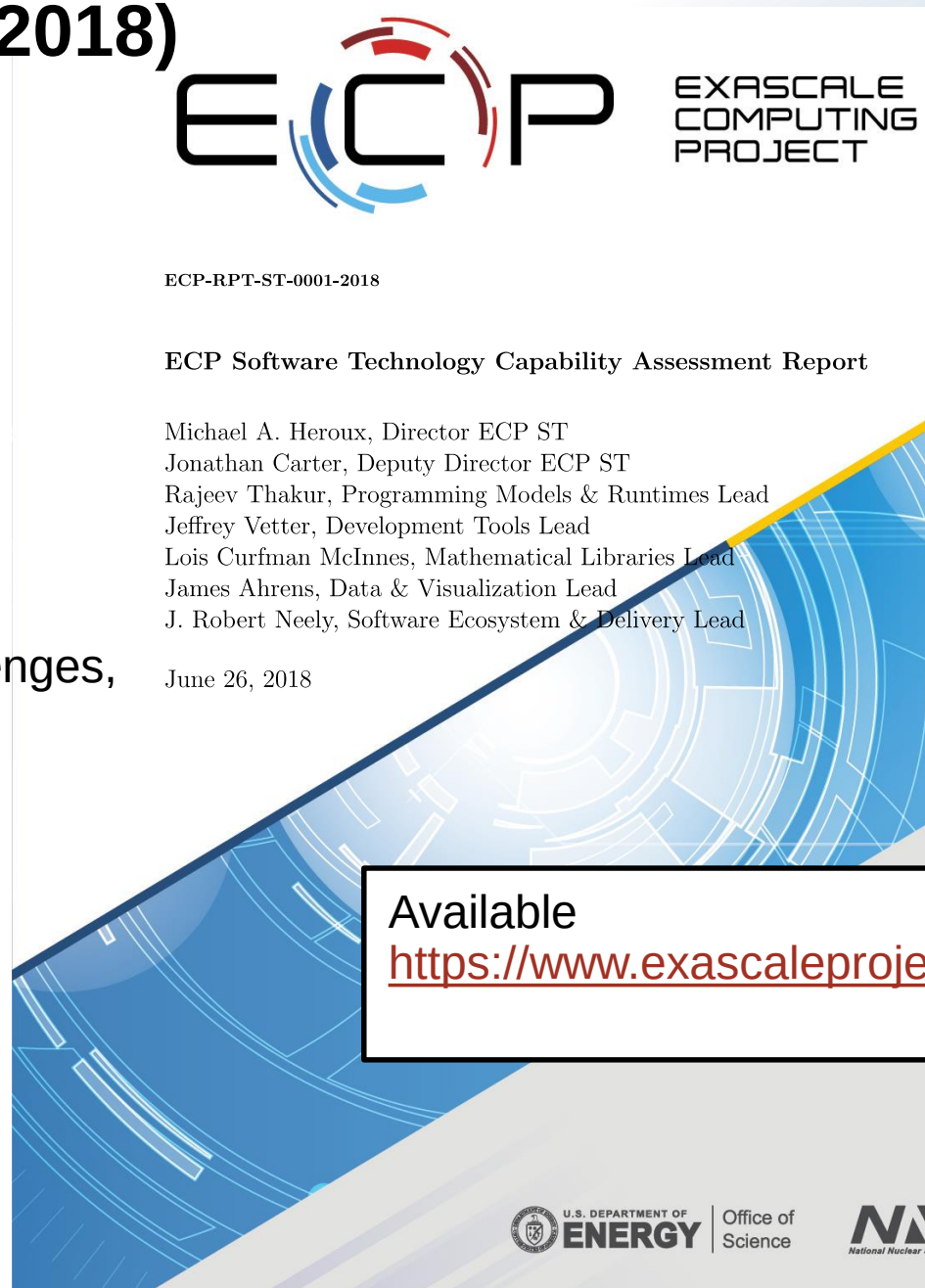
- ECP Scientists from the WDMApp project need in situ visualization to monitor and understand the behavior of coupled fusion codes (GENE and XGC-1)
- The VTK-m ECP project integrated visualization components into the system by leveraging the same ADIOS framework used to couple the fusion codes
- WDMApp scientists were able to monitor and validate the code coupling using in situ visualization, which is the first coupling of continuum and PIC fusion codes
  - See: J. Dominski, et al., *Physics of Plasmas* 2018, 25, DOI: 10.1063/1.5044707
- This technology was run on the Titan supercomputer at the Oak Ridge Leadership Computing Facility



"***In Situ Analysis and Visualization of Fusion Simulations: Lessons Learned***" M. Kim, et al., In Situ Visualization: Introduction and Applications, July 2018

ECP Software Technology Capability Assessment Report (Released July 2018)

- Three document elements:
  1. Executive summary – Public content.
  2. Project Description - Public content.
     - **SDKs, Delivery strategy, project restructuring, new projects.**
     - Technical areas overview.
     - **Deliverables: Products, Standards committees, contributions to external products.**
     - Project two-pages: 55 with description, activities, challenges, next steps.
  3. **Appendix – ECP/Stakeholder content.**
     - **Impact goals/metrics framework.**
     - **Gaps and Overlaps.**
     - **ASC-ASCR leverage tables.**
- LaTeX, separate contributors, easily updated.
- 212 pages (191 public), update twice a year.



ECP ST Products:
55 Projects contribute to 89 Unique Products

Spack Package Support	
Have Spack Package	43
Spack Package in progress	21

Source Build System	
Cmake	44
Configure/Make (autotools)	32
Custom	4

Delivery	
Direct to users from source	81
Vendor stack	11
ALCF	19
OLCF	20
NERSC	20
LLNL	18
LANL	17
OpenHPC	9
Containers (Docker)	3+

User Support	
Documentation	81
Tutorials	50
Support staff training	21
Email/phone contact	70
User-access issue tracking	65

- 48% support Spack.
- 24% Spack in progress.

➢ **Requirement for Q1FY19 participation.**

- Most users directly manage ST software from source.

➢ **Spack packages, SDKs will improve access and management.**

- ST projects have diverse delivery experience with:
  - vendors,
  - leadership facilities,
  - binary release,
  - Containers

➢ **Can leverage across other projects.**

Stats collected April 2018

Programming Models and Runtimes Products (18)

Legion	http://legion.stanford.edu
ROSE	https://github.com/rose-compiler
Kokkos	https://github.com/kokkos
DARMA	https://github.com/darma-tasking
Global Arrays	http://hpc.pnl.gov/globalarrays/
RAJA	https://github.com/LLNL/RAJA
CHAI	https://github.com/LLNL/CHAI
Umpire	
MPICH	http://www.mpich.org
PaRSEC	http://icl.utk.edu/parsec/
Open MPI	https://www.open-mpi.org/
Intel GEOPM	https://geopm.github.io/
LLVM OpenMP compiler	https://github.com/SOLLVE
OpenMP V&V Suite	https://bitbucket.org/crpl_cisc/sollve_vv/src
BOLT	https://github.com/pmodels/argobots
UPC++	http://upcxx.lbl.gov
GASNet-EX	http://gasnet.lbl.gov
Qthreads	https://github.com/Qthreads

Development Tools (19)

SICM	https://confluence.exascaleproject.org/display/STSS07
QUO	https://github.com/lanl/libquo
Kitsune	https://github.com/lanl/kitsune
SCR	https://github.com/llnl/scr
Caliper	https://github.com/llnl/caliper
mpiFileUtils	https://github.com/hpc/mpifileutils
Gotcha	http://github.com/llnl/gotcha
TriBITS	https://tribits.org
Exascale Code Geneneration Toolkit	
PAPI	http://icl.utk.edu/exa-papi/
CHiLL Autotuning Compiler	
Search using Random Forests (SuRF)	
HPCToolkit	http://hpctoolkit.org
The Dyninst Binary Tools Suite	http://www.paradyn.org
Tau	http://www.cs.uoregon.edu/research/tau
Papyrus	https://ft.ornl.gov/research/papyrus
openarc	https://ft.ornl.gov/research/openarc
LLVM	http://llvm.org/
Program Database Toolkit (PDT)	https://www.cs.uoregon.edu/research/pdt/home.php

Mathematical Libraries Products (16)

xSDK	https://xsdk.info
hypre	http://www.llnl.gov/casc/hypre
FleCSI	http://www.flecsi.org
MFEM	http://mfem.org/
Kokkoskernels	https://github.com/kokkos/kokkos-kernels/
Trilinos	https://github.com/trilinos/Trilinos
SUNDIALS	https://computation.llnl.gov/projects/sundials
PETSc/TAO	http://www.mcs.anl.gov/petsc
libEnsemble	https://github.com/Libensemble/libensemble
STRUMPACK	http://portal.nersc.gov/project/sparse/strumpack/
SuperLU	http://crd-legacy.lbl.gov/~xiaoye/SuperLU/
ForTrilinos	https://trilinos.github.io/ForTrilinos/
SLATE	http://icl.utk.edu/slate/
MAGMA-sparse	https://bitbucket.org/icl/magma
DTK	https://github.com/ORNL-CEES/DataTransferKit
Tasmanian	http://tasmanian.ornl.gov/

Data & Visualization Products (25)

HXHIM	http://github.com/hpc/hxhim.git
Cinema	https://datascience.lanl.gov/Cinema.html
MarFS	https://github.com/mar-file-system/marfs
GUFI (The Grand Unified File Index)	https://github.com/mar-file-system/GUFI
Siboka	
ROVER	
C2C	
TuckerMPI	
ParaView	https://www.paraview.org/
Catalyst	https://www.paraview.org/in-situ/
VTK-m	http://m.vtk.org
FAODEL	https://github.com/faodel/faodel
IOSS	https://github.com/gsjaardema/seacas
VeloC	https://xgitlab.cels.anl.gov/ecp-veloc
SZ	https://github.com/disheng222/SZ
UnifyCR	https://github.com/LLNL/UnifyCR
HDF5	https://www.hdfgroup.org/downloads/
ADIOS	https://github.com/ornladios/ADIOS2
Parallel netCDF	http://cucis.ece.northwestern.edu/projects/PnetCDF/
Darshan	http://www.mcs.anl.gov/research/projects/darshan/
ROMIO	http://www.mcs.anl.gov/projects/romio/
Mercury (part of Mochi suite)	http://www.mcs.anl.gov/research/projects/mochi/
zfp	https://github.com/LLNL/zfp
VisIt	https://wci.llnl.gov/simulation/computer-codes/visit
ASCENT	https://github.com/Alpine-DAV/ascent

SW Ecosystem & Delivery Products (11)

BEE	
FSEFI	
Spack	https://github.com/spack/spack
Sonar	
Secure JupyterHub	
Kitten Lightweight Kernel	https://github.com/HobbesOSR/kitten
AML	https://xgitlab.cels.anl.gov/argo/aml
ArgoContainers	https://xgitlab.cels.anl.gov/argo/containers
COOLR	https://github.com/coolr-hpc
NRM	https://xgitlab.cels.anl.gov/argo/nrm
Flang/LLVM Fortran compiler	http://www.flang-compiler.org

Other Important ECP ST Contributions

ECP ST Staff Contribute to ISO and *de facto* standards groups: Assuring Sustainability through standards

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

- **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.

- **ARB*:** Good model for SDKs.

*Architecture Review Board

External Product Impact

Product	Contribution
MAGMA	ECP ST math libraries eorts inform the design, implementation, and optimization of numerical linear algebra routines on NVIDIA GPUs
Vendor/community compilers and runtimes	The Validation and Verication Suite (on-going effort) for the SOLLVE project has helped uncover bugs in OpenMP implementations provided by Cray, LLVM and XL.
SWIG	The ECP ST ForTrilinos eorts contributes the capability to generate automatic Fortran bindings from C++ code.
TotalView Debugger	ECP ST sta are engaged in co-design of OMPD, the new debugging interface for OpenMP programs, along with RogueWave engineers. This eort helps RogueWave improve their main debugging product, TotalView, by making it aware and compatible with recent advances in OpenMP debugging.
MPI Forum	ECP ST sta maintain several chapters of the MPI Forum, effort that require a constant involvement with the other authors, as well as participation to the online discussions related to the chapter and regular attendance of the MPI Forum face-to-face activities. An ECP ST staff member belongs to several working group related to scalability and resilience where, in addition to the discussions, implements proof-of-concept features in OpenMPI.
Cray MPICH MPI-IO	As part of the ExaHDF5 ECP project, the ALCF worked with Cray MPI-IO developers to merge the upstream ROMIO code into the downstream proprietary Cray MPICH MPI-IO, leveraging Crays extensive suite of IO performance tests and further tuning the algorithm. Cray is currently targeting its deployment in an experimental release.
OpenHPC	An ECP ST staff member serves on the OpenHPC Technical Steering Committee as a Component Development representative.
LLVM	An ECP ST staff member is co-leading design discussions around the parallel IR and loop-optimization infrastructure.

Some of our best work is to provide input to software we don't productize or support.

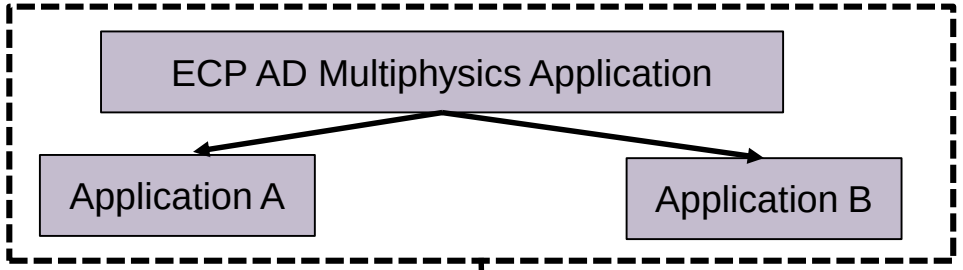
ECP ST SDKs

SW Development Kit (SDK) Overview

- SDK: A collection of related software products (called packages) where coordination across package teams will improve usability and practices and foster community growth among teams that develop similar and complementary capabilities. SDKs have the following attributes:
  - Domain scope: Collection makes functional sense.
  - Interaction model: How package interact; compatible, complementary, interoperable.
  - Community policies: Value statements; serve as criteria for membership.
  - Meta-infrastructure: Encapsulates, invokes build of all packages (Spack), shared test suites.
  - Coordinated plans: Inter-package planning. Does not replace autonomous package planning.
  - Community outreach: Coordinated, combined tutorials, documentation, best practices
- Overarching goal: Unity in essentials, otherwise diversity.

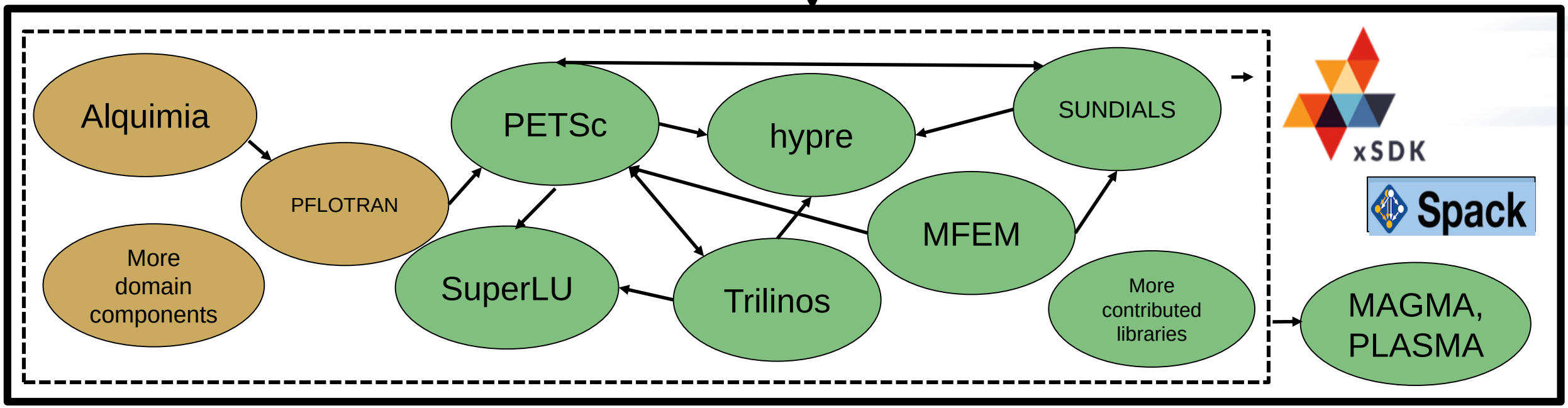
xSDK interactions: Apps, packages, Spack

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



xSDK functionality, Nov 2017

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



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

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.
M6. Respect system resources and settings made by other previously called packages.
M7. Come with an open source license.
M8. Provide a runtime API to return the current version number of the software.
M9. Use a limited and well-defined symbol, macro, library, and include file name space.
M10. Provide an accessible repository (not necessarily publicly available).
M11. Have no hardwired print or IO statements.
M12. Allow installing, building, and linking against an outside copy of external software.
M13. Install headers and libraries under <prefix>/include/ and <prefix>/lib/.
M14. Be buildable using 64 bit pointers. 32 bit is optional.
M15. All xSDK compatibility changes should be sustainable.
M16. The package must support production-quality installation compatible with the xSDK install tool and xSDK metapackage.

Also specify **recommended policies**, which currently are encouraged but 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: Must be an xSDK-compatible package, *and* it uses or can be used by another package in the xSDK, and the connecting interface is regularly tested for regressions.

Prior to defining and complying to 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.

https://xsdk.info/policies

Version 0.3.0, Nov 2017

Software Development Kits Progress: Leadership in place, Spack packaging making rapid progress

ECP software projects

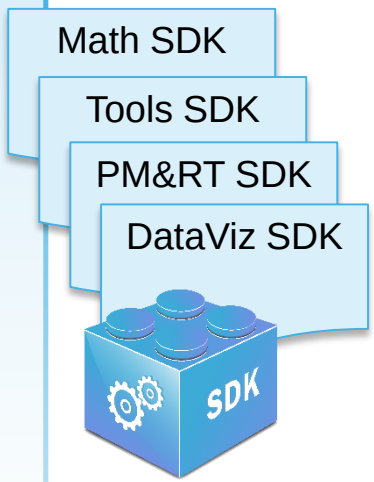
Each project to define (potentially ≥2) release vectors

More projects ← → Fewer projects

SDKs

Reusable software libraries embedded in applications; cohesive/interdependent libraries released as sets modeled on xSDK

- Regular coordinated releases
- Hierarchical collection built on Spack
- Products may belong to >1 SDK based on dependences
- Establish community policies for library development
- Apply Continuous Integration and other robust testing practices



OpenHPC

Potential exit strategy for binary distributions

- Target similar software to existing OpenHPC stack
- Develop super-scalable release targeting higher end systems

Direct2Facility

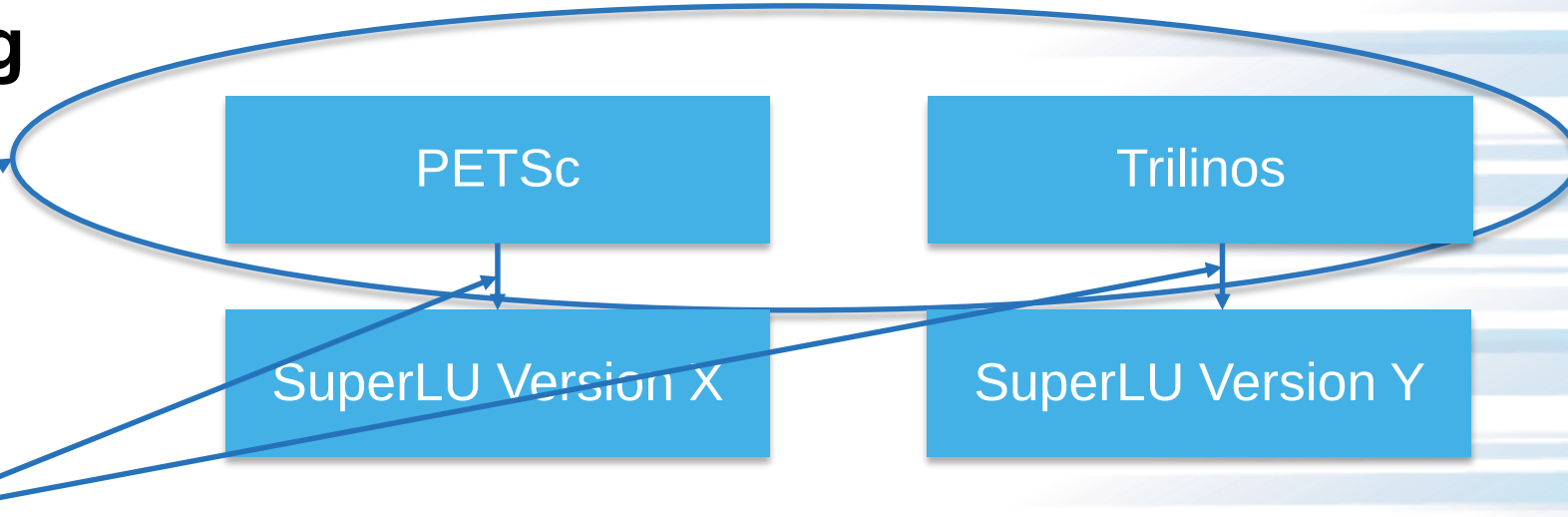
Platform-specific software in support of a specified 2021–2023 exascale system

- Software **exclusively** supporting a specific platform
- System software, some tools and runtimes

SDK Leadership Team: Decades of Software Experience

- **Jim Willenbring – SDK Coordinator and Release Manager**
- **Sameer Shende – Programming Models & Runtimes**
- **Bart Miller – Development Tools**
- **Lois McInnes – Math Libraries**
- **Chuck Atkins - Data & Viz**

SDK “Horizontal” Grouping



- Horizonal (vs Vertical) Coupling
  - Common substrate
  - Similar function and purpose
    - e.x. 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

SuperLU Version Y

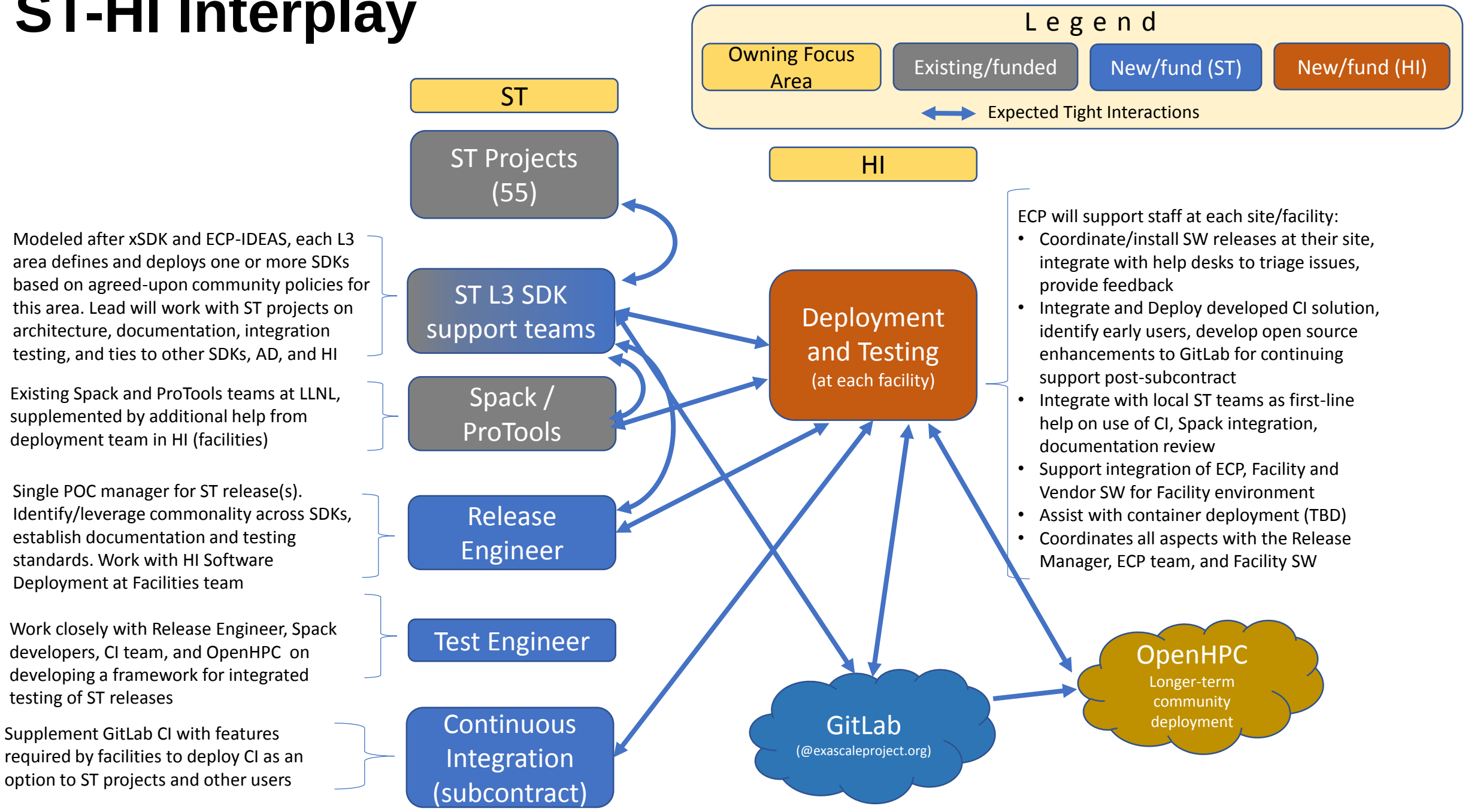
SDK Summary

- 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.
- SDKs will help reduce complexity of delivery:
  - Hierarchical build targets.
  - Distribution of software integration responsibilities.
- Longer term:
  - Establish community policies, enhance best practices sharing.
  - Provide a mechanism for shared infrastructure, testing, training, etc.
  - Enable community expansion beyond ECP.

HI and Facilities Interaction and Integration

- ECP ST has an entire technical area dedicated to HI interactions and integration:
  - ECP ST technical area 5: SW Ecosystem and Delivery.
  - Primary focus is on coordinated software delivery and integration.
- Delivery vs deployment:
  - ST *delivers.*
  - HI, Facilities, vendors, OpenHPC *deploy.*

ST-HI Interplay



Next Steps

Oct 1 – 5, 2018 Review, Virtual (BlueJeans)

Technical Area	External Subject Matter Experts (Advisors)
PMR	1.Sanjay Kale, UIUC 2.Larry Kaplan, Cray 3.Bob Wisniewski, Intel
Dev Tools	1.Laura Carrington, SDSC 2.Felix Wolf, TU-Darmstadt
Math Libs	1.Wolfgang Bangerth, Colorado State 2.Edmond Chow, Georgia Tech
Data & Viz	1.Gary Grider, LANL 2.Paul Navratil, TACC
SW Ecosystem	1.Sadaf Alam, CSCS, Switzerland 2.Karl Schulz, TACC

Review Criteria:

Describe the following:

1. How the project is important or essential to achieving a sustainable, production-level computing capability for ECP.
    - Focus on the new capabilities that ECP is funding, not the general capabilities represented your project.
    - Address what ECP would lack if your project could not deliver.
    - Discuss the project's clients and users, and the importance of your efforts to helping them achieve ECP goals.
2. Project deliverables for 2019 and (assuming continuous funding) deliverables in the ECP timeframe (2021 - 2023).
3. How the project will deliver its capabilities to users.
    - Discuss any project history of production software delivery.
    - Describe planned delivery paths, in particular: open source repositories with production source installation capabilities, third parties (vendors, OpenHPC), direct to facilities.
    - Describe means of documentation, testing, user support, licensing, etc.
4. Progress toward the project's milestones.
    - List project impact goals and impact metrics, and progress toward these goals.
    - Review and highlight completed milestones.
    - Discuss progress toward integration with applications, other ST projects, vendors, or facilities; discuss progress toward platform readiness.
    - Discuss progress on software performance, scalability, and portability.
5. Relationships with other software (from any source) and other ECP efforts.
    - List project dependencies (you depend on others and others depend on you) that are not ECP apps.
    - Discuss projects with similar capabilities, and distinguishing features you want to highlight.
    - Discuss collaborating projects and any relationships with ECP HI (co-design, PathForward, DSE) efforts.
6. Project risks and issues.
    - Discuss technical, programmatic, or personnel risks and issues.
    - Describe plans to address and manage these risks.

Software Practices Polls

- Two polls:
  - The first poll (**Software Practices Survey**) is important for the ECP Independent Project Review. If you can only take time to do one poll, please complete this one. **Responses due COB Wed 9/14.**
  - The second poll (**Software Practices: Lessons Learned**) is intended to capture practices that work and don't work base on the many collective years ECP ST staff have spent writing scientific software. Of particular interest to me are practices that take a long time for the benefit or harm to be realized.
- **Note:**
  - We will not use the information collected in these polls as part of our assessment.
  - We need results from the first poll to baseline the way ECP ST software is developed today.
  - Any reporting of the survey results will not include project-specific connections.
  - Please be as frank as possible.
- Both polls are available from the following Confluence page: https://confluence.exascaleproject.org/display/1ST/Software+Practices+Polls

Where is the most detailed description of the design of new capabilities you plan to develop?

Thinking about the ECP capabilities you have developed in the past year for the product, provide the best response.

Question 1 – 3

- In the funded proposal document or statement of work.
- Undocumented. Part of an informal analysis of what needs to get done.
- In a written design document.
- In a prototype implementation.
- In an issue tracking system (Jira, GitHub Issue, etc).
- Other (Describe at end of survey).

Where is the most detailed description of what is required from new capabilities you plan to develop?

Thinking about the ECP capabilities you have developed in the past year for the product, provide the best response.

- In the funded proposal document or statement of work.
- Undocumented. Part of an informal analysis of what needs to get done.
- In a written requirements document.
- In an issue tracking system (Jira, GitHub Issue, etc).
- Other (Describe at end of survey).

How do you determine what capabilities you plan to develop?

Thinking about the ECP capabilities you have developed in the past year for the product, provide the best response.

	Primary	Secondary	Tertiary
From informal user interactions and my knowledge of the field.	○	○	○
From formal stakeholder engagement (surveys, user sessions, etc.).	○	○	○
From ideas submitted through issue tracking system (Jira, GitHub Issue, etc).	○	○	○
Other (Describe at end of survey).	○	○	○

Questions 4 – 5

Do you receive feedback on your capability design and implementation as you develop it?

Thinking about the ECP capabilities you have developed in the past year for the product, provide the best response.

- No.
- Yes, but informal.
- Yes, through a formal review process.
- Other (Describe at end of survey).

How is development work scheduled and assigned?

Thinking about the ECP capabilities you have developed in the past year for the product, provide the best response.

	Primary	Secondary	Tertiary
Individual project members decide what to do and when.	○	○	○
We coordinate informally within the project.	○	○	○
Work is prioritized and scheduled as staff are available.	○	○	○
Worked is scheduled formally (e.g., in a sprint).	○	○	○
Other (Describe at end of survey).	○	○	○

Questions 6 – 8

How do you test for errors in previously working code (i.e., regressions)?

Thinking about the ECP capabilities you have developed in the past year for the product, provide the best response.

- No independent tests are run.
- We rely on tests from our users' code.
- Manually, as needed.
- Regularly on a schedule.
- As part of integrating our code into the product.
- Other (Describe at end of survey).

When are tests developed?

Thinking about the ECP capabilities you have developed in the past year for the product, provide the best response.

- No tests are written; We rely on external tests.
- As needed to address problems.
- Before writing the source code.
- During or after writing the source code.
- Other (Describe at end of survey).

What drives your development schedule?

Thinking about the ECP capabilities you have developed in the past year for the product, provide th best response.

	Primary	Secondary	Tertiary
We complete work as we have time to do it.	○	○	○
Submission deadlines for papers.	○	○	○
Through a formal work planning process.	○	○	○
Other (Describe at end of survey).	○	○	○

How are new capabilities integrated into the existing product?

Thinking about the ECP capab
best response.

- We develop directly in the product and don't need to integrate changes.
- We submit changes to a repository directly.
- Our changes are submitted via a pull request.
- Other (Describe at end of survey).

How do you document new capabilities?

Thinking about the ECP capabilities you have developed in the past year for the product, provide the best response.

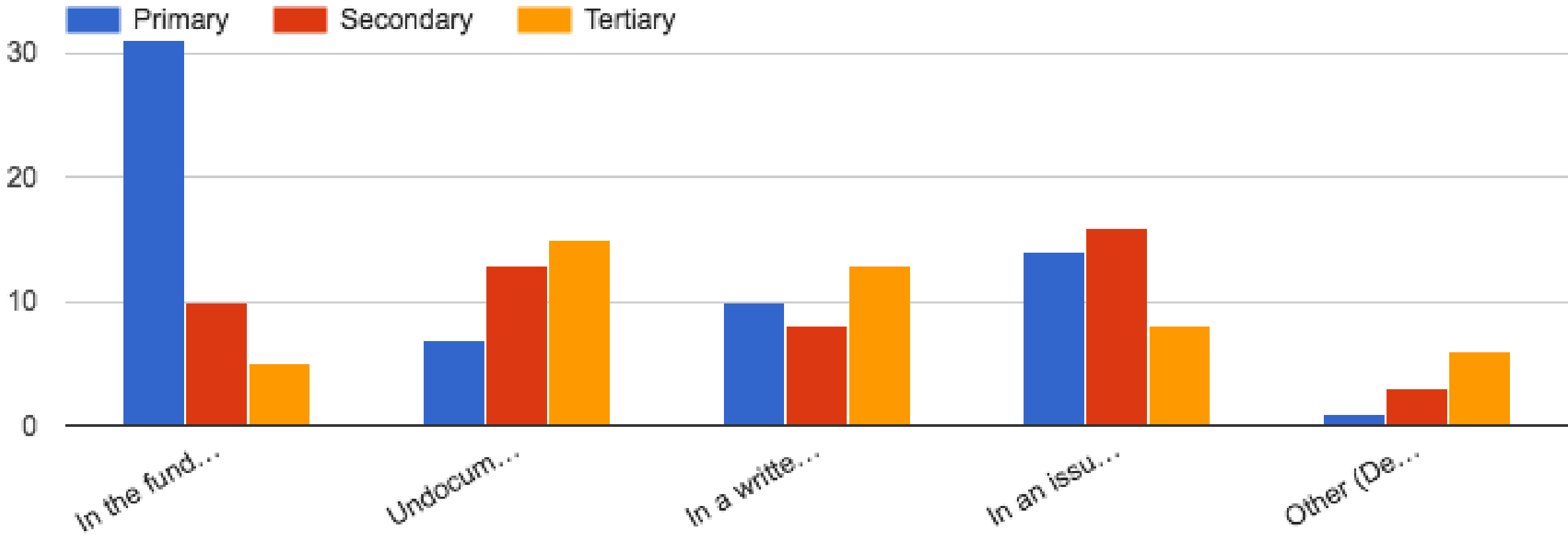
	Primary	Secondary	Tertiary
The source code is the documentation.	○	○	○
Research publications.	○	○	○
From embedded source comments (e.g., Doxygen).	○	○	○
We provide content in a user guide.	○	○	○
In product release notes.	○	○	○
Other (Describe at end of survey).	○	○	

Questions 9 – 10

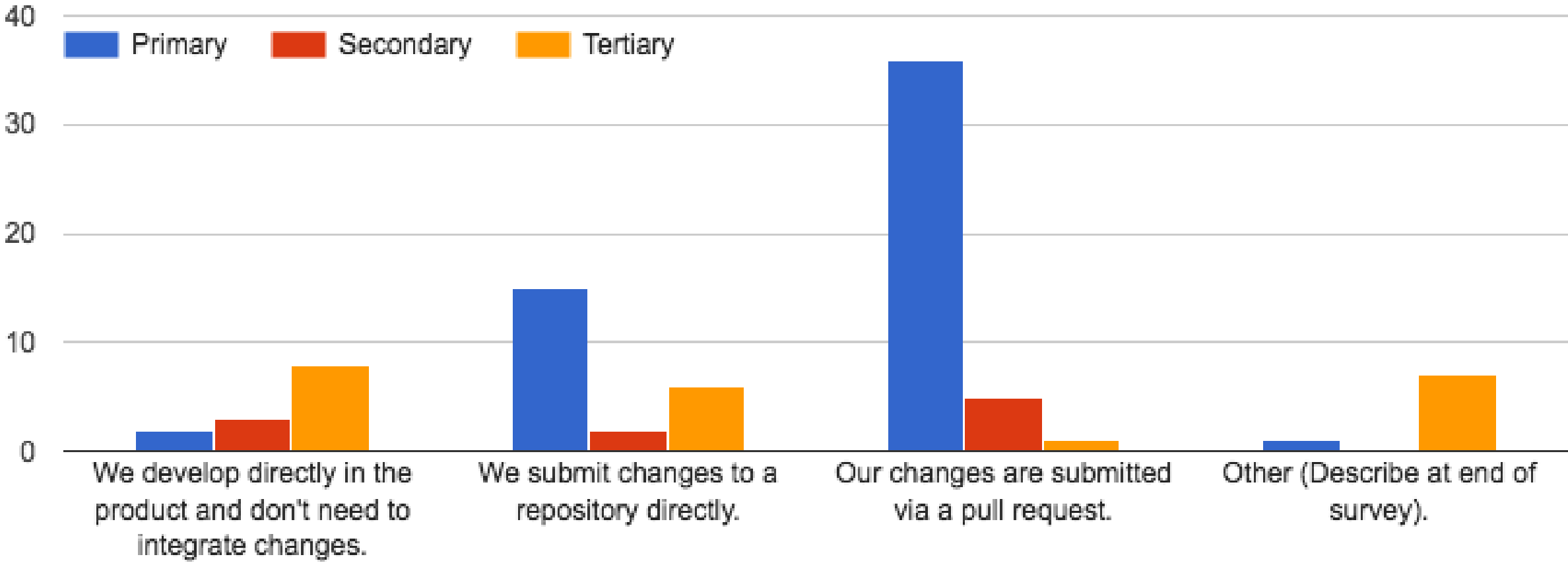
Partial Survey Results

- Results still coming in.
- Next slides are a raw presentation of 48 responses.

Where is the most detailed description of what is required from new capabilities you plan to develop?



How are new capabilities integrated into the existing product?

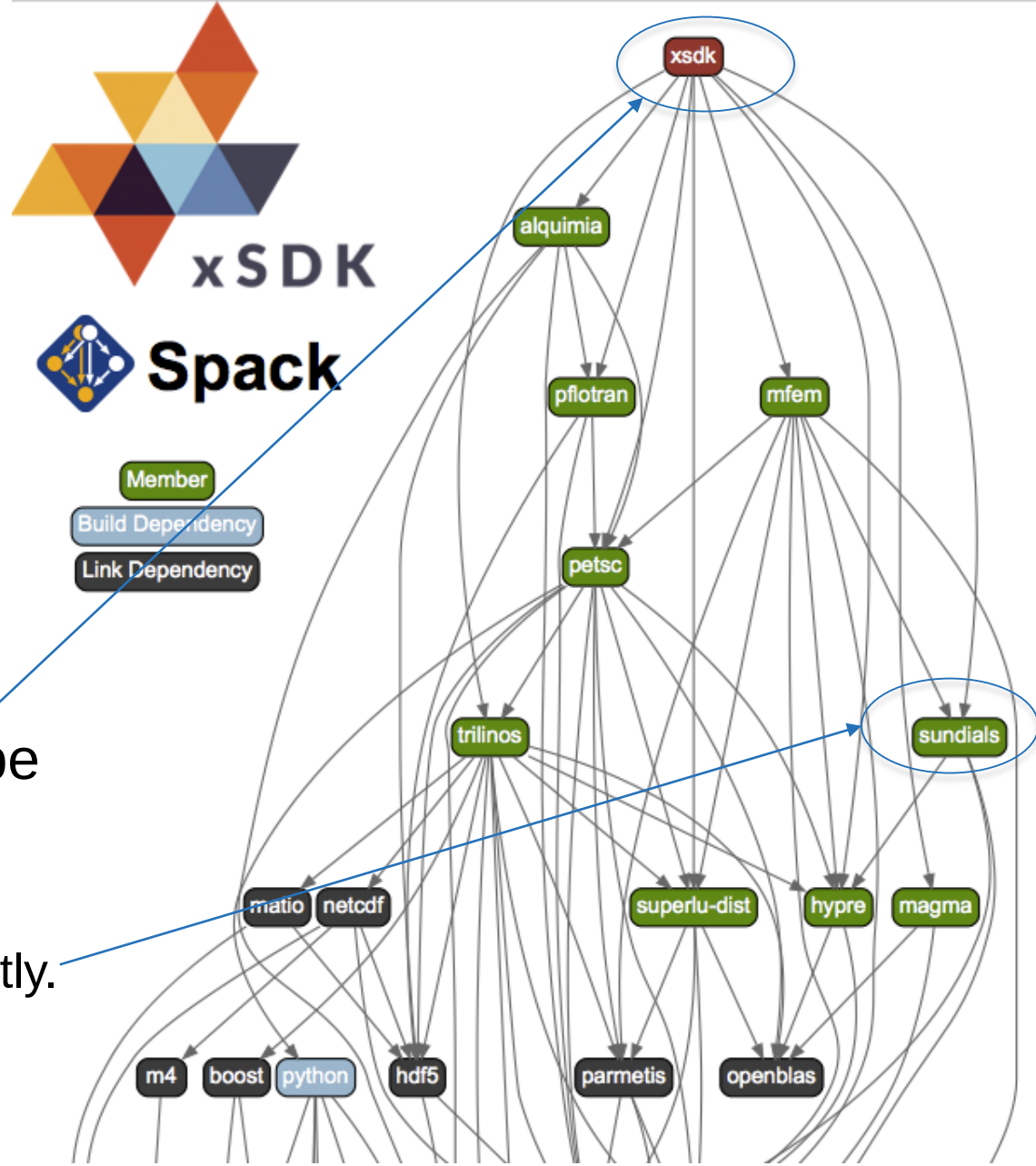


ECP ST Software Release Overview

- ECP ST Delivers Products as Source Code:
  - Tested regularly on target platforms including pre-Exascale.
  - Facilities, Vendors, Users pull from open repositories, coordinated with ECP HI.
- Build from Source
  - Spack build tool:
  - Hierarchical build of independently-developed products.
- Continuous Integration Testing (under construction).
  - ECP ST Products automatically pulled from development repositories.
  - Integrated regression testing performed on key DOE platforms.
- First Releases
  - September 30: Practice Release, Tests integrity of release process.
  - **November 8:** First ECP ST Release, Collection of Products Ready for Release (est. 45).

ECP ST Software Release Goals

- Build All ST Products that are ready.
  - Product readiness is part of success criteria.
  - Number of releasable products increase over time.
- SDKs will provide product suites.
  - Similar products, interoperable.
  - Consistent versions of dependencies.
  - Math SDK (aka, xSDK) is first SDK.
- We build the whole tree, so any subtree will be stable.
  - spack install xsdk – Build entire math SDK.
  - spack install sundials – Guaranteed to build correctly.



ECP ST FY19/20 Preparations for CD2.

- New L4 efforts:
  - Future Technology Directions Project.
    - Tracks external efforts that can inform and influence ECP ST efforts.
    - Reduces risk that ECP ST succeeds in its milestones, but creates unnecessary incompatibilities.
  - Quality Assessment Team:
    - Review and assessment of ST team readiness for Exascale platforms.
    - Assessment of software quality efforts, specifically software practices.

ECP ST L4 Project Adjustment & Augmentation Goals

- Need larger teams merged from existing projects:
  - More uniform size and expectations (presently more than 10x range in size).
  - Need integrated project resources (staffing) for managing CD-2 process expectations.
  - Create opportunities for SDKs to have structural support.
- Anticipated outcomes:
  - Current project teams will retain technical scope integrity and leadership.
  - Technical staff will spend more time on technical work.
  - Cost of implementing CD-2 formality will be amortized and quality improved.
- Details will emerge over the next 4 – 5 months.
- Goals:
  - Use Oct Review output to inform planning.
  - Have project adjustments & augmentations defined in Mar - April 2019, including FY20 milestones.
  - Deploy structure in FY20.

Summary (1)

- ECP Software Technology contributes to a broad spectrum of HPC software products:
  - 89 products total.
  - 33 broadly used in HPC.
    - Require substantial investment and transformation in preparation for exascale architectures
  - Additional 23 important to some existing applications,
    - Typically represent new capabilities that enable new usage models for realizing the potential that Exascale platforms promise.
  - Remaining products are in early development phases
    - Addressing emerging challenges and opportunities that Exascale platforms present.
- SDKs provide a key horizontal coupling to enhance all aspects of ECP software activities.

Summary (2)

- Access to Exascale Systems Info:
  - Structurally hard: Information scoping by lab, ECP is multi-lab (and universities).
  - Working with HI and Intel for A21: Simulator access (small group), Briefing (large group).
- ST Reviews (Oct 1 – 5) & Release (Nov 8).
- Software Practices:
  - Census wrapping up.
  - Explore next steps, consult with industry partners.
- Adjustments/augmentations for CD2:
  - Balanced project sizes.
  - Better support for planning.