xSDK: an Ecosystem of Interoperable Independently Developed Math Libraries

Ulrike Meier Yang

CASC
Center for Applied Scientific Computing

September 20, 2023

Supercomputing Spotlights

LLNL-PRES-854440

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344.
Lawrence Livermore National Security, LLC

Lawrence Livermore National Laboratory

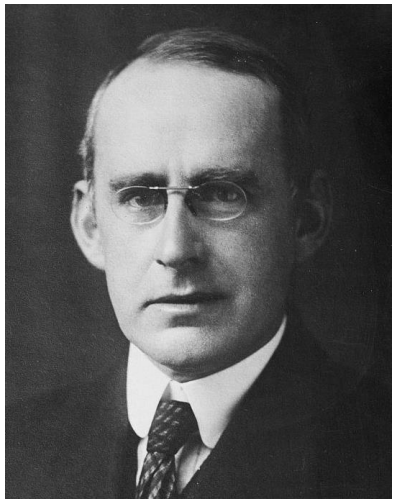
What is an ecosystem?

Ecosystem: A group of independent but interrelated elements comprising a unified whole

Ecosystems are challenging!

“ We used to think that if we knew one, we knew two, because one and one are two. We are finding that we must learn a great deal more about 'and'. ”

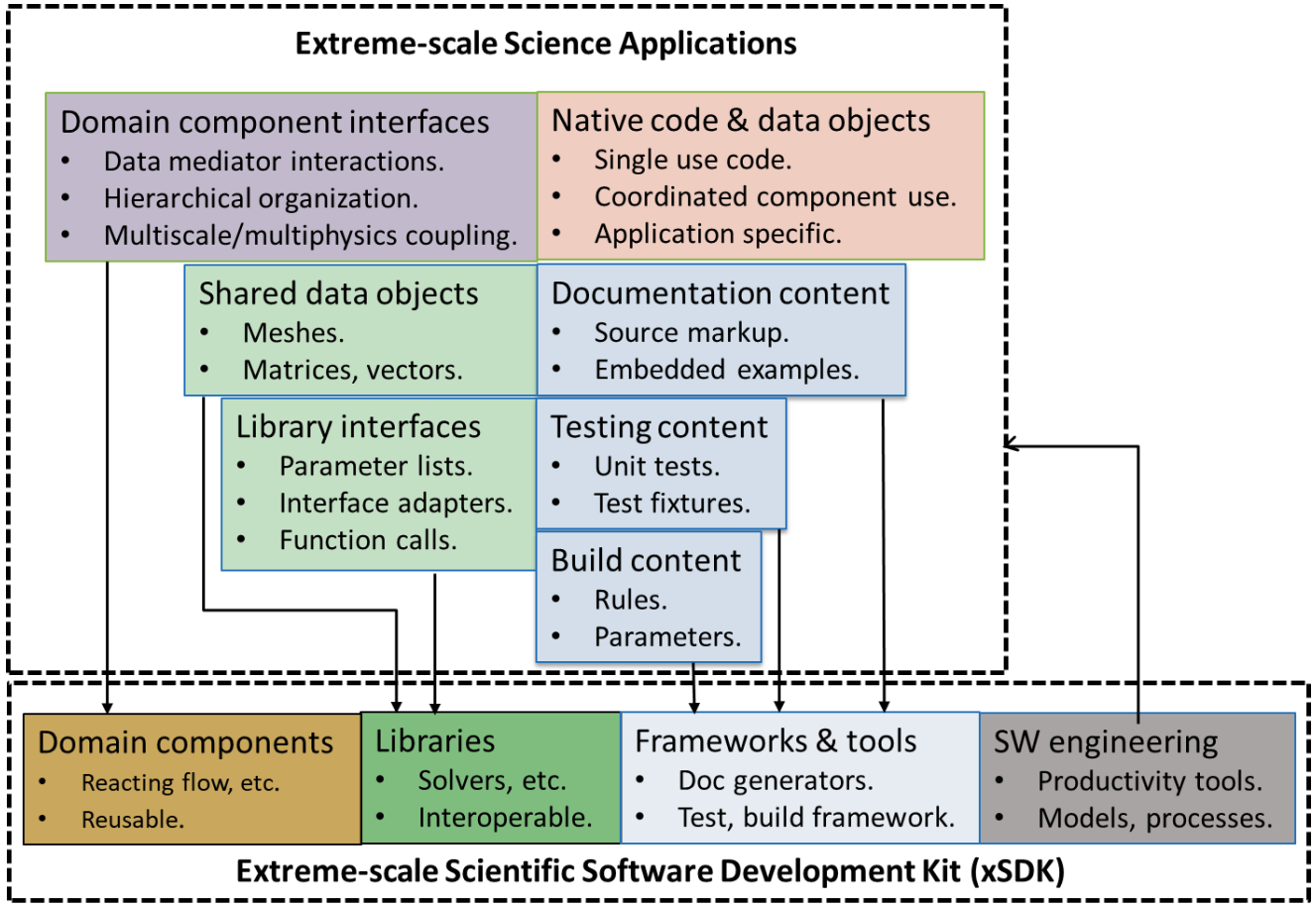
– Sir Arthur Stanley Eddington (1892–1944), British astrophysicist



Effective ecosystem → **Impact(ecosystem) > ∑ Impact (elements)**

Extreme-Scale Scientific Software Development Kit (xSDK)

- Ecosystem of interoperable, but independently developed math libraries
- **Goal:**
  Increase combined usability, standardization and interoperability of libraries, as needed to support large-scale multiphysics and multiscale problems



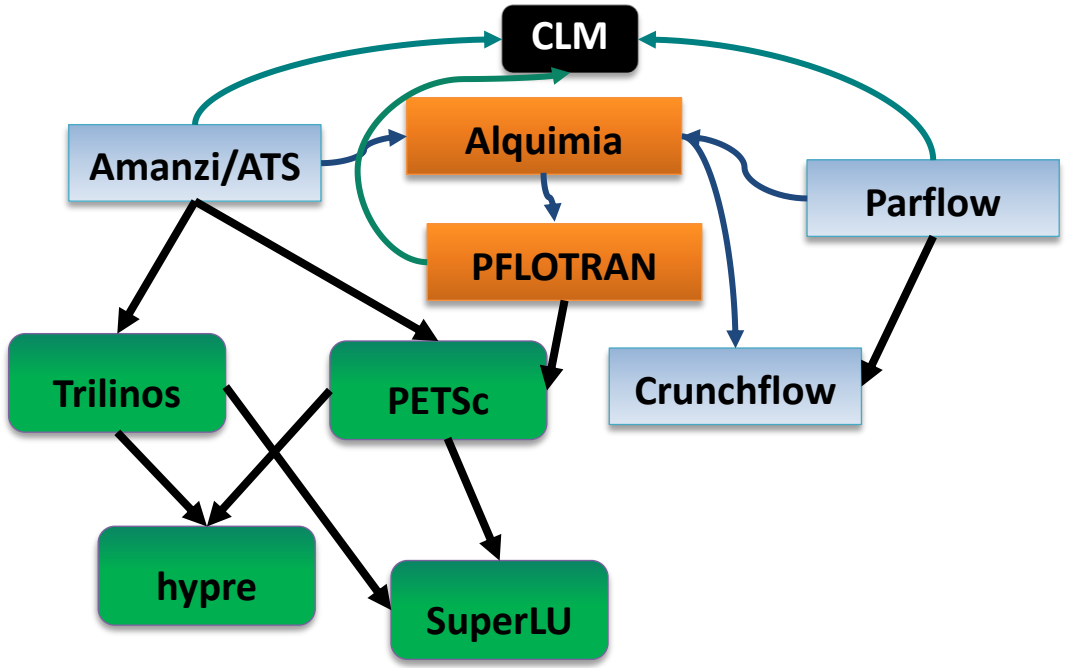
Outline

- Brief History of xSDK
- Software ecosystem and its elements
  - xSDK libraries
  - Spack package manager
  - xSDK community policies
- Achieving an efficient ecosystem
  - High Software Quality
  - Portability
  - Interoperability
  - Sustainability
- Future Plans

History of xSDK

xSDK history: Work began in ASCR/BER partnership, Sept 2014

Needed for BER multiscale, multiphysics integrated surface-subsurface hydrology models



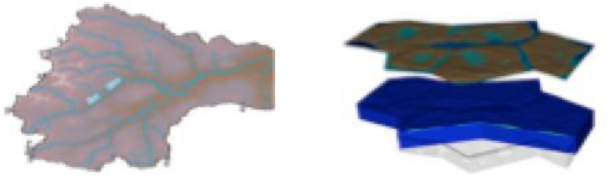
Next-generation scientific simulations require combined use of independent packages

- **Prior to xSDK effort, many difficulties to build required libraries into a single executable due to many incompatibilities**
- Installing multiple independent software packages is tedious and error prone
  - Need consistency of compiler (+version, options), 3rd-party packages, etc.
  - Namespace and version conflicts make simultaneous build/link of packages difficult
- Multilayer interoperability among packages requires careful design and sustainable coordination

IDEAS productivity
www.ideas-productivity.org

Interoperable Design of Extreme-scale Application Software (IDEAS)

First-of-a-kind project:
qualitatively new approach based on making productivity and sustainability the explicit and primary principles for guiding our decisions and efforts.



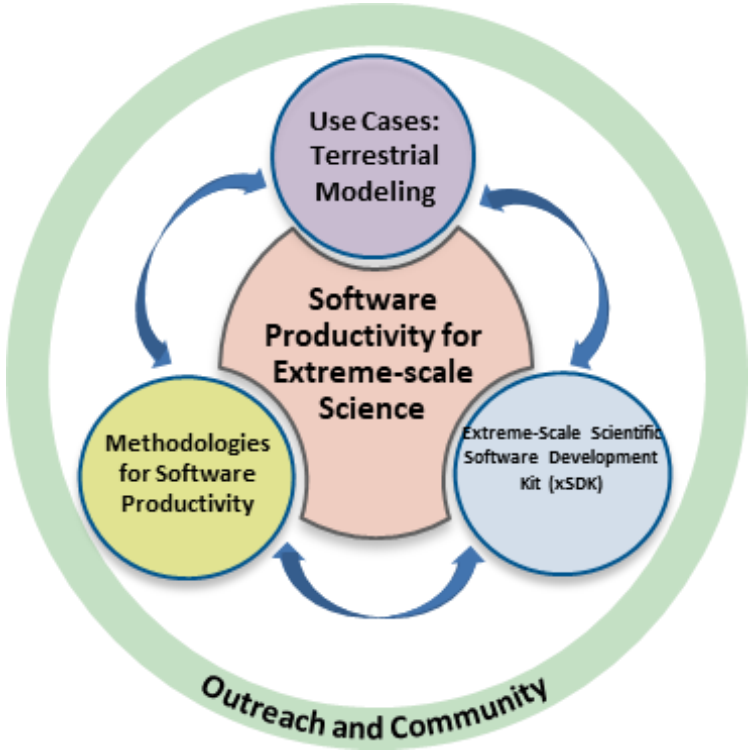
Interdisciplinary multi-lab team (ANL, LANL, LBNL, LLNL, ORNL, PNNL, SNL)

ASCR Co-Leads: Mike Heroux (SNL) and Lois Curfman McInnes (ANL)

BER Lead: David Moulton (LANL)

ASCR/BER partnership ensures delivery of both crosscutting methodologies and metrics with impact on real application and programs.

Integration and synergistic advances in three communities
deliver scientific productivity; outreach establishes a new holistic perspective for the broader scientific community.

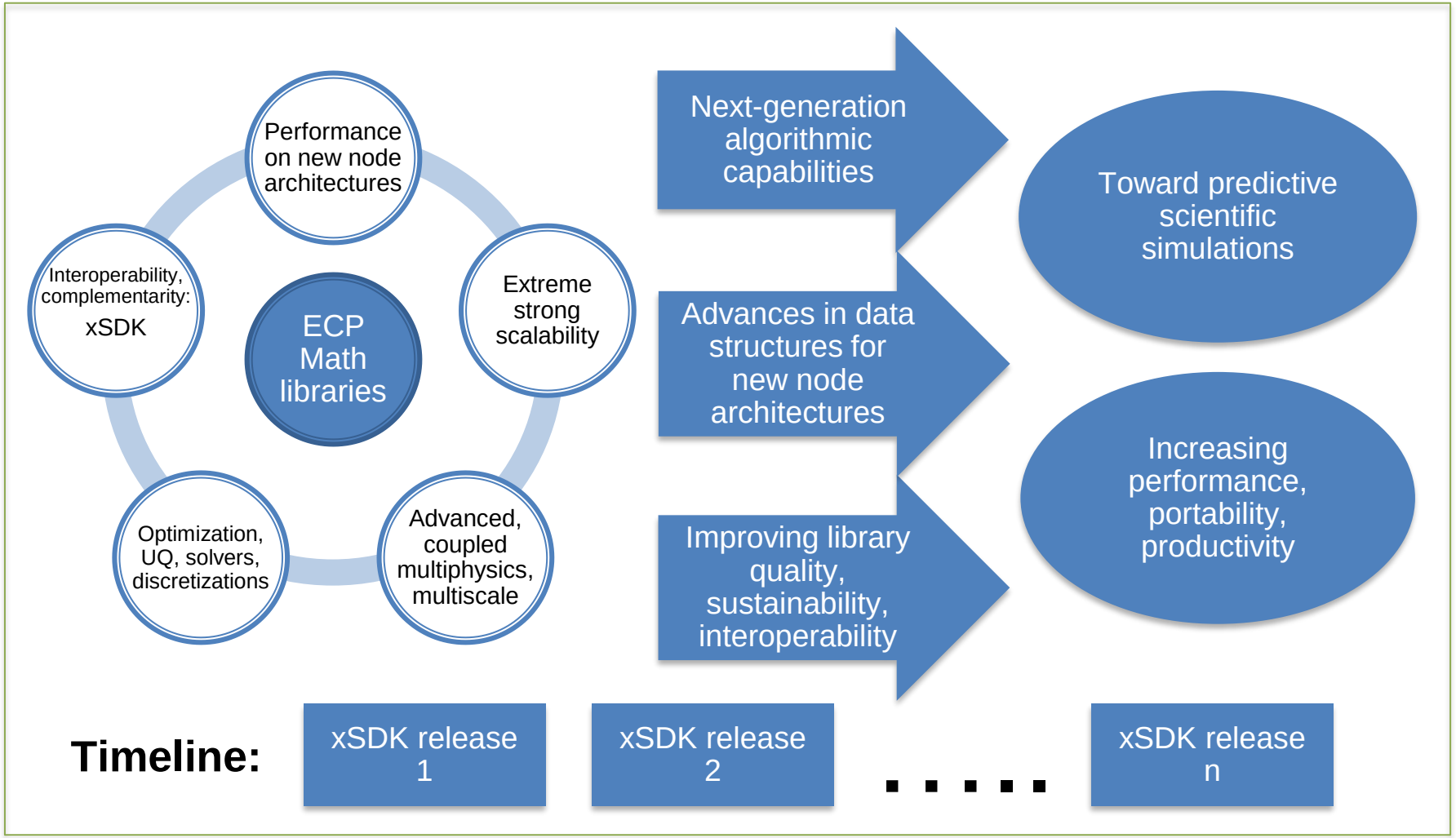


Project began in Sept 2014, ended in Sept 2017

Continuation of xSDK in Exascale Computing Project (ECP)

- ECP: collaborative effort of DOE-SC and NNSA
- Started in Oct 2016
- xSDK-ECP project

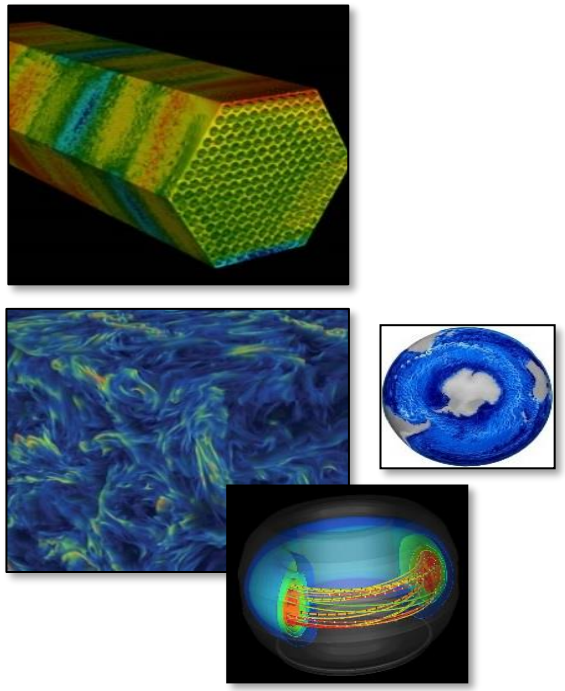
xSDK is key delivery mechanism for ECP math libraries continual advancements toward predictive science



Exascale Computing Project (ECP)'s holistic approach uses co-design and integration to achieve exascale computing

Application Development

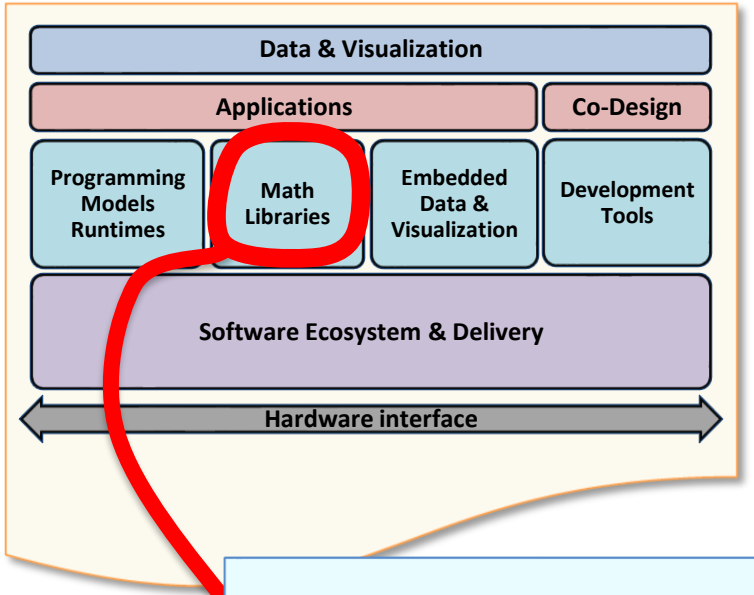
Science and mission applications



Software Technology

Scalable software stack

E4S



Hardware and Integration

Relationships: facilities with AD/ST, with vendors

Emphasis for this presentation

EXASCALE COMPUTING PROJECT

DOE HPC Roadmap to Exascale Systems

FY 2012	FY 2016	FY 2018	FY 2021	FY 2022	FY 2023	FY 2024

ORNL
Cray/AMD/NVIDIA

ORNL
IBM/NVIDIA

ANL
IBM BG/Q

ANL
Cray/Intel KNL

LBNL
Cray/Intel Xeon/KNL

LBNL
HPE/AMD/NVIDIA

ORNL
HPE/AMD

ANL
HPE/Intel

Exascale Systems

LLNL
IBM BG/Q

LANL/SNL
Cray/Intel Xeon/KNL

LLNL
IBM/NVIDIA

LANL/SNL
HPE/TBD

LLNL
HPE/AMD

ECP applications need sustainable coordination among math libraries

ECP AD Teams

Combustion-Pele, EXAALT, ExaAM, ExaFEL, ExaSGD, ExaSky, ExaStar, ExaWind, GAMESS, MFIX-Exa, NWChemEx, Subsurface, WarpX, WDMApp, WarpX, ExaAM, ATDM (LANL, LLNL, SNL) apps, AMReX, CEED, CODAR, CoPA, ExaLearn

Examples:

- **Subsurface:** Chombo, PETSC, hypre, etc, …
- **ExaAM:** DTK**,** SUNDIALS, Tasmanian, hypre, Trilinos, FFT, etc.
- **ExaWind:** hypre, KokkosKernels, SuperLU, Trilinos, AMReX, etc.
- **WDMApp:** PETSc, hypre, SuperLU, STRUMPACK, FFT, etc.
- **CEED:** MFEM, MAGMA, hypre, PETSc, SuperLU, Sundials, etc.
- And many more …

ECP Math Libraries

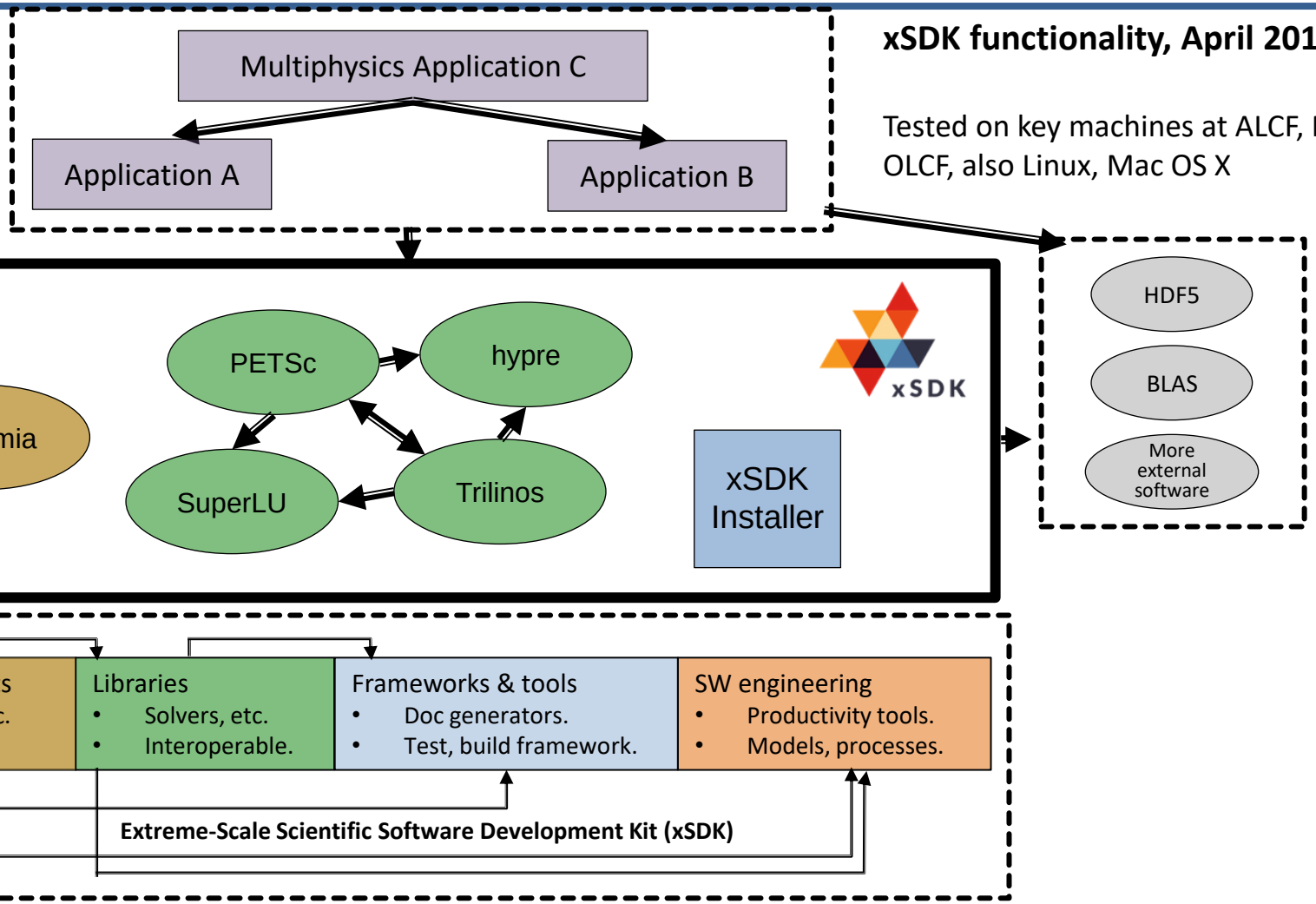
xSDK History: Version 0.1.0: April 2016

https://xsdk.info

Notation: A⇒B:

A can use **B** to provide functionality on behalf of **A**

April 2016
- 4 math libraries
- 1 domain component
- PETSc-based xSDK installer
- 14 mandatory xSDK community policies



xSDK functionality, April 2016

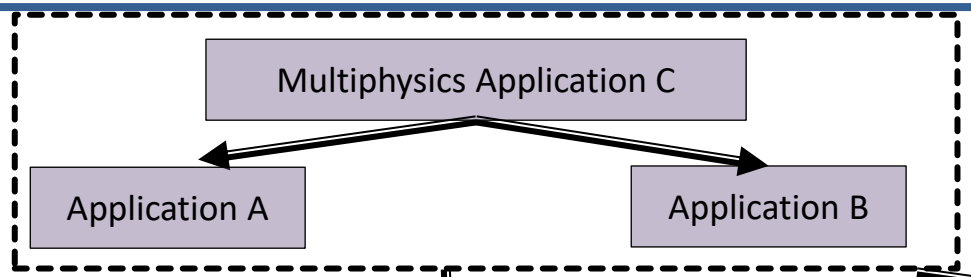
Tested on key machines at ALCF, NERSC, OLCF, also Linux, Mac OS X

xSDK Version 0.8.0: November 2022

https://xsdk.info

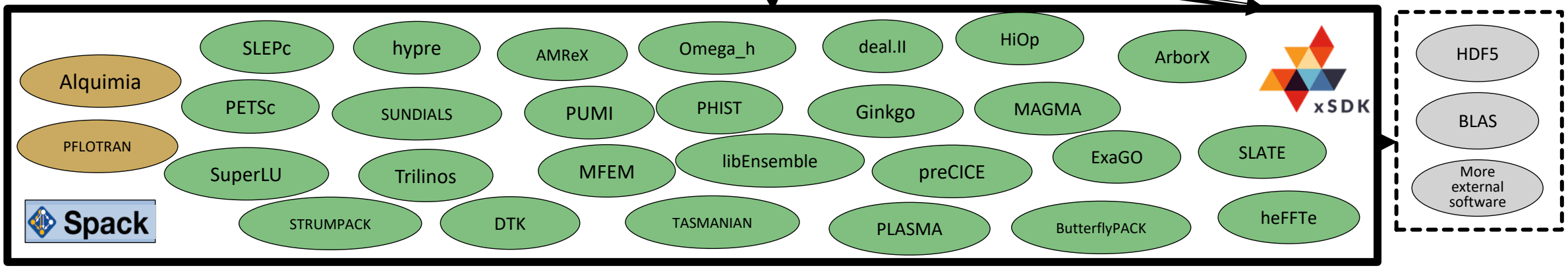
https://xsdk.info

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



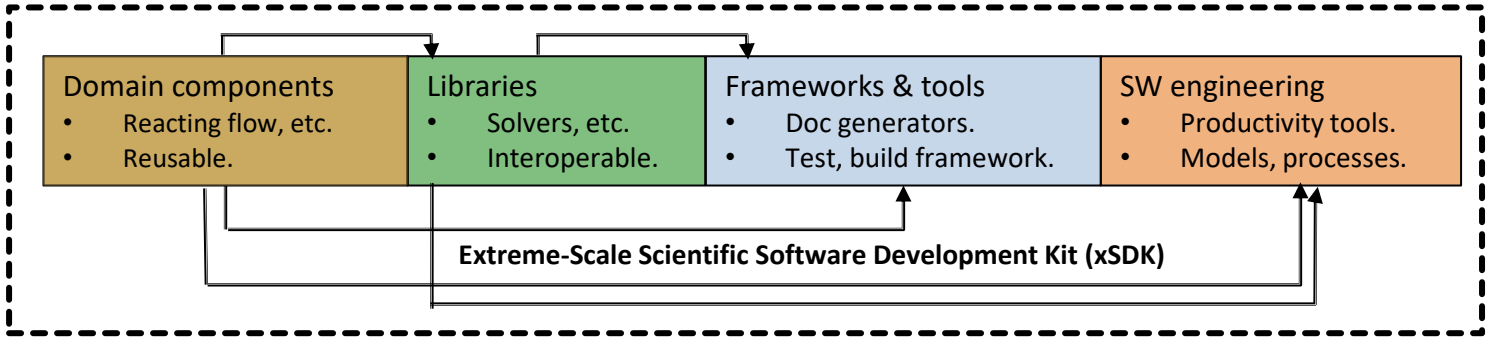
xSDK functionality, Nov 2022

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



November 2022

- 26 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

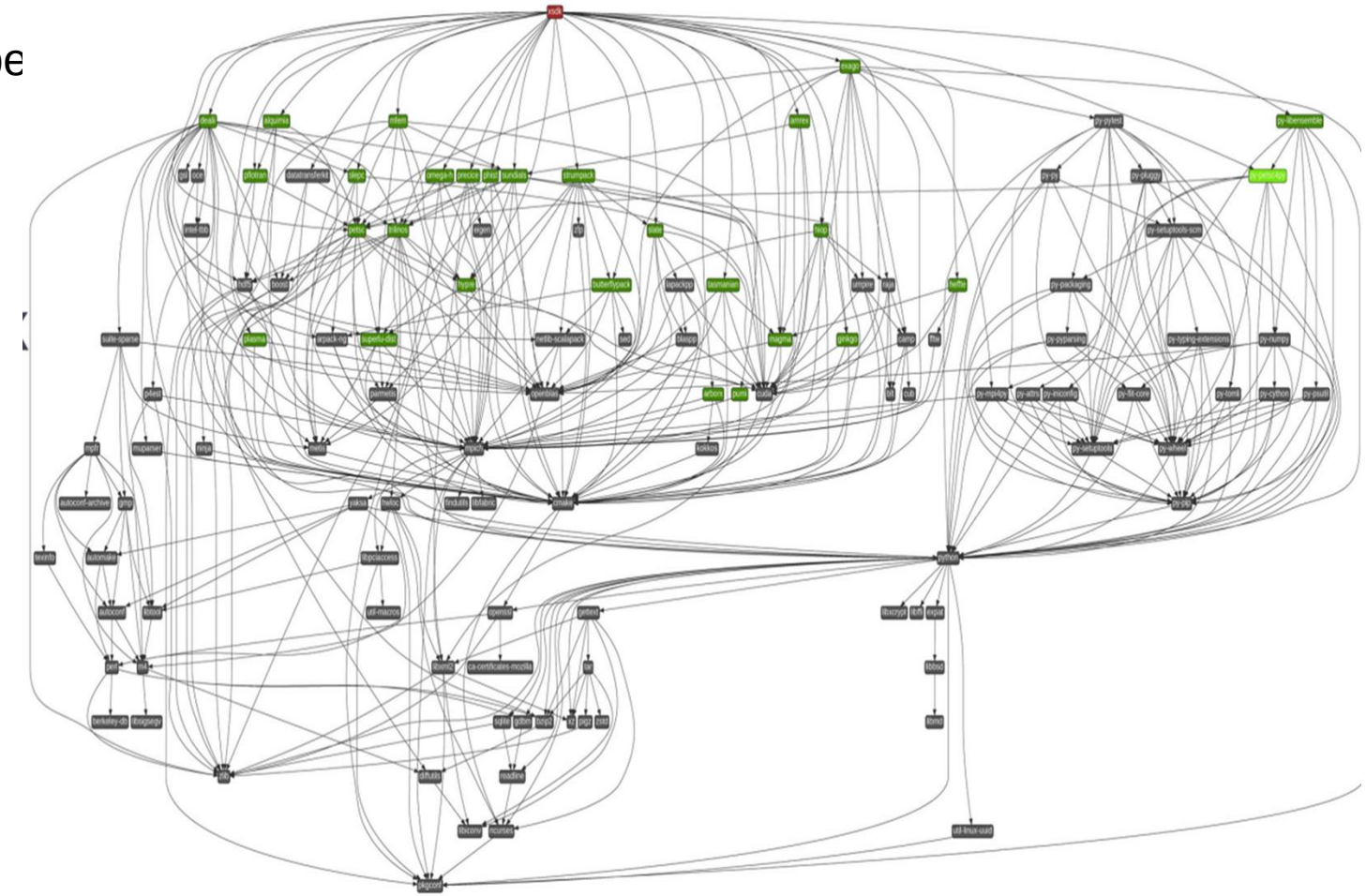
xSDK Elements

- Spack build manager
- Math libraries
- Community policies

The xSDK is using Spack to deploy its software

- The xSDK packages depend on a numbe of open-source libraries
- Spack is a flexible package manager for HPC
- Spack allows the xSDK to be deployed with a single command
  - User can optionally choose compilers, build options, etc.

Spack github.com/spack



xSDK Libraries

- **AMReX**: Ann Almgren (LBNL)
- **ArborX**: Daniel Arndt (ORNL)
- **DTK**: Bruno Turcksin (ORNL)
- **deal.II**: Wolfgang Bangerth (Colorado State University)
- **ExaGO:** Shrirang Abhyankar (PNNL)
- **Ginkgo**: Hartwig Anzt (Karlsruhe Institute of Technology)
- **heFFTe:** Stan Tomov (UTK)
- **HiOp:** Cosmin Petra (LLNL)
- **hypre**: Rob Falgout, Ulrike Yang (LLNL)
- **libEnsemble**: Steve Hudson (ANL)
- **MAGMA** and **PLASMA**: Piotr Luszczek (UTK)
- **MFEM**: Tzanio Kolev (LLNL)
- **Omega_h, PUMI**: Cameron Smith (RPI)
- **PETSc/TAO**: Satish Balay, Todd Munson (ANL)
- **preCICE:** Frederic Simonis (Technical University Munich)
- **SUNDIALS**: Cody Balos, David Gardner, Carol Woodward (LLNL)
- **SuperLU, STRUMPACK, ButterflyPACK**: Sherry Li, Pieter Ghysels, Yang Liu (LBNL)
- **TASMANIAN**: Miroslav Stoyanov (ORNL)
- **Trilinos**: Jim Willenbring (SNL)
- **PHIST**: Jonas Thies (DLR, German Aerospace Center)
- **SLEPc**: José Roman (Universitat Politècnica de València)

xSDK: https://xsdk.info

Building the foundation of an extreme-scale scientific software ecosystem

xSDK community policies: Help address challenges in interoperability and sustainability of software developed by diverse groups at different institutions

https://github.com/xsdk-project/xsdk-community-policies

xSDK compatible package: must satisfy the mandatory xSDK policies (M1, ..., M17)

Topics include configuring, installing, testing, MPI usage, portability, contact and version information, open-source licensing, namespacing, documentation, public repository access

Also specify **recommended policies**, which currently are encouraged but not required (R1, ..., R8)

Topics include error handling, freeing system resources, and library dependencies

xSDK member package:

(1) Must be an xSDK-compatible package, *and*

(2) it uses or can be used by another package in the xSDK, and the connecting interface is regularly tested for regressions.

xSDK policies 1.0.0: Feb 2023

- Facilitate combined use of independently developed packages

Impact:

- Improved code quality, usability, access, sustainability
- Foundation for work on deeper levels of interoperability and performance portability

We encourage feedback and contributions!

xSDK community policies

https://github.com/xsdk-project/xsdk-community-policies

https://xsdk.info/policies

Version 1.0.0, February 2023

Mandatory xSDK policies: must be satisfied

M1. Support portable installation through Spack (includes xSDK Spack variant guildelines)

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 publicly available repository.

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. Have a debug build option.

M17. Provide sufficient documentation to support use and further development.

Recommended xSDK policies: currently encouraged, but not required

R1. Provide at least one validation test that can be invoked through Spack.

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.

R6. Provide versions of dependencies.

R7. Have README, SUPPORT, LICENSE, and CHANGELOG file in top directory.

R8. Provide version comparison preprocessor macros.

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.

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

Adding, Changing, Retiring Community Policies

- xSDK policies are reviewed and, if needed, updated regularly
- Changes in policies maybe needed due to software and/or hardware changes
- Recommended policies may migrate to become mandatory ones
- To maintain community, members have to agree on the set of policies or any changes over time
- xSDK team members seek input from the larger community of users and arrive at consensus (or majority) how to take the feedback into account

Seek community input

Discuss feedback

Consensus vote

https://github.com/xsdk-project/xsdk-community-policies

Compatibility with xSDK community policies

To help developers of packages who are considering compatibility with xSDK community policies, we provide:

- Template with instructions to record compatibility progress
- Examples of compatibility status for xSDK packages
  - Explain approaches used by other packages to achieve compatibility with xSDK policies
- Available at **https://github.com/xsdk-project/xsdk-policy-compatibility**

xSDK Community Policy Compatibility for PETSc

This document summarizes the efforts of current and future xSDK member packages to achieve compatibility with the xSDK community policies. Below only short descriptions of each policy are provided. The full description is available here and should be considered when filling out this form.

Please, provide information on your compability status for each mandatory policy, and if possible also for recommended policies. If you are not compatible, state what is lacking and what are your plans on how to achieve compliance. For current xSDK member packages: If you were not compliant at some point, please describe the steps you undertook to fulfill the policy.This information will be helpful for future xSDK member packages.

Website: https://www.mcs.anl.gov/petsc

Mandatory Policies

Policy	Support	Notes
M1. Support xSDK community GNU Autoconf or CMake options.	Full	PETSc uses the GNU Autoconf options. The implementation is done with python code.
M2. Provide a comprehensive test suite for correctness of installation verification.	Full	PETSc has over 1000 test examples and a test harness that can execute the examples in parallel. It also collects information on the failures and can display them graphically, e.g., see ftp://ftp.mcs.anl.gov/pub/petsc/nightlylogs/archive/2017/09/19/master.html
M3. Employ userprovided MPI communicator (no MPI_COMM_WORLD).	Full	All PETSc objects take a MPI communicator in the constructor, allowing the user complete control over where each object exists and performs its computations.
M4. Give best effort at portability		

What is required for an effective ecosystem?

- High software quality
- Portability
- Interoperability
- Sustainability

Software Quality

Version 1.0.0,
February 2023

Mandatory xSDK policies: must be satisfied

M1. Support portable installation through Spack (includes xSDK Spack variant guildelines)

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 publicly available repository.

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. Have a debug build option.

M17. Provide sufficient documentation to support use and further development.

Recommended xSDK policies: currently encouraged, but not required

R1. Provide at least one validation test that can be invoked through Spack.

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.

R6. Provide versions of dependencies.

R7. Have README, SUPPORT, LICENSE, and CHANGELOG file in top directory.

R8. Provide version comparison preprocessor macros.

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.

Portability

Version 1.0.0,
February 2023

Mandatory xSDK policies: must be satisfied

M1. Support portable installation through Spack (includes xSDK Spack variant guildelines)

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 publicly available repository.

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. Have a debug build option.

M17. Provide sufficient documentation to support use and further development.

Recommended xSDK policies: currently encouraged, but not required

R1. Provide at least one validation test that can be invoked through Spack.

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.

R6. Provide versions of dependencies.

R7. Have README, SUPPORT, LICENSE, and CHANGELOG file in top directory.

R8. Provide version comparison preprocessor macros.

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.

EXASCALE COMPUTING PROJECT

Portability Strategies of xSDK Libraries

- Use of portable programming models that provide abstractions

kokkos SYCL RAJA HIP

- Use of abstraction to limit code that interacts with devices

> **"The way you get programmer productivity is by eliminating lines of code you have to write."**
>
> – Steve Jobs, Apple World Wide Developers Conference, Closing Keynote, 1997

- Use of fast kernel libraries designed for individual architectures

cuBLAS, cuSPARSE
rocBLAS, rocSPARSE
MKL

- Write own CUDA kernels, and use vendor provided tools to port kernels
- Develop new algorithms more suitable for GPUs
  (most challenging, but possibly best results!)

Interoperability

Version 1.0.0,
February 2023

Mandatory xSDK policies: must be satisfied

M1. Support portable installation through Spack (includes xSDK Spack variant guildelines)

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 publicly available repository.

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. Have a debug build option.

M17. Provide sufficient documentation to support use and further development.

Recommended xSDK policies: currently encouraged, but not required

R1. Provide at least one validation test that can be invoked through Spack.

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.

R6. Provide versions of dependencies.

R7. Have README, SUPPORT, LICENSE, and CHANGELOG file in top directory.

R8. Provide version comparison preprocessor macros.

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.**

EXASCALE COMPUTING PROJECT

Interoperability is challenging, particularly for deeper levels!

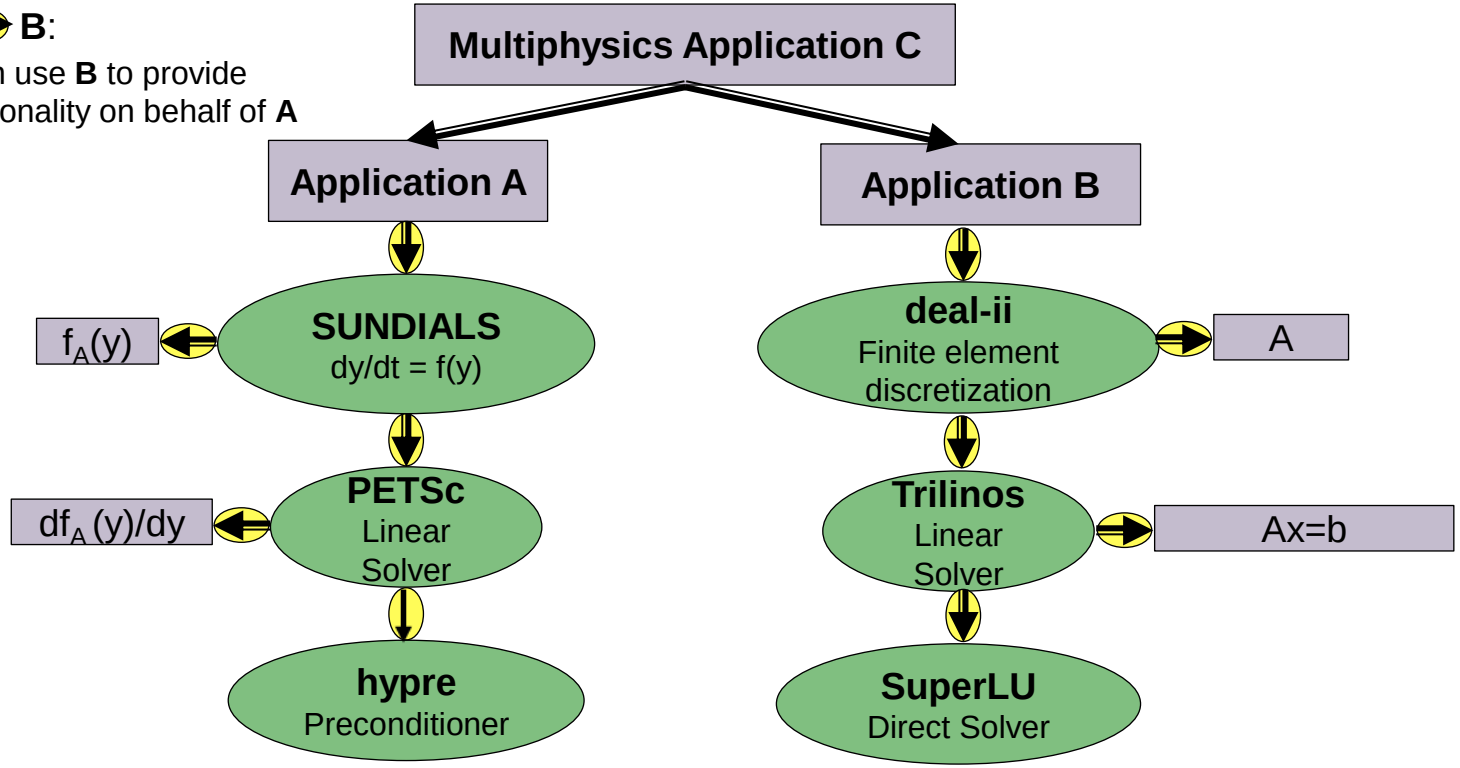
Levels of package interoperability:

- **Interoperability level 1**
  - Both packages can be used (side by side) in an application
- **Interoperability level 2**
  - The libraries can exchange data (or control data) with each other
- **Interoperability level 3**
  - Each library can call the other library to perform unique computations

Notation:

A ⇒ **B**:

A can use **B** to provide functionality on behalf of **A**



xSDK4ECP: Focus on inter-package functionality, denoted by

- Coordinating use of on-node resources
- Integrated execution (control inversion, adaptive execution strategies)

Many more interoperabilities between packages exist!

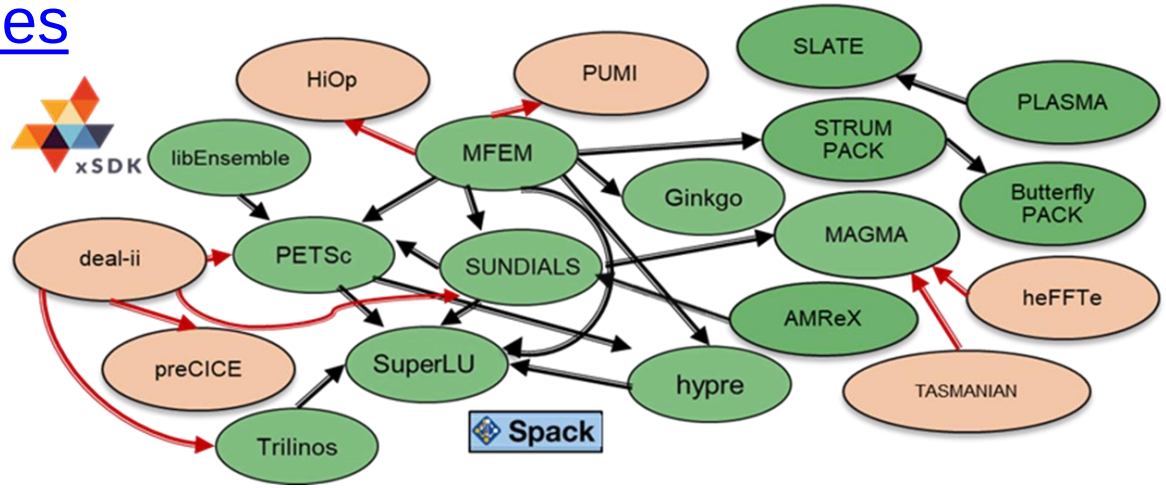
	AMReX	ArborX	ButterflyPACK	deal-ii	DataTransferKit	ExaGO	Ginkgo	heFFTe	HiOp	hypre	libEnsemble	MAGMA	MFEM	Omega_h	PETSc	PHIST	PLASMA	preCICE	PUMI	SLATE	SLEPc	STRUMPACK	SUNDIALS	SuperLU	TASMANIAN	Trilinos
AMReX	■									■					■								■			
ArborX		■																								
ButterflyPACK			■									■														
deal-ii		■		■			■			■					■						■		■	■		■
DataTransferKit		■			■																					■
ExaGO						■			■						■											
Ginkgo							■																			
heFFTe								■			■	■														
HiOp							■		■			■										■				
hypre										■														■		
libEnsemble											■				■											
MAGMA												■														
MFEM							■		■	■			■	■	■				■		■	■	■	■		
Omega_h														■												
PETSc										■					■							■	■	■		
PHIST															■	■										■
PLASMA												■					■									
preCICE				■											■			■								
PUMI														■					■							■
SLATE																				■						
SLEPc															■						■					
STRUMPACK			■									■								■		■				
SUNDIALS										■		■			■								■	■		■
SuperLU																								■		
TASMANIAN												■													■	
Trilinos			■							■	■	■			■				■			■		■		■

■ Interoperability exists

■ Interoperability exists and is enabled in xSDK Spack package

Multi-library example codes demonstrating interoperability

- Suite of example codes has been made available in a github repository and included in the xSDK documentation. : https://github.com/xsdk-project/xsdk-examples
- The example codes are a demonstration of interoperability between xSDK libraries and provide training for xSDK library users interested in using these capabilities.



- Difficulty in building via `spack install xsdk-examples', since new interoperabilities generally not enabled in spack and/or xSDK yet. Provide simple build via `cmake'.
- Test suite important piece of xSDK testing strategy plan

Sustainability

Version 1.0.0,
February 2023

Mandatory xSDK policies: must be satisfied

M1. Support portable installation through Spack (includes xSDK Spack variant guildelines)

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 publicly available repository.

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. Have a debug build option.

M17. Provide sufficient documentation to support use and further development.

Recommended xSDK policies: currently encouraged, but not required

R1. Provide at least one validation test that can be invoked through Spack.

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.

R6. Provide versions of dependencies.

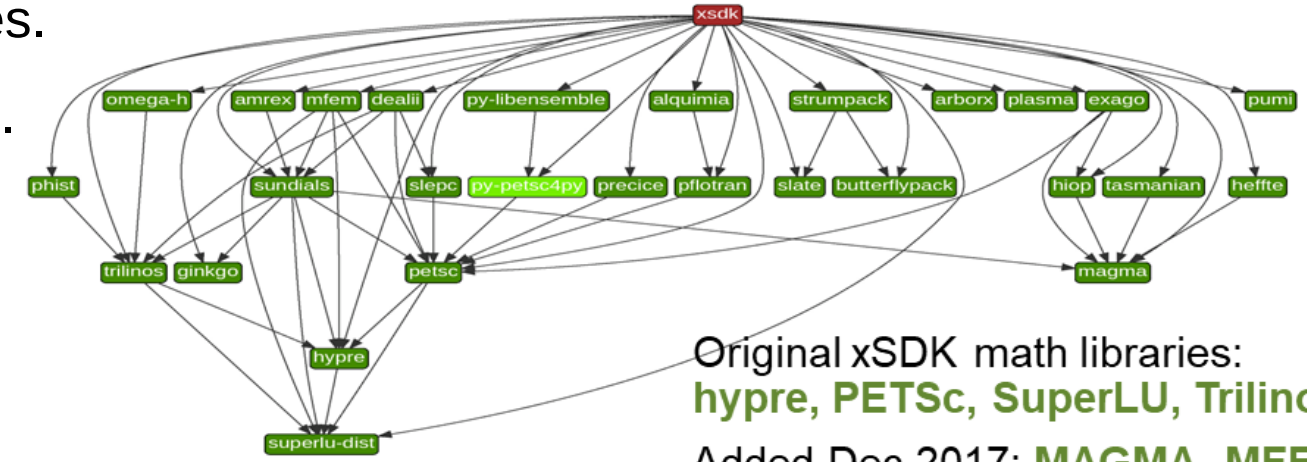
R7. Have README, SUPPORT, LICENSE, and CHANGELOG file in top directory.

R8. Provide version comparison preprocessor macros.

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.

Coordinated releases of complete xSDK with testing, documentation, packaging and deployment

- Demonstrate the impact of community policies to simplify the combined use and portability of independently developed software packages.
- Increase formality of xSDK release process.
- Expand xSDK to include additional key ECP numerical libraries as well as packages in the broader community.
- Pre-exascale environment testing:
  - Summit, Crusher (OLCF)
  - Polaris (ALCF)
  - Perlmutter (NERSC)
- Includes 9 “rocm” and 14 “cuda” enabled libraries.
- Providing specific instructions for these platforms on xSDK website https://xsdk.info/installing-the-software/



xSDK 0.8.0

xSDK

Spack

tested on key platforms at ALCF, NERSC, and OLCF, also Linux and Mac OS X.

Original xSDK math libraries: **hypre, PETSc, SuperLU, Trilinos**

Added Dec 2017: **MAGMA, MFEM, SUNDIALS**

Added Dec 2018: **AMReX, deal.II, DTK, Omega_h, PHIST, PLASMA, PUMI, SLEPc, STRUMPACK, TASMANIAN**

Added Nov 2019: **ButterflyPACK, Ginkgo, libEnsemble, preCICE**

Added Nov 2020: **heFFTe, SLATE**

Added Nov 2021 : **ArborX**

Added Nov 2022 : **ExaGO, HiOp**

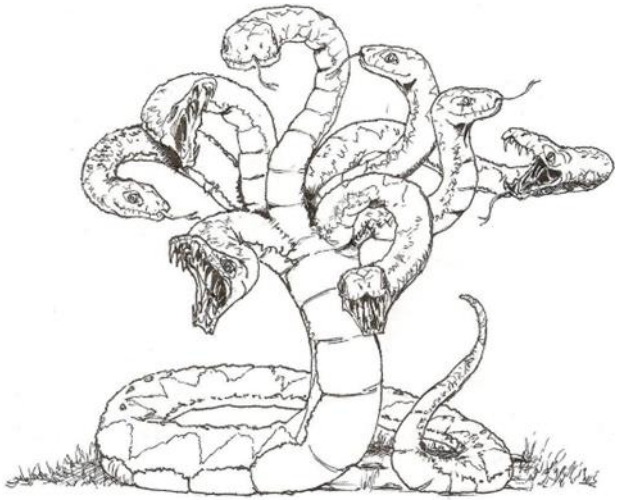
Processes for xSDK release and delivery

- **2-level release process**
  - **xSDK**
    - Ensure and test compatibility of mostly independent package releases
  - **xSDK member packages**
    - Achieve compatibility with xSDK community policies prior to release
      - https://github.com/xsdk-project/xsdk-policy-compatibility
    - Have a Spack package
    - Port to target platforms
    - Provide user support
- **Obtaining the latest release: https://xsdk.info/releases**
- **Draft xSDK package release process checklist:**
  - https://docs.google.com/document/d/16y2bL1RZg8wke0vY8c97ssvhRYNez34Q4QGg4LolEUk/edit?usp=sharing

xSDK delivery process

- Regular releases of software and documentation, primarily through member package release processes
- Anytime open access to production software from GitHub, BitBucket and related community platforms

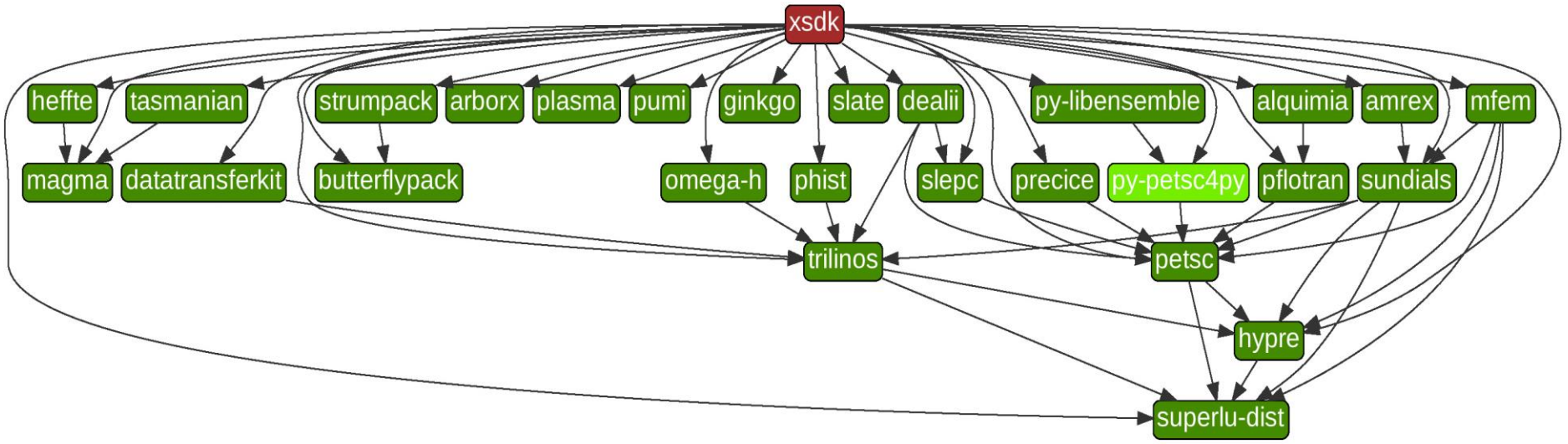
Technical Challenges



- Staying up to date while facing continual changes
  - xSDK release schedule not aligned with individual xSDK library and Spack release schedule
  - Lower dependencies can cause additional problems
- Testing difficulties
  - CI failures need to be investigated to understand what is broken and who should fix it
  - Often there is more than one package causing the issue, but finding the issues is a sequential process, i.e., the first issue needs to be fixed before the next one is discovered
  - The responsible package developers need to be contacted
  - Consistent oversight requires more people to respond to CI failures
- Designed test plan
  - Improve xSDK-examples test suite and integrate it with the xSDK testing process
  - Evaluate and extend current xSDK CI testing through the definition and use of hierarchical test layers, addition of new platforms and increased oversight of test results

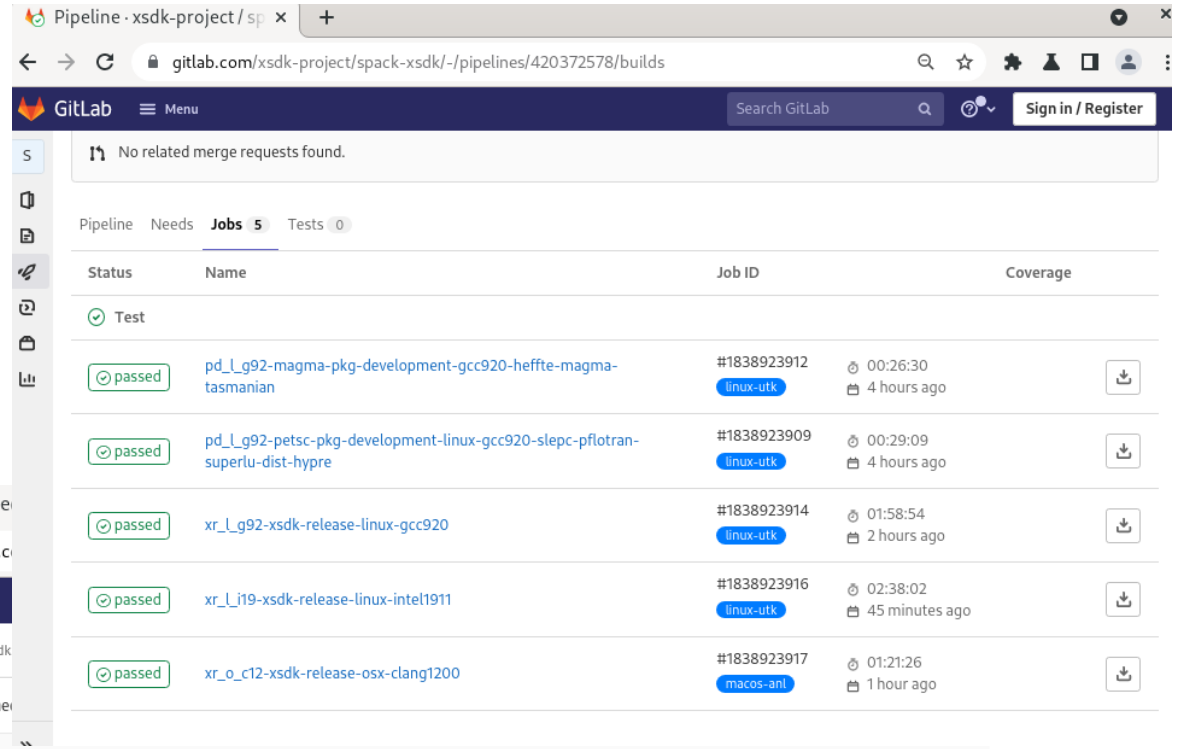
Hierarchical test layers

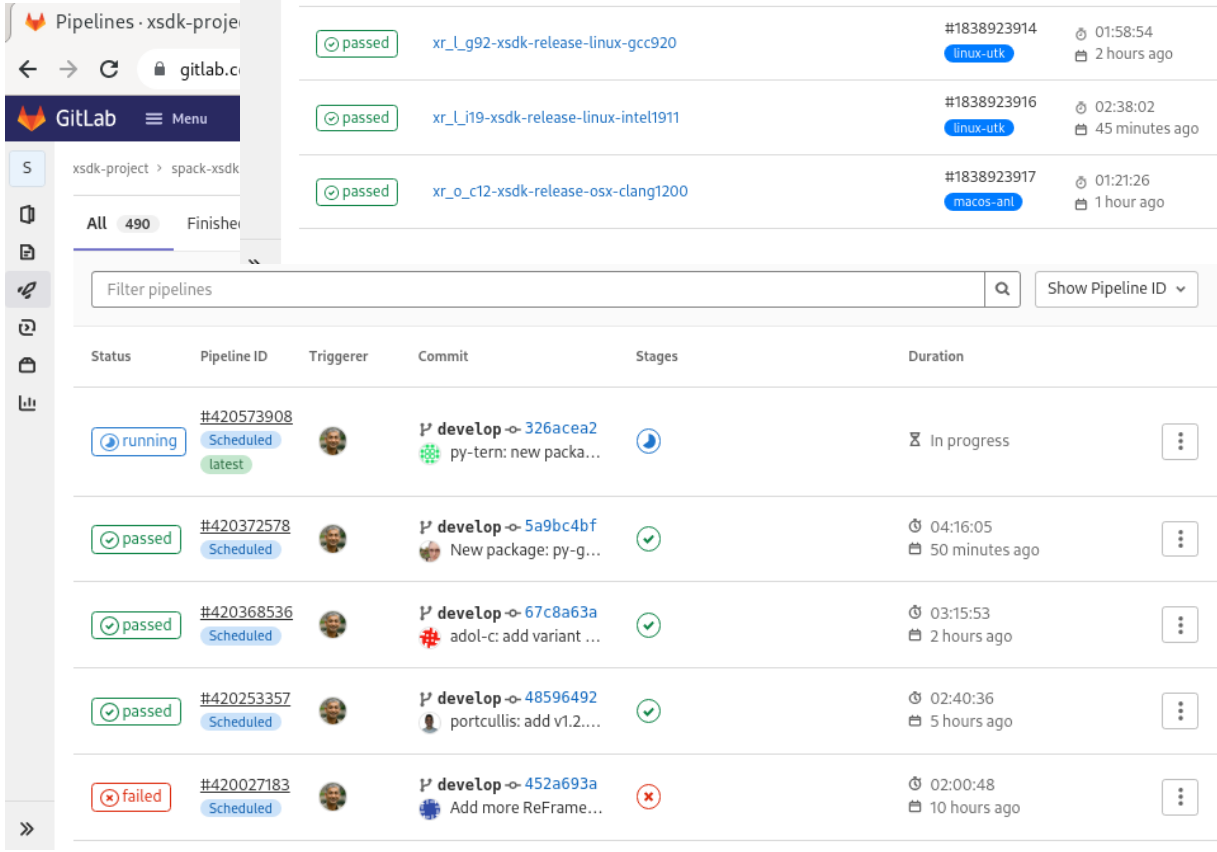
- Multi-layered testing
  - Testing strategies of the individual xSDK libraries
  - Testing of the interfaces between libraries
  - Test subsets of various interoperable packages in combination
  - Define further intermediate levels based on intricacy of library interoperability
  - Testing of the whole xSDK (final level)



xSDK CI/Test setup

- Using Gitlab CI (pipeline) infrastructure at https://gitlab.com/xsdk-project/spack-xsdk/-/pipelines
- Runs multiple tests per pipeline: spack install xsdk
  - MacOS (ANL) with gfortran/clang compilers (xsdk)
  - Linux (UTK) with GNU compilers (xsdk, xsdk-examples+cuda, xsdk-examples)
  - Linux (UTK) with Intel compilers (xsdk)
  - Linux (ANL) with GNU compilers (xsdk-examples)
  - Linux (ANL) with OneAPI(Intel) compilers (xsdk)
- Used as CI for 0.8.0 release work
- Setup for regression testing of 0.8.0 release in spack (scheduled to run a pipeline every day – on latest spack develop branch)
- Added testing of subsets of library development versions to catch build issues early for pre-release testing!

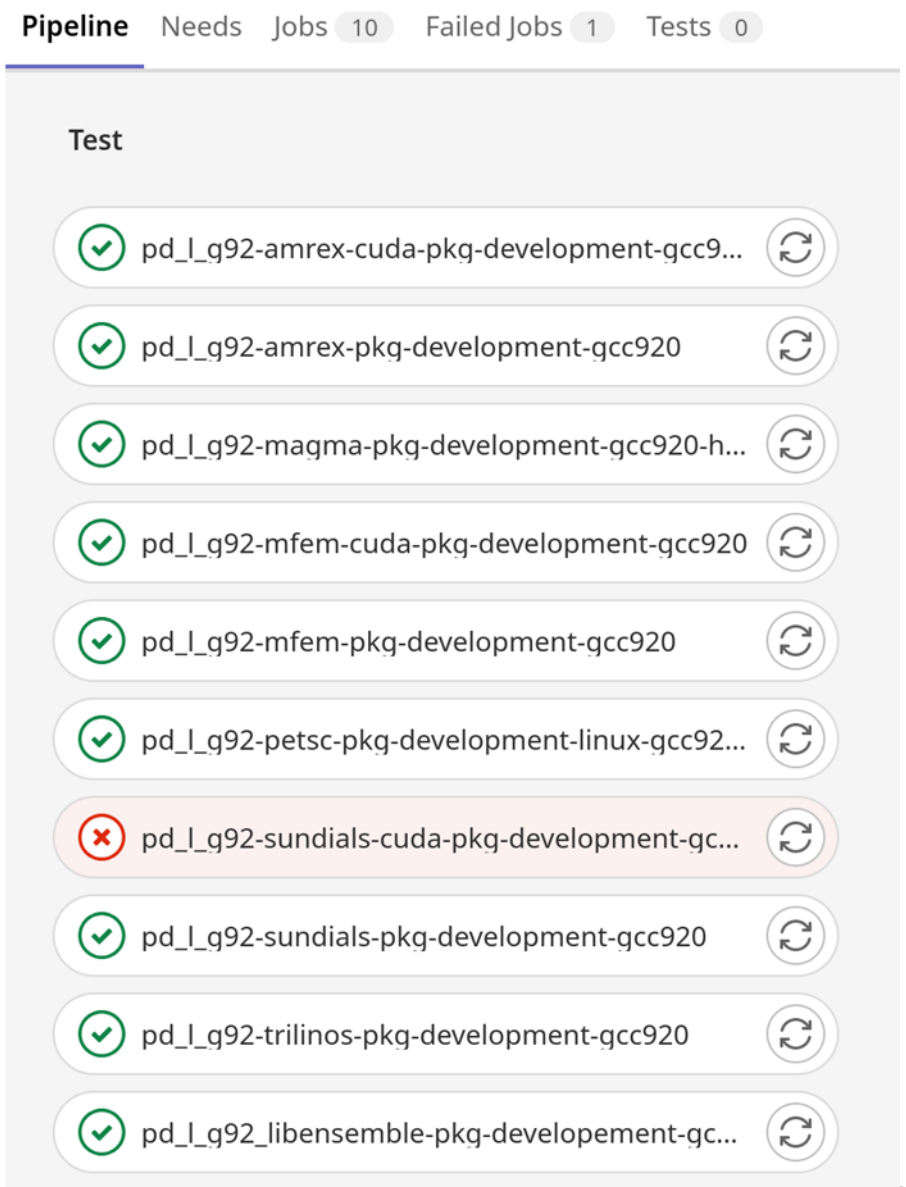




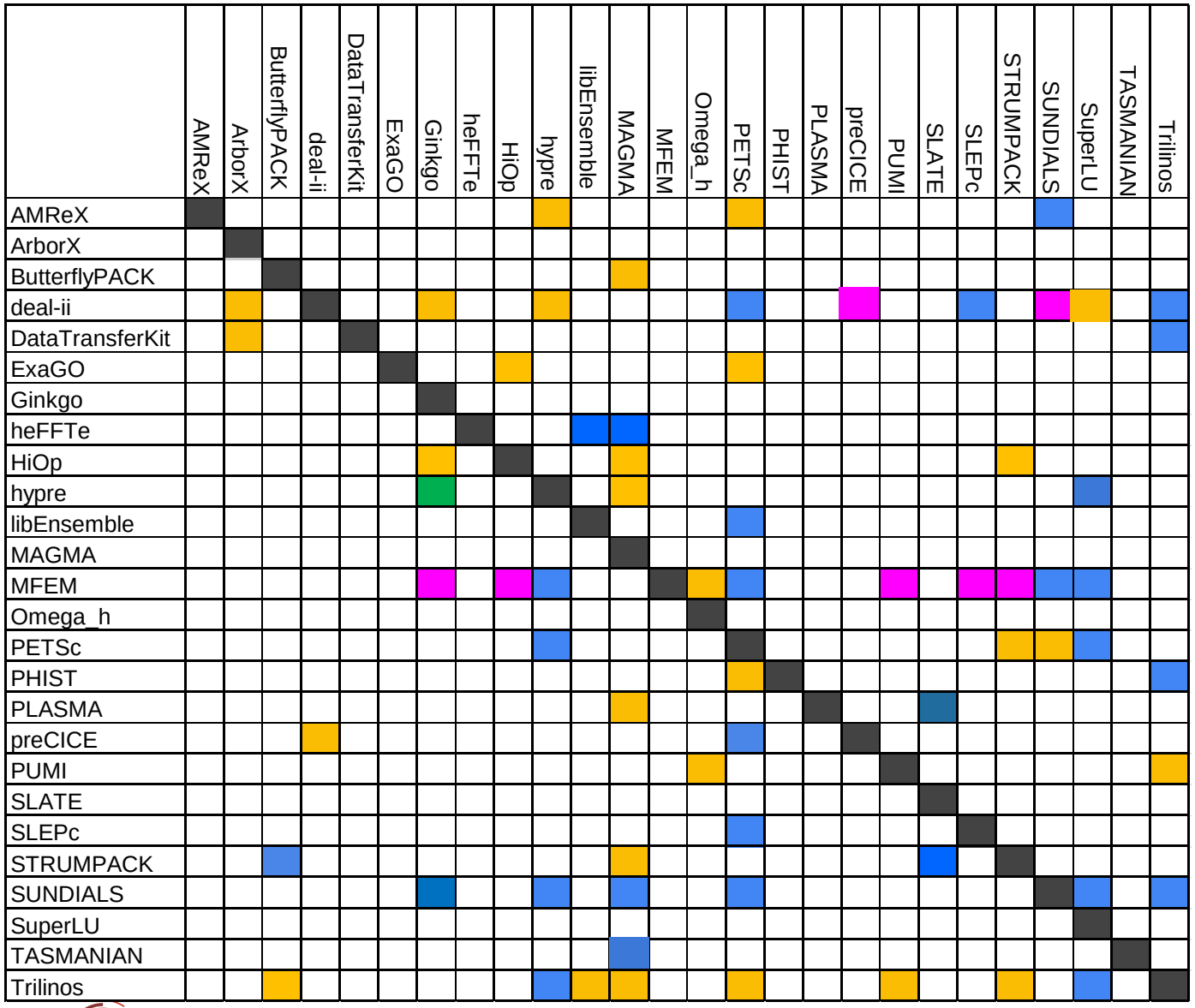
Testing of xSDK subsets with development versions

- Build xSDK subset of development versions of packages - on tSpack development and xSDK (future) branch, on the Linux server at UTK with GNU compilers. It contains test jobs:
  - PETSc-SLEPc-PFLOTRAN-hypre-SuperLU_dist
  - heFFTe-MAGMA-TASMANIAN

  We have added 8 new subset tests with development libraries to this pipeline
  - Libensemble-PETSc-TASMANIAN
  - MFEM-SuperLU-STRUMPACK-PETSc-Slepc-PUMI-SUNDIALS-hypre
  - MFEM-SuperLU-STRUMPACK-PETSc-Slepc-PUMI-SUNDIALS-hypre (CUDA)
  - SUNDIALS-hypre-SuperLU-PETSc
  - SUNDIALS-hypre-SuperLU-PETSc-MAGMA (CUDA)
  - AMReX-SUNDIALS
  - AMRex-SUNDIALS (CUDA)
  - Trilinos-hypre-SuperLU



Updated Interoperability Matrix



	Interoperability exists
	Interoperability exists and is enabled in xSDK Spack package
	Interoperability planned
	Interoperability exists and is enabled in Gitlab subsets job or xsdk-examples

We need to increase subsets to switch more yellow boxes to magenta ones!

Future Plans

- Update xSDK Community Policies
- xSDK 1.0.0 to be released in November 2023
- Increase interoperabilities and example codes
- Continue improving xSDK CI

General xSDK info: https://xsdk.info

- **download**
- **installation,**
- **policies**
  https://github.com/xsdk-project/xsdk-community-policies

We encourage feedback and contributions!

xSDK-ECP Project Members

Ahmad Abdelfattah
Boyana Norris
Carol Woodward
Christian Glusa
Cody Balos
Damien Lebrun-Grandie
Daniel Arndt
Daniel Osei-Kuffuor
David Gardner
Erik Boman
Erin Carson
Gerald Ragghianti
Hartwig Anzt
Ian Mcinerney
Ichi Yamazaki
Ieva Dauzickaite
Jack Dongarra
Jamie Finney
Jennifer Loe
Jim Demmel
Jim Willenbring
Keita Teranishi
Kim Liegeois
Lois Curfman McInnes
Luc Berger-Vergiat
Mark Gates
Mike Heroux
Miroslav K. Stoyanov
Natalie Beams
Nick Higham
Osni Marques
Piotr Luszczek
Pratik Nayak
Richard Mills
Robert Falgout
Samuel Knight
Sarah Osborn
Satish Balay
Sherry Li
Siva Rajamanickam
Stan Tomov
Stephen Hudson
Stuart Slattery
Terry Cojean
Thomas Grützmacher
Tobias Ribizel
Tomas Gergelits
Tzanio Kolev
Ulrike Meier Yang
Veselin Dobrev
Victor Magri
Viktor Reshniak
Wenjun Ge
Yang Liu

And many more …

CASC

Center for Applied Scientific Computing

Lawrence Livermore National Laboratory

Thank you!

This work was supported by the U.S. Department of Energy Office of Science, Office of Advanced Scientific Computing Research (ASCR), and by the Exascale Computing Project, a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration.