xSDK: an Ecosystem of Interoperable Independently Developed Math Libraries

Ulrike Meier Yang

September 20, 2023
**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** →  $\text{Impact(ecosystem)} > \sum \text{Impact (elements)}$
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
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

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

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

![Diagram of software packages and dependencies]

- hypre
- SuperLU
- PETSc
- PFLOTTRAN
- Trilinos
- Amanzi/ATS
- Alquimia
- CLM
- Parflow
- Crunchflow
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

Timeline:
- xSDK release 1
- xSDK release 2
- ... xSDK release n
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
- Data & Visualization
- Applications
- Programming Models
- Runtimes
- Math Libraries
- Embedded Data & Visualization
- Development Tools

**Hardware and Integration**
Relationships: facilities with AD/ST, with vendors

Emphasis for this presentation
### DOE HPC Roadmap to Exascale Systems

<table>
<thead>
<tr>
<th>FY 2012</th>
<th>FY 2016</th>
<th>FY 2018</th>
<th>FY 2021</th>
<th>FY 2022</th>
<th>FY 2023</th>
<th>FY 2024</th>
</tr>
</thead>
<tbody>
<tr>
<td>Titan</td>
<td>Mira</td>
<td>Theta</td>
<td>Summit</td>
<td>ORNL</td>
<td>ORNL</td>
<td>ORNL</td>
</tr>
<tr>
<td>ORNL</td>
<td>IBM/AMD/NVIDIA</td>
<td>IBM/AMD/NVIDIA</td>
<td>ORNL</td>
<td>HPE/AMD</td>
<td>HPE/AMD</td>
<td>HPE/AMD</td>
</tr>
<tr>
<td>Cray/AMD/NVIDIA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ANL</td>
<td>IBM BG/Q</td>
<td>ANL</td>
<td>LBNL</td>
<td>LBNL</td>
<td>LBNL</td>
<td>LBNL</td>
</tr>
<tr>
<td>IBM BG/Q</td>
<td>Cray/Intel KNL</td>
<td>Cray/Intel Xeon/KNL</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mira</td>
<td>Theta</td>
<td>LBNL</td>
<td>LBNL</td>
<td>LBNL</td>
<td>LBNL</td>
<td>LBNL</td>
</tr>
<tr>
<td>Cray/AMD/NVIDIA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sequoia</td>
<td>Trinity</td>
<td>Sierra</td>
<td>LLNL</td>
<td>LLNL</td>
<td>LLNL</td>
<td>LLNL</td>
</tr>
<tr>
<td>LLNL</td>
<td>IBM BG/Q</td>
<td>IBM/AMD/NVIDIA</td>
<td>IBM/AMD/NVIDIA</td>
<td>HPE/AMD</td>
<td>HPE/AMD</td>
<td>HPE/AMD</td>
</tr>
<tr>
<td>Cray/Intel Xeon/KNL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LAN/SNL</td>
<td>LLNL</td>
<td>LAN/SNL</td>
<td>LLNL</td>
<td>LAN/SNL</td>
<td>LAN/SNL</td>
<td>LLNL</td>
</tr>
<tr>
<td>Cray/Intel Xeon/KNL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LLNL</td>
<td>Cray/Intel Xeon/KNL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LAN/SNL</td>
<td>LLNL</td>
<td>LAN/SNL</td>
<td>LLNL</td>
<td>LLNL</td>
<td>LLNL</td>
<td>LLNL</td>
</tr>
<tr>
<td>Cray/Intel Xeon/KNL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LAN/SNL</td>
<td>Cray/Intel Xeon/KNL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LLNL</td>
<td>HPE/AMD</td>
<td>HPE/AMD</td>
<td>HPE/AMD</td>
<td>HPE/AMD</td>
<td>HPE/AMD</td>
<td>HPE/AMD</td>
</tr>
<tr>
<td>Cray/Intel Xeon/KNL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LAN/SNL</td>
<td>Cray/Intel Xeon/KNL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LLNL</td>
<td>HPE/AMD</td>
<td>HPE/AMD</td>
<td>HPE/AMD</td>
<td>HPE/AMD</td>
<td>HPE/AMD</td>
<td>HPE/AMD</td>
</tr>
<tr>
<td>Cray/Intel Xeon/KNL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LAN/SNL</td>
<td>Cray/Intel Xeon/KNL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LLNL</td>
<td>HPE/AMD</td>
<td>HPE/AMD</td>
<td>HPE/AMD</td>
<td>HPE/AMD</td>
<td>HPE/AMD</td>
<td>HPE/AMD</td>
</tr>
<tr>
<td>Cray/Intel Xeon/KNL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LAN/SNL</td>
<td>Cray/Intel Xeon/KNL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LLNL</td>
<td>HPE/AMD</td>
<td>HPE/AMD</td>
<td>HPE/AMD</td>
<td>HPE/AMD</td>
<td>HPE/AMD</td>
<td>HPE/AMD</td>
</tr>
<tr>
<td>Cray/Intel Xeon/KNL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LAN/SNL</td>
<td>Cray/Intel Xeon/KNL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LLNL</td>
<td>HPE/AMD</td>
<td>HPE/AMD</td>
<td>HPE/AMD</td>
<td>HPE/AMD</td>
<td>HPE/AMD</td>
<td>HPE/AMD</td>
</tr>
<tr>
<td>Cray/Intel Xeon/KNL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LAN/SNL</td>
<td>Cray/Intel Xeon/KNL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LLNL</td>
<td>HPE/AMD</td>
<td>HPE/AMD</td>
<td>HPE/AMD</td>
<td>HPE/AMD</td>
<td>HPE/AMD</td>
<td>HPE/AMD</td>
</tr>
<tr>
<td>Cray/Intel Xeon/KNL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LAN/SNL</td>
<td>Cray/Intel Xeon/KNL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LLNL</td>
<td>HPE/AMD</td>
<td>HPE/AMD</td>
<td>HPE/AMD</td>
<td>HPE/AMD</td>
<td>HPE/AMD</td>
<td>HPE/AMD</td>
</tr>
<tr>
<td>Cray/Intel Xeon/KNL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
ECP applications need sustainable coordination among math libraries

**ECP AD Teams**


**Examples:**
- **Subsurface**: Chombo, PETSC, hypre, etc. …
- **ExaAM**: DTK, SUNDIALS, Tasmanian, hypre, Trilinos, FFT, etc.
- **ExaWind**: hypre, KokkosKernels, SuperLU, Trilinos, AMReX, etc.
- **WDMAApp**: 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

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

https://xsdk.info

Multiphysics Application C

Application A

Application B

xSDK functionality, April 2016
Tested on key machines at ALCF, NERSC, OLCF, also Linux, Mac OS X

xSDK

HDF5

BLAS

More external software

xSDK Installer

PETSc

hypre

SuperLU

Trilinos

Alquimia

Domain components
- Reacting flow, etc.
- Reusable.

Libraries
- Solvers, etc.
- Interoperable.

Frameworks & tools
- Doc generators.
- Test, build framework.

SW engineering
- Productivity tools.
- Models, processes.

Extreme-Scale Scientific Software Development Kit (xSDK)

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

HDF5

BLAS

More external software

xSDK Installer

PETSc

hypre

SuperLU

Trilinos

Alquimia

Domain components
- Reacting flow, etc.
- Reusable.

Libraries
- Solvers, etc.
- Interoperable.

Frameworks & tools
- Doc generators.
- Test, build framework.

SW engineering
- Productivity tools.
- Models, processes.

Extreme-Scale Scientific Software Development Kit (xSDK)

https://xsdk.info
Extreme-Scale Scientific Software Development Kit (xSDK)

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

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

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

- Spack build manager
- Math libraries
- Community policies
The xSDK packages depend on a number 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

https://spack.io
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)
- **hype**: 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

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, naming, 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 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://github.com/xsdk-project/xsdk-community-policies)

---

**Mandatory xSDK policies: must be satisfied**

- **M1.** Support portable installation through Spack (includes xSDK Spack variant guidelines)
- **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?

---

[https://xsdk.info/policies](https://xsdk.info/policies)

Version 1.0.0, February 2023
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

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](https://github.com/xsdk-project/xsdk-policy-compatibility)
What is required for an effective ecosystem?

▪ High software quality
▪ Portability
▪ Interoperability
▪ Sustainability
Software Quality

Mandatory xSDK policies: must be satisfied

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

Mandatory xSDK policies: must be satisfied

M1. Support portable installation through Spack (includes xSDK Spack variant guidelines)
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 Strategies of xSDK Libraries

- Use of portable programming models that provide abstractions
- Use of abstraction to limit code that interacts with devices
- Use of fast kernel libraries designed for individual architectures
- 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!)

"The way you get programmer productivity is by eliminating lines of code you have to write."

Interoperability

**Mandatory xSDK policies: must be satisfied**

- **M1.** Support portable installation through Spack (includes xSDK Spack variant guidelines)
- **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 CHANGETLOG 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.
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:**

\[ A \leftrightarrow B: \]

- \( A \) 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!

<table>
<thead>
<tr>
<th>AMReX</th>
<th>ArborX</th>
<th>deal-ii</th>
<th>DataTransferKit</th>
<th>ExaGO</th>
<th>Ginkgo</th>
<th>heFFTe</th>
<th>hypre</th>
<th>libEnsemble</th>
<th>MAGMA</th>
<th>Omega_h</th>
<th>PETSc</th>
<th>PHIST</th>
<th>PLASMA</th>
<th>preCICE</th>
<th>PUMI</th>
<th>SLATE</th>
<th>SLEPE</th>
<th>STRUMPACK</th>
<th>SUNDIALS</th>
<th>SuperLU</th>
<th>TASMANIAN</th>
<th>Trilinos</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

**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

Mandatory xSDK policies: must be satisfied

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

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

tested on key platforms at ALCF, NERSC, and OLCF, also Linux and Mac OS X.
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/16y2bL1RZg8wke0vY8c97ssvhRYNez34Q4QGg4LoIEUk/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 tSpacc development and xSDK (future) branch, on the Linux server at UTK with GNU compilers. It contains test jobs:
  - PETSc-SLEPC-PFLOATAN-hype-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-hype
  - MFEM-SuperLU-STRUMPACK-PETSc-Slepc-PUMI-SUNDIALS-hype (CUDA)
  - SUNDIALS-hype-SuperLU-PETSc
  - SUNDIALS-hype-SuperLU-PETSc-MAGMA (CUDA)
  - AMReX-SUNDIALS
  - AMReX-SUNDIALS (CUDA)
  - Trilinos-hype-SuperLU
Updated Interoperability Matrix

<table>
<thead>
<tr>
<th>AMReX</th>
<th>ArborX</th>
<th>ButterflyPACK</th>
<th>deal-ii</th>
<th>DataTransferKit</th>
<th>ExaGO</th>
<th>Ginkgo</th>
<th>heFFTe</th>
<th>HiOp</th>
<th>hypre</th>
<th>libEnsemble</th>
<th>MAGMA</th>
<th>MFEM</th>
<th>Omega_h</th>
<th>PETSc</th>
<th>PHIST</th>
<th>PLASMA</th>
<th>preCICE</th>
<th>PUMI</th>
<th>SLATE</th>
<th>SLEPc</th>
<th>STRUMPACK</th>
<th>SUNDIALS</th>
<th>SuperLU</th>
<th>TASMANIAN</th>
<th>Trilinos</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interoperability exists</td>
<td>Interoperability exists and is enabled in xSDK Spack package</td>
<td>Interoperability planned</td>
<td>Interoperability exists and is enabled in Gitlab subsets job or xsdk-examples</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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:
- download
- installation,
- policies

https://xsdk.info

We encourage feedback and contributions!

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