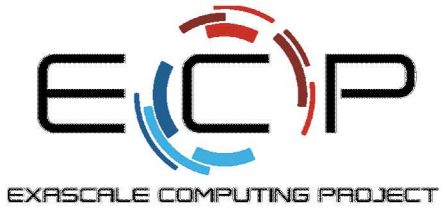
SAND2019-4994C



An introduction to the xSDK, a community of diverse numerical HPC software packages

Keita Teranishi and xSDK Developers

Sandia National Laboratories, Livermore, CA

June 7, 2019

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Who are we?

- Developers of **high-quality, robust, portable high-performance math libraries**

hypre high performance preconditioners
PETSc/TAO
PHIST
MAGMA
mfem
AMReX
SuperLU
TRILINOS
sundials
PLASMA
PUMI
STRUMPACK
Omega_h
DTK
TASMANIAN
SLEPc
deal.II

Why are we leading this tutorial?

- Explain how the xSDK's approach toward a scientific software ecosystem **improves quality, sustainability, and combined use of independent packages**, as needed for extreme-scale computational science and engineering
- Encourage discussions with package developers: **How you can leverage math libraries for extreme-scale computational science**

xSDK
ECP EXASCALE COMPUTING PROJECT

xSDK collaborators

xSDK Release 0.4.0, Dec 2018

- **xSDK release lead:** Jim Willenbring, SNL
- **xSDK planning**
  - Lois Curfman McInnes (ANL)
  - Ulrike Meier Yang (LLNL)
- **Leads for xSDK testing**
  - Satish Balay, ANL: ALCF testing
  - Piotr Luszczek, UTK: OLCF testing
  - Aaron Fischer, LLNL: NERSC testing
  - Cody Balos, LLNL: general testing
  - Keita Teranishi, SNL: general testing
- **Spack liaison:** Todd Gamblin, LLNL

- **Package compatibility with xSDK community policies and software testing:**
  - **AMReX**: Ann Almgren, Michele Rosso (LBNL)
  - **DTK**: Stuart Slattery, Bruno Turcksin (ORNL)
  - **deal.II**: Wolfgang Bangerth (Colorado State University)
  - **hypre**: Ulrike Meier Yang, Sarah Osborn, Rob Falgout (LLNL)
  - **MAGMA** and **PLASMA**: Piotr Luszczek (UTK)
  - **MFEM**: Aaron Fischer, Tzanio Kolev (LLNL)
  - **Omega_h**: Dan Ibanez (SNL)
  - **PETSc/TAO**: Satish Balay, Alp Denner, Barry Smith (ANL)
  - **PUMI**: Cameron Smith (RPI)
  - **SUNDIALS**: Cody Balos, David Gardner, Carol Woodward (LLNL)
  - **SuperLU** and **STRUMPACK**: Sherry Li and Pieter Ghysels (LBNL)
  - **TASMANIAN**: Miroslav Stoyanov, Damien Lebrun Grandie (ORNL)
  - **Trilinos**: Keita Teranishi, Jim Willenbring, Sam Knight (SNL)
  - **PHIST**: Jonas Thies (DLR, German Aerospace Center)
  - **SLEPc**: José Roman (Universitat Politècnica de València)
  - **Alquimia**: Sergi Mollins (LBNL)
  - **PFLOTRAN**: Glenn Hammond (SNL)

and many more …

Who are you?

- **Extreme-scale computational science community**
  - Developers of extreme-scale scientific applications
  - Developers of high-performance software packages and tools
  - Project leaders, stakeholders, program managers
  - Others

Learning objectives:

- Understand
  - Why a **software ecosystem perspective** is essential for extreme-scale computational science
    - Strategic objectives: **building community and building sustainability**
  - How to download and **use the xSDK and member packages**
  - How ECP **math library capabilities** can help your science

Tutorial outline:
xSDK Approach and Experiences

- **Introduction**
  - Math libraries and scientific software ecosystems
  - Building community and sustainability
  - xSDK history and goals to fulfill ECP needs
- **About the xSDK**
  - xSDK community policies
  - Short introduction to Spack
  - xSDK release process
  - Installing the xSDK
  - Using the xSDK in ECP applications
- **Lessons learned**
- **xSDK package overviews**

Software libraries facilitate progress in computational science and engineering

- **Software library:** a high-quality, encapsulated, documented, tested, and multiuse software collection that provides functionality commonly needed by application developers
  - Organized for the purpose of being reused by independent (sub)programs
  - User needs to know only
    - Library interface (not internal details)
    - When and how to use library functionality appropriately

- **Key advantages** of software libraries
  - Contain complexity
  - Leverage library developer expertise
  - Reduce application coding effort
  - Encourage sharing of code, ease distribution of code

- **References:**
  - https://en.wikipedia.org/wiki/Library_(computing)
  - What are Interoperable Software Libraries? Introducing the xSDK

Why is reusable scientific software important for you?

User perspective:

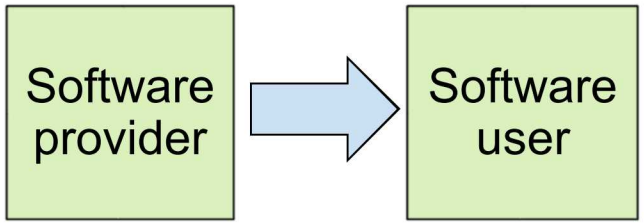
Focus on primary interests

- Reuse algorithms and data structures developed by experts
- Customize and extend to exploit application-specific knowledge
- Cope with complexity and changes over time

Provider perspective:

Share your capabilities

- Broader impact of your work
- Motivate new directions of research



- **More efficient, robust, reliable, sustainable software**
- **Improve developer productivity**
- **Better science**

Software libraries are not enough

- Well-designed libraries provide critical functionality … But alone are not sufficient to address all aspects of next-generation scientific simulation and analysis.

- Applications need to use software packages **in combination** on ever evolving architectures

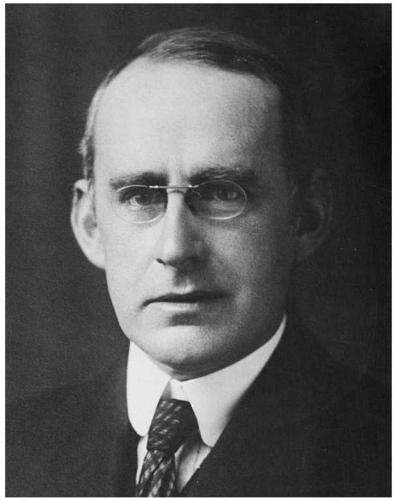
> **"The way you get programmer productivity is by eliminating lines of code you have to write."**
>
> – Steve Jobs, Apple World Wide Developers Conference, Closing Keynote, 1997

Need software ecosystem perspective

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

Ecosystems are challenging!

"We often think that when we have completed our study of one we know all about two, because 'two' is 'one and one.' We forget that we still have to make a study of 'and.' "



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

Difficulties in combined use of independently developed software packages

Challenges:

- Obtaining, configuring, and installing multiple independent software packages is tedious and error prone.
  - Need consistency of compiler (+version, options), 3rd-party packages, etc.
- Namespace conflicts
- Incompatible versioning
- And even more challenges for deeper levels of interoperability

Ref: What are Interoperable Software Libraries? Introducing the xSDK

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

Ecosystem imperative for math software

Dialogue with the broader CSE / HPC community

- **Classic application development approach:**
  - Application developers write most code; source code considered private
  - Occasionally use libraries, but typically only those “baked into” the OS
  - Portability challenges, unmanaged disruptions: Low risk. But…
- **Ecosystem-based application development approach:**
  - App developers use composition, write glue code & unique functionality
  - Source code includes substantial 3rd party packages
  - Risks (if 3rd party code is poor):
    - Dependent on portability of 3rd party code
    - Upgrades of 3rd party package can be disruptive (interface changes, regressions)
  - Opportunities (if 3rd party code is good):
    - 3rd party improvements are yours (for free!)
    - Portability to new architectures is seamless

BLUE WATERS
SUSTAINED PETASCALE COMPUTING

Webinar track launched Nov 2017

Scientific Software Ecosystems

https://bluewaters.ncsa.illinois.edu/webinars

Objectives:

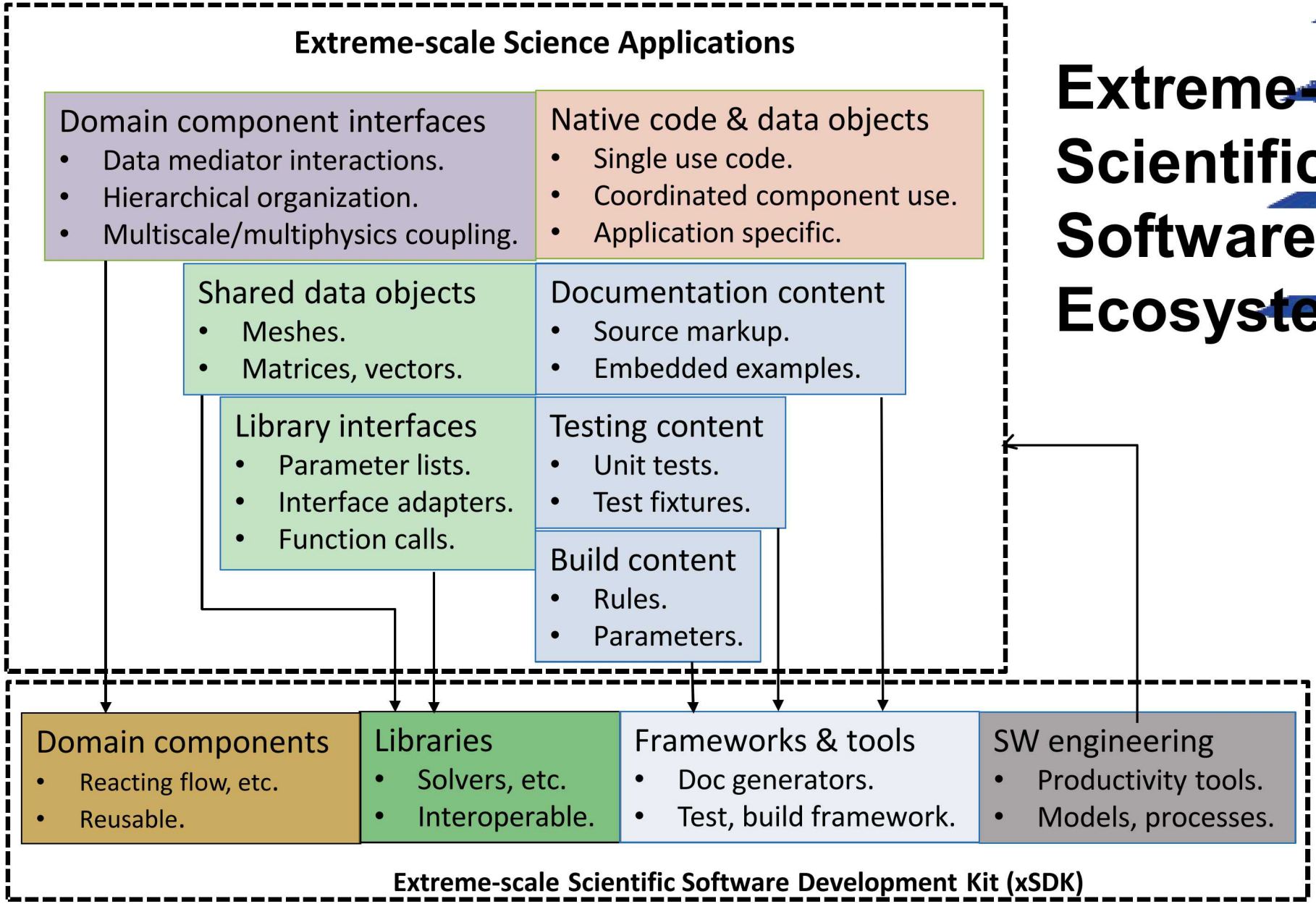
- Promote quality reusable research software for computational and data-enabled discovery
- Promote community efforts to improve research software quality, culture, credit, collaboration, …

While considering issues in scientific software ecosystems

xSDK

ECP EXASCALE COMPUTING PROJECT

Extreme-scale Scientific Software Ecosystem



IDEAS productivity
www.ideas-productivity.org

Interoperable Design of Extreme-scale Application Software (IDEAS)

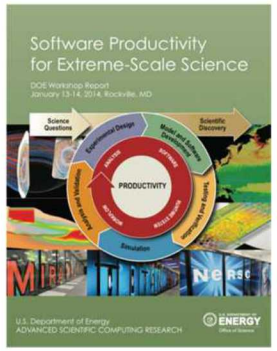
Motivation

Enable ***increased scientific productivity,*** realizing the potential of extreme- scale computing, through ***a new interdisciplinary and agile approach to the scientific software ecosystem***.

Objectives

Address confluence of trends in hardware and increasing demands for predictive multiscale, multiphysics simulations.

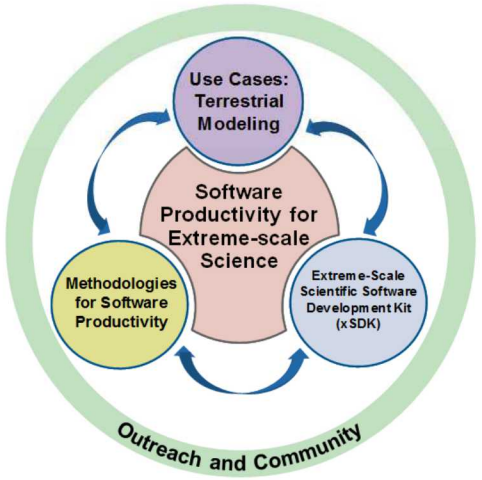
Respond to trend of continuous refactoring with efficient agile software engineering methodologies and improved software design.



Impact on Applications & Programs

Terrestrial ecosystem ***use cases tie IDEAS to modeling and simulation goals*** in two Science Focus Area (SFA) programs and both Next Generation Ecosystem Experiment (NGEE) programs in DOE Biologic and Environmental Research (BER).

Approach



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

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)

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

IDEAS history

ASCR/BER partnership began in Sept 2014

Program Managers:

- Paul Bayer, David Lesmes (BER)
- Thomas Ndousse-Fetter (ASCR)

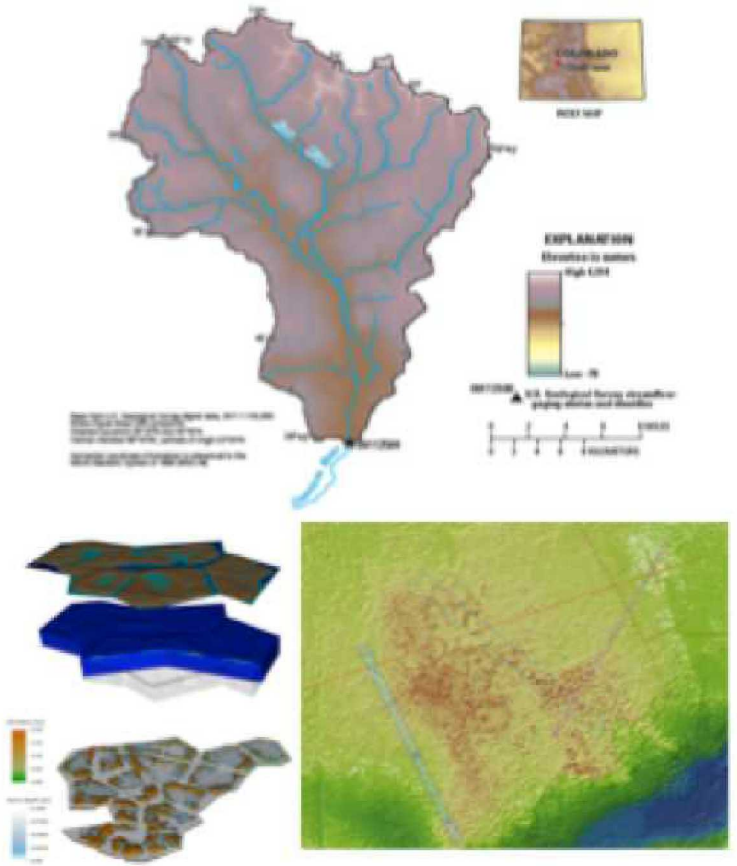
First-of-a-kind project: qualitatively new approach based on making productivity and sustainability the explicit and primary principles for guiding our decisions and efforts.

xSDK

ECP EXASCALE COMPUTING PROJECT

Use cases: Multiscale, multiphysics modeling of watershed dynamics

- **Use Case 1:** Hydrological and biogeochemical cycling in the Colorado Rive System
- **Use Case 2:** Thermal hydrology and carbon cycling in tundra at the Barrow Observatory
- **Use Case 3:** Hydrologic, land surface, and atmospheric process coupling over CONUS
- **Leverage & complement SBR, TES programs:**
  - LBNL and PNNL SFAs; NGEE Arctic and Tropics
- **Approach:**
  - Leverage existing open source apps
  - Improve software development practices
  - Targeted refactoring of interfaces, data structures, and key components to facilitate interoperability
  - Modernize management of multiphysics integration and multiscale coupling



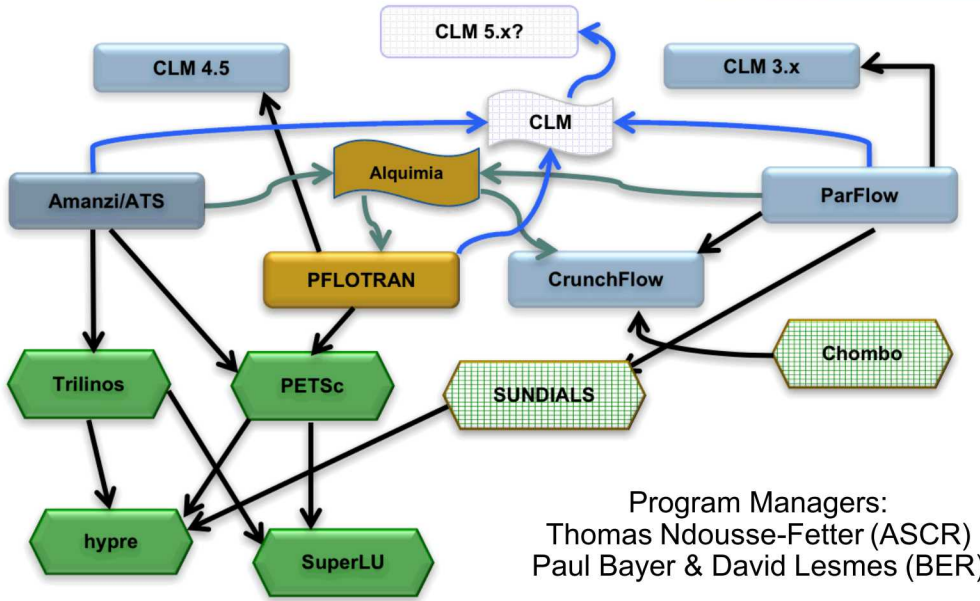
Motivation and history of xSDK

Next-generation scientific simulations require combined use of independent packages

- 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
- **Prior to xSDK effort, could not build required libraries into a single executable due to many incompatibilities**

xSDK history: Work began in ASCR/BER partnership, IDEAS project (Sept 2014)

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



Program Managers:
Thomas Ndousse-Fetter (ASCR)
Paul Bayer & David Lesmes (BER)

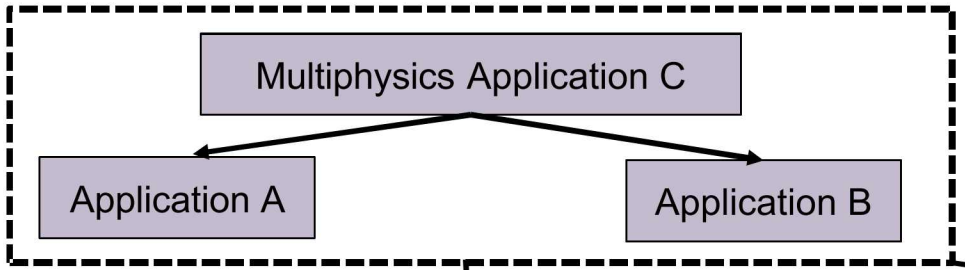
xSDK

ECP EXASCALE COMPUTING PROJECT

xSDK History: Version 0.1.0: April 2016

https://xsdk.info

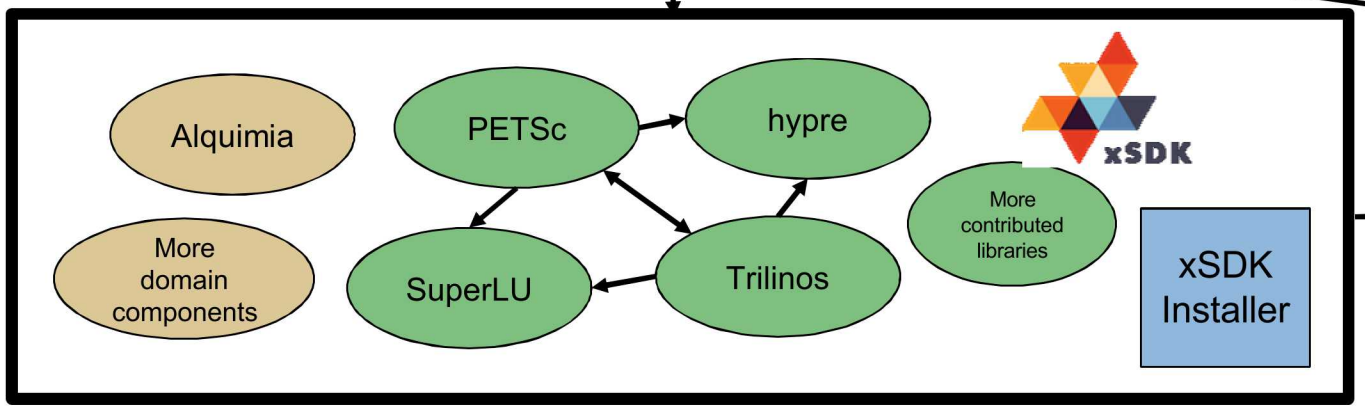
Notation: A → B:
A can use **B** to provide functionality on behalf of **A**

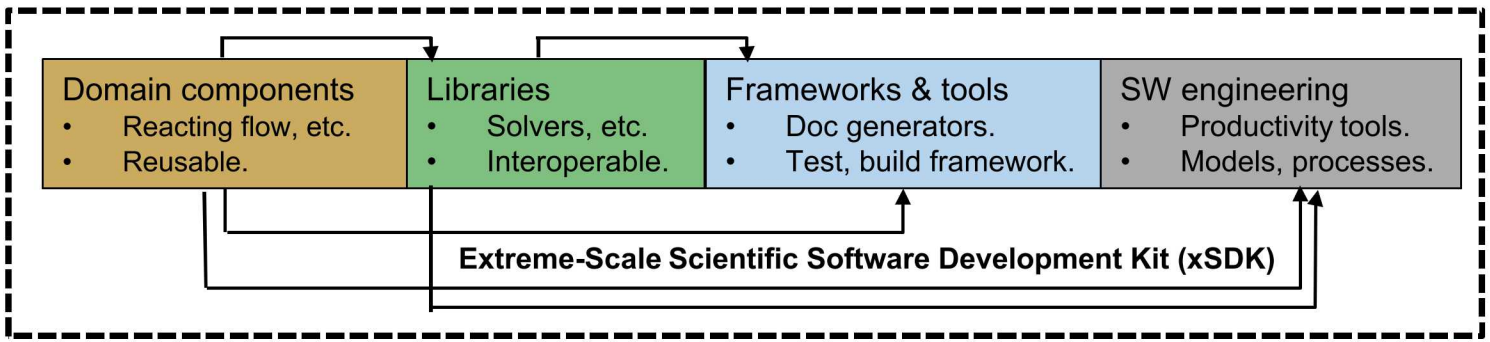


xSDK functionality, April 2016

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

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





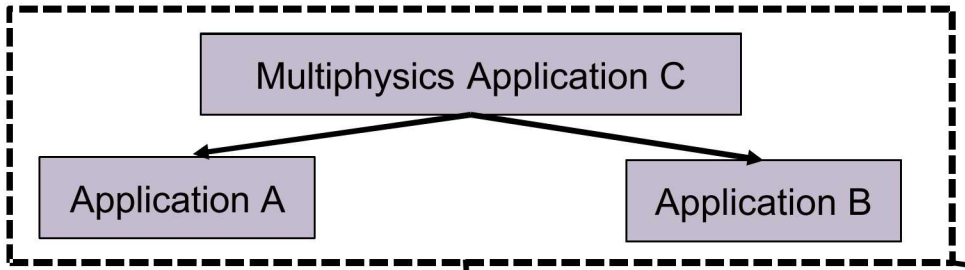
xSDK

ECP EXASCALE COMPUTING PROJECT

xSDK History: Version 0.2.0: February 2017

https://xsdk.info

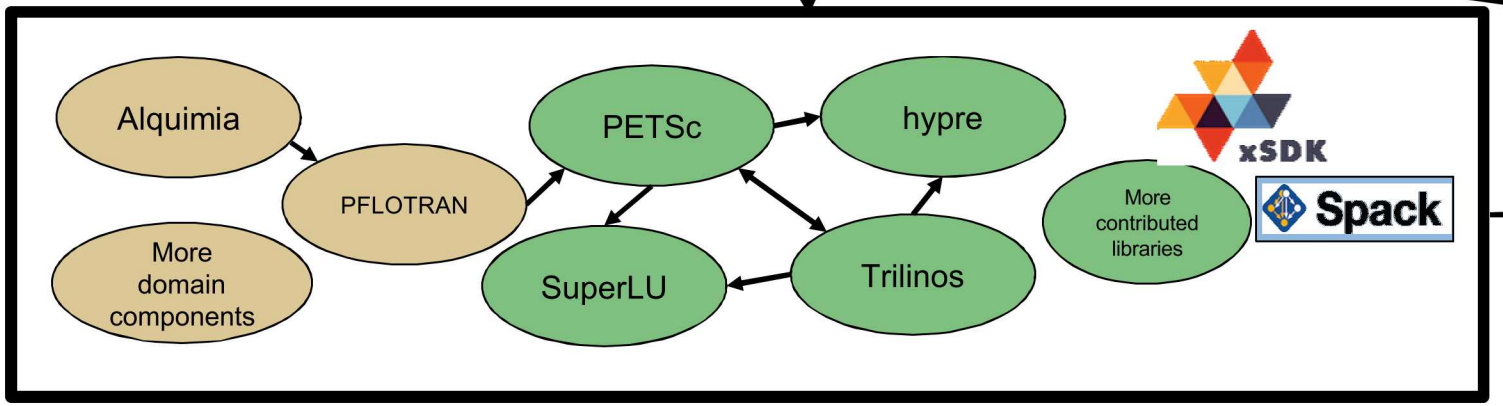
Notation: A → B:
A can use **B** to provide functionality on behalf of **A**

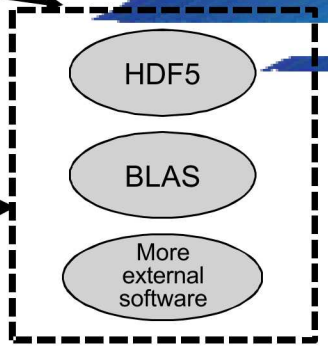


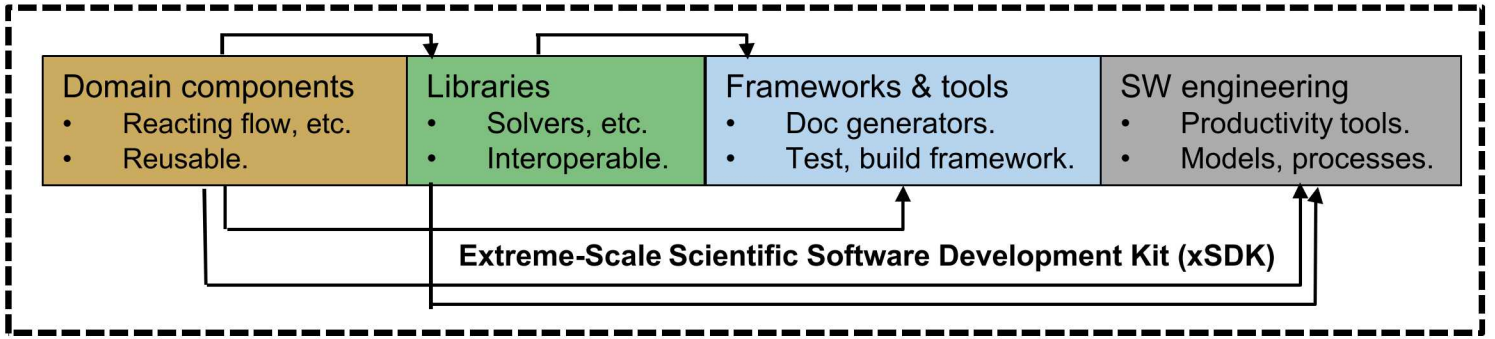
xSDK functionality, Feb 2017

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

February 2017
- 4 math libraries
- 2 domain components
- Spack xSDK installer
- 14 mandatory xSDK community policies







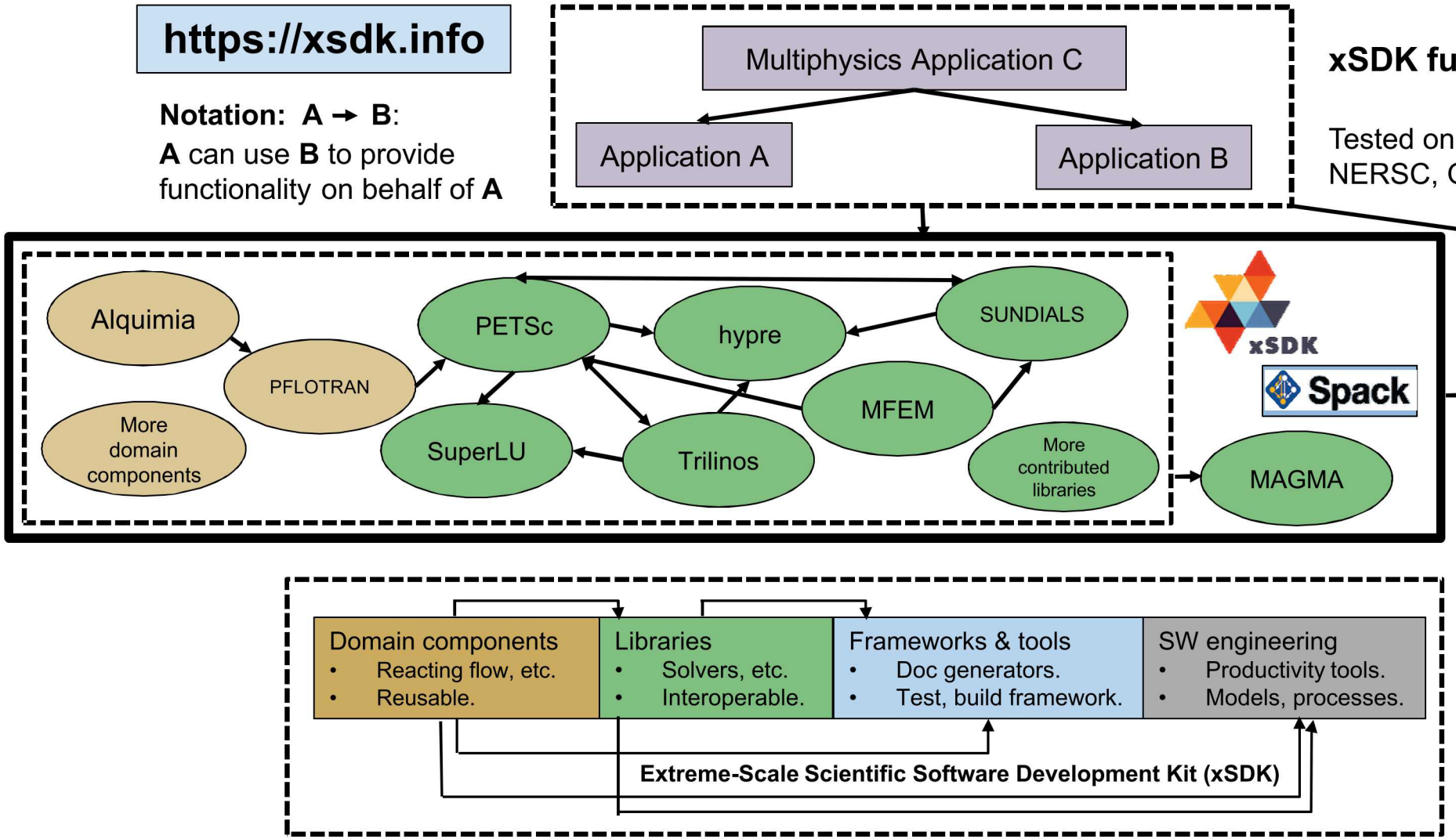
xSDK

EXASCALE COMPUTING PROJECT

xSDK History: Version 0.3.0: December 2017

https://xsdk.info

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



xSDK functionality, Dec 2017

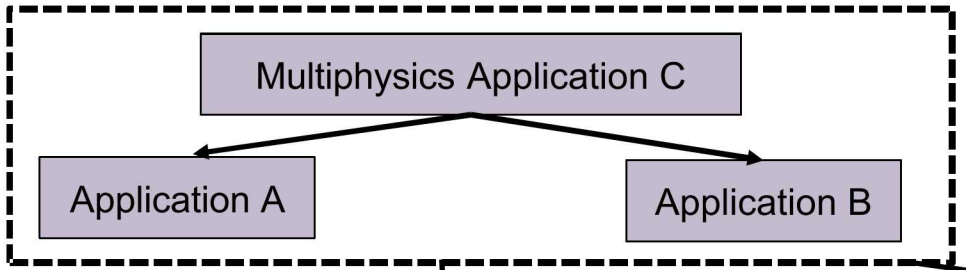
Tested on key machines at ALCF, NERSC, OLCF, also Linux, Mac OS X

December 2017
- 7 math libraries
- 2 domain components
- Spack xSDK installer
- 16 mandatory xSDK community policies

xSDK History: Version 0.4.0: December 2018

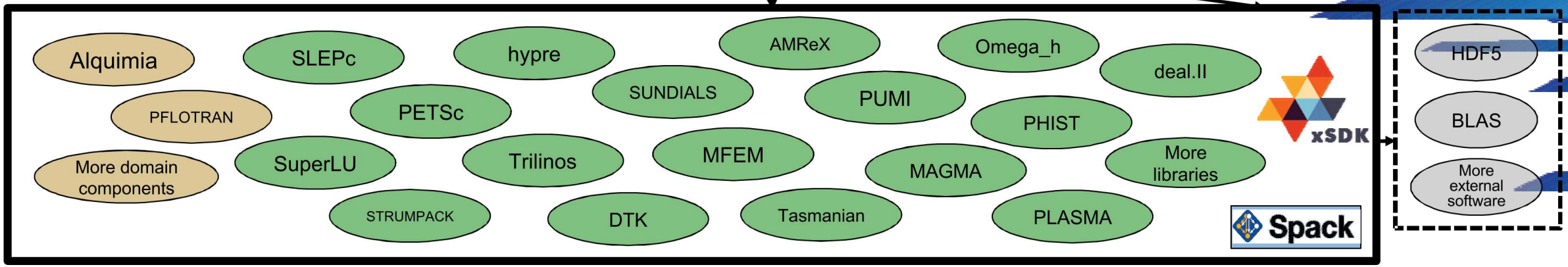
https://xsdk.info

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



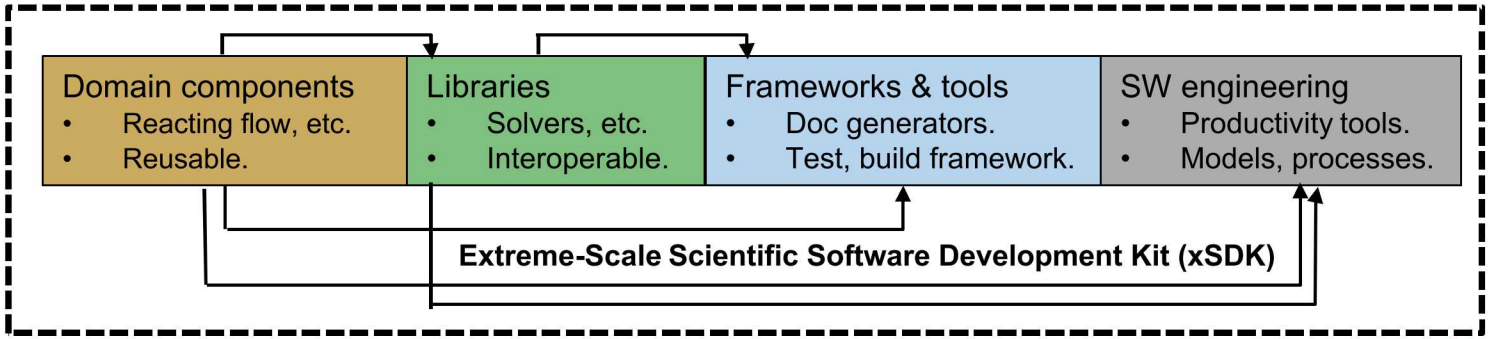
xSDK functionality, Dec 2018

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



December 2018

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



Impact: Improved code quality, usability, access, sustainability

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

xSDK

ECP EXASCALE COMPUTING PROJECT

xSDK for ECP: Project goals, description, scope

Goals: Create a value-added aggregation of ECP mathematics libraries, to increase the combined usability, standardization and interoperability of these libraries, as needed to support large-scale multiphysics and multiscale problems.

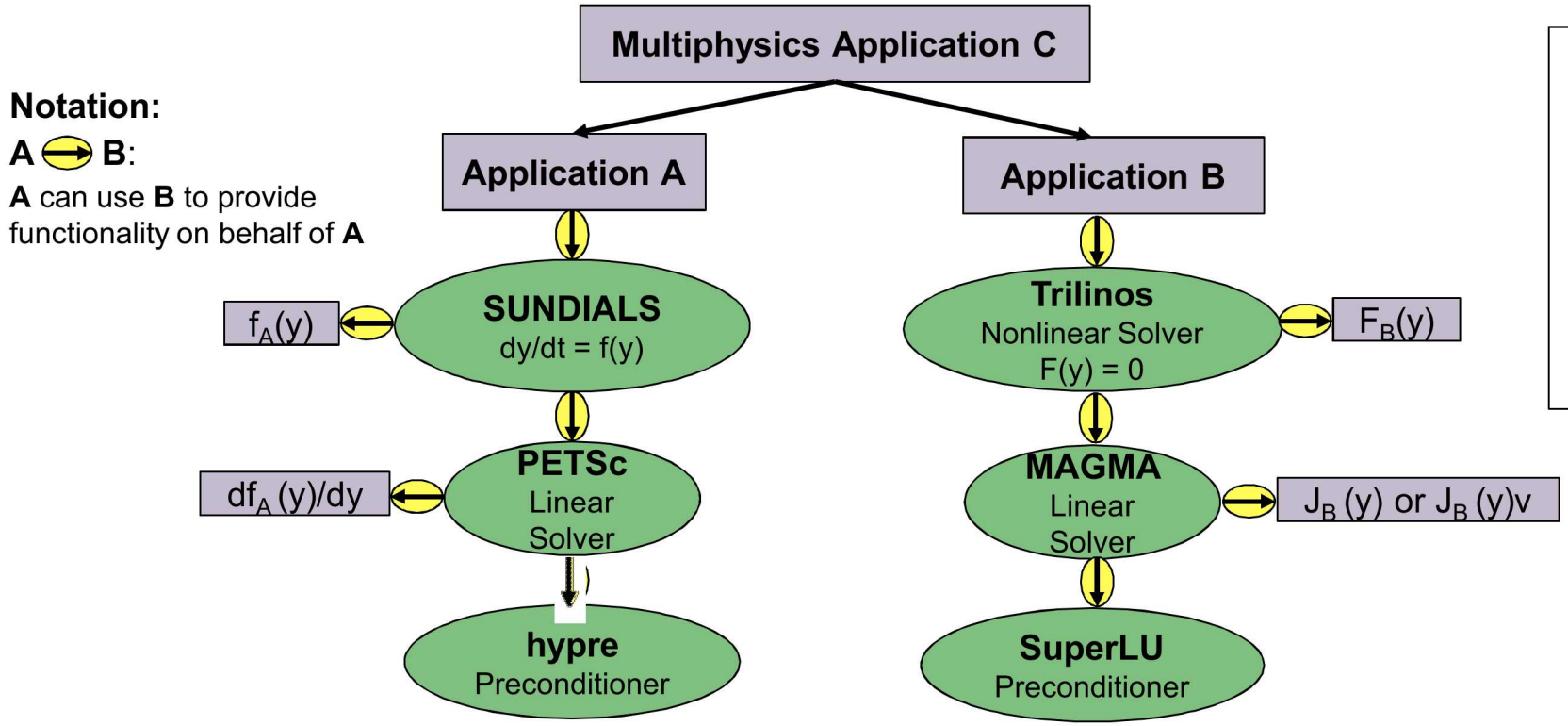
Project Description

- Develop **community policies** and **interoperability** layers among xSDK component packages
- Determine xSDK sustainability strategy for ECP
- Work with ECP applications to motivate and test xSDK

Project Scope

- Enable the seamless combined use of diverse, independently developed software packages as needed by ECP applications
  - **coordinated use of on-node resources**
  - **integrated execution**
  - **coordinated & sustainable documentation, testing, packaging, and deployment**

Using math libraries in combination for next-generation apps



One example of xSDK package interoperability; many more xSDK package interconnections exist

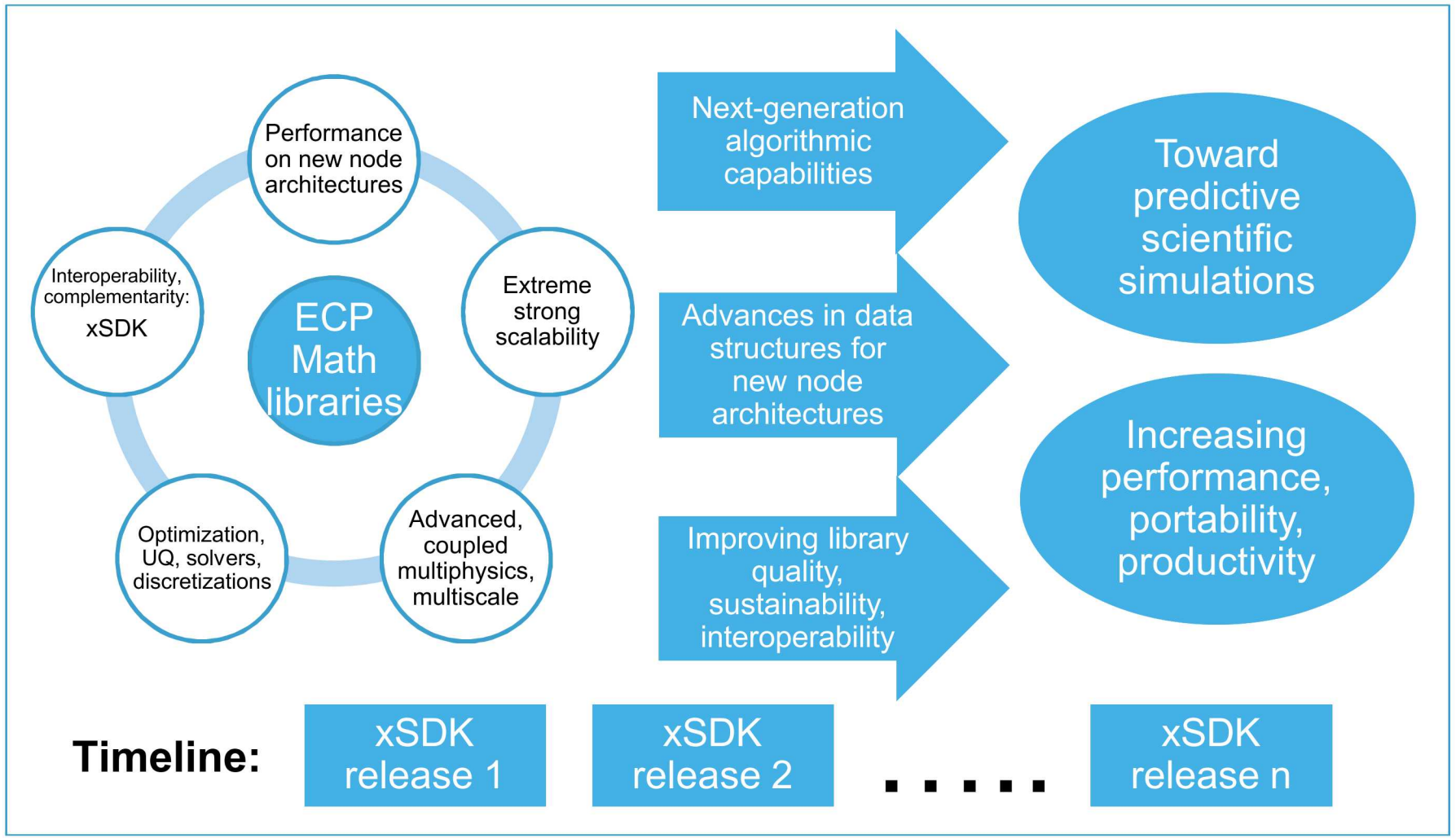
xSDK4ECP: Focus on inter-package functionality, denoted by ⟷

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

xSDK

ECP EXASCALE COMPUTING PROJECT

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



Math Libraries Approach:

As motivated & validated by the needs of ECP applications:

- Establish performance baselines
- Refactor, revise algorithms and data structures for new architectures
- Research into new numerical algorithms for next-generation predictive science

xSDK

ECP EXASCALE COMPUTING PROJECT

Tutorial outline:
xSDK Approach and Experiences

- Introduction
  - Math libraries and scientific software ecosystems
  - Building community and sustainability
  - xSDK history and goals to fulfill ECP needs
- **About the xSDK**
  - xSDK community policies
  - Short introduction to Spack
  - xSDK release process
  - Installing the xSDK
  - Using the xSDK in ECP applications
- **Lessons learned**
- **xSDK package overviews**

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

xSDK compatible package: must satisfy the mandatory xSDK policies (M1, ..., M16)

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

Also specify **recommended policies**, which currently are encouraged but not required (R1, ..., R6)

Topics include: public repository access, 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 0.4.0: Dec 2018

- 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

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

https://xsdk.info/policies

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 **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.

R6. Provide versions of 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.

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		

Introduction to Spack for xSDK Users

- For people who missed yesterday's full day Spack Tutorial (or need a refresh).
- More documentation available in Spack repo
  - $ ls lib/spack/docs/tutorial

Basic Usage

- The basic usage of Spack that installs latest xSDK
  - $ git clone https://github.com/spack/spack.git
  - $ cd spack
  - $ bin/spack install xsdk
- xSDK and all its dependencies will be installed into git repo (may be changed)
  - spack/opt/spack/OS-ARCH/compiler-X.Y.Z/
- Customization
  - You can track xSDK development by switching to a different branch:
    - $ git checkout develop
- See the following slides for more detailed instructions
  - Documentation is also available in lib/spack/docs

Download the latest Spack and the package files.
One of them is xSDK package.

Install the latest xSDK release

Package Information: spack info xsdk

- Description, homepage
- Versions
  - Preferred version is installed by default: 0.4.0
- Variants with default value and description
  - In 0.4.0: cuda, dealii, debug, omega-h
- Dependencies
  - Build
  - Link
  - Run

Customized Builds (Required on Some DOE Systems)

- To request a specific version of xSDK, use @version notation
  - $ spack install xsdk@0.4.0
- To request a specific dependency of xSDK, use ^dep@version
  - $ spack install xsdk^perl@5.16.3
- To select one of the compilers, use %compiler
  - $ spack install xsdk%gcc@7.2.0
- To enable/disable variants, use + (plus) and ~ (tilde); might require shell escape
  - $ spack install xsdk+cuda~dealii
- The full details on how xSDK was installed may be listed with
  - $ spack find –very-long xsdk

Displaying Dependency Tree (Try before Install)

- A full install of xSDK could take a long time
  - Even with multicore builds, some DOE system have few slow cores
- It is possible to show full dependence tree with Spack's "spec" command
  - $ spack spec xsdk
- The syntax is similar almost identical to "install"
  - $ spack spec xsdk@0.4.0
  - $ spack spec xsdk^perl@5.16.3
  - $ spack spec xsdk^python@2.7.15
  - $ spack spec xsdk%gcc@7.2.0
  - $ spack spec xsdk arch=cray-cnl5-interlagos

Processes for xSDK release and delivery

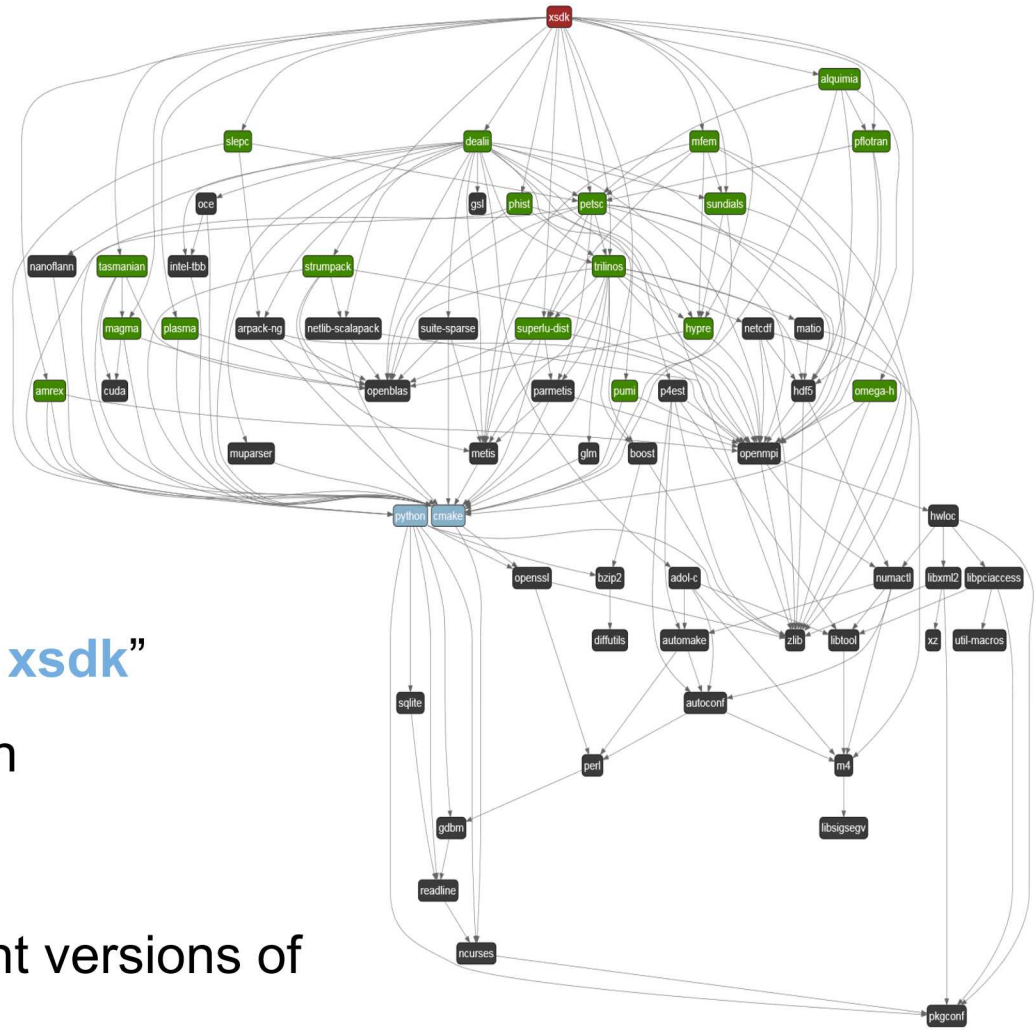
- **2-level release process**
  - **xSDK member packages**
    - Achieve compatibility with xSDK community policies prior to release
      - https://github.com/xsdk-project/xsdk-policy-compatibility
    - Have a Spack package
    - Port to target platforms
    - Provide user support
  - **xSDK**
    - Ensure and test compatibility of mostly independent package releases
- **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

xSDK release process - Spack/git workflow

- **Packages**
  - Follow the standard workflow for a Spack package
  - Submit pull requests with the “xSDK” label
  - Provide package candidate and final releases for xSDK releases
- **xSDK meta-package**
  - Depends on xSDK member packages: “**spack install xsdk**”
  - Maintain xsdk branch in spack for release coordination
- **Coordinate development:**
  - Via ‘development’ version of xsdk – using development versions of some of the individual packages



Downloading

https://xsdk.info/download

1. Obtain xSDK using Spack.

xSDK is distributed primarily with the Spack package manager.

You can obtain Spack from the github repository using this command:

```
git clone https://github.com/spack/spack.git
```

Installing xSDK

https://xsdk.info/installing-the-software

1. After cloning spack git repo, setup spack environment

```
# For bash users
$ export SPACK_ROOT=/path/to/spack
$ . $SPACK_ROOT/share/spack/setup-env.sh

# For tcsh or csh users (note you must set SPACK_ROOT)
$ setenv SPACK_ROOT /path/to/spack
$ source $SPACK_ROOT/share/spack/setup-env.csh
```

2. Setup spack compilers

```
spack compiler find
```

Spack compiler configuration is stored in **$HOME/.spack/$UNAME/compilers.yaml** and can be checked with

```
spack compiler list
```

xSDK

EXASCALE COMPUTING PROJECT

Installing xSDK

https://xsdk.info/installing-the-software

3. Edit/update packages.yaml file to specify any system/build tools needed for xSDK installation.

Although Spack can install required build tools, it can be convenient to use preinstalled tools – if already installed. On macOS, in addition to Xcode, we've used packages from Homebrew. Such preinstalled packages can be specified to spack in **$HOME/.spack/packages.yaml** config file. The following is an example for macOS

```
packages:
  autoconf:
    paths:
      autoconf@2.69: /usr/local
    buildable: False
  automake:
    paths:
      automake@1.15.1: /usr/local
    buildable: False
  libtool:
    paths:
      libtool@2.4.6: /usr/local
    buildable: False
  m4:
    paths:
      m4@1.4.18: /usr/local
    buildable: False
  python:
    paths:
      python@2.7.13: /usr
    buildable: False
  all:
    providers:
      mpi: [mpich]
    compiler: [clang@9.0.0-apple]
```

[See website for example]

4. Install xSDK

After the edit, xSDK packages and external dependencies can be installed with a single command:

```
spack install xsdk
```

5. Install environment modules.

Optionally one can install environment-modules package to access xsdk packages as modules.

```
spack install environment-modules
```

After installation, the modules can be enabled with the following command line.

For bash:

```
# For bash users
$ source `spack location -i environment-modules`/Modules
/init/bash
# For tcsh or csh users
$ source `spack location -i environment-modules`/Modules
/init/tcsh
```

Installing xSDK (cont.)

6. Load xSDK module and its sub-modules.

```
spack load -r xsdk
```

Then, **module list** generates the following output, for example:

```
Currently Loaded Modulefiles:
 1) mpich-3.3b2-gcc-8.2.1-hexpnjh
 2) numactl-2.0.12-gcc-8.2.1-iazvfzp
 3) zlib-1.2.11-gcc-8.2.1-cwve3xs
 4) hdf5-1.10.4-gcc-8.2.1-pryqzbp
 5) openblas-0.3.4-gcc-8.2.1-thneimy
 6) hypre-2.15.1-gcc-8.2.1-c7h4gtr
 7) metis-5.1.0-gcc-8.2.1-f6wzzhi
 8) parmetis-4.0.3-gcc-8.2.1-gwbojrk
 9) superlu-dist-6.1.0-gcc-8.2.1-p2olnk3
 10) bzip2-1.0.6-gcc-8.2.1-dmheqyf
 11) boost-1.69.0-gcc-8.2.1-tpes32u
 12) glm-0.9.7.1-gcc-8.2.1-ozt2r2z
 13) matio-1.5.13-gcc-8.2.1-3oidyje
 14) netcdf-4.6.2-gcc-8.2.1-u7ywiqz
 15) trilinos-12.14.0-rc1-gcc-8.2.1-kass6i7
 16) petsc-3.10.3-gcc-8.2.1-eko2h5x
 17) pflotran-xsdk-0.4.0-gcc-8.2.1-hemk2tj
 18) alquimia-xsdk-0.4.0-gcc-8.2.1-oerexyg
 19) amrex-18.10.1-gcc-8.2.1-r5f4jnv
 20) adol-c-develop-gcc-8.2.1-bjm4wr2
 21) arpack-ng-3.6.3-gcc-8.2.1-wyahvso
 22) gsl-2.5-gcc-8.2.1-qzonakz
 23) intel-tbb-2019.2-gcc-8.2.1-spudoqk
 24) muparser-2.2.6.1-gcc-8.2.1-mf6kph5
 25) nanoflann-1.2.3-gcc-8.2.1-vivp3qi
 26) netlib-scalapack-2.0.2-gcc-8.2.1-45so534
 27) oce-0.18.3-gcc-8.2.1-o4dtwgt
 28) p4est-2.0-gcc-8.2.1-5ks4hsq
 29) suite-sparse-5.3.0-gcc-8.2.1-zr1pqbs
 30) sundials-3.2.1-gcc-8.2.1-bt6b3j5
 31) dealii-9.0.1-gcc-8.2.1-pkrq311
 32) mfem-3.4.0-gcc-8.2.1-xq55orp
 33) phist-1.7.5-gcc-8.2.1-z6bvhby
 34) plasma-18.11.1-gcc-8.2.1-g4kvnvg
 35) pumi-2.2.0-gcc-8.2.1-dd674x6
 36) slepc-3.10.1-gcc-8.2.1-w7otlnl
 37) strumpack-3.1.1-gcc-8.2.1-zcntogv
 38) tasmanian-6.0-gcc-8.2.1-qsse4mb
 39) xsdk-0.4.0-gcc-8.2.1-vq54mbz
```

[See website for example]

https://xsdk.info/installing-the-software

xSDK 0.4.0 platform testing

xSDK 0.4.0 has been updated/fixed on a regular basis on various workstations

- linux-fedora29-x86_64 / clang@7.0.0
- linux-fedora29-x86_64 / gcc@8.2.1
- linux-ubuntu18.04-x86_64 / clang@6.0.0-1ubuntu2
- linux-ubuntu18.04-x86_64 / gcc@7.3.0
- darwin-mojave-x86_64 / clang@10.0.0-apple
- darwin-mojave-x86_64 / gcc@8.2.0
- linux-ubuntu16.04-x86_64 / gcc@5.4.0
- linux-ubuntu14.04-x86_64 / gcc@4.8
- linux-centos7-x86_64 / intel@18.0.2
- linux-fedora29-aarch64 / gcc@8.2.1

See website for platform-specific details, including compilers.yaml and packages.yaml files for:

- **ALCF**:**Theta**: Cray XC40 with Intel compilers
- **NERSC**: **Cori**: Cray with Gnu compilers
- **OLCF**: **Titan**: Cray with Gnu compilers

Upcoming xSDK releases for ECP

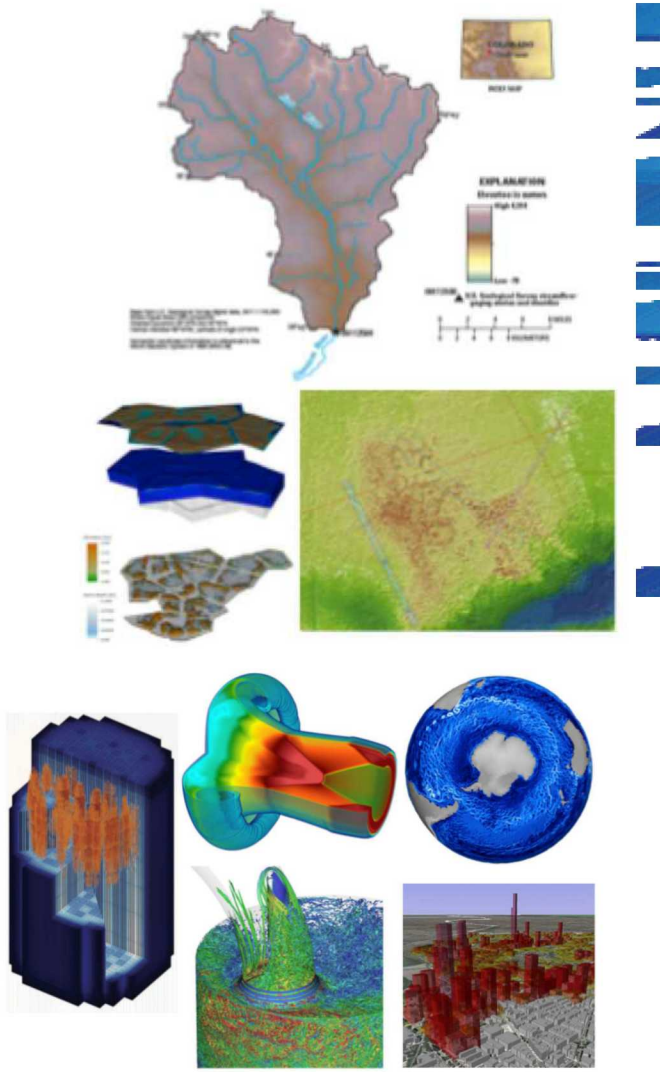
FY19-FY20: Regular releases of xSDK for ECP

Theme throughout ECP timeframe: **Expanding ECP math library capabilities for predictive science:** Sustainable coordination and delivery of math libraries across independent development efforts, with enhanced capabilities as needed by ECP applications

- Additional math packages compatible with xSDK community policies
- Deeper multilevel interoperability, including control inversion and adaptive execution
- Coordination with broader ECP software ecosystem

Applications using xSDK

- PFLOTRAN and Alquimia
  - Multiphysics & multiscale modeling of watershed dynamics
  - Provided as part of xSDK
  - Spack script for individual application packages
- Nalu in ExaWind
  - Call hypre from Trilinos (xSDK Trilinos)
- Laghos in CEED
  - MFEM and hypre
  - Planning to use SuperLU, SUNDIALS and PUMI
- AMPE and Truchas in ExaAM
  - SUNDIALS and hypre
  - Wrote Spack script for AMPE and Truchas



Tutorial outline:
xSDK Approach and Experiences

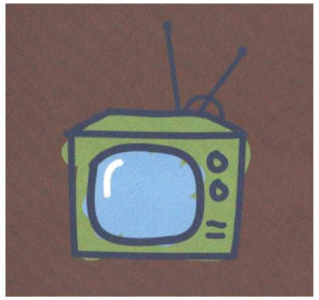
- Introduction
  - Math libraries and scientific software ecosystems
  - Building community and sustainability
  - xSDK history and goals to fulfill ECP needs
- About the xSDK
  - xSDK community policies
  - Short introduction to Spack
  - xSDK release process
  - Installing the xSDK
  - Using the xSDK in ECP applications
- **Lessons learned**
- **xSDK package overviews**

xSDK Lessons Learned: General Observation

- **Working toward shared understanding of issues and perspectives is essential and takes time**
  - Need regular opportunities for exchanging ideas, persistence, patience, informal interaction
  - Must establish common vocabulary

- **Lots of fun, too … xSDK: Life is good ☺**

It takes all kinds.


Think outside the box.


Face the bumps with a smile.


The pursuit is the reward.

xSDK Lessons Learned: Users' Perspective

- Building the whole xSDK takes time and produces a very large executable.
  - Future releases should allow building of a subsection.
- Need better document for Spack and xSDK
- Application developers might use **their own versions** of xSDK.
  - Some capabilities might no longer be supported, but necessary for their applications.
  - It will be important to provide flexibility through the xSDK to allow users to use their own versions of some xSDK libraries.
- xSDK member libraries should also pursue improved compatibilities where possible to **avoid for users to have building their own versions**.
  - New version typically provides improvement performance and interoperability (compilers, and other libraries)

xSDK Lessons Learned: Developers' Perspective

- Requires some code modifications to **eliminate naming conflicts**
  - Namespaces
  - Unique prefix for function names and preprocessor macros
- Maintaining interoperability needs close communication with the developers of other packages
  - Coordination for release scheduling is challenging
- Work toward better, faster, more people-efficient workflow for development and testing is important!
  - Continuous and integrated testing
  - Multiple compilers
  - Multiple parallel runtime setting (OpenMP, CUDA, etc.)

xSDK Lessons Learned

Background:

- deal.II was already compliant with almost all of xSDK's Community Policy Compatibilities.
- In particular, it has a very large testsuite (10,000+ tests) that covers all of the interfaces we have with other libraries; in some cases, we seem to have better coverage of these external libraries through the interfaces than the package's test suite itself.

Lessons learned:

- Avoid unprefixed macros or provide a way to disable them. Avoid unprefixed preprocessor variables.
- Don't use MPI_COMM_WORLD, but user provided MPI communicators.

xSDK Lessons Learned

Background:

- HYPRE had various issues that needed to be addressed.
  - Name space conflicts (some functions with simple names)
  - Overlooked prints of error messages
  - No exhaustive test suite that could be run on arbitrary computers

Lessons learned:

- Giving all functions the prefix ‘hypre_’ avoids namespace conflicts.
- Allowing for error messages to only be printed for a higher print level avoids undesired printouts.
- A test suite that allows users to test hypre solvers on any platform and check for errors is now available.

xSDK Lessons Learned

Background:

- MAGMA's solvers rely unconditionally on hardware accelerators (AMD, Intel, NVIDIA).
- Accelerators are optional for xSDK packages.

Lessons learned:

- Having established software practices helps with xSDK integration.
  - Continual maintenance is a must.
    - This is enforced for MAGMA with vibrant accelerator hardware market and frequent product releases.
  - Code documentation is required.
    - Large MAGMA user base made good documentation a must to ease the burden of answering user questions.
- User contributions might not meet xSDK requirements.
  - Adjustments were needed for user-contributed Spack package for MAGMA.
- New variant added to support xSDK builds with CUDA present on the installation system.

xSDK Lessons Learned

MFEM

Background:

- MFEM interfaces with numerous libraries in the XSDK including HYPRE, SUNDIALS, SuperLU, PETSc, and STRUMPACK.

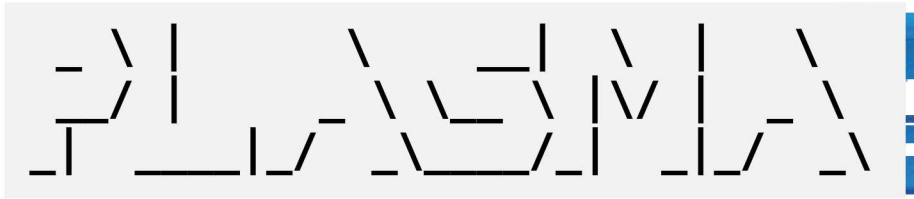
Lessons learned:

- Maintaining connections to all existing libraries requires significant effort and communication.
- Synchronizing releases with interoperating libraries that have API changes is difficult.
- Library version interoperability needs to be added to our testing suite.

xSDK

ECP EXASCALE COMPUTING PROJECT

xSDK Lessons Learned



Background:

- PLASMA relies on BLAS and LAPACK for low-level HPC optimizations.
  - Varied implementations and interfaces across hardware platforms
  - Test suites for BLAS and LAPACK are usually not available in most implementations
  - Additional set of basic routines (Core BLAS) did not gain community acceptance

Lessons learned:

- Must use name space for all interface entry points even if it might be an older project.
- Ease of installation is a necessity. Using established software configuration frameworks (autoconf, CMake, etc.) enforce portable code and adaptation to a variety of software environments.

xSDK Lessons Learned

PETSc/TAO

Background:

- PETSc development was being slowed down by backlog of branches waiting to be fully tested and moved into master.
- Through work on xSDK, we refactored PETSc tests to improve overall test functionality and better support continuous integration testing.

Lessons learned:

- Result: Work toward better, faster, more people-efficient workflow for testing has enabled moving developer branches into master branch without breaking master branch or requiring hand holding.
  - Introduced parallelism (and other techniques) to decrease time for running complete test suite while still providing high coverage
  - Simplified the addition of new tests into suite
  - Introduced finer grain control over what is tested
  - Making testing process more robust to random system, hardware, or software failures
  - Working toward making it easier for anyone to automatically run full PETSc test suite, including dashboard

xSDK Lessons Learned

PHIST

Background:

- PHIST already adhered to most of the xSDK policies, but still required a few changes.

Lessons learned:

- Had to duplicate some CMake flags although Spack is the high-level build system now
- Had to switch off performance optimizations (e.g. no OpenMP, no –march flag) to get it to compile on all platforms
- No easy way to uninstall all xSDK packages via Spack

xSDK Lessons Learned

Background:

- PUMI was not compatible with the requirement for runtime control of output - there were over 700 calls to functions from the printf ‘family’.

Lessons learned:

- Design your library from the beginning with a print statement wrapper so it can run in silent mode, or with various levels of output for performance information, developer level debugging, etc..
  - Use grep/sed to automate replacement of the printf family functions with the wrapper API.
  - The handful of C++ cout/cerr uses were manually replaced with the wrapper functions. In some cases stringstream was used to compose the strings and then those strings were passed into the API.

xSDK Lessons Learned

Background:

- SUNDIALS has interfaces to several external libraries e.g., PETSc, hypre, KLU, LAPACK, …
  - Existing CMake options did not align with xSDK policies.
  - Added redundant options that overwrite existing variables to maintain options for current users.
- MFEM has interfaces to SUNDIALS time integrators and nonlinear solvers.
  - Updates to SUNDIALS for xSDK compatibility were introduced along side a new linear solver API.
  - The new API broke compatibility with MFEM and required updating the MFEM interface to SUNDIALS.

Lesson learned:

- Packages working toward xSDK compatibility should adopt xSDK conventions early to ease user transition to new options.
- Maintaining interfaces between xSDK packages requires regular communication and testing with in-development versions.

xSDK Lessons Learned

SuperLU

Background:

- SuperLU initially faced challenges with build system, revision control, namespacing.

Lessons learned:

- Migration from manual editing make.inc to CMake/Ctest increases build-test productivity and robustness.
  - Easier to manage dependencies (ParMetis, machine-dependent files), and platform-specific versions (_MT, _DIST, GPU) and correctness
  - Better accommodate special build requirements (e.g., disable third-party software like ParMetis)
- Migration from svn to git improves distributed contributions and bug fixes. E.g., users have contributed:
  - Working with Windows environment, building as both static and shared libraries simultaneously
- Proper namespacing allows 3 versions of the library (serial, multithreaded and distributed) to be used simultaneously and to be used by other packages in xSDK.
- Improved productivity of new code development:
  - Wrote comprehensive regression unit test code
  - UseTravis CI for continuous integration on each git commit

xSDK Lessons Learned

Background:

- Trilinos had interfaces to both PETSc and hypre, but those interfaces were
  - Poorly documented (e.g. – no hypre interface document)
  - Not tested regularly (e.g. – the PETSc-Trilinos interface was broken in recent releases)

Lessons learned:

- Interfaces supported for the xSDK require regular testing and clear documentation
- Continual maintenance of code and documentation will be required; occasional fixes are insufficient

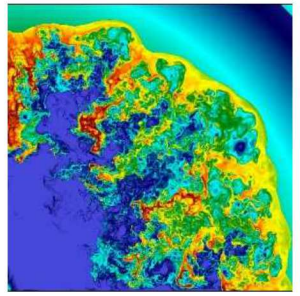
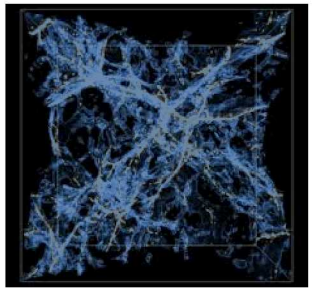
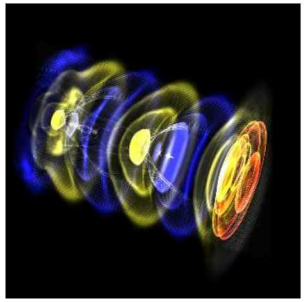
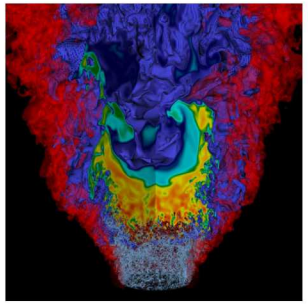
xSDK Package Overviews

AMReX

Block-structured adaptive mesh refinement framework. Support for hierarchical mesh and particle data with embedded boundary capability.

- **Capabilities**
  - Support for solution of PDEs on hierarchical adaptive mesh with particles and embedded boundary representation of complex geometry
  - Support for multiple modes of time integration
  - Support for explicit and implicit single-level and multilevel mesh operations, multilevel synchronization, particle, particle-mesh and particle-particle operations
  - Hierarchical parallelism –
    - hybrid MPI + OpenMP with logical tiling on multicore architectures
    - MPI + CUDA/OpenACC for CPU/GPU systems with support for launching kernels on GPUs based on use of C++ lambda functions
  - Native multilevel geometric multigrid solvers for cell-centered and nodal data
  - Highly efficient parallel I/O for checkpoint/restart and for visualization – native format supported by Visit, Paraview, yt
  - Tutorial examples available in download
- **Open source software**
  - Used for a wide range of applications including accelerator modeling, astrophysics, combustion, cosmology, multiphase flow, phase field modeling…
  - Freely available on github

Examples of AMReX applications

FASTMATH ECP

https://www.github.com/AMReX-Codes/amrex

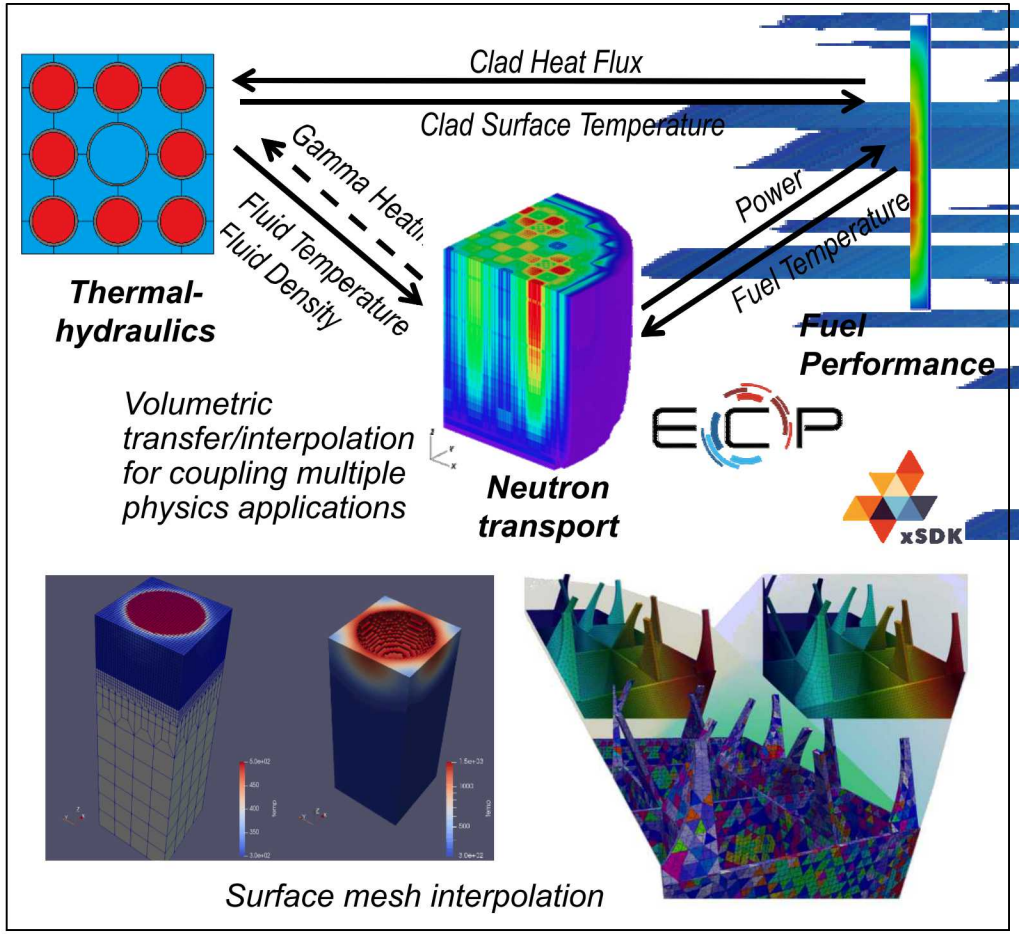
xSDK ECP EXASCALE COMPUTING PROJECT

DataTransferKit

OAK RIDGE National Laboratory

Open source library for parallel solution transfer.
Support for grid-based and mesh-free applications.

- **Overview**
  - Transfers application solutions between grids with differing layouts on parallel accelerated architectures
  - Coupled applications frequently have different grids with different parallel distributions; DTK is able to transfer solution values between these grids efficiently and accurately
  - Used for a variety of applications including conjugate heat transfer, fluid structure interaction, computational mechanics, and reactor analysis
- **Capabilities**
  - Support for DOE leadership class machines through MPI+Kokkos programming model
  - Algorithms demonstrated scalable to billions of degrees of freedom
  - General geometric search algorithms
    - Comparable serial performance to Boost r-Tree and NanoFlann
    - Also thread scalable on many core CPU and GPUs and distributed via MPI
  - Grid interpolation operators and mesh-free transfer operators



https://github.com/ORNL-CEES/DataTransferKit

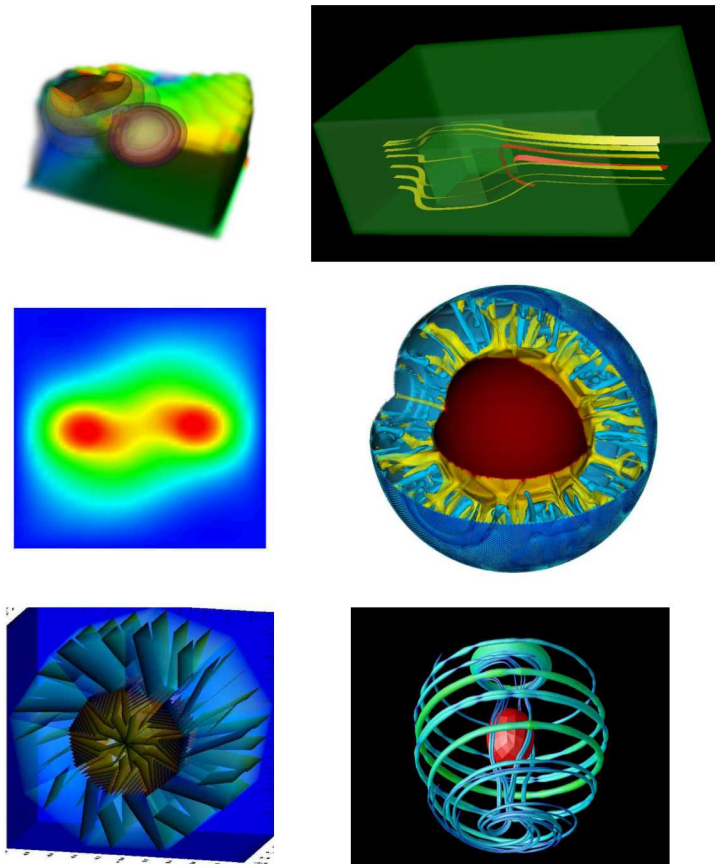
xSDK

ECP EXASCALE COMPUTING PROJECT

deal.II

deal.II — an open source finite element library. Modern interface to the complex data structures and algorithms required for solving partial differential equations computationally using state-of-the-art programming techniques.

- **Meshes and elements:**
  - Supports h- and p-adaptive meshes in 1d, 2d, and 3d
  - Easy ways to adapt meshes: Standard refinement indicators already built in
  - Many standard finite element types (continuous, discontinuous, mixed, Raviart-Thomas, Nedelec, ABF, BDM,...)
  - Full support for coupled, multi-component, multi-physics problems
- **Linear algebra:**
  - Has its own sub-library for dense and sparse linear algebra
  - Interfaces to PETSc, Trilinos, UMFPACK, ScaLAPACK, ARPACK
- **Pre- and postprocessing:**
  - Can read most mesh formats
  - Can write almost any visualization file format
- **Parallelization:**
  - Uses threads and tasks on shared-memory machines.
  - Uses up to 100,000s of MPI processes for distributed-memory machines.
  - Can use CUDA
- **Open-source software:**
  - Used for a wide range of applications including heart muscle fibers, microfluidics, oil reservoir flow, fuel cells, aerodynamics, quantum mechanics, neutron transport, numerical methods research, fracture mechanics, damage models, sedimentation, biomechanics, root growth of plants, solidification of alloys, glacier mechanics, and many others.
  - Freely available on GitHub



https://www.dealii.org/

xSDK

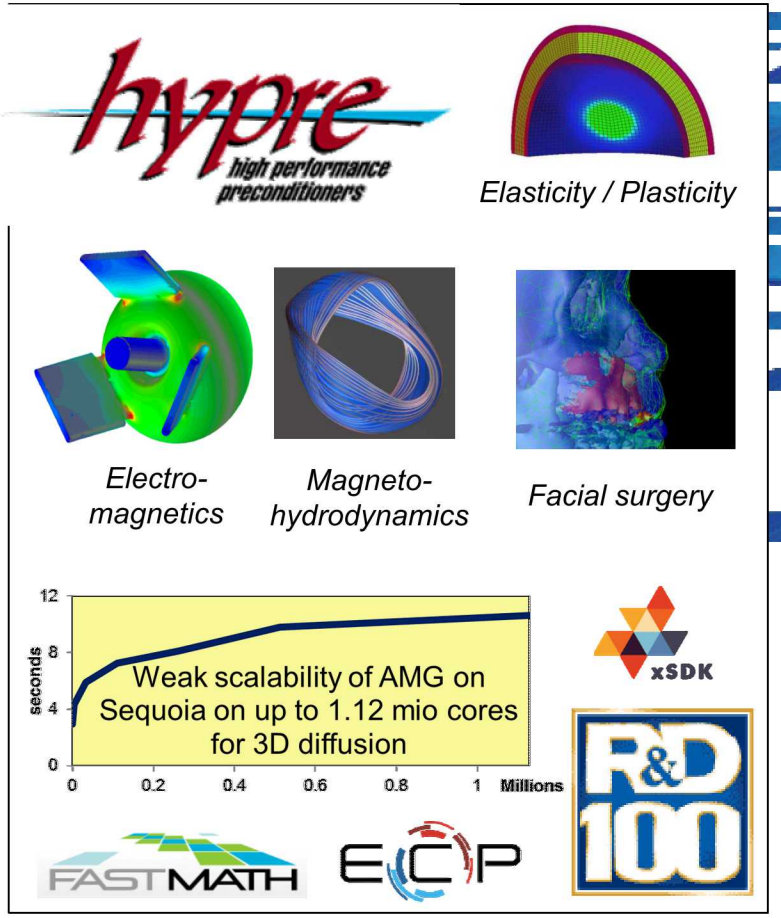
ECP EXASCALE COMPUTING PROJECT

hypre

Lawrence Livermore National Laboratory

Highly scalable multilevel solvers and preconditioners. Unique user-friendly interfaces. Flexible software design. Used in a variety of applications. Freely available.

- **Conceptual interfaces**
  - Structured, semi-structured, finite elements, linear algebraic interfaces
  - Provide natural "views" of the linear system
  - Provide for more efficient (scalable) linear solvers through more effective data storage schemes and more efficient computational kernels
- **Scalable preconditioners and solvers**
  - Structured and unstructured algebraic multigrid solvers
  - Maxwell solvers, H-div solvers
  - Multigrid solvers for nonsymmetric systems: pAIR, MGR
  - Matrix-free Krylov solvers
- **Open source software**
  - Used worldwide in a vast range of applications
  - Can be used through PETSc and Trilinos
  - Available on github: https://www.github.com/hypre-space/hypre



http://www.llnl.gov/CASC/hypre

xSDK ECP EXASCALE COMPUTING PROJECT

MAGMA

THE UNIVERSITY OF TENNESSEE KNOXVILLE

r algebra solvers and spectral decompositions for hardware accelerators.
ble dense direct and sparse iterative solvers for GPUs and coprocessors.

- **Dense Linear Algebra Solvers**
  - Linear systems of equations
  - Linear least squares
  - Singular value decomposition
- **Matrix spectrum methods**
  - Symmetric and non-symmetric eigenvalues
  - Generalized eigenvalue problems
  - Singular Value Decomposition
- **Sparse Solvers & Tensor Computations**

MAGMA SPARSE

ROUTINES	BiCG, BiCGSTAB, Block-Asynchronous Jacobi, CG, CGS, GMRES, IDR, Iterative refinement, LOBPCG, LSQR, QMR, TFQMR
PRECONDITIONERS	ILU / IC, Jacobi, ParILU, ParILUT, Block Jacobi, ISAI
KERNELS	SpMV, SpMM
DATA FORMATS	CSR, ELL, SELL-P, CSR5, HYB

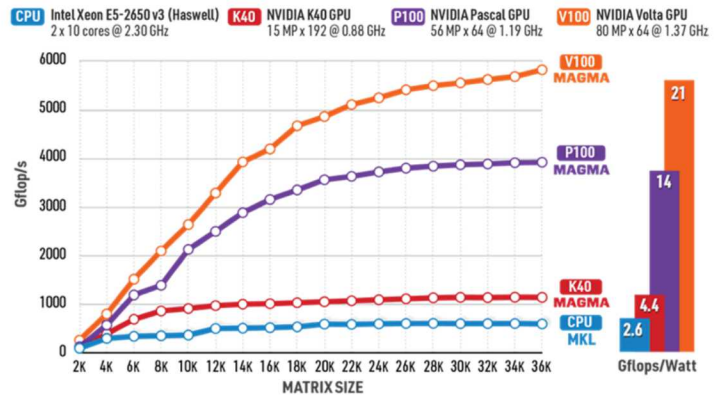
FEATURES AND SUPPORT

- **MAGMA 2.3** FOR CUDA
- **clMAGMA 1.4** FOR OpenCL
- **MAGMA MIC 1.4** FOR Intel Xeon Phi

CUDA	OpenCL	Intel Xeon Phi	
●	●	●	Linear system solvers
●	●	●	Eigenvalue problem solvers
●	●		Auxiliary BLAS
●			Batched LA
●		●	Sparse LA
●	●	●	CPU/GPU Interface
●	●	●	Multiple precision support
●			Non-GPU-resident factorizations
●	●	●	Multicore and multi-GPU support
● NEW			MAGMA Analytics/DNN
●	●	●	LAPACK testing
●	●	●	Linux
●	●		Windows
●	●		Mac OS

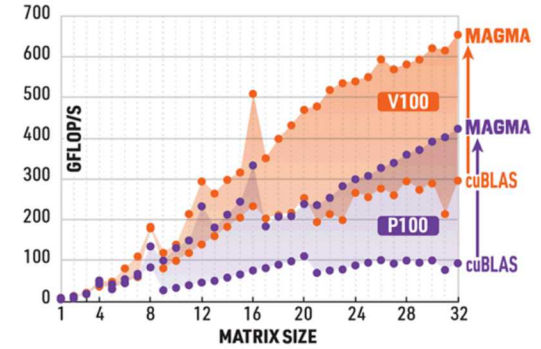
PERFORMANCE & ENERGY EFFICIENCY

MAGMA LU factorization in double precision arithmetic



PERFORMANCE OF BATCHED LU

in double precision arithmetic on 1 million matrices



http://icl.utk.edu/magma/

xSDK

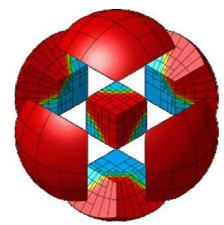
EXASCALE COMPUTING PROJECT

MFEM

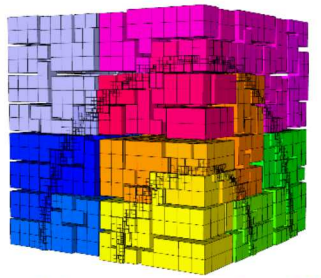
Lawrence Livermore National Laboratory

Free, lightweight, scalable C++ library for finite element methods. Supports arbitrary high order discretizations and meshes for wide variety of applications.

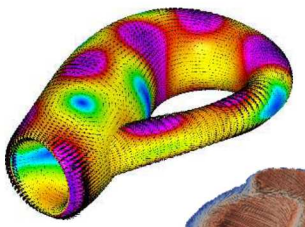
- **Flexible discretizations on unstructured grids**
  - Triangular, quadrilateral, tetrahedral and hexahedral meshes.
  - Local conforming and non-conforming refinement.
  - Bilinear/linear forms for variety of methods: Galerkin, DG, DPG, …
- **High-order and scalable**
  - Arbitrary-order H1, H(curl), H(div)- and L2 elements. Arbitrary order curvilinear meshes.
  - MPI scalable to millions of cores. Enables application development on wide variety of platforms: from laptops to exascale machines.
- **Built-in solvers and visualization**
  - Integrated with: HYPRE, SUNDIALS, PETSc, SUPERLU, …
  - Accurate and flexible visualization with VisIt and GLVis
- **Open source software**
  - LGPL-2.1 with thousands of downloads/year worldwide.
  - Available on GitHub, also via OpenHPC, Spack. Part of ECP's CEED co-design center.



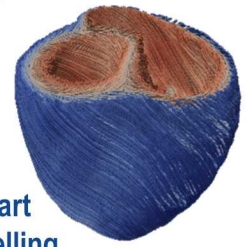
High order curved elements



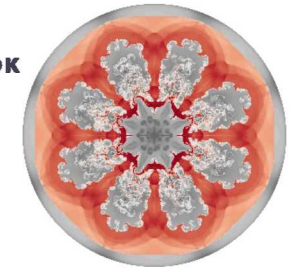
Parallel non-conforming AMR



Surface meshes



Heart modelling



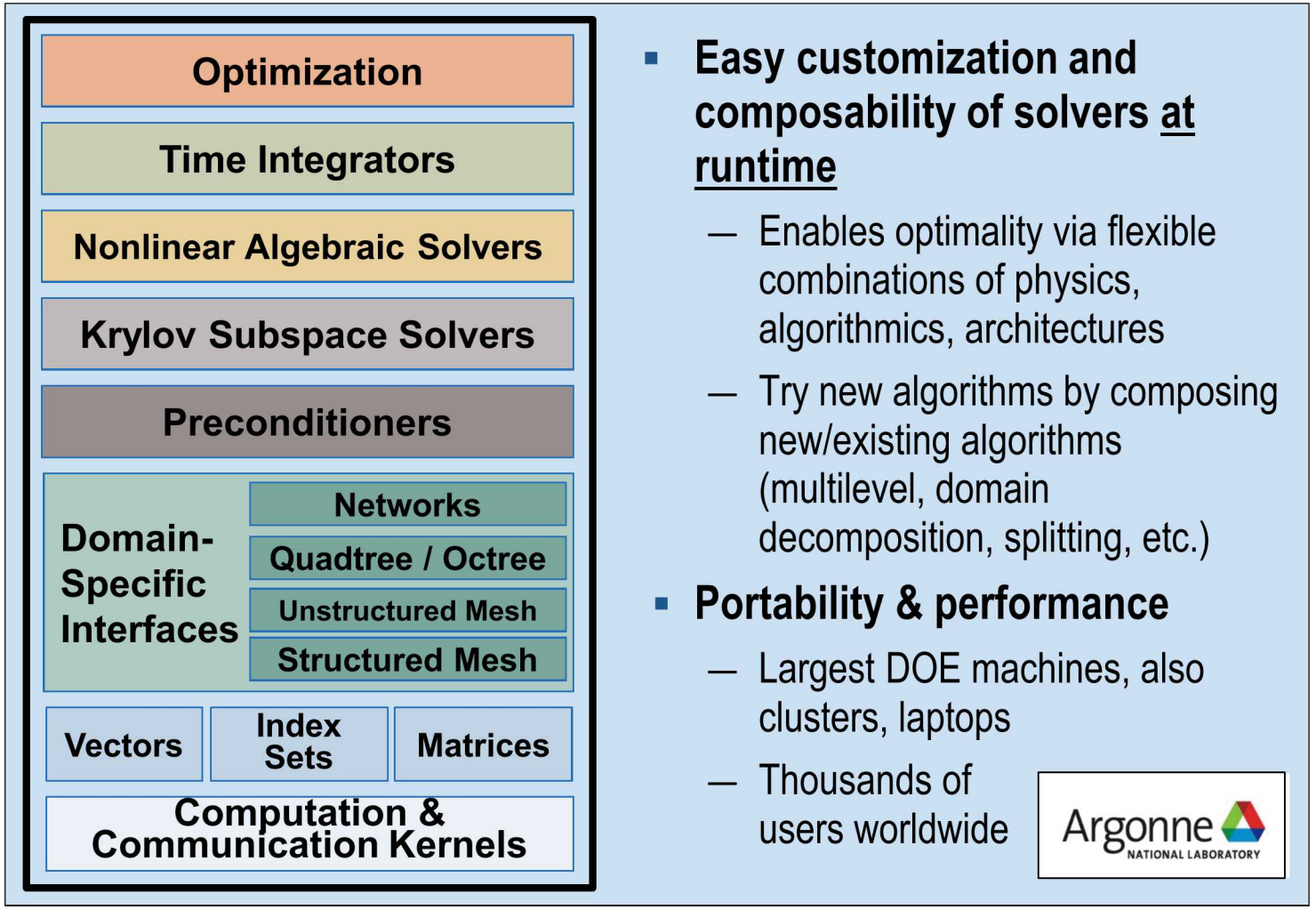
Compressible flow ALE simulations

http://mfem.org

PETSc/TAO

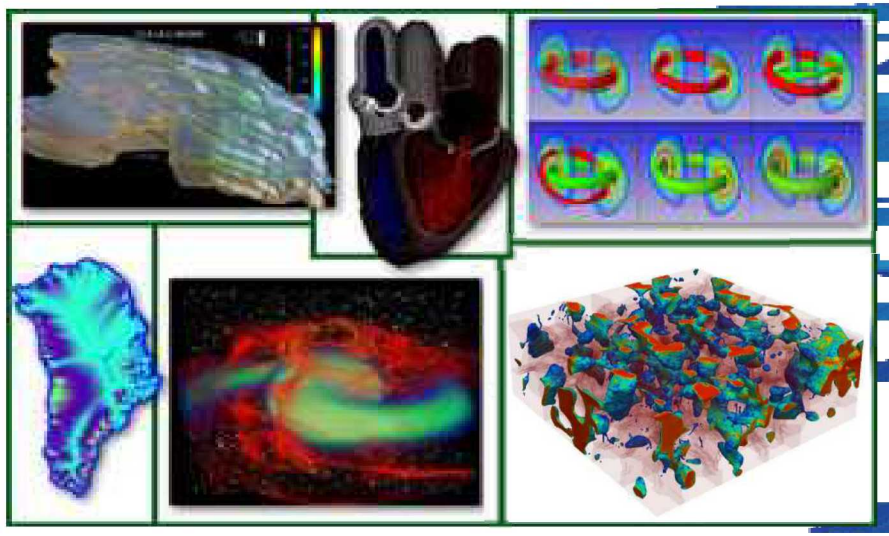
Portable, Extensible Toolkit for Scientific Computation / Toolkit for Advanced Optimization

Scalable algebraic solvers for PDEs. Encapsulate parallelism in high-level objects. Active & suppported user community. Full API from Fortran, C/C++, Python.



- **Easy customization and composability of solvers <u>at runtime</u>**
  - Enables optimality via flexible combinations of physics, algorithmics, architectures
  - Try new algorithms by composing new/existing algorithms (multilevel, domain decomposition, splitting, etc.)
- **Portability & performance**
  - Largest DOE machines, also clusters, laptops
  - Thousands of users worldwide

Argonne NATIONAL LABORATORY



PETSc provides the backbone of diverse scientific applications.

clockwise from upper left: hydrology, cardiology, fusion, multiphase steel, relativistic matter, ice sheet modeling

R&D 100

xSDK

FASTMATH

<u>https://www.mcs.anl.gov/petsc</u>

xSDK

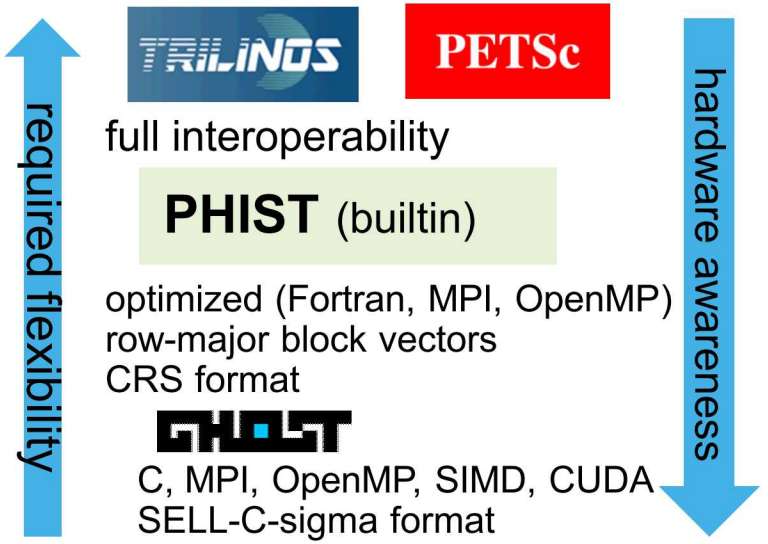
ECP EXASCALE COMPUTING PROJECT

PHIST Pipelined, Hybrid-parallel Iterative Solver Toolkit

Hybrid-parallel Iterative Sparse Eigenvalue and linear solvers
Integration with different linear algebra backends and preconditioners

- **Sparse Eigenvalue Solver: Block Jacobi-Davidson QR**
  - Hermitian or non-Hermitian matrices
  - Generalized problems $\boldsymbol{A}x = \lambda \boldsymbol{B}x$ (for Hermitian pos. def. matrix $\mathbf{B}$)
  - Blocked iterative linear solvers like GMRES, BiCGStab and CGMN
  - Can be accelerated by preconditioning
  - Matrix-free interface
  - Supported data types: D, Z, S, C
- **Algorithmic Building Blocks**
  - block orthogonalization
  - Eigenvalue counting (kernel polynomial method/KPM)
  - Fused basic operations for better performance
- **Various interfaces**
  - C, C++, Fortran 2003, Python

Can choose from several backends at compile time (Trilinos/Tpetra in xSDK 0.4.0)



Funded by the DFG project ESSEX
SPPEXA

https://bitbucket.org/essex/phist/

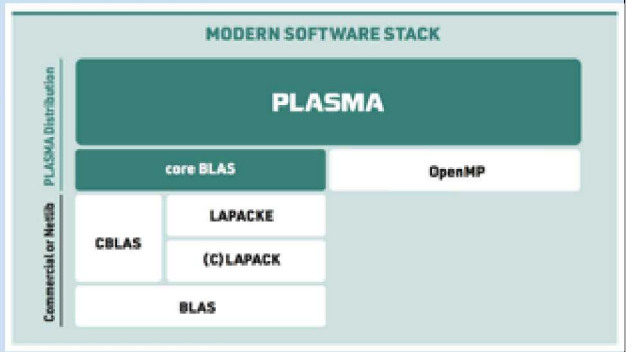
xSDK
ECP EXASCALE COMPUTING PROJECT

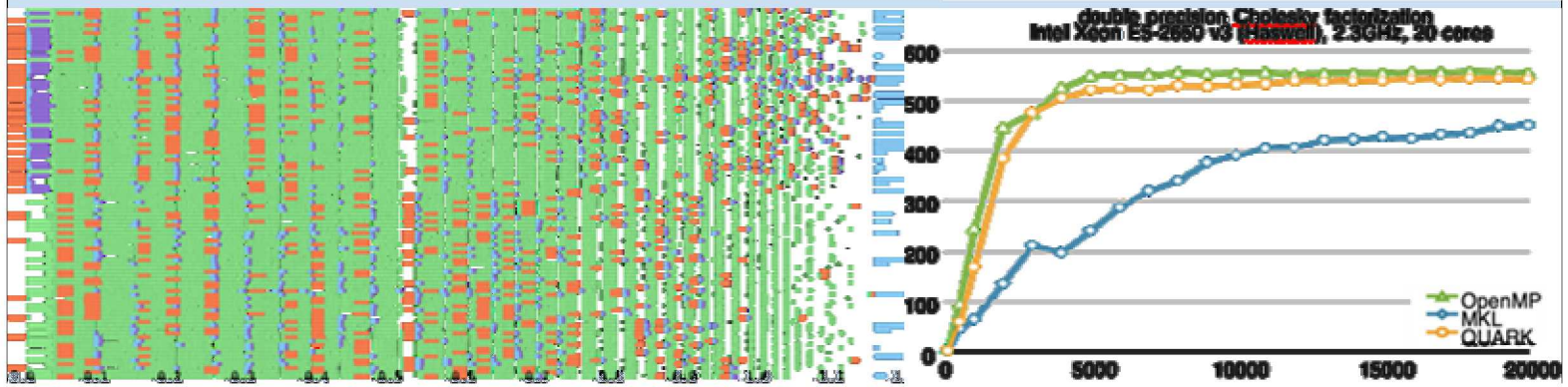
PLASM

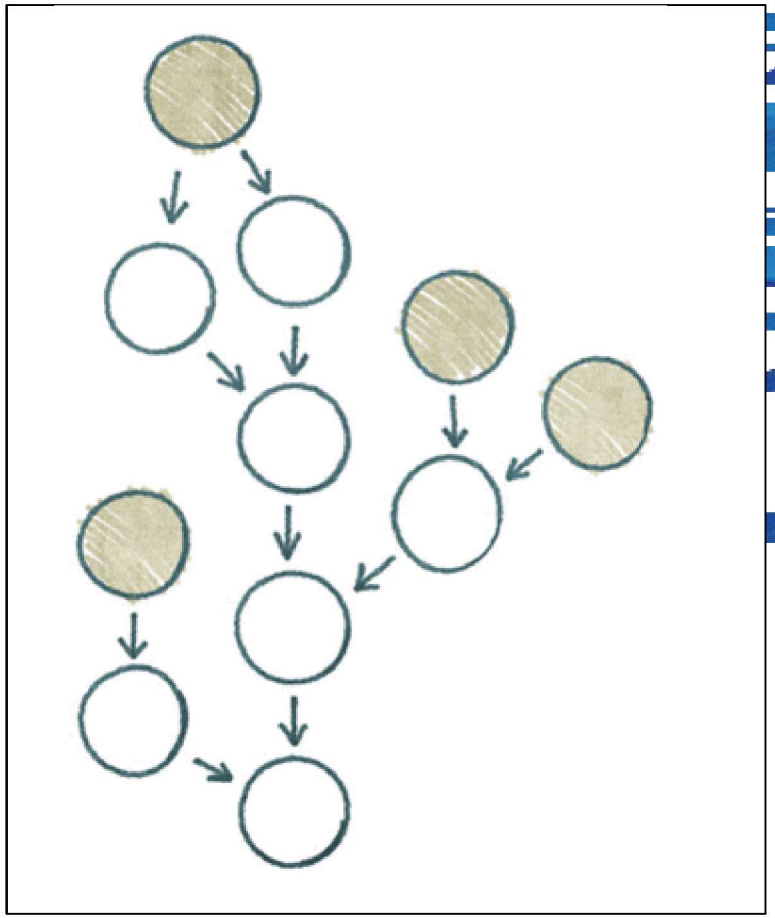
THE UNIVERSITY OF TENNESSEE KNOXVILLE

r algebra solvers and spectral decompositions for multicore processors.
ble and scalable dense solvers for large core counts.

- **Dense Linear Algebra Solvers**
  - Linear systems of equations
  - Linear least squares
  - Positive/Hermitian definitive solvers
- **Matrix spectrum methods**
  - Symmetric and non-symmetric eigenvalues
  - Generalized eigenvalue problems
  - Singular Value Decomposition
- **Data conversion and thread control**







http://icl.utk.edu/plasma/

xSDK

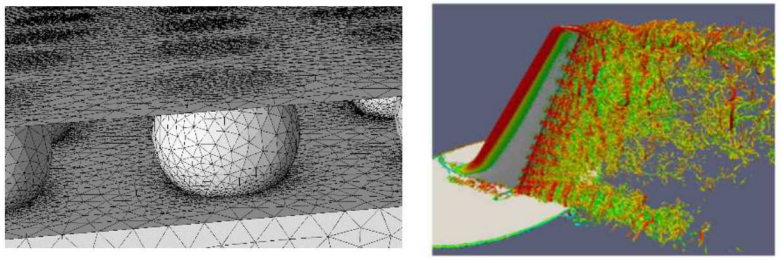
ECP EXASCALE COMPUTING PROJECT

PUMi: Parallel Unstructured Mesh Infrastructure

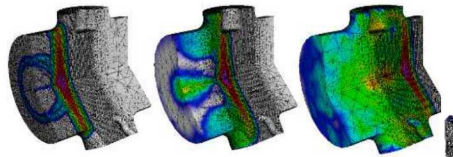
Parallel management and adaptation of unstructured meshes.
Interoperable components to support the development of unstructured mesh simulation workflows

- **Core functionality**
  - Distributed, conformant mesh with entity migration, remote read only copies, fields and their operations
  - Link to the geometry and attributes
  - Mesh adaptation (straight and curved), mesh motion
  - Multi-criteria partition improvement
  - Distributed mesh support for Particle In Cell methods
- **Designed for integration into existing codes**
  - Conformant with XSDK
  - Permissive license enables integration with open and closed-source codes
- **In-memory integrations developed**
  - MFEM: High order FE framework
  - PHASTA: FE for turbulent flows
  - FUN3D: FV CFD
  - Proteus: Multiphase FE
  - ACE3P: High order FE for EM
  - M3D-C1: FE based MHD
  - Nektar++: High order FE for flow
  - Albany/Trilinos: Multi-physics FE

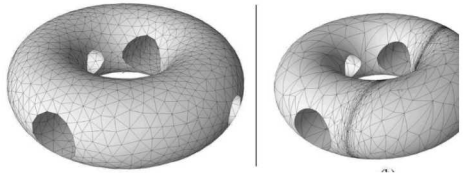
PUMi SCOREC ECP Rensselaer FASTMATH



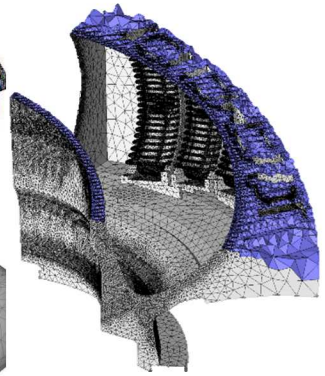
Applications with billions of elements: flip-chip (L), flow control (R)



Mesh adaptation for evolving features



Anisotropic adaptation for curved meshes



RF antenna and plasma surface in vessel.

Source Code: github.com/SCOREC/core
Paper: www.scorec.rpi.edu/REPORTS/2014-9.pdf

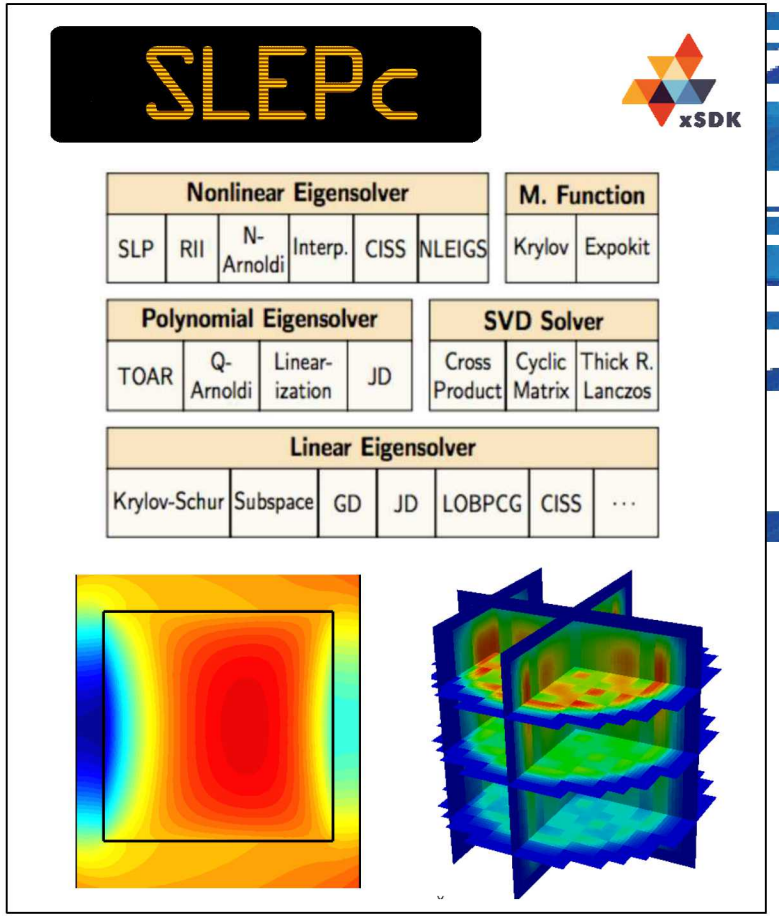
xSDK ECP EXASCALE COMPUTING PROJECT

SLEPc

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Scalable Library for Eigenvalue Problem Computations. Parallel solvers for linear and nonlinear eigenproblems. Also functionality for matrix functions.

- **Linear eigenvalue problems and SVD**
  - Standard and generalized eigenproblem, $Ax=\lambda x$, $Ax=\lambda Bx$; singular values $Au=\sigma v$
  - Easy selection of target eigenvalues, shift-and-invert available for interior ones
  - Many solvers: Krylov, Davidson, LOBPCG, contour integral, …
- **Nonlinear eigenvalue problems**
  - Polynomial eigenproblem $P(\lambda)x=0$, for quadratic or higher-degree polynomials
  - Solvers: Krylov with compact basis representation; Jacobi-Davidson
  - General nonlinear eigenproblem $T(\lambda)x=0$, for any nonlinear function incl. rational
- **Matrix functions**
  - Parallel Krylov solver to evaluate $y=f(A)v$
  - Support for matrix exponential, square root, etc. and combinations thereof
- **Extension of PETSc**
  - Runtime customization, portability and performance, C/C++/Fortran/python
  - Can use any PETSc linear solvers and preconditioners



http://slepc.upv.es

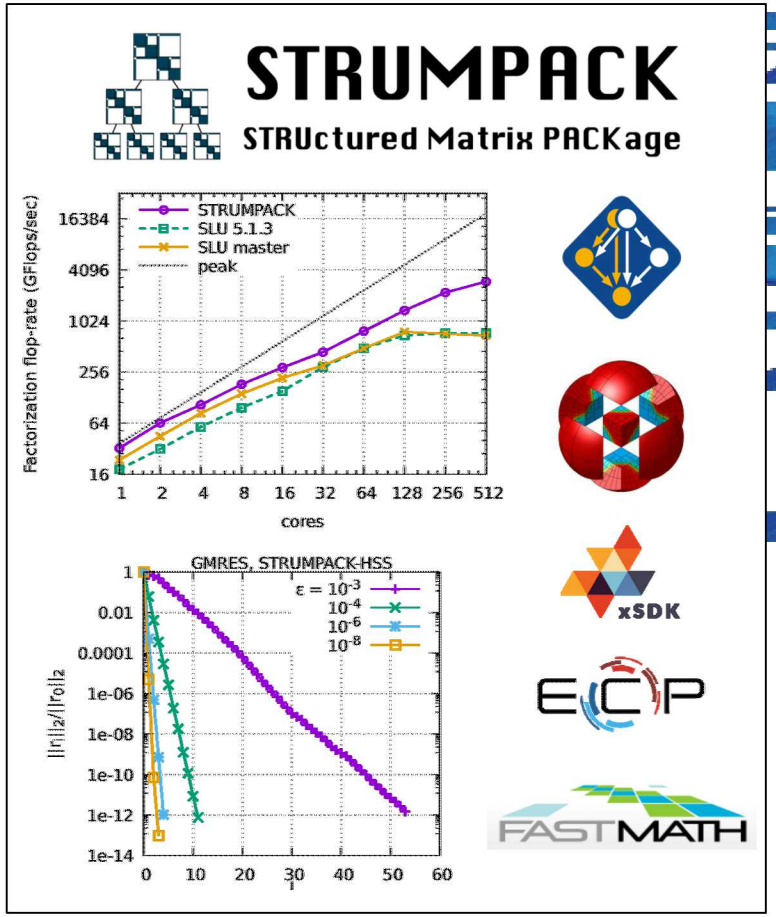
xSDK

ECP EXASCALE COMPUTING PROJECT

STRUMPACK

STRUctured Matrix PACKage. Hierarchical solvers for dense rank-structured matrices; fast sparse solver and robust and scalable preconditioners.

- **Dense Matrix Solvers, Hierarchical Approximations**
  - Hierarchical partitioning, low-rank approximations
  - Formats: Hierarchically Semi-Separable (HSS), Hierarchically Off-Diagonal Block Low-Rank (HODLR), Block Low-Rank (BLR)
  - Applicable to integral equations discretized with boundary element methods, structured matrices such as Cauchy or Toeplitz, kernel matrices, covariance matrices, …
  - Algorithms with much lower asymptotic complexity than corresponding ScaLAPACK routines
- **Sparse Direct Solver**
  - Multifrontal algorithm, Fill-reducing orderings: Par-METIS, PT-Scotch, RCM, spectral
  - Good scalability, fully distributed, parallel symbolic phase
- **Sparse Preconditioners**
  - Sparse direct solver with dense hierarchical (low-rank) approximations
  - Scalable and robust, aimed at PDE discretizations, indefinite systems, …
  - Iterative solvers: GMRES, BiCGStab, iterative refinement
- **Software**
  - BSD License, MPI+OpenMP, scalable to 10K+ cores
  - Interfaces from PETSc, MFEM (Trilinos coming), available in Spack
  - Under very active development



github.com/pghysels/STRUMPACK

xSDK ECP EXASCALE COMPUTING PROJECT

SUNDIALS

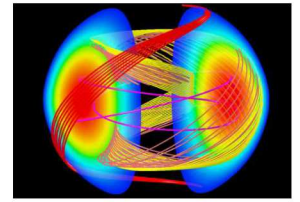
SUite of Nonlinear DIfferential /ALgebraic equation Solvers

Adaptive time integrators for ODEs and DAEs and efficient nonlinear solvers
Used in a variety of applications. Freely available. Encapsulated parallelism

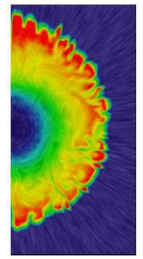
- **ODE integrators:**
  - CVODE(S): variable order and step BDF (stiff) and Adams (non-stiff)
  - ARKode: variable step implicit, explicit, additive IMEX, and multirate Runge-Kutta
- **DAE integrators**: IDA(S) - variable order and step BDF integrators
- **Sensitivity Analysis (SA):** CVODES and IDAS provide forward and adjoint SA
- **Nonlinear Solvers:** KINSOL - Newton-Krylov, Picard, and accelerated fixed point
- **Modular Design**
  - Written in C with interfaces to Fortran
  - Users can supply their own data structures and solvers
  - Optional use structures: serial, MPI, threaded, MPI+CUDA, MPI+RAJA, OpenMP Device, hypre, and PETSc
- **Open Source Software**
  - Freely available (BSD License) from LLNL site, github, and Spack
  - Can be used from MFEM and PETSc

Lawrence Livermore National Laboratory

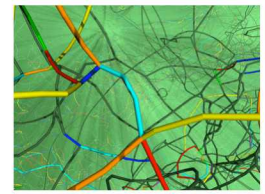
SUNDIALS is used by thousands worldwide in applications from research and industry



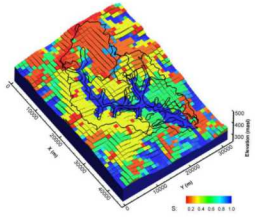
Magnetic reconnection



Core collapse super-nova



Dislocation dynamics



Subsurface flow

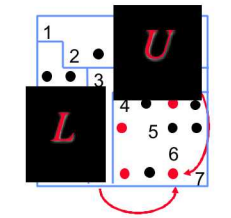
SUNDIALS is supported by extensive documentation, a sundials-users email list, and an active user community

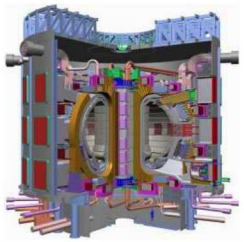
http://www.llnl.gov/CASC/sundials

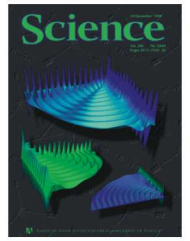
SuperLU

Supernodal Sparse LU Direct Solver. Unique user-friendly interfaces. Flexible software design. Used in a variety of applications. Freely available.

- **Capabilities**
  - Serial (thread-safe), shared-memory (SuperLU_MT, OpenMP or Pthreads), distributed-memory (SuperLU_DIST, hybrid MPI+ OpenM + CUDA).
    - Implemented in C, with Fortran interface
  - Sparse LU decomposition, triangular solution with multiple right-hand sides
  - Incomplete LU (ILU) preconditioner in serial SuperLU
  - Sparsity-preserving ordering:
    - Minimum degree ordering applied to A^TA or A^T+A
    - Nested dissection ordering applied to A^TA or A^T+A [(Par)METIS, (PT)-Scotch]
  - User-controllable pivoting: partial pivoting, threshold pivoting, static pivoting
  - Condition number estimation, iterative refinement.
  - Componentwise error bounds
- **Performance**
  - Factorization strong scales to 24,000 cores (IPDPS'18)
  - Triangular solve strong scales to 4000 cores (CSC'18)
- **Open source software**
  - Used worldwide in a vast range of applications, can be used through PETSc and Trilinos, available on github







ITER tokamak quantum mechanics

Widely used in commercial software, including AMD (circuit simulation), Boeing (aircraft design), Chevron, ExxonMobile (geology), Cray's LibSci, FEMLAB, HP's MathLib, IMSL, NAG, SciPy, OptimaNumerics, Walt Disney Animation.

FASTMATH xSDK

http://crd-legacy.lbl.gov/~xiaoye/SuperLU

xSDK ECP EXASCALE COMPUTING PROJECT

Trilinos

Sandia National Laboratories

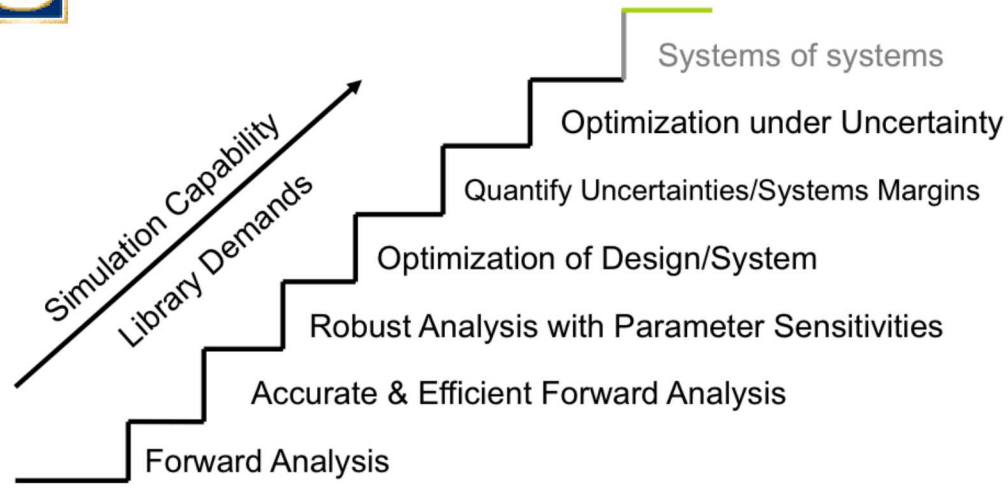
Optimal kernels to optimal solutions. Over 60 packages. Laptops to leadership systems. Next-gen systems, multiscale/multiphysics, large-scale graph analysis.

- **Optimal kernels to optimal solutions**
  - Geometry, meshing
  - Discretization, load balancing
  - Scalable linear, nonlinear, eigen, transient, optimization, UQ solvers
  - Scalable I/O, GPU, manycore
  - Performance Portability Across Multiple Platforms provided by **Kokkos**
- **60+ packages**
  - Other distributions: Cray LIBSCI, Github repo
  - Thousands of users, worldwide distribution
  - Laptops to leadership systems



R&D 100

Transforming Computational Analysis To Support High Consequence Decisions



Each stage requires *greater performance* and *error control* of prior stages:
Always will need: more accurate and scalable methods.
more sophisticated tools.

https://trilinos.org

xSDK FASTMATH TRILINOS

xSDK ECP EXASCALE COMPUTING PROJECT

More info

- **xSDK Foundations: Toward an Extreme-scale Scientific Software Development Kit,** R. Bartlett, I. Demeshko, T. Gamblin, G. Hammond, M. Heroux, J. Johnson, A. Klinvex, X. Li, L.C. McInnes, D. Osei-Kuffuor, J. Sarich, B. Smith, J. Willenbring, U.M. Yang, https://arxiv.org/abs/1702.08425, *Supercomputing Frontiers and Innovations*, vol 4 No 1 (2017), pp.69-82.
- https://xsdk.info

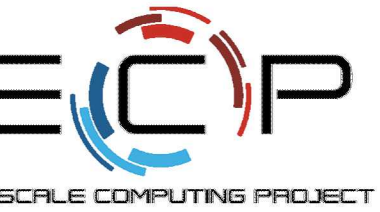
License, citation and acknowledgements

License and Citation

- This work is licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0).
- Requested citation: xSDK Developers, *An introduction to the xSDK, a community of diverse numerical HPC software packages* , ECP 3rd Annual Meeting, Houston, TX, January 15, 2019

Acknowledgements

- 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 (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration.



Thank you!