LAWRENCE LIVERMORE NATIONAL LABORATORY

LLNL-TR-820178

Summarizing the interoperabilities between xSDK members

C. Balos, S. Balay, A. Fisher, P. Lusczcek, S. Osborn, U. Yang

March 4, 2021

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Summarizing the interoperabilities between xSDK members

Report in fulfillment of Milestone STMS05-31: Document summarizing the interoperabilities between xSDK members

Cody Balos, Satish Balay, Aaron Fisher, Piotr Lusczcek, Sarah Osborn, Ulrike Meier Yang, in collaboration with other xSDK project members

Table of Contents

Introduction

Rapid, efficient production of high-quality, sustainable extreme-scale scientific applications is best accomplished using a rich ecosystem of state-of-the art reusable libraries, tools, lightweight frameworks, and defined software methodologies, developed by a community of scientists who are striving to identify, adapt, and adopt best practices in software engineering. The vision of the xSDK is to provide infrastructure for and interoperability of a collection of related and complementary software elements — developed by diverse, independent teams throughout the high-performance computing (HPC) community — that provide the building blocks, tools, models, processes, and related artifacts for rapid and efficient development of high-quality applications. A primary component, and challenge, of the xSDK is to improve interoperability among software libraries and domain components.

This document summarizes the current and planned interoperabilities of the twenty-three xSDK member packages as of March 2021. Additionally, the status of the xSDK example codes that demonstrate and test the interoperabilities within the xSDK is provided.

Current and Planned xSDK Interoperabilities

To gather information about planned updates to interoperability support, each member package answered a brief questionnaire. The survey questions are provided in Appendix A. The objective was to evaluate the current status of interoperability support, including if the functionality exists but is not yet enabled within the xSDK Spack package, and query for any plans or in-progress work for expanding interoperability support with another xSDK member package. Table 1 shows currently supported interoperabilities, as well as planned support, and whether the feature is enabled with the xSDK Spack package.

Table 1: This table shows the status of package interoperabilities as of March 2021. Row entries indicate the status of the interoperability where the row-labeled package USES the column-labeled package. Column entries indicate the status of the interoperability where the column-labeled package IS USED BY the row-labeled package.

A yellow box indicates that the package has support for using some part of the other. A blue box indicates that additionally, the interoperability is enabled in the xSDK Spack package. A green box indicates that the interoperability is in development or planned.

(Abbreviations used: AMR = AMReX, BFP = ButterflyPACK, dea = deal.II, DTK = DataTransferKit, GKO = Ginkgo, hff = heFFTe, HYP = hypre, lEn = libEnsemble, MAG = MAGMA, MFE = MFEM, Omg = Omega_h, PET = PETSc, PHI = PHIST, PLA = PLASMA, PRE = preCICE, PUM = PUMI, SLA = SLATE, SLE = SLEPc, STR = STRUMPACK, SUN = SUNDIALS, SLU = SuperLU_DIST, TAS = TASMANIAN, Tri = Trilinos)

	AMR	BFP	dea	DTK	GKO	hFF	HYP	lEn	MAG	MFE	Omg	PET	PHI	PLA	PRE	PUM	SLA	SLE	STR	SUN	SLU	TAS	Tri
AMR	⬛						🟨					🟨								🟩			
BFP		⬛							🟨														
dea			⬛		🟦		🟨					🟦						🟦		🟦	🟨		🟦
DTK				⬛																			🟦
GKO					⬛																		
hFF						⬛			🟦														
HYP							⬛														🟦		
lEn								⬛				🟦										🟨	
MAG									⬛														
MFE					🟦		🟦			⬛	🟩	🟦				🟦		🟨	🟦	🟦	🟦		
Omg											⬛												
PET							🟦					⬛							🟨	🟨	🟦		🟦
PHI												🟨	⬛										🟦
PLA									🟨					⬛			🟩						
PRE			🟨									🟦			⬛								
PUM											🟨					⬛							🟨
SLA																	⬛						
SLE												🟦						⬛					
STR		🟦							🟨								🟦		⬛				
SUN					🟩		🟦		🟨			🟦								⬛	🟦		🟦
SLU																					⬛		
TAS								🟦	🟦													⬛	
Tri							🟦	🟨	🟨			🟨				🟨			🟨		🟦		⬛

Key	
🟨 Yellow	Interoperability exists
🟦 Blue	Interoperability exists and is enabled in xSDK Spack
🟩 Green	Interoperability is planned
⬛ Black(ish)	Diagonal

Many xSDK member packages have enabled the existing interoperabilities through the xSDK Spack package; however, bottlenecks can prevent an existing interoperability to not be enabled within the xSDK Spack package, e.g., version issues, etc. 5 existing interoperabilities are planned to be enabled within the xSDK Spack package (PETSc + STRUMPACK, PETSc + SUNDIALS, PLASMA + MAGMA, PUMI + Omega_h, SUNDIALS + MAGMA), 3 have performance-based reasons for not enabling the existing interoperability with another xSDK package (ButterflyPACK + MAGMA, Trilinos + hypre and Trilinos + PETSc), and 1 has a Spack installation issue that prevents the interoperability being enabled (libEnsemble + Tasmanian) where `*lib1 + lib2*' indicates that *lib1* calls *lib2.* The specific responses from the questionnaire on the status of enabling an interoperability connection that exists but is not yet enabled within the xSDK Spack package (I.e. moving from Yellow to Blue in Table 1) are given below, as well as details on bottlenecks that make this difficult:

- ButterflyPACK + MAGMA: Interoperability works but performance needs to be carefully evaluated. We will enable interoperability within xSDK Spack if we observe good performance.
- libEnsemble + Tasmanian: Tasmanian is one we use in examples. But it must use the Python interface, and this currently requires building with the '--user' option. It may be worth looking at whether installation could be simplified and possibly we could include as a Spack variant.
- PETSc + STRUMPACK: Plan to work on exposing petsc+strumpack in Spack
- PETSc + SUNDIALS: Currently PETSc interfaces with an old SUNDIALS-2.5.0 version. Will update to the latest version in FY21.
- PLASMA + MAGMA: Planning for enabling interoperability in xSDK Spack in 2021 to be included in the next xSDK release.
- PUMI + Omega_h: Within the next year we hope to enable this.
- SUNDIALS + MAGMA: Sundials interoperability with MAGMA is not enabled in the xSDK Spack package yet (since it was added after xsdk-0.6.0) but we plan to enable it.

- Trilinos + hypre and Trilinos + PETSc:
    - hypre interface was once activated, but the user did not like it because of extra memory copies hurting the performance.
    - PETSc (Eptra and AztecOO) interface has not been tested for a long time, and the code is part of the old 1st generation packages that are under "maintenance mode."

New interoperabilities among xSDK members (and proposed new members) are planned during the remainder of FY21 and beyond (see green boxes in Table 1). The status and timeline for the individual interoperabilities among libraries is given below:

- MFEM + Omega_h: Interoperability is enabled for some limited use cases. Significant work is left to enable the full capability and is currently underway.
- SUNDIALS + MAGMA: This integration exists, and will be exposed in the next release cycle.
- DTK + ArborX: The required dependency ArborX is proposed as a new member package on its own.
- Deal.II + ArborX: There is a pull request to allow using ArborX (a proposed new xSDK member) with deal.II.
- Deal.II + SUNDIALS: The SUNDIALS interoperability existed already but not with the recent version. This is in the process of getting fixed.
- Trilinos + STRUMPACK: STRUMPACK is supported by Amesos2. This feature has been already integrated in the master branch.
- SUNDIALS + Ginkgo: Early planning stage of interoperability with Ginkgo.
- SLEPc + MAGMA: We would like to add an interface to MAGMA, but don't have resources for this now.
- ButterflyPACK + SLATE: We plan to evaluate the possibility of interoperability with Slate starting in FY21.

The xSDK project will continue to provide a medium of communication among these teams to help support development of such interfaces and their continual testing.

xSDK Example Test Suite: Status and Planned Updates

The xSDK example codes are a demonstration of interoperability between xSDK libraries and provide training for xSDK library users interested in using these capabilities. The first version of the suite of example codes demonstrating the use of various xSDK packages in combination was released in March 2020. The example codes in xSDK Examples v0.1.0 feature examples exhibiting the following interoperabilities (`*lib1* + *lib2*' indicates that *lib1* calls *lib2)*:

- hypre + SuperLU_DIST
- libEnsemble + PETSc/TAO
- MFEM + hypre + SuperLU_DIST
- MFEM + PETSc
- MFEM + SUNDIALS
- PETSc + hypre + SuperLU_DIST
- SUNDIALS + SuperLU_DIST
- SUNDIALS + PETSc
- Trilinos + SuperLU_DIST
- MFEM + Ginkgo (not enabled in release v0.1.0)

For the new release of the xSDK-examples test suite (v0.2.0), the xSDK examples have been updated to use the current xSDK release – xSDK v0.6.0. Additionally, the following examples have been added to the test suite to demonstrate increased interoperability features within the xSDK, including examples using GPU functionality:

- MFEM + Ginkgo (now enabled within Spack package)
- MFEM + SUNDIALS using CUDA
- SUNDIALS + MAGMA using CUDA
    - This example is not enabled in v0.2.0 as this interoperability is not in xSDK 0.6.0, but can be built with a provided cmake script.

Additional examples to be included in future versions of the xSDK examples test suite have been identified to demonstrate use-cases for utilizing various xSDK libraries in tandem. A planned addition to the test suite is another example of demonstrating MFEM + Ginkgo interoperability, using advanced preconditioners for matrix-free linear solvers. This work is scoped for Q2/Q3 2021.

Other potential examples that would be beneficial to add for future releases of the xSDK examples include demonstrating the following interoperabilities:

- PUMI + Omega_h (Once PUMI + Omega_h is enabled within xSDK)
- AMReX + hypre/PETSc
- PETSc + SuperLU_DIST
- MFEM + STRUMPACK.

In addition to providing potential users examples of how different xSDK libraries can work together to solve problems of interest, the xSDK examples are used for testing as well. Adding more examples not only provides users with additional demonstrations, but also enhances the testing of interoperabilities within the xSDK.

Lessons Learned: Outlook for Increasing xSDK Interoperabilities

When evaluating new examples to add to the xSDK example suite, a common issue arose where new interoperability features have been added to individual xSDK libraries, but those features are not yet in a released version of xSDK. Therefore, either the examples must be built separately (not with the Spack package for the xSDK-examples) or more frequent releases of the xSDK are necessary. This is problematic as synchronizing releases with interoperating libraries that have API changes is difficult and time-consuming. Additionally, the individual libraries have independent release schedules that make coordinating for a new xSDK release challenging.

Steps are being taken to work toward a faster, and more people-efficient workflow for development and testing within the xSDK. Currently a Gitlab CI testing capability for the xSDK is being employed to test the entire xSDK on several different platforms. Additions are being made to the testing capability by adding new tests, and layers of testing, e.g., only testing a subset of xSDK packages to more easily diagnose potential issues. The primary goal of the CI testing is to ensure the current release version of the xSDK is stable. However, an important secondary opportunity is to test with the development versions of various libraries. This will enable potential issues, like API changes, to be identified sooner.

Additional survey responses

The following responses were given to the survey question, “Are there other interoperability needs or new features that would be useful for ECP applications?”:

- PLASMA: Useful integration would be OpenMP offload including C++ with native vendor libraries.
- Trilinos: We request the users to introduce the 2nd generation packages for a long term solution. I have heard that there are still several applications and libraries using the old 1st generation packages. We are not planning to add new features and performance improvement (such as GPU support).
- PETSc: Likely exploring GPU features from the current packages.
- libEnsemble: Variable propagation in Spack could be useful.

Appendix A: Planned updates of interoperability between xSDK Packages Survey Questions

1) If interoperability exists but is NOT enabled in the xSDK Spack package, do you have plans to enable it within the xsdk Spack package? If so, what is your time frame for that? If not, are there any issues that prevent it from being enabled within Spack (dependencies, variants, conflicts, etc.)?

2) Do you plan additional interoperability with other xSDK packages? If so, what is the status (E.g., in development, will start work in FY21, planned future work, etc.)?

3) Are there other interoperability needs or new features that would be useful for ECP applications?

4) Are any interoperabilities of your package using another demonstrated in the xsdk-examples suite? If not, could any examples be added to the test suite to demonstrate existing or new interoperabilities (even if they are not built with Spack)?