SAND2015-7037C

Tpetra Project Overview
HPCOR Workshop, September 2015

Mark Hoemmen, Sandia National Laboratories[1]

Introduction

Tpetra, a C++ library that lives in the Trilinos software project, implements parallel linear algebra objects and parallel data (re)distribution. "Linear algebra objects" include sparse graphs and matrices and dense vectors. Many applications and other Trilinos packages use Tpetra's linear algebra objects, or depend on its parallel data redistribution facilities. Tpetra was based on Trilinos' popular Epetra package, and extends it with the following design goals:

1. Permit graphs, matrices, and vectors with huge dimensions and numbers of entries. "Huge" means "not limited by 32-bit integer indices." Let users choose smaller integer types for performance reasons.
2. Support matrices and vectors with entries of many different types, including different precisions, complex arithmetic, and other types for embedded analysis (e.g., automatic differentiation and uncertainty quantification).
3. Enable both distributed- and shared-memory parallelism, in both coarse-grained computational kernels (like sparse matrix-vector multiply) and fine-grained data access by users. Do this in as platform-independent a way as possible.

Development Team

About ten developers have contributed to Tpetra over its lifetime. It has one package lead and full-time core developer, another part-time core developer, and two to three core contributors. The latter develop other Trilinos packages that depend heavily on Tpetra, in particular the algebraic multigrid package MueLu. The largest single concentration of developers is at Sandia National Laboratories in Albuquerque, NM and Livermore, CA. Additional contributors and advanced users are at Oak Ridge National Laboratory and various universities and research laboratories in the United States and Europe. Team members include staff and faculty at these institutions as well as postdocs and graduate students. Tpetra accepts contributions from many other communities. All software is open source and uses Trilinos' distribution mechanism.

Other statistics

1. **Languages:** Tpetra is written entirely in C++, with configuration logic using the open-source TriBITS CMake library (like most other Trilinos packages).
2. **Lines of code:** Tpetra contains about 250k C++ lines and about 4k CMake lines.
3. **Primary methods:** Tpetra provides parallel sparse linear algebra data structures and computational kernels, and parallel data distribution and communication.
4. **Types of problems/domains/science application problems:** Tpetra's capabilities enable linear solvers (iterative and direct), preconditioners, nonlinear solvers, optimization, graph analytics, embedded analysis (automatic differentiation and stochastic PDE discretizations), and many other numerical methods.

[1] Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

5. **Scale of resources commonly used for production runs:** Applications based on Tpetra run on systems from laptops to all of the largest computing systems in the world. Some applications have used Tpetra at full scale on leadership systems (e.g., 512k cores of BlueGene/Q).
6. **Supercomputers regularly used:** All DOE systems, all European supercomputing systems, several Japanese systems. Tpetra comes with Trilinos, which is packaged as part of the Cray LIBSCI product.
7. **Libraries/tools for prototyping:** Tpetra's generic parallel data distribution and redistribution facilities make it easy for developers to move data between different libraries. Projects like CASL's Data Tool Kit (DTK) build on this. Tpetra lets users wrap their own data structures, or wrap a solver in a third-party library; for example, this lets us use PETSc sparse matrices in Trilinos' solvers, and SuperLU resp. Hypre to solve resp. precondition Tpetra linear systems.
8. **Libraries/tools for production science campaigns:** Many production science campaigns built on Trilinos' Epetra data structures. Tpetra offers them a path forward for exploiting maximum parallelism on current and future architectures. Several NNSA integrated codes efforts either already use or plan to start using Tpetra. The Albany finite-element analysis application and the CASL project depend on Tpetra as well.

Performance Portability
Tpetra serves as the basis for applications to begin adopting performance portability. It provides the fundamental scalable data classes and computational kernels on which applications and other Trilinos packages can build. Applications that do not yet use shared-memory parallelism can start with Tpetra's parallel computational kernels, and then use Tpetra's data structures and the Kokkos programming model to expand parallelism into their own code. Tpetra has a history of early adoption of abstractions over different shared-memory parallel programming models, using standard C++ technology. In fact, in 2008 its lead developer at the time came up with an abstraction that later inspired the Kokkos shared-memory parallel programming model. Over the past two years, we have gradually refactored Tpetra to make more use of Kokkos. This also makes it compatible with OpenMP, CUDA, and all back-ends that Kokkos supports.

Exascale challenges
Exascale introduces uncertainty about *distributed*-memory parallel programming models. First, the increased likelihood of component failure may force models that replace the MPI / PGAS idea of a fixed number of parallel processes with permanent unique identifiers, with a "key-value store" that hides data ownership. Tpetra abstracts over MPI itself, but its data distribution abstractions will need revision in this case. Second, Tpetra currently assumes uses an MPI + threads approach where only one thread per MPI process communicates. If running one MPI process per node, this approach does not exploit all the available network bandwidth, but the current MPI_THREAD_MULTIPLE model does not scale. Tpetra will need to track and even contribute to ongoing MPI developer discussions. Tpetra mitigates these challenges by its strong connections to programming model efforts at Sandia National Laboratories in both areas of uncertainty.