Solving Large Eigenvalue problems with Trilinos

Michael O. Atambo

Università degli Studi di Modena e Reggio Emilia,

2016

Table of Contents

The Trilinos Framework

- Framework providing algorithms and other neccessary technologies required for large scale scientific/engineering problems.
- Provides abstractions over distributed and shared memory paradigms, and parallel data objects as well as computational kernels.
- Cutting-edge capabilities in linear algebra, focusing on sparse graphs and matrices.

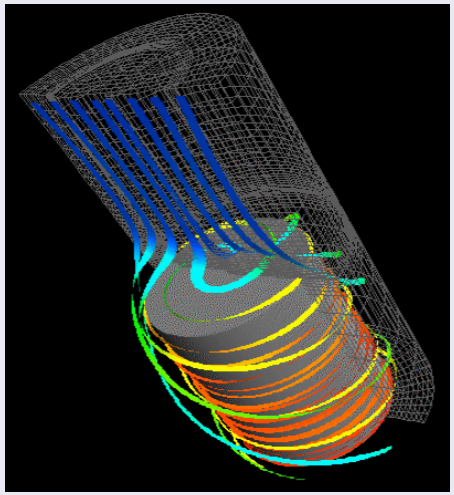
WHY USE IT

- Large library of algorithms, (linear+ non linear solvers, preconditioners).
- Support for multi-core, hybrid CPU/GPU computation, distributed memory.
- Support for >2B unknowns in a problem (sparse matrices with >2B rows...)
- Comes with some batteries included.
- No need to reinvent the wheel

WHERE IS IT APPLIED?

- Condensed matter
- Fluid-flow research
- Structural mechanics

from MPsalsa



I Have a problem: Where do i apply Trilinos?

- Problem setup:

$$y_1'' = -5y_1 + 2y_2$$
$$y_2'' = 2y_1 - 2y_2$$

I Have a problem: Where do i apply Trilinos?

- Problem setup:

$$y_1'' = -5y_1 + 2y_2$$
$$y_2'' = 2y_1 - 2y_2$$

- Variables:

$$y_1, y_2$$

I Have a problem: Where do i apply Trilinos?

- Problem setup:

$$y_1'' = -5y_1 + 2y_2$$
$$y_2'' = 2y_1 - 2y_2$$

- Variables:

$$y_1, y_2$$

- Solution:

$$y'' = \begin{bmatrix} y_1'' \\ y_2'' \end{bmatrix} = A\mathbf{y} = \begin{bmatrix} -5 & 2 \\ 2 & -2 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

$$\vdots \text{some mathematics}$$

I Have a problem: Where do i apply Trilinos?

- Problem setup:

$$y_1'' = -5y_1 + 2y_2$$
$$y_2'' = 2y_1 - 2y_2$$

- Variables:

$$y_1, y_2$$

- Solution:

$$y'' = \begin{bmatrix} y_1'' \\ y_2'' \end{bmatrix} = A\mathbf{y} = \begin{bmatrix} -5 & 2 \\ 2 & -2 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

$$\vdots \text{some mathematics}$$

$$y_1 = a_1 cos(t) + b_1 sin(t) \cdots$$

$$y_2 = a_2 cos(t) + b_2 sin(t) \cdots$$

I Have a problem: Where do i apply Trilinos?

- Problem setup:

$$y_1'' = -5y_1 + 2y_2$$
$$y_2'' = 2y_1 - 2y_2$$

- Variables:

$$y_1, y_2$$

- Solution:

$$y'' = \begin{bmatrix} y_1'' \\ y_2'' \end{bmatrix} = A\mathbf{y} = \begin{bmatrix} -5 & 2 \\ 2 & -2 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

$$\vdots \text{some mathematics}$$

$$y_1 = a_1 cos(t) + b_1 sin(t) \cdots$$

$$y_2 = a_2 cos(t) + b_2 sin(t) \cdots$$

- Intepretation: System is a harmonic oscillator with... $\cdots$

I Have a problem: Where do i apply Trilinos?

- Problem setup:
  Usually unique to your domain of science/engineering
- Solve problem:
  Use a Math Library **TRILINOS COMES IN HERE**
- Interpret solution:
  This will usually be unique to your domain

WHATS IN TRILINOS

- Non-Monolothic framework, composed of multiple packages
- 55 packages as of 12.x
- Packages can be used independently,
- some may have dependencies on other packages.

HOW IS IT USED

- Trilinos is primarily written in C++, using modern (C++11) features
- Extensive use of C++ templates*
- C++ is the normative language of choice
- Python interface through PyTrilinos
- Fortran and C through ForTrilinos and CTrilinos

SELECT PACKAGES IN TRILINOS:

EPETRA

- Serial and parallel linear algebra
- (construction/use of sparse matrices/ dense vectors)
- Data distribution/redistribution facilities
- Provides uniform API to the rest of the trilinos stack

SELECT PACKAGES IN TRILINOS:

TPETRA

- Parallel linear algebra objects like sparse matrices, dense vectors.
- Works with OpenMP, posix threads and CUDA paradigms
- Native support for >2B unknowns
- Templated, allowing support for varied data types, Multiple and mixed precision computing.
- Hybrid parallelism supported (MPI+X)

SELECT PACKAGES IN TRILINOS:

KOKKOS

- Provides shared memory/manycore programming model.
- Abstractions for parallel execution and data management.
- Can use OpenMP, CUDA, Pthreads.
- Allows for performance optimization provided by arch specific memory acccess patterns.
- Linear algebra, mesh/grid libraries, discretization can be build on top of Kokkos.

SELECT PACKAGES IN TRILINOS: Solvers

- iterative linear : AztecOO, Belos..
- Direct sparse linear: Amesos, Amesos2...
- Direct dense linear: Epetra, Teuchos, Pliris
- Iterative eigenvalue: Anasazi..

SELECT PACKAGES IN TRILINOS: Preconditioners

- ILU : Ifpack, ShyLU, AztecOO
- Multilevel: ML, CLAPS..
- Block preconditioners: Meros

SELECT SOLVER PACKAGES:

ANASAZI

- Separates the algorithms from the linear algebra objects.
- Solves Ax=Mx or Ax=MxB
- Supports hermitian/non hermitian/real/complex.
- Uses Kyrolv-shur/Davison/LOBPCG/
- interoperable framework for large scale eigenvalue problems
- provides a generic interface to a collection of algorithms
- Available solvers include: BKS, Block-Davison, LOBPCG, TraceMin family
- Solvers vary in their support for preconditioners, support for complex hermitian/ non hermitian matrices.

SELECT SOLVER PACKAGES:

AMESOS/2

- Set of interfaces to serial and parallel direct solvers
- Has two implementations: KLU(serial) and Paraklete(parallel)
- Provides interface to several widely used solvers: Lapack, UMFPACK, PARDISO,MUMPS,
- Abstracts away from the user details such as the matrix format, data distibution for the matrix, setting parmeters and other direct solver details.
- Above means a uniform interface to several direct solvers.

SELECT SOLVER PACKAGES:

Ifpack2

- provides incomplete factorizations, domain decomposition operators
- Support for Tpetra, and multiple data types using Templating.
- Can be used by other packages: i.e. provide preconditioners to Anasazi solvers.
- Works using Tpetra objects, and thus can take advantage of Tpetras hybrid parallelism

SELECT PACKAGES:

TEUCHOS

- Provides general purpose utilities including:
- Light weight smart pointers
- Array class, similar to C++ STL, but integrated with other Trilinos array types.
- Output support
- Exception handling
- Containers
- Parameter lists
- Performance monitoring

SELECT PACKAGES:

Other packages you will meet

- NOX: non linear solvers.
- LOCA: continuation problems.
- Rythmos: Numeric time integration methods.
- Sacado: Automatic differentiation.

Maps in Trilinos, Implemented in Tpetra.

- Maps define a list of global elements
- Defines the distribution of global elements accross distributed memory
- Defines mapping between global elements and local elements
- Separate local and global indexes
- Local indexes by default 32 bit ints.
- Global indexes usually 64 bit ints.

Notable Features

- Distributed liner algebra objects
- This means trilinos can store your matrix for you on distributed memory.
- Trilinos also handles the communcation and data layout during the computation.
- The framework optimized the data distribution, communication to suit the architecture and solver algorithms.
- These linear algebra objects can be dense vectors or more usefully, sparse matrices.

Other overlapping Math libraries

- PETSc (MPI +OpenMP/CUDA, limited threading.)
- PARALUTION: (partly proprietary).
- Hypre
- Eigen
- LAMA, GHOST, ARPACK,

Comparison to PETSc

- Both support data parallelism (using MPI)
- Use BLAS/LAPACK for dense Linear algebra
- Have own implementations for Sparse LA
- Both have preconditioners and solvers (liner/nonlinera)
- Both support interfaces to 3rd party solvers (MUMPS, UMFPACK...)
- Petsc recommended for non OO design in code.
- Trilinos is well designed OOP code, experience in C++ neccessary.
- Trilinos design ephasizes reusability and extensiblity.
- Strong Emphasis on user experience (Documentation/Tutorials)

BACK TO TRILINOS: CAPABILITY

HOW LARGE IS LARGE FOR TRILINOS?

- Trilinos has been used for solving large systems:
- 2014: 27 billion row matrix (sparse), 500,000 Cores (IBM B/Q).
- Demonstrated weak scalling up to 9 Billion elements.

physics engineering and scientific problems. While the original version of Trilinos was designed for highly scalable solutions for large problems, the need for increasingly higher fidelity simulations has pushed the problem sizes beyond what could have been envisioned two decades ago. When problem sizes exceed a billion elements even highly scalable applications and solver stacks require a complete revision. The next-generation Trilinos employs C++ templates in order to solve arbitrarily large problems and enable extreme-scale simulations. We present a case study that involves integration of Trilinos with an engineering application (Sierra low Mach module/Nalu), involving the simulation of low Mach fluid flow for problems of size up to nine billion elements. Through the use of improved algorithms and better software engineering practices, we demonstrate

that both supports arbitrarily large problem sizes and provides a path forward for good performance on future architectures.

Many challenging scientific and engineering problems at Sandia National Laboratories (SNL) require ever increasing fidelity for computational simulations. One example is the accurate simulation of fire environment, for example, large scale hydrocarbon pool fires that occur in accident scenarios. Figure 1a depicts an example hydrocarbon JP-8 pool fire experiment and Figure 1b depicts a numerical simulation.

SNL uses the SIERRA/Fuego [2] engineering application, built on top of the SIERRA

BACK TO TRILINOS: Actual Code

What does code using trilinos look like?

```
#include "AnasaziTraceMinDavidsonSolMgr.hpp"
#include "AnasaziBasicEigenproblem.hpp"
...
int main(int argc, char *argv[]) {
  // Initialize the MPI session ...
  Teuchos::GlobalMPISession mpiSession(&argc,&argv,&blackhole);
  // Read the matrices from a file
  RCP<const CrsMatrix> K = Reader::readSparseFile(filenameA, comm, node);
  ...
  // Create the eigenproblem
  RCP<Anasazi::BasicEigenproblem<Scalar,MV,OP> > MyProblem =
    rcp (new Anasazi::BasicEigenproblem<Scalar,MV,OP> (K, M, ivec));
  ...
  // Inform the eigenproblem that you are finished passing it information
  bool boolret = MyProblem->setProblem();
  ...
  // Initialize the TraceMin-Davidson solver
  Anasazi::Experimental::TraceMinDavidsonSolMgr<Scalar, MV, OP> MySolverMgr(MyProblem, MyPL);
  // Solve the problem to the specified tolerances
  Anasazi::ReturnType returnCode = MySolverMgr.solve();
  // Get the eigenvalues and eigenvectors from the eigenproblem
  Anasazi::Eigensolution<Scalar,MV> sol = MyProblem->getSolution();
  return 0;
}
```

Research Problem

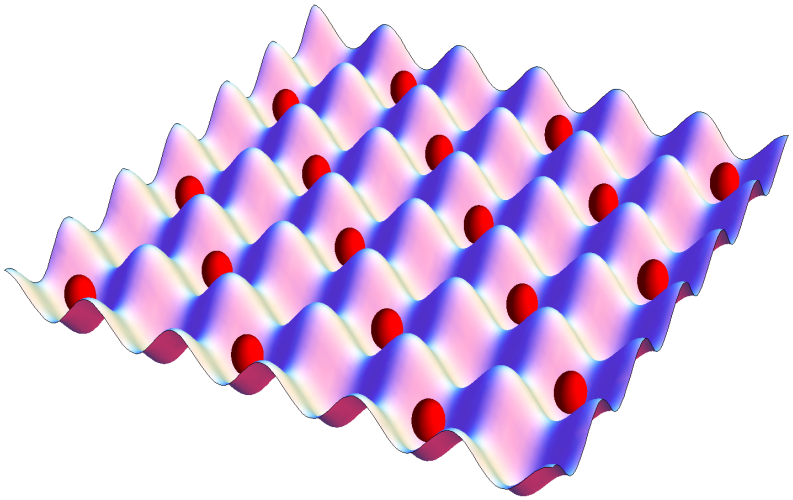
Our use case

- In Condensed matter reasearch: Understanding bosons + possibly application to superconductivity
- Involves modeling non interacting bosons on a lattice.
- Lattice size has to be big enough to reflect physically/scientifically meaning full systems.
-

Research Problem

Our use case: The theory

- Hamiltonian H of hardcore bosons with nearest neighbour interaction V and hopping chemical potentialis set to 0.
- Bosons are on a lattice, and matrix elements $H_{mn} =< n|H|m >$ are computed for each segment.
- Eigenvalues are then computed.



Research Problem

Some of our challenges

- Problem model leads to *very large* matrices
- Matrices for slightly expanded lattice are exponentially larger
- We needed to make efficient use of modern HPC facilities at scientists disposal
- Solution needed to be easily extended by the scientist.

Research Problem

Some of our challenges

- Filling the matrix was costly
- Matrix dimension grew as $\frac{n!}{L!(n-L)!}$
- We needed to use distributed memory at small lattice sizes
- Serial/threaded python and c++ (ARPACK/ARNOLDI) were already insufficient

Research Problem

Final representation

- Problem represented as $\mathbf{Ax} = \lambda\mathbf{Mx}$
- Theory dictated it was hermitian, and positive definite

Research Problem

Solution with trilinos

- With trilinos crsmatrix, we could fill the matrix up in parallel from different mpi process.
- With trilinos crsmatrix objects (sparse matrices in distributed memory) we could hold large matrices (>200GB).
- Trilinos iterative solvers understood these linear algebra objects, interfacing and calling a solver was straighforward
- Very few lines of code! (Important for maintenability, training,...)

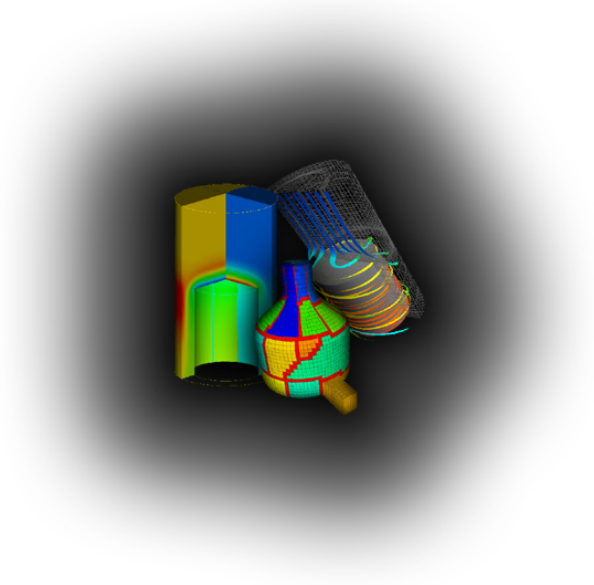
Research Problem

Algorithms provided to us by Trilinos for Eigenvalue problems

- LOBPCG
- Krylov-shur (recommended in the literature, als in hindsight better default choice)
- Davidson

Known Trilinos Known Users

- SALINAS: Finite element analysis
- MPSalsa: fluid flow+ chemical kinetics modelling
- Sundance: finite element solutions to PDE's
- DAKOTA:



Useful resources

- Documentation page on Trilinos.org
- Tutorials/Exercises from user groups(some with videos)
- Hands on training
- Help from the google group (Moderate traffic)

Thank you.