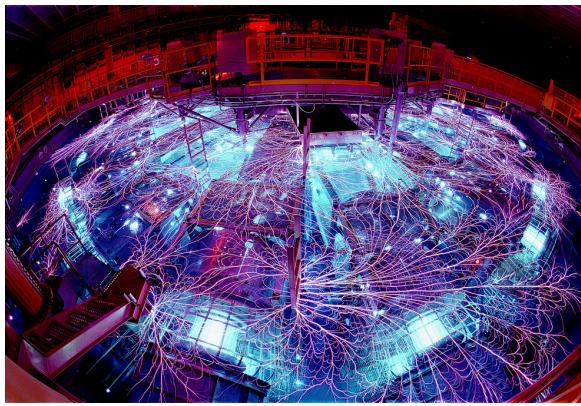
Exceptional service in the national interest

Sandia National Laboratories

SAND2016-3200C





Parallel Algebraic Multigrid Preconditioned Krylov Solvers using Trilinos

Tobias Wiesner

April 4, 2016

U.S. DEPARTMENT OF ENERGY | NNSA National Nuclear Security Administration

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. SAND NO. 2014-00000

Parallel AMG preconditioned Krylov Solvers using Trilinos

Sandia National Laboratories

1) How to set up linear problems using Trilinos?

- Linear Algebra packages in Trilinos
- How to assemble a Tpetra matrix using Kokkos

2) Parallel Algebraic Multigrid kernels

- Algorithmic kernels of Algebraic Multigrid methods

3) How to use AMG preconditioned Krylov solvers from Trilinos

- Overview of Krylov solver packages in Trilinos
- How to use MueLu as AMG preconditioner with Belos

Sandia National Laboratories

How to set up linear problems?

Linear algebra frameworks and parallel assembly of linear systems using Trilinos

Linear Algebra frameworks

Sandia National Laboratories

Epetra

- Standard implementation of Petra framework
- Supports `SC=double` only
- Supports OpenMP (configuration flag)
- Epetra64: `GO=long long`

Tpetra

- New implementation of Petra framework
- Tpetra objects use C++ templates to let you specify `SC` and `GO` on any device
- Uses `Kokkos` underneath

Xpetra

Thin abstraction layer with unified user interface for `E/Tpetra`

Xpetra-sup

Extensions: support for blocked operators, helper routines, ...

Which linear algebra framework should i use?

Sandia National Laboratories

Use Epetra if...

you need a code which only needs `SC=double` and `GO=int` on CPU systems. **Not actively developed any more.**

Use Tpetra if...

you start writing a new code based on the second Trilinos stack which is supposed to run on next generation HPC systems (including GPUs, Xeon Phi, ...).

Use Xpetra if...

- you have an `Epetra` based code that you want to migrate step-by-step to `Tpetra` or
- you want to write your own block preconditioners

Xpetra versus Thyra

Sandia National Laboratories

Common features

- General abstraction layer for `Epetra` or `Tpetra`
- Provides support for blocked operators

Thyra: for application interfaces

- Designed following strict mathematical principles
- Access to different linear solvers through `Stratimikos` (by setting run-time options)
- Rather complicated direct access to matrix/vector data

Xpetra: for algorithms

- Replicates the Tpetra interface for Epetra
- **Easy access to matrix/vector data**

How to assemble a Tpetra matrix using Kokkos

Sandia National Laboratories

1) Local part using Kokkos

- Determine number of unknowns per row in graph/matrix
- Fill `Kokkos::View` objects with CSR data
  - Use local ids for columns

2) Global part using Tpetra

- Generate row and column map (containing global column IDs) for MPI communication
- Construct `Tpetra::CrsMatrix` using row and column map (`Tpetra::Maps`) and local CSR data (`Kokkos::Views`).

Example - 1a) Count entries per row

```
1  // Do a reduction over local elements to count the total number
2  // of (local) entries in the graph.  While doing so, count the
3  // number of (local) entries in each row, using Kokkos' atomic
4  // updates.
5  LO numLclRows = 10000;
6  Kokkos::View<size_t*> rowCounts ("row counts", numLclRows);
7  size_t numLclEntries = 0;
8  Kokkos::parallel_reduce (numLclElements,
9    KOKKOS_LAMBDA (const LO elt, size_t& curNumLclEntries) {
10     const LO lclRows = elt;
11
12     // Always add a diagonal matrix entry.
13     Kokkos::atomic_fetch_add (&rowCounts(lclRows), 1);
14     curNumLclEntries++;
15     ...
16 }, numLclEntries /* reduction result */);
```

- Use the default device defined in `Kokkos`
- We need `Kokkos::atomic_fetch_add` since other threads might also increment row counter at the same time

Example - 1b) Generate row offset data

```
1  // Use a parallel scan (prefix sum) over the array of row
2  // counts, to compute the array of row offsets for the
3  // sparse graph. Use an 'exclusive scan'
4  Kokkos::View<size_t*> rowOffsets ("row offsets",numLclRows+1);
5  Kokkos::parallel_scan (numLclRows+1,
6    KOKKOS_LAMBDA (const LO lclRows, size_t& update, const bool
         final) {
7      if (final) {
8        // Kokkos uses a multipass algorithm to implement scan.
9        // Only update the array on the final pass.  Updating
10       // the array before changing 'update' means that we do
11       // an exclusive scan. Update the array after for an
12       // inclusive scan.
13       rowOffsets[lclRows] = update;
14     }
15     if (lclRows < numLclRows) {
16       update += rowCounts(lclRows);
17     }
18   });
```

Example - 1c) Fill CSR data

```
1  // zero out row counter
2  Kokkos::deep_copy (rowCounts, static_cast<size_t> (0));
3  Kokkos::View<LO*> colIndices ("column indices", numLclEntries);
4  Kokkos::View<double*> matrixValues ("matrix values",
       numLclEntries);
5
6  // Iterate over elements in parallel to fill the matrix
7  Kokkos::parallel_for (numLclElements,
8    KOKKOS_LAMBDA (const LO elt) {
9      const int lclRows = elt;
10
11     // Always add a diagonal matrix entry.
12     const size_t count =
13       Kokkos::atomic_fetch_add (&rowCounts(lclRows), 1);
14     colIndices(rowOffsets(lclRows) + count) = lclRows;
15     Kokkos::atomic_fetch_add (&matrixValues(rowOffsets(lclRows)
           + count),2.0);
16     ...
17   });
```

Example - 2a) Create global maps for MPI communication

Sandia National Laboratories

Tpetra objects

To generate `Tpetra` objects using local data (e.g. a local `Kokkos::Graph`), we have to provide the corresponding global communication pattern in form of `Tpetra::Map` objects.

For a `Tpetra::CrsMatrix` we need

- a (non-overlapping) row map
- a (overlapping) column map

Example - 2a) Create global maps for MPI communication

Sandia National Laboratories

Generate a global non-overlapping row map

```
1  // Wrap the "raw" MPI communicator for use in Tpetra.
2  RCP<const Teuchos::Comm<int> > comm =
3    rcp (new Teuchos::MpiComm<int> (MPI_COMM_WORLD));
4
5  const GO indexBase = 0;
6  RCP<const Tpetra::Map<> > rowMap =
7    rcp (new Tpetra::Map<> (numGblElements, numLclElements,
         indexBase, comm));
```

Use `Tpetra::Map` with default template parameters

Example - 2b) Create global maps for MPI communication

Sandia National Laboratories

Generate a global overlapping column map

```
1  // Create column map (using global column indices)
2  const LO num_col_inds = numLclElements;
3  Kokkos::View<GO*> colInds ("Col Map", num_col_inds);
4  auto lclRowMap = rowMap->getLocalMap ();
5  Kokkos::parallel_for (num_col_inds, KOKKOS_LAMBDA (const LO k) {
6      colInds(k) = lclRowMap.getGlobalElement (k);
7    });
8    ...
9
10 // Flag to tell Tpetra::Map to compute the global number of
11 // indices in the Map.
12 const Tpetra::global_size_t INV =
13   Teuchos::OrdinalTraits<Tpetra::global_size_t>::invalid ();
14 // Soon, it will be acceptable to pass in a Kokkos::View here,
15 // instead of a Teuchos::ArrayView.
16 RCP<const Tpetra::Map<> > colMap =
17   rcp (new Tpetra::Map<> (INV, Kokkos::Compat::getConstArrayView
         (colInds), indexBase, comm));
```

Example - 2c) Construct `Tpetra::CrsMatrix`

Create the `Tpetra::CrsMatrix` providing the row and column map information and the local CSR data

```
1 Tpetra::CrsMatrix<> A (rowMap, colMap,
2   rowOffsets, colIndices, matrixValues);
3 A.fillComplete ();
```

The global matrix `A` is ready to use (e.g. for solvers)

Example - 2c) Construct `Tpetra::CrsMatrix`

Create the `Tpetra::CrsMatrix` providing the row and column map information and the local CSR data

```
1 Tpetra::CrsMatrix<> A (rowMap, colMap,
2   rowOffsets, colIndices, matrixValues);
3 A.fillComplete ();
```

The global matrix `A` is ready to use (e.g. for solvers)

More information

The detailed example can be found here:
`packages/tpetra/core/examples/Lesson07-Kokkos-Fill`
A similar example can be found here:
`packages/xpetra/test/CrsMatrix/CrsMatrix_UnitTests.cpp`

Sandia National Laboratories

Parallel Algebraic Multigrid

Algorithmic kernels of Algebraic Multigrid Methods

Parallel Algebraic Multigrid

Sandia National Laboratories

Definition: Parallel (computer science)

Distribute problem over machine

$$\textbf{Parallel} = \textbf{MPI} + \textbf{X}$$

Parallel Algebraic Multigrid

Sandia National Laboratories

Definition: Parallel (computer science)

Distribute problem over machine

$$\textbf{Parallel} = \textbf{Tpetra} + \textbf{Kokkos}$$

Parallel Algebraic Multigrid

Sandia National Laboratories

Definition: Parallel (computer science)

Distribute problem over machine

$$\textbf{Parallel} = \textbf{Tpetra} + \textbf{Kokkos}$$

Tpetra: Tpetra handles MPI communication and inter-node parallelism

Kokkos: Kokkos handles on-node parallelism

Parallel Algebraic Multigrid

Sandia National Laboratories

Definition: Parallel (computer science)

Distribute problem over machine

$$\textbf{Parallel} = \textbf{Tpetra} + \textbf{Kokkos}$$

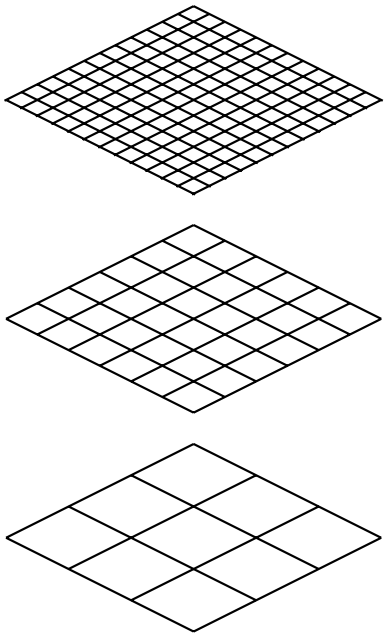
Tpetra: Tpetra handles MPI communication and inter-node parallelism

Kokkos: Kokkos handles on-node parallelism

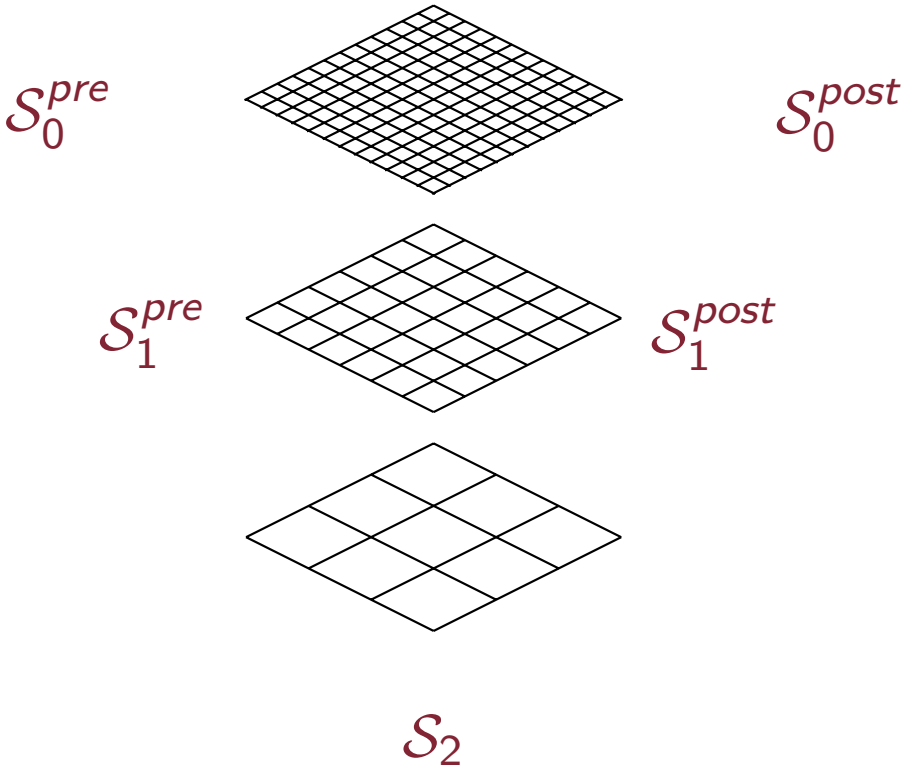
Definition: Parallel (algorithms)

Parallel = Solve problems simultaneously instead of consecutively.

Parallel *Algebraic Multigrid*



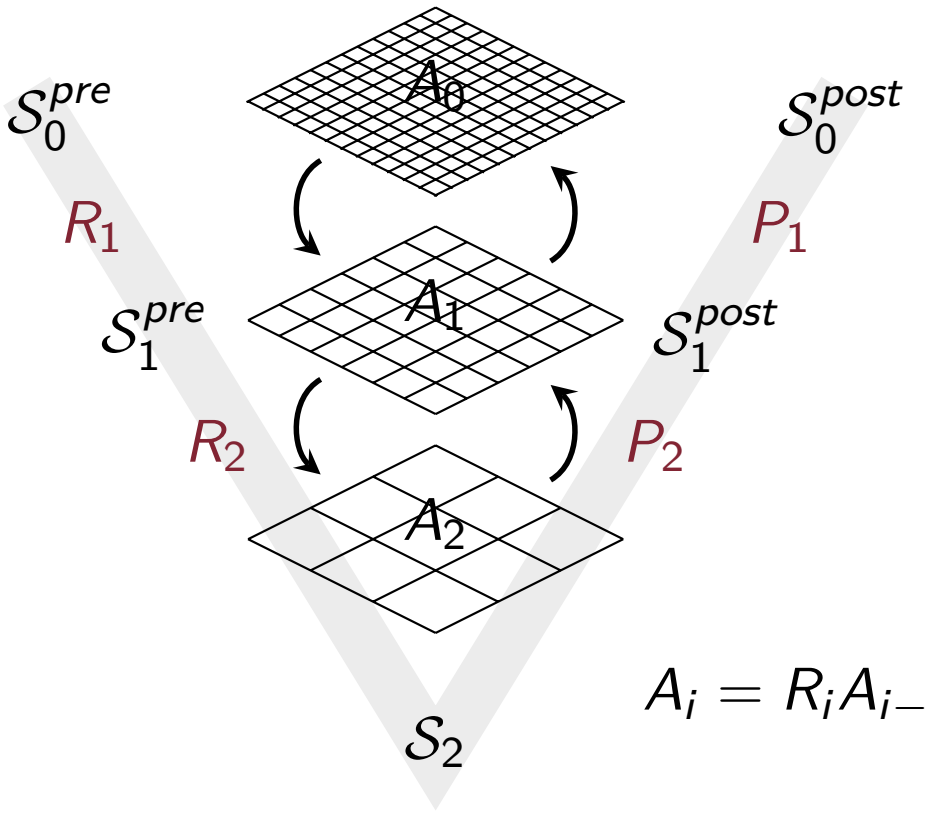
Parallel *Algebraic Multigrid*



Two main components

- Smoothers
  - "Cheap" reduction of oscillatory error (high energy)
  - $\mathcal{S}_L \approx A_L^{-1}$ on the coarsest level L

Parallel *Algebraic Multigrid*



$$A_i = R_i A_{i-1} P_i$$

Two main components

- Smoothers
  - "Cheap" reduction of oscillatory error (high energy)
  - $\mathcal{S}_L \approx A_L^{-1}$ on the coarsest level L
- Grid transfers (prolongators and restrictors)
  - Data movement between levels
  - Definition of coarse level matrices.

Parallel Algebraic Multigrid - Algorithmic kernels

Sandia National Laboratories

Smoothers

- Matrix-Vector products
- Polynomial evaluation
- Matrix operations (ILU)
- Residual evaluation

Transfers

- Coarsening/aggregation
- Matrix operations
  - Matrix Matrix (MM) multiplication
  - Scaling operations

Galerkin product

- Matrix Matrix Matrix (MMM) product
  - Two Matrix Matrix (MM) multiplications
- Scaling operations

Linear Algebra kernels for multigrid

- Matrix-Vector multiplication
- Vector operations (norms,...)
- Matrix-Matrix multiplication

Sandia National Laboratories

AMG preconditioned Krylov solvers

How to use AMG preconditioned Krylov solvers from Trilinos

Krylov solvers in Trilinos

Sandia National Laboratories

AztecOO:

- Object-oriented interface for Aztec 2.1 package
- Support for `Epetra` only
- Contains CG, GMRES, BiCGstab, TFQMR

Belos:

- Modern C++ implementation of Krylov solvers
- Native support for `{E,T,X}petra` through traits and specializations
- Contains CG, GMRES, BiCGstab, TFQMR
- Actively supported

AMG packages in Trilinos

Sandia National Laboratories

ML:

- Smoothed aggregation AMG in Trilinos
- Support for `Epetra` only
- Not actively developed any more

MueLu:

- Modern C++ multigrid framework in Trilinos
- Native support for `{E,T}petra` through `Xpetra`
- Runs on next generation HPC systems (MPI+X)

MueLu as a preconditioner in Belos

1) Prepare input

```
1 // Create A, B, X ...
2 Teuchos::RCP<Tpetra::CrsMatrix<> >    A = ...;
3 Teuchos::RCP<Tpetra::MultiVector<> > B = ...;
4 Teuchos::RCP<Tpetra::MultiVector<> > X = ...;
```

The multi vector `B` contains the RHS vector(s). The multi vector `X` contains the initial guess.

2) Construct preconditioner

3) Construct problem

4) Set solver parameters

5) Solve problem

MueLu as a preconditioner in Belos

Sandia National Laboratories

1) Prepare input

2) **Construct preconditioner**

```
1 std::string optionsFile = "mueluOptions.xml";
2 Teuchos::RCP<MueLu::TpetraOperator>
      mueLuPreconditioner =
3 MueLu::CreateTpetraPreconditioner(A, optionsFile);
```

The XML file contains the MueLu multigrid parameters (see MueLu user guide for all options).

3) Construct problem

4) Set solver parameters

5) Solve problem

MueLu as a preconditioner in Belos

1) Prepare input
2) Construct preconditioner
3) **Construct problem**

Put all information together for Belos.

```
1 Belos::LinearProblem<> problem(A, X, B);
2 problem->setLeftPrec(mueLuPreconditioner);
3 bool set = problem.setProblem();
```

Belos supports left and/or right-preconditioning.

4) Set solver parameters
5) Solve problem

MueLu as a preconditioner in Belos

1) Prepare input
2) Construct preconditioner
3) Construct problem
4) **Set solver parameters**

Set the `Belos` parameters. Refer to the `Belos` documentation for all available options.

```
1 Teuchos::ParameterList belosList;
2 belosList.set("Maximum Iterations", 100);
3 belosList.set("Convergence Tolerance", 1e-7);
```

5) Solve problem

MueLu as a preconditioner in Belos

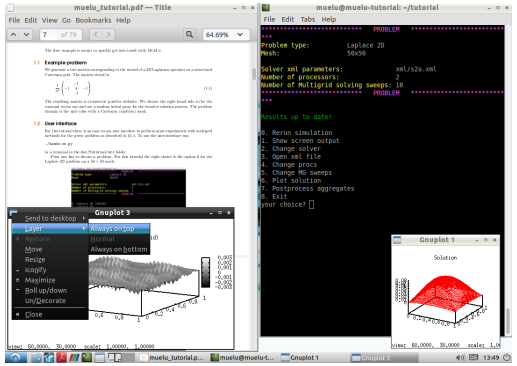
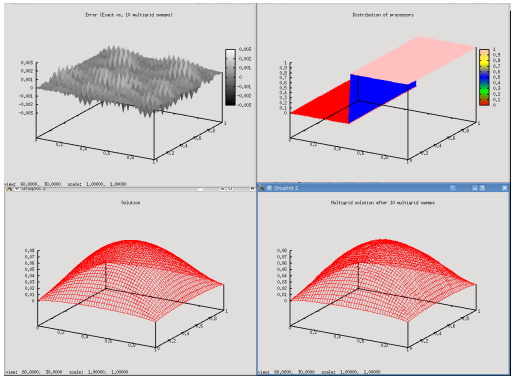
1) Prepare input
2) Construct preconditioner
3) Construct problem
4) Set solver parameters
5) **Solve problem**

```
1 Belos::BlockCGSolMgr<> solver(rcp(&problem,false),
      rcp(&belosList,false));
2 Belos::ReturnType ret = solver.solve();
```

Variable `X` contains the solution vector.

MueLu Documentation

- **User's Guide** (`packages/muelu/doc/UsersGuide`)
  - Geared towards new users
  - Complete list of user options (new options are caught automatically)
- **Tutorial:** Download:
  `trilinos.org/packages/muelu/muelu-tutorial`

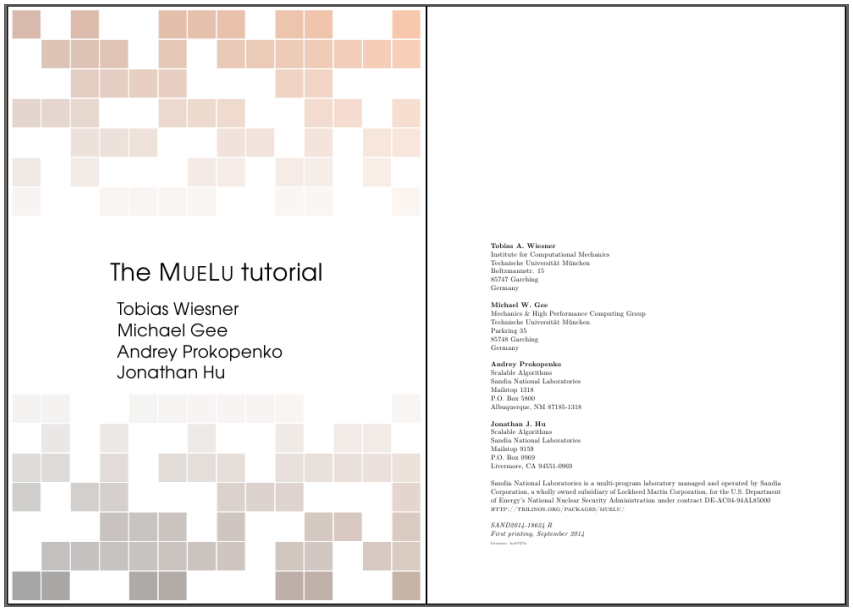



- **Examples and tests** (`packages/muelu/{examples,tests}`)
- **Doxygen**

  Best used as reference for developers

MueLu Tutorial and virtual machine

- PDF guide along with interactive Python script
- Provides a step-by-step tutorial for new MueLu users with practical examples
- Easy to try multigrid methods
- Comes with a VirtualBox image, no Trilinos compilation



The MueLu tutorial, SAND2014-18624 R