MueLu: The Trilinos Multigrid Framework

Andrey Prokopenko
Jonathan Hu

Sandia National Labs

October 29, 2014

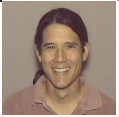
SAND2014-19281 PE

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. 2011-XXXP

MueLu

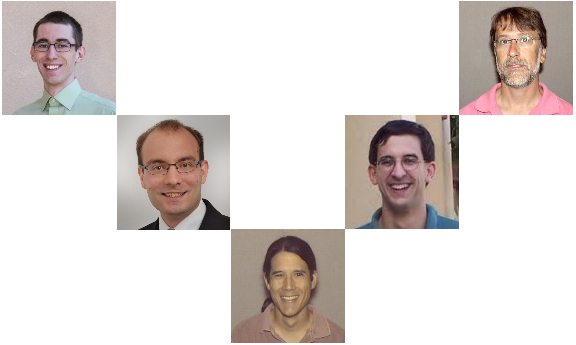
- Team
  - Andrey Prokopenko (SNL)
  - Tobias Wiesner (TUM)
  - Jonathan Hu (SNL)
  - Chris Siefert (SNL)
  - Ray Tuminaro (SNL)
  - Paul Tsuji (SNL)





MueLu

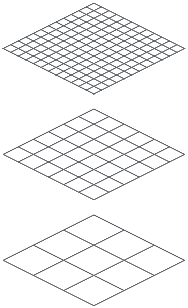
- Team
  - Andrey Prokopenko (SNL)
  - Tobias Wiesner (TUM)
  - Jonathan Hu (SNL)
  - Chris Siefert (SNL)
  - Ray Tuminaro (SNL)
  - Paul Tsuji (SNL)



- First public release

  Trilinos 11.12, October 2014

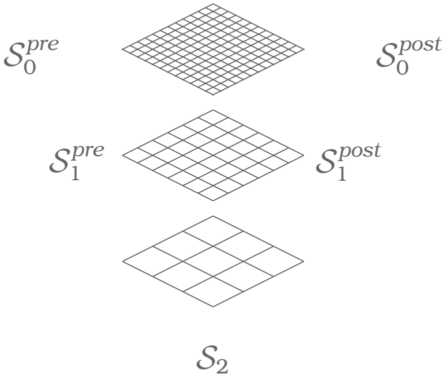
Algebraic Multigrid (AMG)



Main idea

Capture errors at multiple resolutions.

Algebraic Multigrid (AMG)



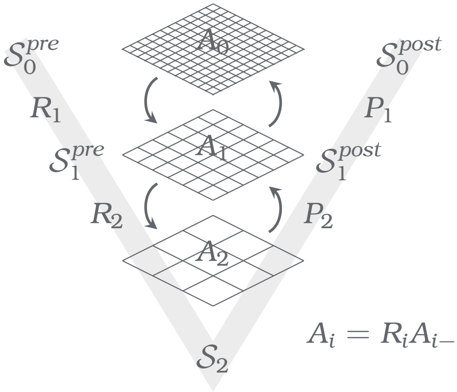
Two main components

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

Main idea

Capture errors at multiple resolutions.

Algebraic Multigrid (AMG)



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

Two main components

- Smoothers
  - Approximate solve on each level
  - ``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
  - Reduction of smooth error (low energy)

Main idea

Capture errors at multiple resolutions.

Capabilities

- Can use either Epetra (32-bit) or Tpetra

  Template types: Local and global indices, scalar, compute node
- Grid transfers
  - Smoothed and unsmoothed aggregation
  - Petrov-Galerkin
  - Energy minimization
  - Maxwell
- Smoothers (Ifpack/Ifpack2)
  - Relaxation: Jacobi, SOR, $l1$ Gauss-Seidel
  - Incomplete factorizations: ILU(k), ILUT, ILUTP*
  - Others: Chebyshev, additive Schwarz, Krylov, Vanka, . . .
- Direct solvers (Amesos/Amesos2)

  KLU2, SuperLU, . . .
- Load balancing (Zoltan/Zoltan2)

  RCB, multijagged (Zoltan2 only)

MueLu/ML Feature Comparison

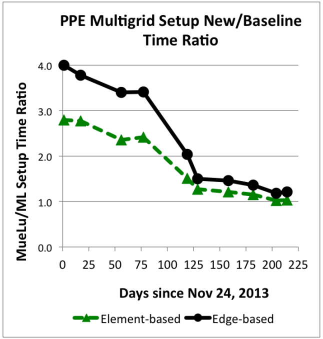
Similarities

- Algorithmic capabilities
- Performance (with some caveats)
- Simple application interfaces
- Simple input decks

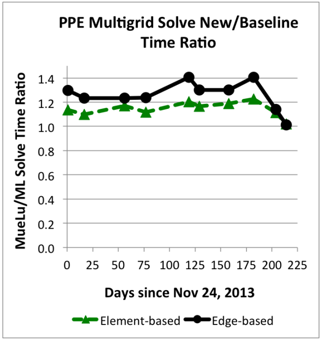
Differences

- MueLu can solve problems with > 2.1b DOFs
- MueLu can use Kokkos (MPI+X)
- MueLu has much stronger unit testing than ML
- ML has a better scaling SPGEMM (slower in serial)

Relative setup performance

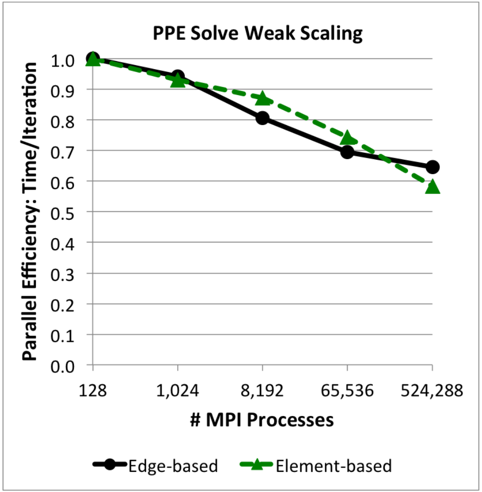


Relative solve performance

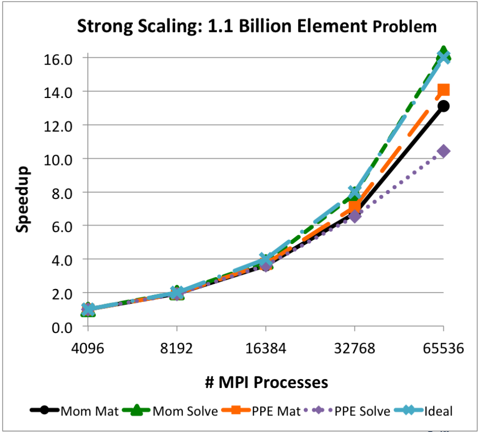


Some Performance Results

Weak scalability of GMRES/SA-AMG pressure solve on BG/Q



Strong scalability of GMRES/SA-AMG pressure solve on BG/Q



User interfaces

- Natural parameter lists (**recommended**)
  - Suitable for beginners and experts
  - Support most common use-cases
  - Provide a reasonable subset of all MueLu parameters
  - Fully validated
- Hierarchical parameter lists
  - Suitable for experts
  - Reflect module dependencies in MueLu
- ML-style parameter lists
  - Oriented toward former ML users
  - Strive to provide some backwards compability with ML
  - But: MueLu and ML have different defaults
- C++ API
- Through Stratimikos

Natural interface

```xml
<ParameterList name="MueLu">
  <Parameter name="verbosity" type="string" value="high"/>
  <Parameter name="max levels" type="int" value="10"/>
  <Parameter name="coarse: max size" type="int" value="2000"/>
</ParameterList>
```

- Uses reasonable defaults
- Generates smoothed aggregation AMG

Natural interface

```
<ParameterList name="MueLu">
  <Parameter name="verbosity" type="string" value="high"/>
  <Parameter name="max levels" type="int" value="10"/>
  <Parameter name="coarse: max size" type="int" value="2000"/>
  <Parameter name="multigrid algorithm" type="string"
    value="unsmoothed"/>
</ParameterList>
```

- Generates unsmoothed aggregation AMG

Natural interface

Sandia National Laboratories

```
<ParameterList name="MueLu">
  <Parameter name="verbosity" type="string" value="high"/>
  <Parameter name="max levels" type="int" value="10"/>
  <Parameter name="coarse: max size" type="int" value="2000"/>
  <Parameter name="multigrid algorithm" type="string"
    value="unsmoothed"/>
  <Parameter name="smoother: type" type="string"
    value="CHEBYSHEV"/>
  <ParameterList name="smoother: params">
    <Parameter name="chebyshev: degree" type="int" value="3"/>
  </ParameterList>
</ParameterList>
```

- Generates unsmoothed aggregation AMG
- Use third degree polynomial smoother

Natural interface

Sandia National Laboratories

```xml
<ParameterList name="MueLu">
  <Parameter name="verbosity" type="string" value="high"/>
  <Parameter name="max levels" type="int" value="10"/>
  <Parameter name="coarse: max size" type="int" value="2000"/>
  <Parameter name="multigrid algorithm" type="string"
    value="unsmoothed"/>
  <ParameterList name="level 2">
    <Parameter name="smoother: type" type="string"
      value="CHEBYSHEV"/>
    <ParameterList name="smoother: params">
      <Parameter name="chebyshev: degree" type="int" value="3"/>
    </ParameterList>
  </ParameterList>
</ParameterList>
```

- Generates unsmoothed aggregation AMG
- Use third degree polynomial smoother on level 2
- Use default smoother (symmetric Gauss-Seidel) for all other levels

MueLu's master list

Single place for all MueLu parameters.

```
<parameter>
  <name>smoother: type</name>
  <type>string</type>
  <default>"RELAXATION"</default>
  <Poisson>"CHEBYSHEV"</Poisson>
  <description>Smoother type</description>
  <visible>true</visible>
</parameter>
```

XSL transformations to

- ParameterList

  Used internally in MueLu
- LaTeX

  Used in User's Manual
- HTML

  Used for website

MueLu as a preconditioner in Belos

```
// Create A, B, X ...
Teuchos::RCP<Tpetra::CrsMatrix<> > A;
Teuchos::RCP<Tpetra::MultiVector<> > B, X;
```

MueLu as a preconditioner in Belos

```
// Create A, B, X ...
Teuchos::RCP<Tpetra::CrsMatrix<> > A;
Teuchos::RCP<Tpetra::MultiVector<> > B, X;
// Construct preconditioner
std::string optionsFile = "mueluOptions.xml";
Teuchos::RCP<MueLu::TpetraOperator> mueLuPreconditioner =
  MueLu::CreateTpetraPreconditioner(A, optionsFile);
```

MueLu as a preconditioner in Belos

```
// Create A, B, X ...
Teuchos::RCP<Tpetra::CrsMatrix<> > A;
Teuchos::RCP<Tpetra::MultiVector<> > B, X;
// Construct preconditioner
std::string optionsFile = "mueluOptions.xml";
Teuchos::RCP<MueLu::TpetraOperator> mueLuPreconditioner =
  MueLu::CreateTpetraPreconditioner(A, optionsFile);
// Construct problem
Belos::LinearProblem<> problem(A, X, B);
problem->setLeftPrec(mueLuPreconditioner);
bool set = problem.setProblem();
```

MueLu as a preconditioner in Belos

```
// Create A, B, X ...
Teuchos::RCP<Tpetra::CrsMatrix<> > A;
Teuchos::RCP<Tpetra::MultiVector<> > B, X;
// Construct preconditioner
std::string optionsFile = "mueluOptions.xml";
Teuchos::RCP<MueLu::TpetraOperator> mueLuPreconditioner =
  MueLu::CreateTpetraPreconditioner(A, optionsFile);
// Construct problem
Belos::LinearProblem<> problem(A, X, B);
problem->setLeftPrec(mueLuPreconditioner);
bool set = problem.setProblem();
// Set Belos parameters
Teuchos::ParameterList belosList;
belosList.set("Maximum Iterations", 100);
```

MueLu as a preconditioner in Belos

```
// Create A, B, X ...
Teuchos::RCP<Tpetra::CrsMatrix<> > A;
Teuchos::RCP<Tpetra::MultiVector<> > B, X;
// Construct preconditioner
std::string optionsFile = "mueluOptions.xml";
Teuchos::RCP<MueLu::TpetraOperator> mueLuPreconditioner =
  MueLu::CreateTpetraPreconditioner(A, optionsFile);
// Construct problem
Belos::LinearProblem<> problem(A, X, B);
problem->setLeftPrec(mueLuPreconditioner);
bool set = problem.setProblem();
// Set Belos parameters
Teuchos::ParameterList belosList;
belosList.set("Maximum Iterations", 100);
// Solve the problem
Belos::BlockCGSolMgr<> solver(rcp(&problem,false), rcp(&
    belosList,false));
Belos::ReturnType ret = solver.solve();
```

Documentation

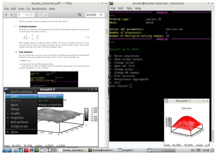
- User's Guide `(packages/muelu/doc/UsersGuide)`
  - Geared towards new users
  - Complete list of user options (new options are caught automatically)
- Tutorial `(packages/muelu/doc/Tutorial)`
- Examples and tests `(packages/muelu/{examples,tests})`
- Mailing lists

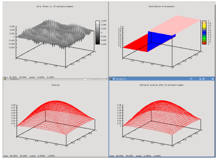
  {muelu-users,muelu-developers}@software.sandia.gov
- Doxygen

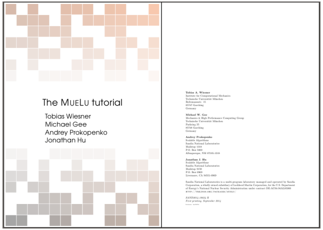
  Best used as reference

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
Tobias Wiesner
Michael Gee
Andrey Prokopenko
Jonathan Hu
SAND2014-18624 R

Future Plans

- Support for 64-bit EPETRA
- Incorporation of KOKKOS kernels directly into MUELU
- Setup cost reduction, improved solver robustness
- Reuse of algorithmic components between solves
- New algorithms research for
  - higher order methods
  - semistructured problems (i.e., extruded meshes)
  - UQ ensembles

Backup slides

C++ interface

Sandia National Laboratories

```
Hierarchy H(fineA);    // generate hierarchy using fine level
                       // matrix

H.Setup();             // call multigrid setup (create hierarchy)

H.Iterate(B, nIts, X); // perform nIts iterations with multigrid
                       // algorithm (V-Cycle)
```

- Uses reasonable defaults
- Generates smoothed aggregation AMG

C++ interface

Sandia National Laboratories

```
Hierarchy H(fineA);     // generate hierarchy using fine level
                        // matrix
RCP<TentativePFactory> PFact = rcp(new TentativePFactory());
FactoryManager M;           // construct factory manager
M.SetFactory("P", PFact);   // define tentative prolongator
                            // factory as default factory for
                            // generating P
H.Setup(M);             // call multigrid setup (create hierarchy)

H.Iterate(B, nIts, X);// perform nIts iterations with multigrid
                        // algorithm (V-Cycle)
```

- Generates unsmoothed aggregation AMG

C++ interface

Sandia National Laboratories

```cpp
Hierarchy H(fineA);     // generate hierarchy using fine level
                        // matrix
Teuchos::ParameterList smootherParams;
smootherParams.set("chebyshev: degree", 3);
RCP<SmootherPrototype> smooProto =
  rcp(new TrilinosSmoother("CHEBYSHEV", smootherParams);
RCP<SmootherFactory>   smooFact  =
  rcp(new SmootherFactory(smooProto));
FactoryManager M;
M.SetFactory("Smoother", smooFact);

H.Setup(M);             // call multigrid setup (create hierarchy)

H.Iterate(B, nIts, X);// perform nIts iterations with multigrid
                        // algorithm (V-Cycle)
```

- Generates smoothed aggregation AMG
- Use third degree polynomial smoother

References

1. P. Lin, M. Bettencourt, S. Domino, T. Fisher, M. Hoemmen, J.J. Hu, E. Phipps, A. Prokopenko, S. Rajamanickam, C. Siefert, S. Kennon, *Towards extreme-scale simulations for low Mach fluids with second- generation Trilinos* (to appear)
2. L. Olson, J. Schroder, and R.S. Tuminaro, *A general interpolation strategy for algebraic multi- grid using energy minimization* SIAM Journal on ScientiïňĄc Computing, 33(2):966âĂŞ991, 2011.
3. M. Sala and R.S. Tuminaro, *A new Petrov-Galerkin smoothed aggregation preconditioner for nonsymmetric linear systems* SIAM Journal on ScientiïňĄc Computing, 31(1):143âĂŞ166, 2008.
4. T.A. Wiesner, *Flexible aggregation-based algebraic multigrid methods for contact and flow problems*, PhD thesis, 2014.
5. T.A. Wiesner, M.W. Gee, A. Prokopenko, and J.J. Hu, *The MueLu tutorial*, http://trilinos.org/packages/muelu/muelu-tutorial, 2014. SAND2014- 18624R.
6. T.A. Wiesner, R.S. Tuminaro, W.A. Wall and M.W. Gee, *Multigrid transfers for nonsymmetric systems based on Schur complements and Galerkin projections*, Numer. Linear Algebra Appl., 2013
7. A. Popp, Ph. Farah, T.A. Wiesner, W.A. Wall, *Efficient parallel solution methods for mortar finite element discretizations in computational contact mechanics*, 11th World Congress on Computational Mechanics (WCCM XI), Barcelona/Spain, 2014
8. T.A. Wiesner, A. Popp, M.W. Gee, W.A. Wall, *Aggregation based algebraic multigrid methods for mortar methods in contact mechanics*, (in preparation)