Introduction to MueLu

The Trilinos Multigrid Framework

Tobias Wiesner
Andrey Prokopenko
Jonathan Hu

Sandia National Labs

March 2, 2015

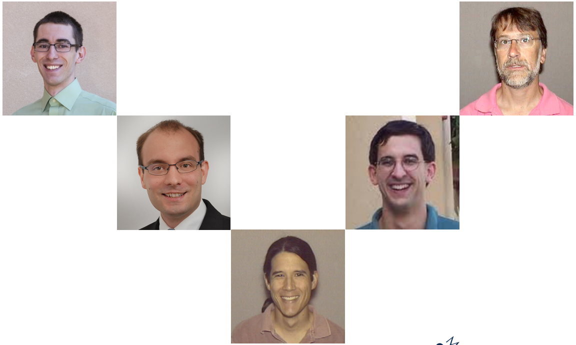
Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. SAND NO. 2011-XXXP

MueLu

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

  Jeremie Gaidamour

  (SNL: 2010-2013, CNRS: 2013-2014, 2014-now: Inria)

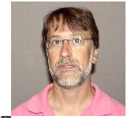


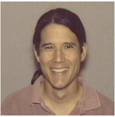
MueLu

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

  Jeremie Gaidamour

  (SNL: 2010-2013, CNRS: 2013-2014, 2014-now: Inria)
- MueLu provides a flexible and extensible fully object-oriented framework for designing application-specific AMG preconditioners

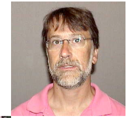


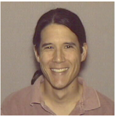


MueLu

- Team
  - Andrey Prokopenko (SNL)
  - Tobias Wiesner (TUM)
  - Jonathan Hu (SNL)
  - Chris Siefert (SNL)
  - Ray Tuminaro (SNL)
  - Paul Tsuji (SNL)
- Former team members:
  Jeremie Gaidamour

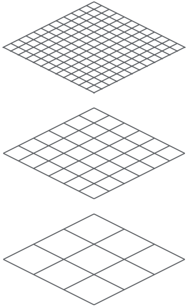
  (SNL: 2010-2013, CNRS: 2013-2014, 2014-now: Inria)
- MueLu provides a flexible and extensible fully object-oriented framework for designing application-specific AMG preconditioners
- First public release
  Trilinos 11.12, October 2014





Algebraic Multigrid Methods

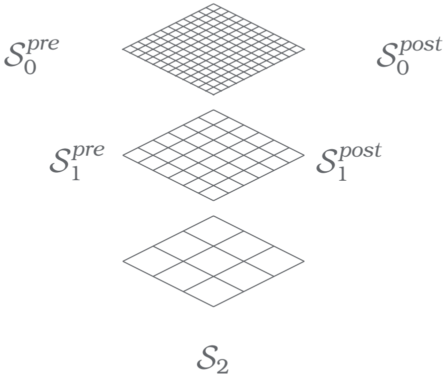
Algebraic Multigrid (AMG)



Main idea

Capture errors at multiple resolutions.

Algebraic Multigrid (AMG)



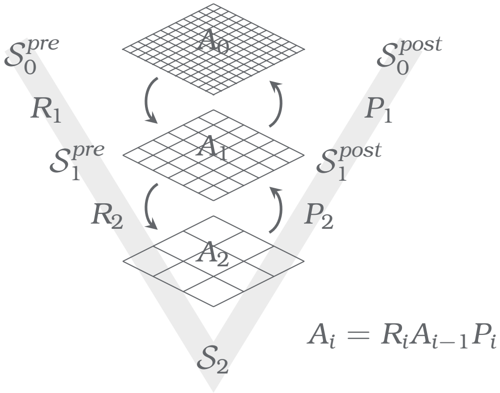
Two main components

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

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)
  - $S_L \approx A_L^{-1}$ on the level L
- Level transfers
  - Data movement between levels
  - Reduction of smooth error (low energy)

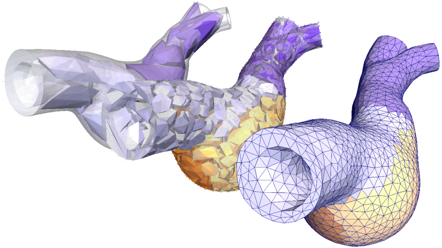
Main idea

Capture errors at multiple resolutions.

Algebraic Multigrid Methods

Sandia National Laboratories

- build a multigrid hierarchy using the fine level matrix information
  - ⇒ ideal for complicated geometries and unstructured meshes
  - ⇒ the user does not have to create coarse meshes
- known for efficiency and optimal scaling properties for certain problem classes.
- no black-box methods!



Level	rows	nnz	aggs	procs
0	21237	834405	--	4
1	2154	373338	359	2
2	132	13176	22	1

Requirements for modern multigrid code

- **Flexibility:** Need for highly flexible problem-specific preconditioners
- **Performance:** Latest developments in hard- and software
- **Usability:** Reasonable results also for non-expert users

MueLu - the new multigrid package in Trilinos

MueLu at glance

- Integration with Trilinos library
- Modern object-oriented software architecture

  Written completely in C++ as a modular object-oriented multigrid framework
- Open source

  Available through a simplified BSD license
- Easy-to-use interface

  User-friend parameter input deck
- Extensibility

  Experienced users have full access to the underlying framework through an advanced XML based interface
- Broad range of supported platforms

  MueLu runs on wide variety of architectures, from desktop workstations to parallel Linux clusters and supercomputers

Capabilities

- Can use either Epetra or Tpetra

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

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

  RCB, multijagged (Zoltan2 only)

MueLu- The next-generation Multigrid Framework

MueLu can be interesting for

Mathematicians: due to

- its **modularity** and **flexibility**
- optimal for **research** on **new multigrid concepts**

Computer scientists: due to

- its **advanced software architecture**
- targeting **extremely large problems** (HPC)
- support for latest hardware (CPU, GPU, threads)

Engineers:

- applicability to **real world problems**
- **problem-specific adaptions** with minimal effort

MueLu/ML Feature Comparison

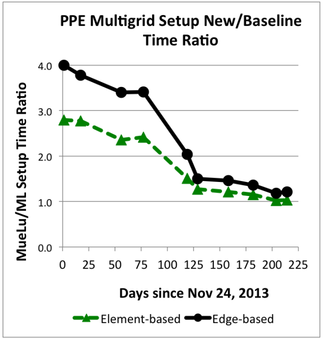
Similarities

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

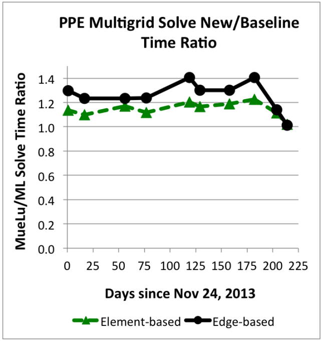
Differences

- 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

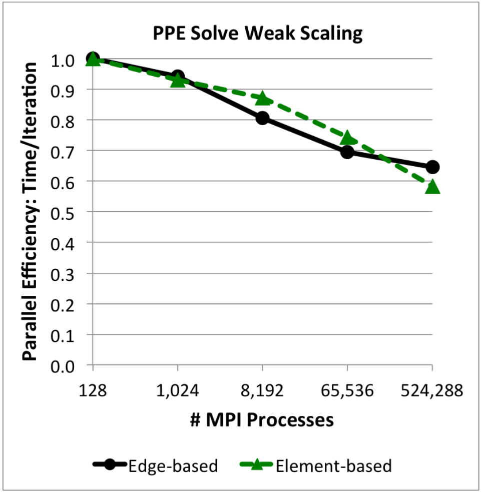


Results provided by

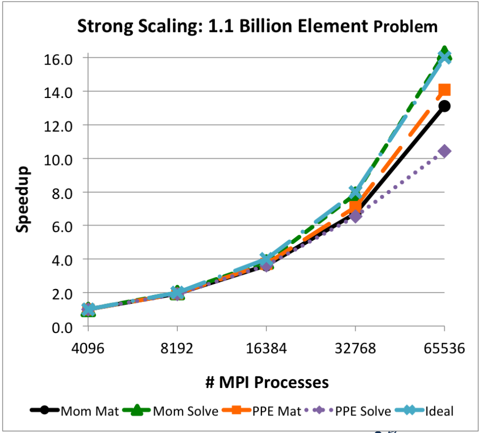
Paul Lin

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



Results provided by

Paul Lin

Usage of MueLu

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

```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>
```

- Generates unsmoothed 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"/>
  <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

```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

```cpp
// 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

```cpp
// 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

```cpp
// 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

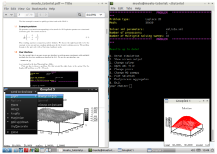
  Best used as reference

Pre-compiled users's guide and tutorial

```
http://wiesner.userweb.mwn.de/sandia/muelututorial.pdf
http://wiesner.userweb.mwn.de/sandia/mueluguide.pdf
```

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



muelu@muelu-tutorial: ~/tutorial

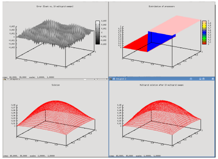
File Edit Tabs Help

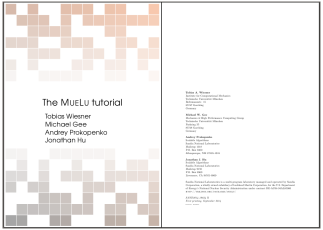
```
**************************   PROBLEM   **************************
***
Problem type:          Laplace 2D
Mesh:                  50x50

Solver xml parameters:              xml/s2a.xml
Number of processors:               2
Number of Multigrid solving sweeps: 10
**************************   PROBLEM   **************************
***

Results up to date!

0. Rerun simulation
1. Show screen output
2. Change solver
3. Open xml file
4. Change procs
5. Change MG sweeps
6. Plot solution
7. Postprocess aggregates
8. Exit
your choice? 
```





The MueLu tutorial

Tobias Wiesner
Michael Gee
Andrey Prokopenko
Jonathan Hu

Tobias A. Wiesner
Institute for Computational Mechanics
Technische Universität München
Boltzmannstr. 15
85747 Garching
Germany

Michael W. Gee
Mechanics & High Performance Computing Group
Technische Universität München
Parkring 35
85748 Garching
Germany

Andrey Prokopenko
Scalable Algorithms
Sandia National Laboratories
Mailstop 1318
P.O. Box 5800
Albuquerque, NM 87185-1318

Jonathan J. Hu
Scalable Algorithms
Sandia National Laboratories
Mailstop 9159
P.O. Box 0969
Livermore, CA 94551-0969

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
HTTP://TRILINOS.ORG/PACKAGES/MUELU/

SAND2014-18624 R
First printing, September 2014

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

The MueLu tutorial

- Download MueLu tutorial from here

  http://trilinos.org/wordpress/wp-content/uploads/2014/11/mt.pdf

  or

  http://wiesner.userweb.mwn.de/sandia/muelututorial.pdf

  (high quality)
- Log in to the local workstations
- Open a terminal and execute the following commands

  ```
  cd tuto_muelu
  ./hands-on.py
  ```
- The MueLu tutorial covers
  - Natural parameter lists (chapters 1-5)
  - Hierarchical parameter lists (chapters 6-11)
  - ML-compatibility and C++ interface (chapters 12-13)