MueLu - AMG Design and Extensibility

Tobias Wiesner
Andrey Prokopenko
Jonathan Hu

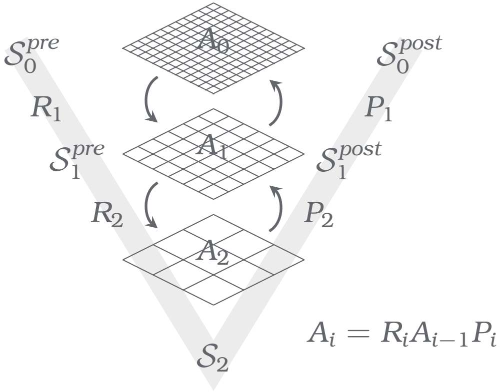
Sandia National Labs

March 3, 2015

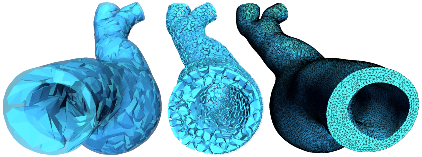
Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. SAND NO. 2011-XXXP

Main components of Algebraic Multigrid

V-cycle (3 levels)



Example for aggregates



Main components

- Smoothers $\mathcal{S}_i$
- Level transfers P_i and R_i
  - Based on aggregates
  - Transfer operator smoothing
- Galerkin product $A_i = R_i A_{i-1} P_i$

Setting up multigrid hierarchies with MueLu

Features:

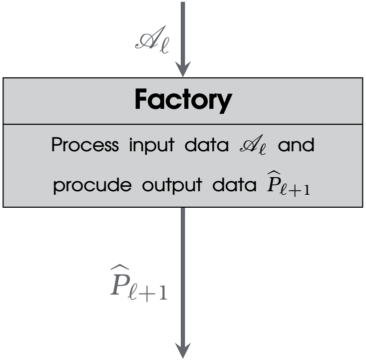
- MueLu implements all building blocks for state-of-the-art aggregation-based AMG methods
  - Parallel aggregation algorithms
  - Transfer operators
  - Level smoothers
  - Rebalancing
- Flexibility through modularity
  - Strict splitting of data and algorithms
  - Advanced software patterns (e.g. Factory classes,...)
- Usability through use of xml files
  - Easy-to-use xml format (for beginners and advanced users)
  - The multigrid hierarchy is defined at runtime through xml files

T. Wiesner, *MueLu: The next-generation Trilinos multigrid package*, 1st European Trilinos User Group Meeting, Lausanne, 2012
available here: `http://trilinos.org/oldsite/events/eurotug_2012/presentations/wiesner.pdf`

Factory concept

Data processing

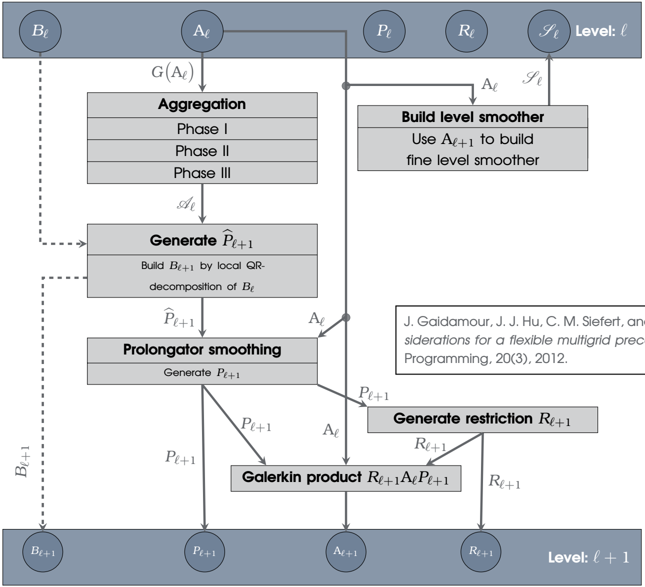
- Each building block is represented by a factory
- Each factory knows which input is needed to produce the corresponding output
- Each factory requests one or more input variables
- Each factory produces one or more output variables



Data storage

- All input and output data is (temporarily) stored in data container classes
- There is one data container for each multigrid level
- The hierarchical data dependencies are automatically handled by MueLu

Hierarchical setup process




```xml
<!-- Factory collection -->
<ParameterList name="Factories">

  <!-- Note that ParameterLists must be defined prior to being used -->


  <!-- sub block factories -->
  <!-- BLOCK 1 -->
  <ParameterList name="mySubBlockAFactory1">
    <Parameter name="factory" type="string" value="SubBlockAFactory"/>
    <Parameter name="block row"                 type="int"     value="0"/>
    <Parameter name="block col"                 type="int"     value="0"/>
  </ParameterList>

  <ParameterList name="myAggFact1">
    <Parameter name="factory" type="string" value="UncoupledAggregationFactory"/>
    <Parameter name="MinNodesPerAggregate" type="int" value="6"/>
    <Parameter name="MaxNodesPerAggregate" type="int" value="12"/>
    <Parameter name="MaxNeighAlreadySelected" type="int" value="0"/>
  </ParameterList>

  <ParameterList name="myCoarseMap1">
    <Parameter name="factory" type="string" value="CoarseMapFactory"/>
    <Parameter name="Striding info" type="string" value="{ 2,1 }"/>
    <Parameter name="Strided block id" type="int" value="0"/>
  </ParameterList>

  <ParameterList name="myTentativePFact1">
    <Parameter name="factory" type="string" value="TentativePFactory"/>
    <Parameter name="A" type="string" value="mySubBlockAFactory1"/>
    <Parameter name="Aggregates" type="string" value="myAggFact1"/>
    <Parameter name="CoarseMap" type="string" value="myCoarseMap1"/>
  </ParameterList>

"string" value="NullspaceFactory"/>
lspace" type="string" value="Nullspace1"/>
pe="string" value="myTentativePFact1"/>

  <!-- BLOCK 2 -->
  <ParameterList name="mySubBlockAFactory2">
```

J. Gaidamour, J. J. Hu, C. M. Siefert, and R. S. Tuminaro. *Design considerations for a flexible multigrid preconditioning library.* Scientific Programming, 20(3), 2012.

FactoryManager

- It is very difficult to define all inter-factory dependencies by hand
- A `FactoryManager` makes default choices for missing inter-factory dependencies
- The default information provided by the `FactoryManager` may or may not be optimal

Example

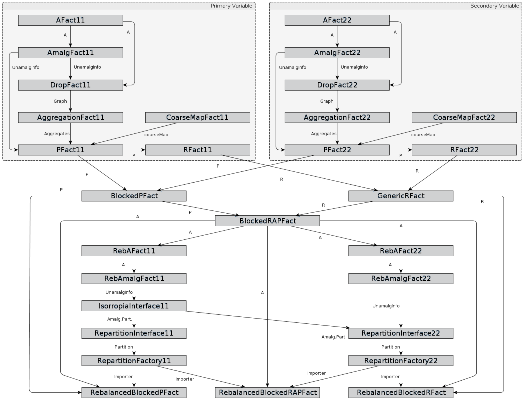
- The user declares a **`TentativePFactory`** object which needs **`Aggregates`** as input.
- The user does not declare an **`AggregationFactory`** which produces **`Aggregates`**.
- The **`FactoryManager`** provides a **default `AggregationFactory`** which is used by the **`TentativePFactory`**

Hierarchical xml files

Sandia National Laboratories

- Use hierarchical XML files to exactly describe the dependency graph of the factories
- Details on the hierarchical XML files can be found in the MueLu tutorial (chapters 6-11)
- Use the hierarchical framework to plug-in your application-specific factories in the existing framework

Example for 2×2 block matrix with rebalancing:



Minimal example

Hierarchical XML file for 5 level AMG with symmetric Gauss-Seidel level smoother:

```
<ParameterList name="MueLu">
  <!-- Factory collection -->
  <ParameterList name="Factories">

    <ParameterList name="Sym.Gauss-Seidel">
      <Parameter name="factory" type="string" value="TrilinosSmoother"/>
      <Parameter name="type"    type="string" value="RELAXATION"/>
      <ParameterList name="ParameterList">
        <Parameter name="relaxation: type"           type="string"          value="Symmetric Gauss-Seidel"/>
        <Parameter name="relaxation: sweeps"         type="int"             value="1"/>
        <Parameter name="relaxation: damping factor" type="double"          value="0.7"/>
      </ParameterList>
    </ParameterList>

  </ParameterList>

  <!-- Definition of the multigrid preconditioner -->
  <ParameterList name="Hierarchy">
    <Parameter name="max levels"                                 type="int"      value="5"/>
    <Parameter name="coarse: max size"                           type="int"      value="1000"/>
    <Parameter name="verbosity"                                  type="string"   value="High"/>
    <ParameterList name="AllLevel">
      <Parameter name="Smoother"                                 type="string"   value="Sym.Gauss-Seidel"/>
    </ParameterList>
  </ParameterList>
</ParameterList>
```

Natural interface

Sandia National Laboratories

Natural XML file for 5 level AMG with symmetric Gauss-Seidel level smoother:

```xml
<ParameterList name="MueLu">
  <Parameter name="max levels" type="int" value="5"/>
  <Parameter name="coarse: max size" type="int" value="1000"/>
  <Parameter name="verbosity" type="string" value="high"/>
  <Parameter name="smoother: type" type="string"   value="RELAXATION"/>
  <ParameterList name="smoother: params">
    <Parameter name="relaxation: type"            type="string"         value="Symmetric Gauss-Seidel"/>
    <Parameter name="relaxation: sweeps"          type="int"            value="1"/>
    <Parameter name="relaxation: damping factor" type="double"         value="0.7"/>
  </ParameterList>
</ParameterList>
```

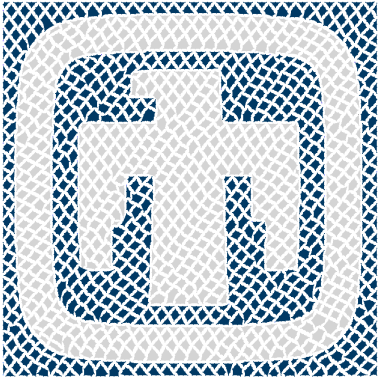
Natural versus hierarchical XML files:

- Hierarchical XML parameter file longer than natural XML file
- Full flexibility with hierarchical XML file

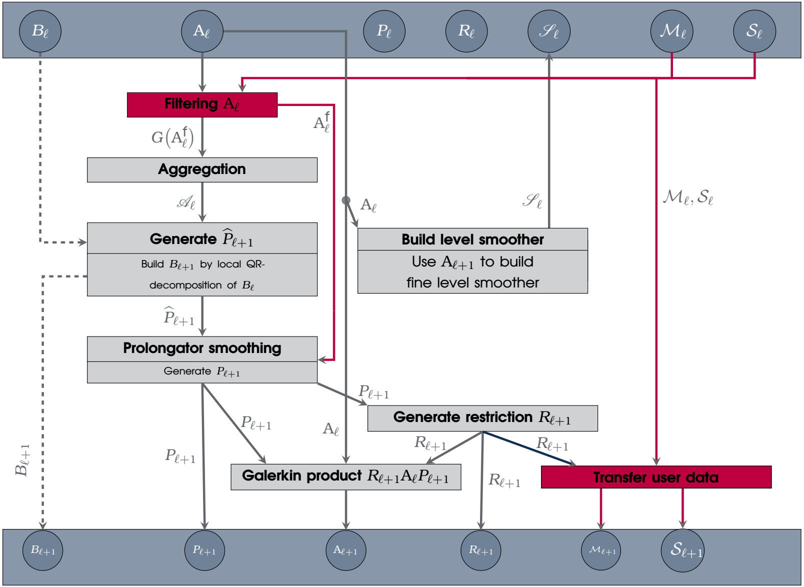
MueLu Multigrid Framework

Demonstration

- Build special aggregates by modifying input for aggregation algorithm
- Create a special filter factory
  - Input: fine level matrix A_ℓ, splitting information $\mathcal{M}, \mathcal{S}$
  - Output: modified matrix A_ℓ^{f}
- Use filtered A_ℓ^{f} as input for aggregation and transfer operator smoothing (but not for level smoothers)
- Transfer splitting information using a user-provided transfer factory

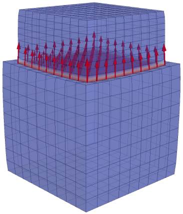


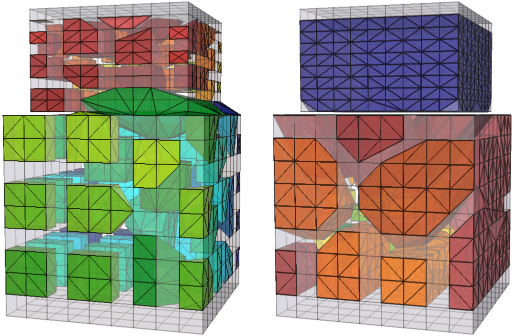
Extend setup phase by new factories



Computational contact mechanics

- Two-solid bodies example
- Aggregates should not cross contact interface
- Modify input for aggregation algorithm accordingly





Standard aggregation technique: aggregates cross the contact interface

Adapted aggregation technique: aggregates do not cross the contact interface

T.A. Wiesner, *Flexible aggregation-based algebraic multigrid methods for contact and flow problems*, PhD thesis, 2015.

C++ API in MueLu

C++ interface

```cpp
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

MueLu Multigrid Framework

C++ interface

```
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

Access MueLu hierarchy data from C++

- Access data containers in `Hierarchy` object
- `Hierarchy::GetNumLevels()` returns number of multigrid levels
- Use keyword `Keep` to access temporary data

```
Hierarchy H(fineA);     // generate hierarchy using fine level
                        // matrix
H.Setup();              // call multigrid setup (create hierarchy)
```

Access MueLu hierarchy data from C++

- Access data containers in `Hierarchy` object
- `Hierarchy::GetNumLevels()` returns number of multigrid levels
- Use keyword `Keep` to access temporary data

```
Hierarchy H(fineA);    // generate hierarchy using fine level
                       // matrix
H.Setup();             // call multigrid setup (create hierarchy)

// access data container for level 0 and level 1
RCP<Level> fineLevel   = H.GetLevel(0);
RCP<Level> coarseLevel = H.GetLevel(1);
```

Access MueLu hierarchy data from C++

- Access data containers in `Hierarchy` object
- `Hierarchy::GetNumLevels()` returns number of multigrid levels
- Use keyword `Keep` to access temporary data

```
Hierarchy H(fineA);      // generate hierarchy using fine level
                         // matrix
H.Setup();               // call multigrid setup (create hierarchy)

// access data container for level 0 and level 1
RCP<Level> fineLevel   = H.GetLevel(0);
RCP<Level> coarseLevel = H.GetLevel(1);
// extract data (fine level matrix and transfers)
RCP<Matrix> A = fineLevel->Get  < RCP<Matrix> >("A");
RCP<Matrix> P = coarseLevel->Get< RCP<Matrix> >("P");
RCP<Matrix> R = coarseLevel->Get< RCP<Matrix> >("R");
```

⇒ **Use data, e.g., to implement your own V-cycle**

Reuse data for several multigrid setups

Sandia National Laboratories

- Call `Keep` routine for data that you want to keep stored in the level class (and not automatically freed if possible).
- Call `Setup` and use Hierarchy object
- Overwrite input data (fine level A)
- Redo `Setup` for new fine level matrix and use (new) Hierarchy
- Do not forget to delete data in level containers as soon as possible to save memory.

```
FactoryManager M;

Hierarchy H(A1);

RCP<Factory> PtentFact = rcp(new TentativePFactory());
M.SetFactory("Ptent", PtentFact);
H.Keep("P",           PtentFact.get());

RCP<Factory> AcFact = rcp(new RAPFactory());
M.SetFactory("A", AcFact);

H.Keep("RAP Pattern", AcFact.get());

// first setup call
H.Setup(M);

// -> use H

// Change the problem
RCP<Level> finestLevel = H.GetLevel(0);
finestLevel->Set("A", A2);

// Redo the setup
H.Setup(M);

// -> use H

H.Delete("P",           M.GetFactory("Ptent").get());
H.Delete("RAP Pattern", M.GetFactory("A").get());
```

Current research projects (using MueLu)

Current research projects (using MueLu)

- Nonlinear multigrid methods (FAS):
  - Implementation of multigrid FAS scheme
  - Use MueLu aggregation and transfer operators
  - General framework for nonlinear problems based on callback functions (NOX compatible)

Matthias Mayr, MHPC, TUM

- Use MueLu with different types of operators (e.g., high order discretization operators on the finest level and lowest order discretization on coarse levels)

Chris Siefert, SNL

Current research projects (using MueLu)

- MueLu is used within the EQUINOX framework
  - EQUINOX = Environment for quantifying Uncertainty: Integrated aNd Optimized at the eXtreme-scale
  - EQUINOX contains advanced multi-level methods for alleviating the complexity and accelerating the solutions of both deterministic and stochastic extreme-scale solvers
  - MueLu is applied as a preconditioner to an ensemble of samples
  - Use flexibility of Xpetra/Tpetra for a scalar type representing an ensemble (diagonal matrix)
  - Scalability shown up to 512 nodes (BG/Q machine)
  - See also **`http://equinox.ornl.gov`**

Eric Phipps, SNL

Prof. M. Gunzburger (FSU)
John Burkardt (FSU)
Prof. J. Wu (Georgia Tech)
C. Webster (ORNL)
R. Archibald (ORNL)
C. Hauck (ORNL)
S. Pannala (ORNL)
E. Phipps (SNL)
C. Edwards (SNL)
J. Hu (SNL)
S. Rajamanickam (SNL)
and others. . .

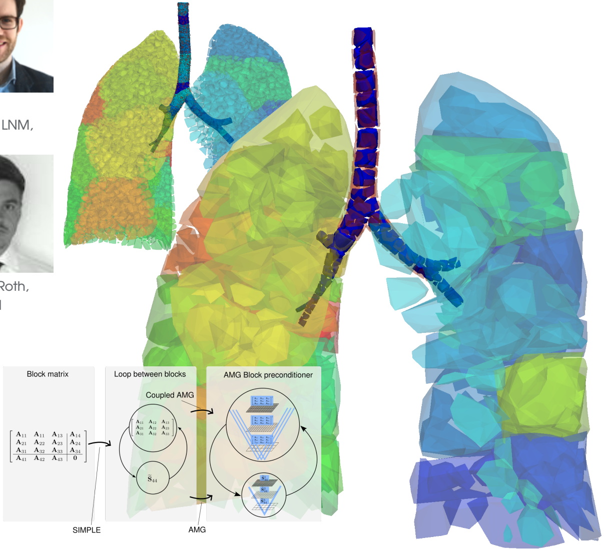
Current research projects (using MueLu)

- General multigrid framework for multiphysics problems
  - Implementation of general multigrid framework for $n \times n$ block matrices (e.g., FSI problems with constraints, TSI problems,. . .)
  - Use MueLu level smoothers, aggregation methods and single-field transfer operators
  - Own V-cycle implementation and block smoothing strategies

see also: M.W. Gee, U. Küttler and W.A. Wall, *Truly monolithic Algebraic Multigrid for fluid-structure interaction*, Int. J. Numer. Meth. Engng., 85 (2011)

Francesc Verdugo, LNM, TUM

Christian Roth, LNM, TUM



Thank you for your attention

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, 2015.
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)