Amesos2: Common Interface to Direct Solvers

Eric Bavier and Erik Boman and Siva Rajamanickam

Sandia National Laboratories

July 6, 2011

Sandia is a multiprogram laboratory operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin, for the United States Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Amesos2 Motivation

- Uniform Interface to common third-party direct solvers.
- Support a variety of scalar and ordinal datatypes
- Support Epetra and Tpetra data structures and allow a design that can extend to other matrix types (for eg.: PetSc matrix, compressed column matrix)
- Revisit Amesos design choices and redesign to easily support more solvers and matrix types.

Use Case 1: Simple Solve.

A, X, B all known when creating the solver and the user requires one direct solve.

```
RCP<MAT> A; RCP<MV> X; RCP<MV> B;
// initialize A and B
RCP<Solve<MAT,MV> > solver = Amesos::create(A, X, B);
solver->solve(); // solution placed in X
```

Typical Usage: Preorder, Symbolic, Numeric, Solve

A, X, B all known when creating the solver and the user requires one or multiple solves. Different steps of the factorization could be called in different places.

```
RCP<MAT> A; RCP<MV> X; RCP<MV> B;
// initialize A and B
RCP<Solve<MAT,MV> > solver = Amesos::create(A, X, B);
solver->preOrdering();
solver->symbolicFactorization();
solver->numericFactorization();
solver->solve();
```

Solver Interface and Creating a Solver

- The basic solver interface
  - `preOrdering()`
  - `symbolicFactorization()`
  - `numericFactorization()`
  - `solve()` or `solve(X,B)`
- Creation of an Amesos2 solver
  1. Solver name (optional, defaults to "KLU2")
  2. The matrix A (RCP or pointer)
  3. X and B (multi)vectors (optional if using `solve(X,B)` interface)

Solver Interface and Creating a Solver

Summary of creation options

- `create("SuperLU", A, X, B)`
- `create("SuperLU", A)`
- `create(A, X, B)`
- `create(A)`

Amesos2 Parameters vs Solver Parameters

```
<ParameterList name="Amesos2">
  <Parameter name="Tranpose" type="bool" value="true" />
  <ParameterList name="SuperLU_MT">
    <Parameter name="nprocs" type="int" value="8" />
  </ParameterList>
  <ParameterList name="SuperLU">
    <Parameter name="DiagPivotThresh" type="double" value=".1" />
  </ParameterList>
</ParameterList>
```

Amesos2 Parameters vs Solver Parameters

```
<ParameterList name="Amesos2">
  <Parameter name="Tranpose" type="bool" value="true" />
  <ParameterList name="SuperLU_MT">
    <Parameter name="nprocs" type="int" value="8" />
  </ParameterList>
  <ParameterList name="SuperLU">
    <Parameter name="DiagPivotThresh" type="double" value=".1" />
  </ParameterList>
</ParameterList>
```

Amesos2 Parameters vs Solver Parameters

```
<ParameterList name="Amesos2">
  <Parameter name="Tranpose" type="bool" value="true" />
  <ParameterList name="SuperLU_MT">
    <Parameter name="nprocs" type="int" value="8" />
  </ParameterList>
  <ParameterList name="SuperLU">
    <Parameter name="DiagPivotThresh" type="double" value=".1" />
  </ParameterList>
</ParameterList>
```

Solvers supported in Amesos2

These solvers will be supported by the Sept. release:

- KLU2 (in progress)
- SuperLU
- SuperLU_MT
- SuperLU_DIST (in progress)

We are also considering adding LAPACK (for almost dense matrices in CrsMatrix format) and Pardiso.

Support for New Solvers

- Set up necessary internal data structures to interface with the TPL
- Implement
  - `preOrdering_impl()`
  - `symbolicFactorization_impl()`
  - `numericFactorization_impl()`
  - `solve_impl(X, B)`
  - `matrixShapeOK_impl()`
  - `setParameters_impl()`
  - `getValidParameters_impl()`
- Do not need to worry about
  - creating timers
  - checking compatibility of A, X, and B
  - keeping track of solver status

Support for New Matrices and Multivectors

The amount of effort required depends on what object is being added.

- Extends `Epetra_RowMatrix` or `Tpetra::RowMatrix`? Very little, only a method that imports the object into a new object with a given map.
- Otherwise, implement functions relating to
  - getting a compressed row or column copy
  - getting global/local matrix statistics
  - getting row/col map
  - import method
- Multivectors require methods for
  - getting global/local statistics
  - getting the map
  - getting a contiguous 1-D copy
  - setting/globalizing a 1-D array

Expert Usage

Releasing A after numeric factorization

For preconditoners or smoothers when there is one often one numeric factorization and multiple solves.

```
RCP<MAT> A;
// Get A from somewhere
RCP<Solver<MAT,MV> > solver = Amesos::create("SuperLU", A);
solver->symbolicFactorization().numericFactorization();
A = Teuchos::null;                  // no longer need A
solver.setA(Teuchos::null); // tell solver to release A
RCP<MV> X; RCP<MV> B;
// do some other work, finally get B's values
solver->solve(X, B);                // solution placed in X
```

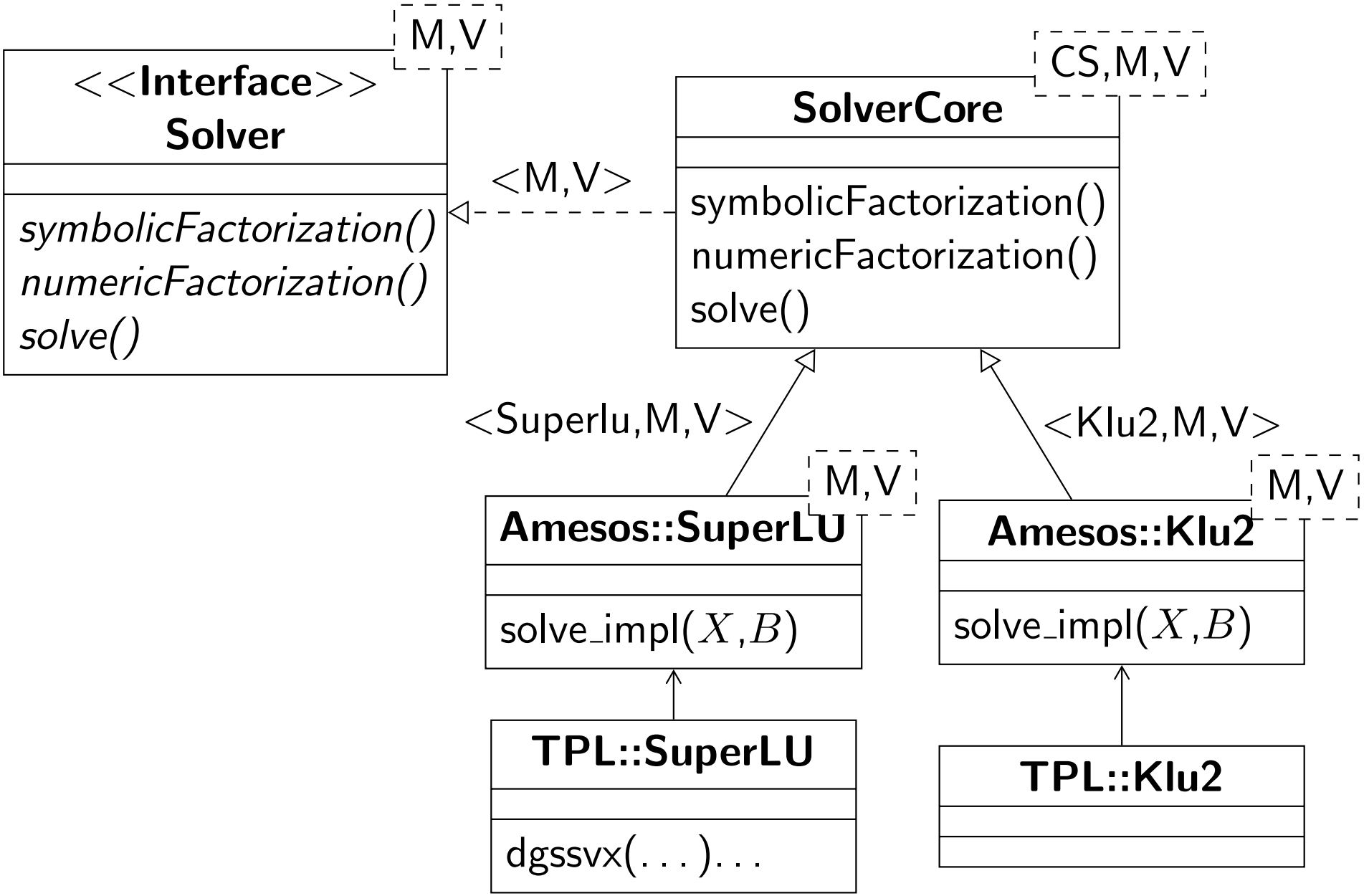
Expert Usage

Reusing the solver for a different matrix

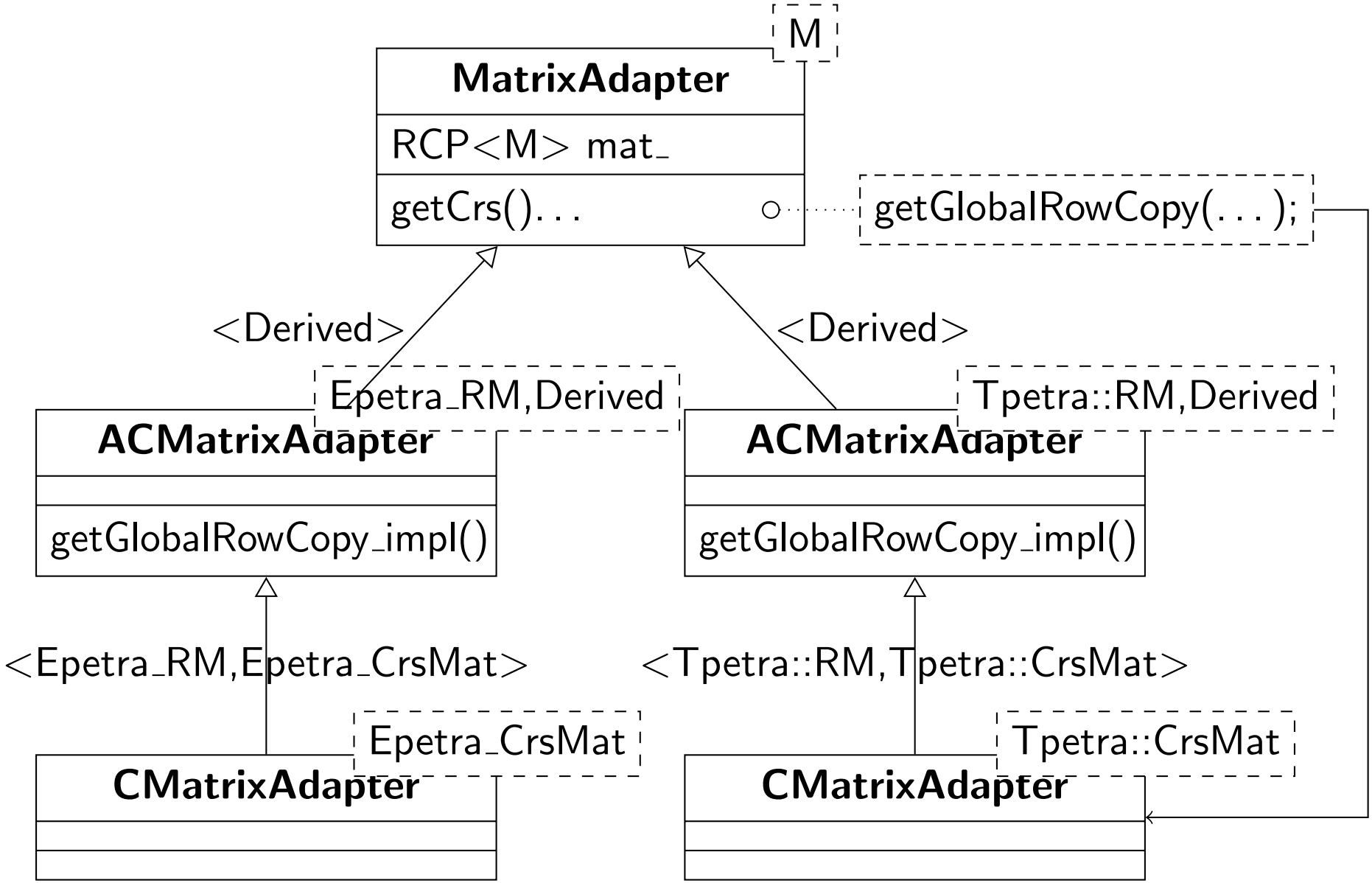
Amesos does not support this expert-only usecase, but Amesos2 does: Use the same solver instance for solving different linear systems.

```
RCP<MAT> A1, A2;
// initialize A1, A2, and B
RCP<Solver<MAT,MV> > solver = Amesos::create(A1,X,B);
solver->solve(); // solution in X
solver->setA(A2);
solver->solve(); // refactorizes A2 first
```

Internal Design: Solver Hierarchy



Internal Design: MatrixAdapter Hierarchy



Future Work

- Need better support for pre-ordering: Zoltan2 will provide both graph partitoning based and minimum degree based orderings.
- Currently we use SuperLU's internal orderings. KLU2 uses the orderings from Amesos.
- Need to support more solvers and matrix types.