AMESOS: General Interfaces to Direct Solver Libraries

Marzio Sala ETHZ/D-INFK
K. Stanley (Oberlin College),
M. Heroux (SNL), R. Hoekstra (SNL)

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Outline

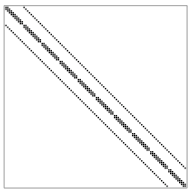
- Design of the AMESOS project, an abstract framework for solving

  $$A x = b$$

  with distributed, sparse direct methods
- Advantages and disadvantages
- Supported libraries
- Python interface (through PyTrilinos)

NOTE: we consider the usage of direct solvers, not their implementation

Background



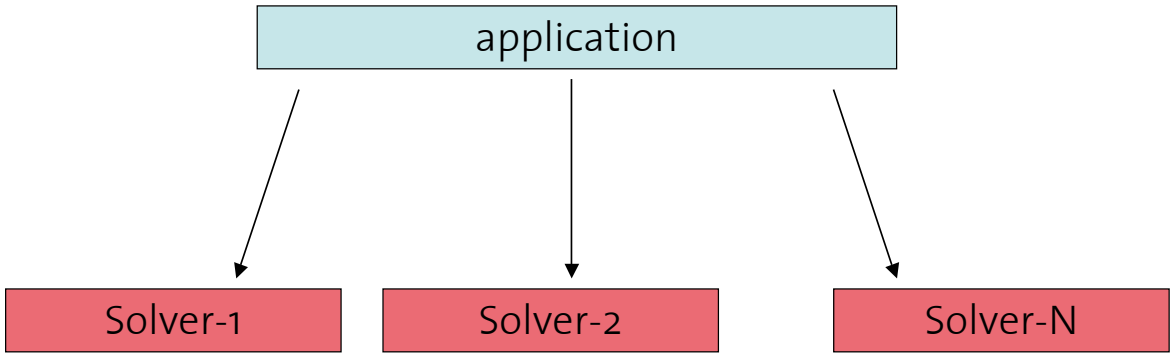
- An application has to solve

$$A\,x = b$$

- The linear system matrix A is:
  - Square, double-precision
  - Serial or distributed
  - Sparse
- Very good libraries of direct solution methods available
  - Not trivial to implement
  - Parallel even more difficult
  - Public domain or commercial

Background (2)

- What is the best method/library?
  - No absolute winner, experimentation needed
- Requires custom-made interfaces
  - A library can be tested only if an interface exists
  - Code to write, debug, maintain



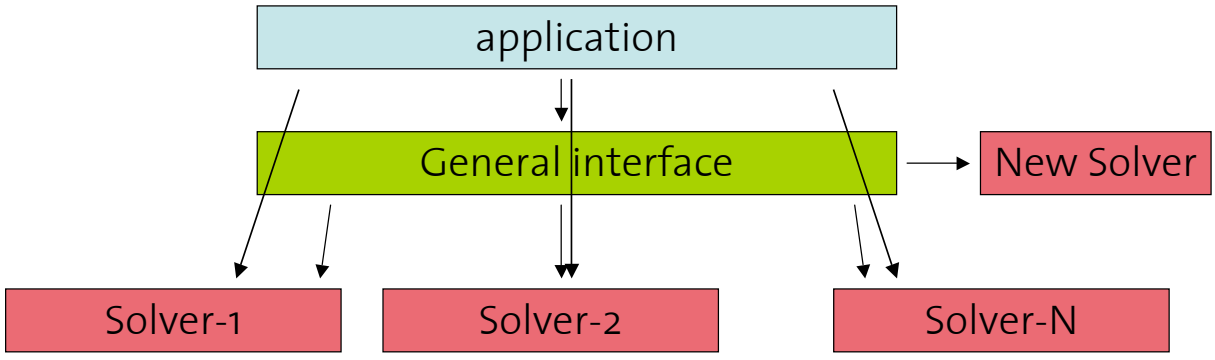
Background (3)

- Can we improve this process?
- OO solution: keep the application and the solver as separate as possible
- Requires an additional software layer...

➢ this is the main goal of the AMESOS project

Objectives

- AMESOS defines an additional layer to the linear system solver libraries
- Takes care of dealing with each solver's data distribution and format, calling sequence, ...



Objectives (2)

- Flexibility:
  - More than one algorithm/library must be available, allow easy testing
- Simplicity of usage:
  - MATLAB's solution is simply `x = A\b`, it should not be more difficult in a production code
  - Mathematical code should be simple to use!
- Efficiency:
  - The final software framework must be as efficient as possible
  - The overhead must be minimal

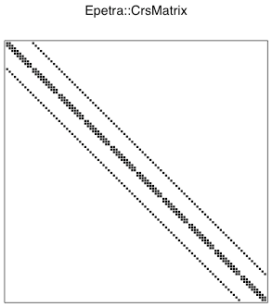
Design

Two basic (pure virtual) classes:

1. RowMatrix class to query for matrix elements
   - Contained in the Epetra package
   - "Adaptor" design pattern

2. Solver class to manage all the internal operations of the supported library
   - Concrete implementations are the core of Amesos
   - Decouples operations and low-level operations
   - "Facade" design pattern; also use "factory"

The `RowMatrix` class



- We don't want a matrix format, rather a matrix interface
- Each solver typically requires a (slightly) different format
- We allow queries for matrix rows:
  - The RowMatrix class must provide a `getrow()` method that returns the nonzero indices and values for a given (locally owned) row
  - Each row is wholly owned by one processor
  - The Solver class will query the matrix and reallocate it in the supported solver's format

The `RowMatrix` class (2)

Advantages:

- The matrix format used by the application becomes inessential
- Easy to modify the matrix with dropping, reordering, ...
- Separate the application and the solver (good OO practice)

Disadvantages:

- Possible memory overhead

The `RowMatrix` class (3)

- However:
  - The matrix formats of the application is often different from the matrix format of the solver; memory overhead unavoidable
  - The matrix layout sometimes is not supported from the solver; requires data redistribution (not easy!)
- The RowMatrix gives efficient solutions to these problems
- Memory overhead can be reduced by downcast to specific matrix classes

The `Solver` class

- The generic interface to a direct solver library is encapsulated in the `Solver` class:
- Contains methods:
  - `SymbolicFactorization()`
  - `NumericFactorization()`
  - `Solve()`
  - `SetParameters()`
- The calling sequence of the library is hidden to the user (high-level view)
- Tuning with `SetParameters()`
  - Solver-specific

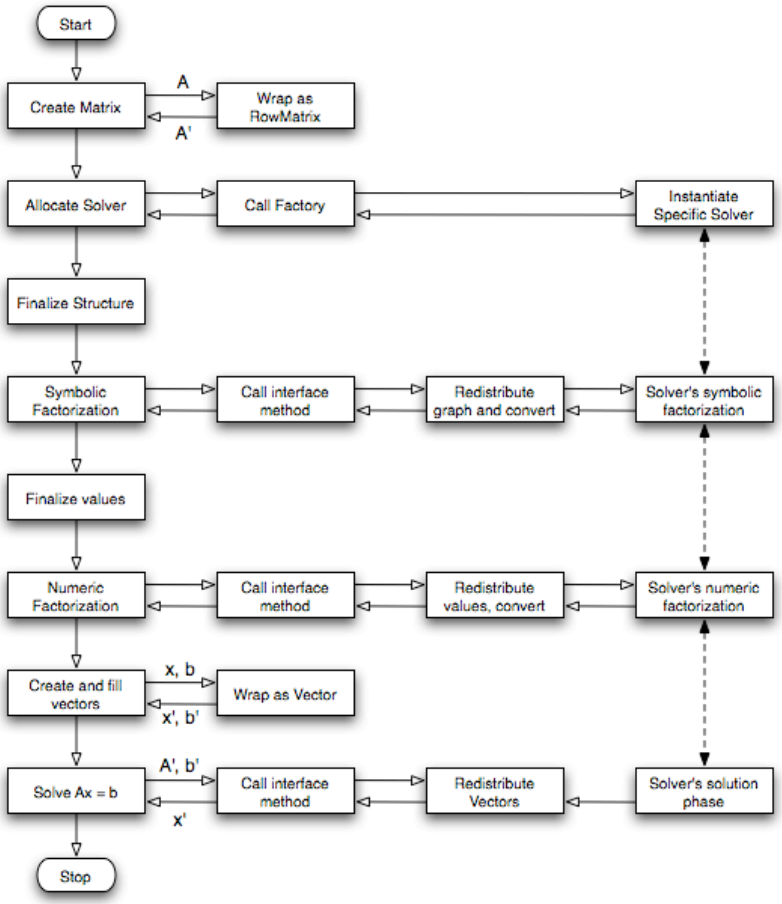
Supported libraries

name	model	language
LAPACK	Serial, dense	F77
DSCPACK	Parallel (dist)	C
KLU	serial	C
MUMPS	Serial, parallel (dist)	F90
PARDISO	Parallel (shared)	sources
TAUCS	Serial, out-of-core	C
UMFPACK	serial	C
SuperLU	Serial	C
SuperLU_DIST	parallel	C
SCALAPACK	Parallel, dense	F77

Supported Libraries (2)

- Some solvers are serial
- The concrete implementations of the Solver class redistribute objects as necessary:
- Serial solvers can be used in parallel
- Some solvers require different distributions for matrix and vectors
- A different number of processors may be required by the solver (e.g., coarse solver in multilevel preconditioners)

➢ Users do not care about data distribution

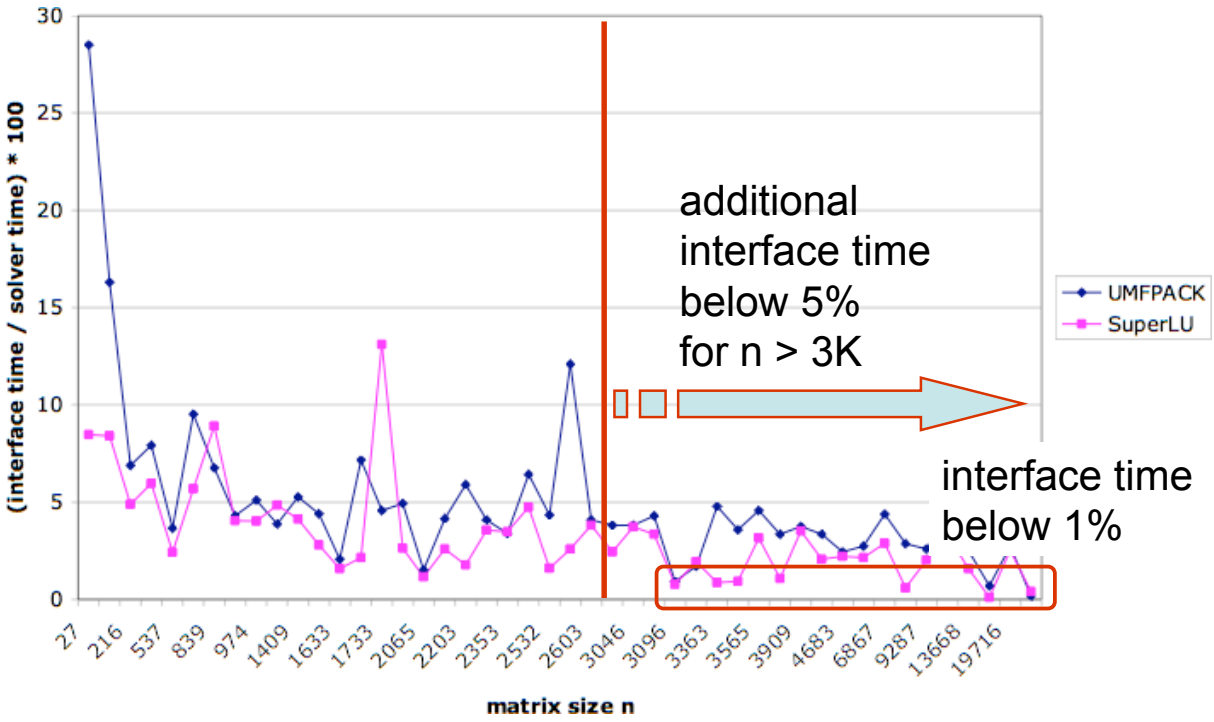


Example of Code

```
#include "Amesos.h"
#include "mpi.h"
#include "Epetra_MpiComm.h"
...

int main(int argc, char *argv[])
{
  MPI_Init(&argc, &argv);
  Epetra_MpiComm Comm(MPI_COMM_WORLD);
  <create A, x, b>
  Epetra_LinearProblem Problem(A, x, b);

  Amesos Factory;                  // factory class
  string SolverType = "Mumps";     // selected interface
  Amesos_BaseSolver* Solver;       // generic solver object
  Solver = Factory.Create(SolverType, Problem);

  Solver->SymbolicFactorization(); // symbolic factorization
  Solver->NumericFactorization();  // numeric factorization
  Solver->Solve();                 // linear system solution
  delete Solver;

  MPI_Finalize();
  return(EXIT_SUCCESS);
}
```

- Implements a "virtual constructor"
- The application code only deals with abstract classes
- Details about the implementation are contained in the library only

Overhead



Extension to Python

- Few, well-defined interfaces are easy to wrap
- This is done by the PyTrilinos project:
  - Developers: MS, Bill Spotz, Mike Heroux, Eric Phipps

- SWIG is used to generate the "glue" code
- See next talk for more details

Summary

- AMESOS is a set of interfaces to direct solvers:
  - Easy-to-use
  - (almost) as efficient as the underlying solver
  - Easy to add new solvers
  - Standard C++ API from applications to all solvers for (almost) any matrix format/distribution
- Disadvantages:
  - Fine tuning can be problematic
  - Some solvers are not compatible (same function name, different parameters); same for different versions

Summary (2)

- Web page, download, info
  http://software.sandia.gov/trilinos/
- Future developments:
  - Add new interfaces (HSL MAxx, OBLIO, PaStiX, ...)
  - Generalizing the framework with templates (float, double, complex<double>)
  - Feel free to ask!
- Amesos Developers:
  - Ken Stanley, MS, Mike Heroux, Rob Hoekstra, Tim Davis