Coupling Multi-level Component Interfaces for Parallel Sparse Linear System Solvers

Fang Liu, Masha Sosonkina, [1]
Dane Coffey[2]

[1]Ames Laboratory and Iowa State University, USA
[2]University of Minnesota

Component-Based High-Performance Computing,
Portland, OR, November 2009

Outline

Sparse linear algebra in multi-scale simulations

- Ubiquitous: Sparse matrices arise from near-neighbor and long-range interactions.
  - Matrices have different characteristics and may be structured and unstructured, affect the numerical methods applied.
- Sparse linear system solution (or eigen-value computation) is a significant cost factor.
  - Numerous implementations exist and depend on the problem and hardware architecture at hand.
  - Sequences of matrices are often to be solved during simulation cycles.

Switching of solution methods may be advantageous from one cycle to another in a simulation.

Universe of existing sparse linear system solvers

Dichotomy into sparse direct and iterative solvers

- Direct: solve linear system by performing Gaussian elimination directly while applying sparse matrix techniques.
- Iterative: find solution with a desired accuracy by improving solution one step at a time, say, by a projection method.
  - Accelerator iterative procedure to obtain approximate solution in a particular subspace.
  - Preconditioner helps the accelerator to converge by transforming the original problem into one that is easier to solve and has the same solution.
- Large number of implementations:
  - Example, a list of freely available solver software is at http://www.netlib.org/utk/people/JackDongarra/la-sw.html.
    - Contains 12 direct and 22 iterative solver entries.

Design choices

- Low-level (LI): User expresses all sparse matrix operations with components.
  - Beneficial for very large matrices when conversion is prohibitively expensive.
- Medium-level (MI): Major solver parts are separate components.
  - Useful for expert tuning of solution;
  - User needs to know matrix representation.
- High-level (HI): Entire linear system solution is encapsulated into a component.
  - Treats solver as “black box”;
  - Easy switching of solver packages;
  - Many solvers, such as PETSc and Trilinos, may be linked via high-level interfaces.

High-level design: CCA-LISI

CCA LInear system Solver Interfaces [Indiana University]

Design goals:

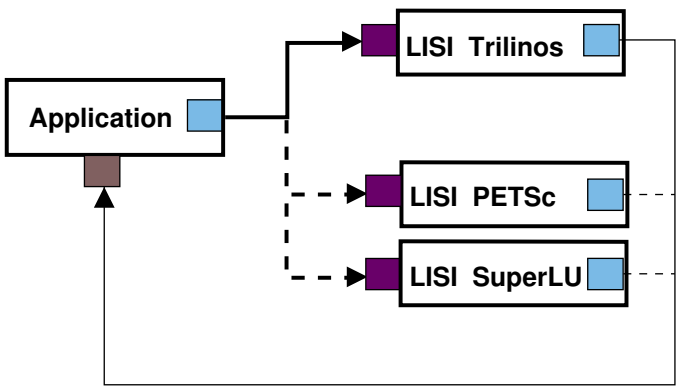
Hide the underlying implementation while preserving functionality and user flexibility.

- Encapsulate different sparse matrix formats into interface implementations.
- Handle parallelism assuming block-row matrix distribution and programming model of each underlying solver.
- Provide for user-defined matrix-vector operations similar in spirit to “reverse communication”.
- Ease and simplicity of use: Allow user to seamlessly switch linear system solver packages.

High-level design: CCA-LISI

LISI architecture

- `SparseSolver` interface has *provides* port for application.
- `MatrixFree` interface is to be implemented by application and is *used* by the solver.
- Matrix is passed to LISI solver as multiple-arrays to reduce complexity of matrix object construction on the application side.
- Solver parameters are set as (***key***, ***value***) pair, while explicit methods are proposed for matrix data input.



Low- and Medium-level design: SPARSKIT-CCA

What is SPARSKIT?

- Well-known library of sparse matrix kernels by Yousef Saad (University of Minnesota).
- Provides a range of functions for sparse matrix computations with a focus on Iterative Methods.
- pARMS is a parallel embodiment of (extended) SPARSKIT's ideas.
- BLASSM is a suite of BLAS-like *low-level* operations on sparse matrices.

Goal: Abstract iterative method interfaces from sparse matrix format

- Especially important for plug-and-play components.

Separation of concerns in medium-level design

- Separate component `MatrixFormat` encapsulating matrix format transformations and inquiries as well as storing the matrix.
- Feasible since MI is designed *a priori* for expert users.

Benefits:

- Increases component reuse.
- Easy addition of new matrix formats.
- Simplification of MI design (more light-weight components).
- Clean connection to HI.

Parallel SPARSKIT Components

- Builds on SPARSKIT-CCA
  - `DistributedMatrixFormat` simply extends the MatrixFormat interface with matrix data distribution: `getDistribution` and `setDistribution`.
- Provides generic partitioner interface.
- Implements pARMS suite of iterative methods.
- Performs communication set-up each time a parallel component is created that needs it.
  - Example: Accelerator, preconditioner, matrix-vector multiply.
  - Allows reuse of the interfaces developed for sequential SPARSKIT.

Motivation: Benefits to users on different levels

- HI→MI[a]: Provide a high-level view to MI components for novice users.
  - standard way of componentizing MI driver component.
- MI→HI: Access to state-of-the-art iterative method components
  (e.g., preconditioners).
  - no need to interface MI into each SpLA package.

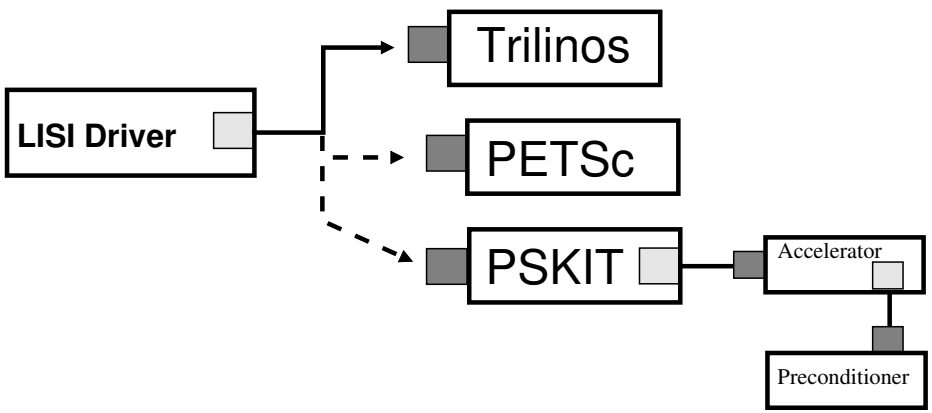
[a] Let "→" show the direction of the benefit

Design:

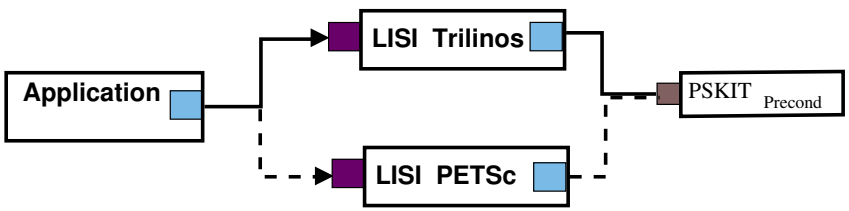
- HI→MI: Provide `SparseSolver` interface by the MI driver component.
- MI→HI: Use the `GenericPreconditioner` interface similar to "matrix-free" (reverse-communication) in solvers.
  - Example: via LISI `MatrixFree` interface.

Design diagrams

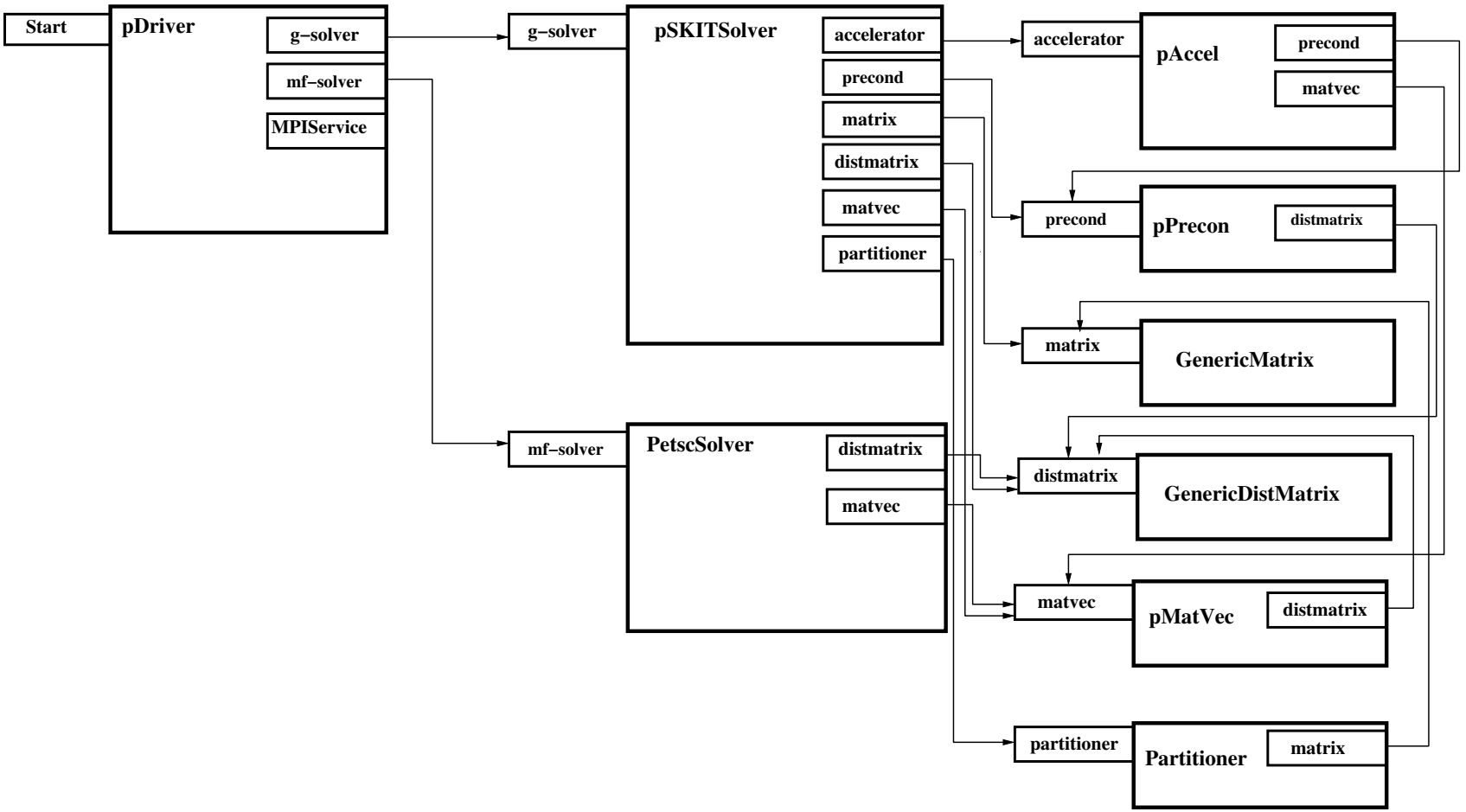
HI→MI:



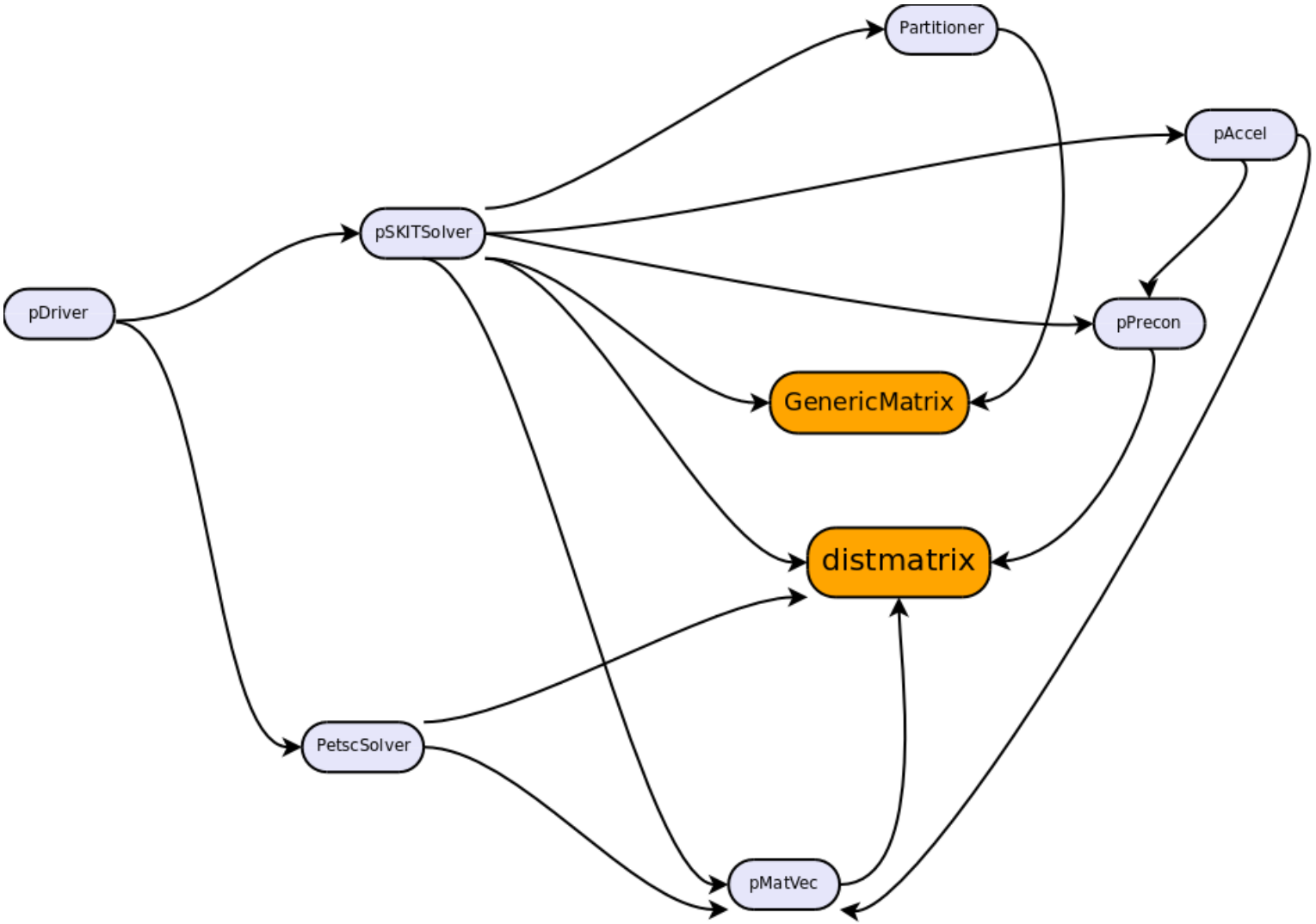
MI→HI:



Wire diagram for hierarchical interface connections



Wire diagram for star topology connections



1. Experiments: HI→MI

Table: Comparison of parallel MI components and their encapsulation using LISI HI interfaces.

Matrix	proc	its	PSKIT-MI	LISI-PSKIT
mat100x100	1	23	0.33	0.34
mat100x100	4	129	0.30	0.31
mat100x100	16	209	0.41	0.41
sherman5	1	17	0.083	0.091
sherman5	4	38	0.075	0.083
sherman5	16	114	0.145	0.168

mat100x100 represents a Poisson equation on a 2D 100×100 mesh with Dirichlet boundary conditions

2. Experiments: MI→HI

Table: PETSc solution using the `MatrixFree` port in LISI and the matrix-vector multiplication component

(a) from SPARSKIT

Matrix	its	mf-P	mf-MI
sherman5	100	0.17	0.17
parmobil	22	0.44	0.44
bcsstk16	76	0.71	0.71

(b) from PSPARSKIT on *mat100x100*

proc	its	mf-MI
1	113	0.29
4	189	0.74

Matrix row sizes:

parmobil: 8185; *sherman5*: 3312; and *bcsstk16*: 4884.

Summary

- MI interfaces are extended to include the ones for parallel solvers
  - based on SPARSKIT interfaces;
  - proposed partitioner interfaces;
  - abstracted distributed matrix into a separate component
- Coupling HI and MI interfaces makes iterative method parts accessible as a (whole) *black box*.
- Hierarchy of SpLA interfaces essentially bridges expert and novice user communities.