

# Hierarchical Usability Levels for Sparse Linear System Solver Components

Masha Sosonkina, Dane Coffey<sup>1</sup>  
Fang Liu, Randall Bramley<sup>2</sup>

<sup>1</sup>Ames Laboratory and Iowa State University, USA

<sup>2</sup>Indiana University, USA

Component-Based High-Performance Computing, 2008

# Outline

- 1 Introduction
  - Sparse linear algebra in multi-scale simulations
  - Universe of existing sparse linear system solvers
- 2 Usability requirements
  - Design choices
  - Examples of interface levels
  - SPARSKIT components
- 3 Hierarchy of interface levels
  - Design rationale
  - Experiments
- 4 Summary

# 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.

# Solver integration via Trilinos and PETSc

## **Trilinos** from Sandia National Laboratories.

- Targets multi-physics complex simulations.
- Object-oriented framework for inclusion of packages:
  - Each package is self-contained software.
  - Minimal set of interfaces/add-ons to the package to add it to the Trilinos framework.
- <http://trilinos.sandia.gov>

## **PETSc** from Argonne National Laboratory.

- Provides software for the scalable solution of systems of equations arising from PDEs (original goal).
- Has object-oriented programming style leveraging structured and unstructured matrices.
- Interfaces with many existing solvers as well as optimization techniques.
- <http://www.mcs.anl.gov/petsc>

# 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:

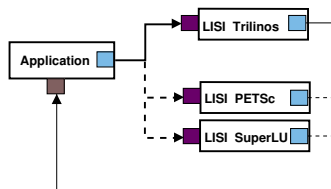
Hide the underlying implementation while preserving functionality and user flexibility.

- 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.



# High-level design: TOPS

## Interfaces for solvers developed in the Center “Towards Optimal Petascale Simulations” (TOPS)

### Design Goals:

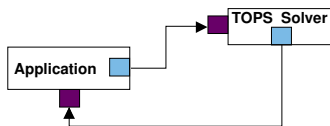
Provide scalable solution of linear and nonlinear systems arising from structured or unstructured meshes. Allow maximum flexibility to the application as to the data structures and solution choice.

- Enable experimentation with solvers without changing matrix data structures.
- Provide for structured and unstructured solvers.
- Construct the system on the application side and use it by the connected TOPS solver.

# High-level design: TOPS

## TOPS architecture

- `TOPS.System` interface is to be implemented in the application code and *used* by the `TOPS.Solver` component.
- `TOPS.Structured.Solver` and `TOPS.Unstructured.Solver` *provide* solution methods for the matrix objects implemented in `TOPS.System`.
- Separate methods exist for such functionality as residual and initial guess computations, construction of right-hand side.



# 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

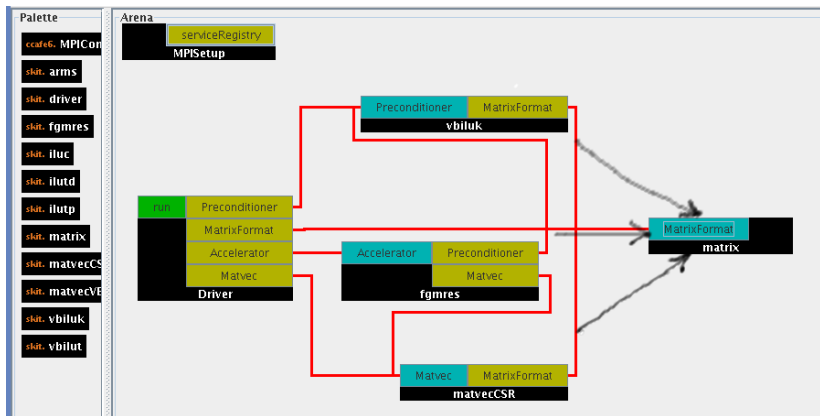
- Separate component `MatrixFormat` encapsulating matrix format transformations and inquiries and storing the matrix.
- Feasible since MI is designed *a priori* for expert users.

## Benefits:

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

# Sample diagram in medium-level design

Resemble **star** topology, centered around `MatrixFormat`.



## Motivation: Benefits to users on different levels

- HI→MI<sup>a</sup>: 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.

---

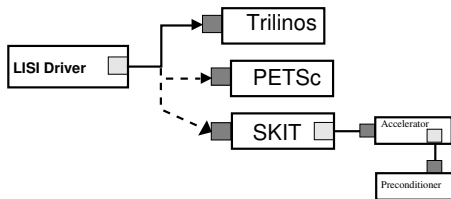
<sup>a</sup>Let “→” show the direction of the benefit

## Design:

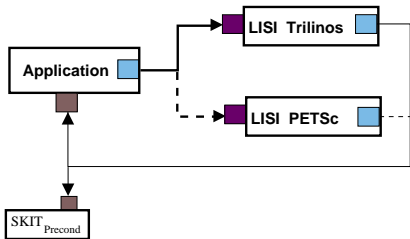
- HI→MI: Provide `SparseSolver` interface by the MI driver component.
- MI→HI: Use the `GenericPreconditioner` interface similar to “matrix-free” (reverse-communication) in solvers.
  - Example: via LISI `MatrixFree` interface.

# Design diagrams

HI→MI:



MI→HI:



# 1. Experiments: HI→MI

**Table:** Linear system *parmobil* solution with MI components encapsulated by LISI interfaces.

Precon	its	SKIT	SKIT-MI	SkitSolver
VBILUT	*	6.39	6.44	6.43
VBILUK	23	78.19	77.79	77.57
ILUC	*	2.54	2.6	2.61
ARMS+ILUTP	31	0.83	0.83	0.84
ARMS+ILUTD	32	0.86	0.87	0.86
ARMS+ILUC	31	0.84	0.85	0.85

PARMOBIL matrix: number of rows is 8,185.

## 2. Experiments: MI $\rightarrow$ HI

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

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

# Summary

- Linear system solver components enable HPC applications to immediately benefit from a vast knowledge and code bases in the field of numerical sparse linear algebra.
- Components is a viable programming model for developing sparse linear system solvers.
  - Incurs negligible overhead for coarse-grained componentization.
  - Provides an easy access to legacy codes.
  - Integrates existing and new solver packages.
- Implementing HI in the MI driver makes iterative method parts accessible as a (whole) *black box*.
- **Hierarchy of SpLA interfaces essentially bridges expert and novice user communities.**