SAND2011-1225C

A Hybrid Parallel Sparse Solver (Preconditioner)

Siva Rajamanickam

Erik Boman

Michael Heroux

Scalable Algorithms Department

Sandia National Laboratories

SIAM CS&E, Feb. 2011

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

National Nuclear Security Administration

Sandia National Laboratories

Sandia National Laboratories

Outline

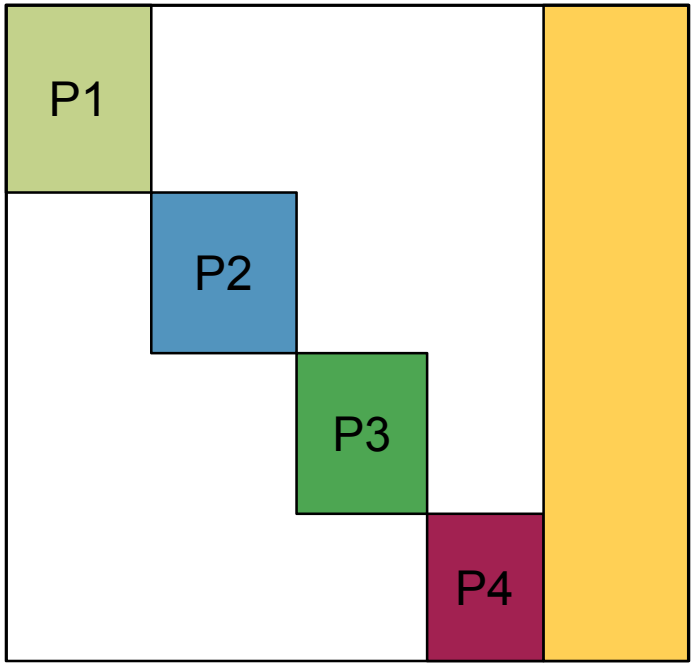
- Motivation for HyperLU
- Partitioning and block bordered form
- Schur complement and Probing
- Algorithm summary
- Numerical results
- Future work

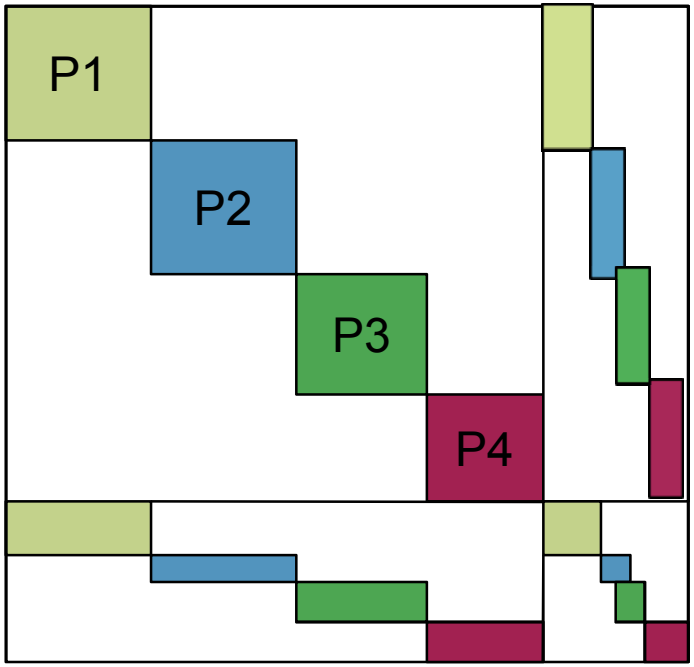
Sandia National Laboratories

HyperLU: Parallel Hybrid Sparse Solver

- Goal: Solve large-scale sparse linear systems on modern architectures
- Leverage existing parallel iterative solvers across nodes (MPI-based)
  - Multigrid, domain decomposition, etc.
- Need better sparse solver on the node
  - Core counts increase rapidly
  - Use threaded or hybrid programming model
- HyperLU is a new hybrid solver. It can be used
  - As a stand-alone iterative solver
  - As a preconditioner for subdomains

Decomposition by Partitioning





- Bordered block diagonal form exposes parallelism.
- We use hypergraph partitioner (Zoltan) to permute system to singly (unsym.) or doubly (sym.) bordered block form.
- Diagonal blocks can be solved independently, but couplings along borders remain.
- P1 … P4 can correspond to either cores or UMA regions.

Decomposition by Partitioning

Sandia National Laboratories

- Why limit the subdomains to UMA regions ?
  - Threaded solvers can take over in UMA regions without NUMA concerns.
  - Limits the size of the border
  - Less work for partitioning
- Solver framework
  - Symmetric – Factor the square diagonal blocks
  - Unsymmetric – Factor the “square” part of the rectangular diagonal blocks (partial pivoting across all rows/maximum matching)

Schur Complement Strategy

- Let $A = \begin{pmatrix} D & C \\ R & G \end{pmatrix}$
  - where D is block diagonal
- Schur complement: $S = G - RD^{-1}C$
- Exact S is almost dense, so must be approximated by sparse S'
  - Common approach: Sparsify by dropping entries
  - We choose probing method instead

Probing

- Probing is a method to cheaply build S'≈S
  - Chan & Mathew (1992)
- First select sparsity pattern of S'
  - For example, tridiagonal, pattern(G)+ few diagonals
- Then compute values of S' by applying the operator $RD^{-1}C$ to a set of probing vectors.
  - The probing vectors can be found by coloring the pattern of S'.

$$\begin{pmatrix} m_{11} & m_{12} & & & & & \\ m_{21} & m_{22} & m_{23} & & & & \\ & m_{32} & m_{33} & m_{34} & & & \\ & & m_{43} & m_{44} & m_{45} & & \\ & & & m_{54} & m_{55} & \ddots & \\ & & & & \ddots & \ddots & \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ \vdots & \vdots & \vdots \end{pmatrix} = \begin{pmatrix} m_{11} & m_{12} & 0 \\ m_{21} & m_{22} & m_{23} \\ m_{34} & m_{32} & m_{33} \\ m_{44} & m_{45} & m_{43} \\ m_{54} & m_{55} & m_{56} \\ \vdots & \vdots & \vdots \end{pmatrix}.$$

Algorithm Summary

Sandia National Laboratories

Setup:

- Partition matrix into block bordered form
- Factor diagonal blocks
- Form S' by probing

Apply:

- Xs=S'\b
  - Solve for S' using a few inner iterations
- Xi = Di\(bi-C * Xs)
  - Matvec and forward/back substitution

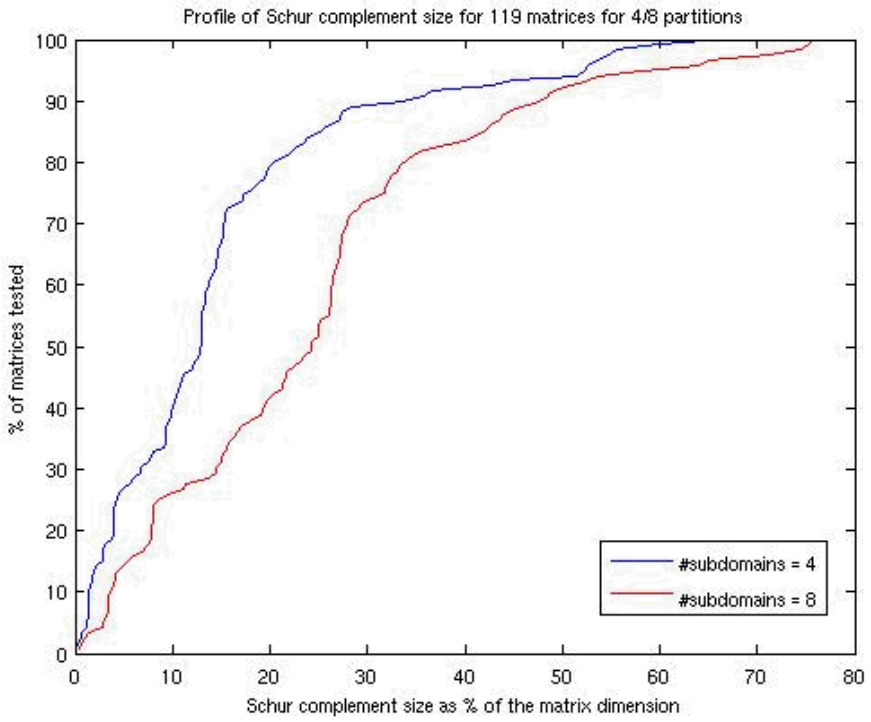
All steps can be done in parallel!

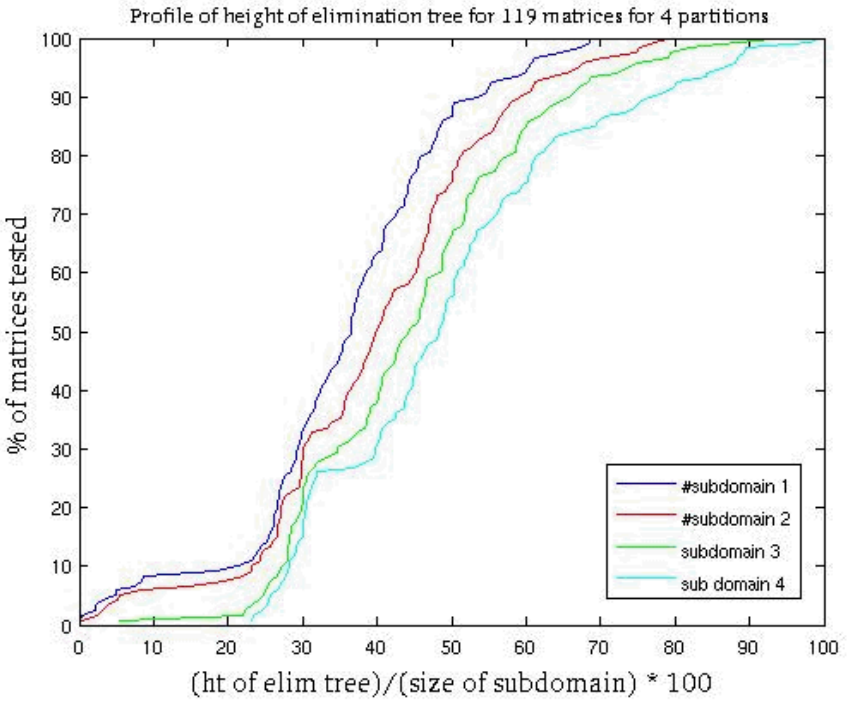
HyperLU Implementation Details

Sandia National Laboratories

- Implementation based on the Trilinos Framework
- Epetra based implementation for symmetric permutations with lfpack wrappers.
- Amesos/AztecOO for the solvers
- All the Matvecs are distibuted and the solves are local
- Currently iterations on the whole matrix even though the iteration over S is sufficient.
  - New interface will almost look like a direct solver (which will use an iterative solve underneath)

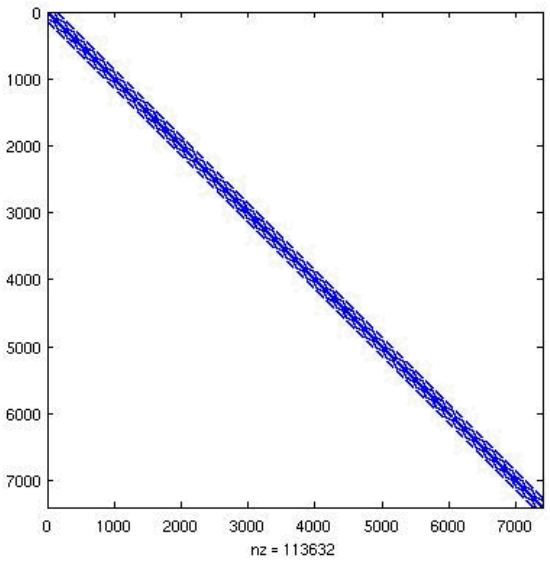
HyperLU: Preliminary results





- 180 matrices from UF sparse matrix collection, size 1k-10K
- Each local blocks are of the order of 2500x2500 – Need to exploit the parallelism in the UMA cores.
- The pessimistic elimination tree can provide only limited parallelism.

HyperLU preliminary results



- Finite Element matrix from MATLAB gallery.
- Compute partitions with Hypergraph partitioning with symmetric permutation.
- Use AztecOO solver with Ifpack wrapper of HyperLU

Dimension	nproc = 2	nproc = 4	nproc = 8
7500x7500	30	32	35
15000x15000	29	32	34
30000x30000	32	32	35

HyperLU preliminary results

- **Matrices of size 10K to 20K from UF sparse matrix collection, Tramanto, MATLAB (synthetic FEM)**
- **Number of iterations == 0 => No convergence**
- **HyperLU as good as other incomplete factorization preconditioners**

Matrix Name	HyperLU	ML	ILU	ILUT
Cage11	13	14	12	12
Crystm02	16	15	16	52
cbuckle	101	0	0	0
Lourakis	28	20	42	38
FIDAPex35	16	0	0	0
Oberwolfach	0	27	0	0
fem_3d_thermal	25	23	26	25
Dubkova1	56	55	189	154
Tramanto	112	0	0	0
wathenLarge	35	14	36	37

Work in Progress

- Two-level parallelism
  - Coarse-grain: Block structure from partitioning
  - Fine-grain: Multithreaded sparse direct solver
    - Can use Pardiso, SuperLU-MT, KLU2, …
- Incomplete factorization of blocks
  - With an outer iteration over the entire matrix, no need to solve subproblems exactly
- Open Question: How do we load balance the inner iteration?

Summary

- **HyperLU with two-level parallelism under development.**
  - **We intend to use this within a single node, so it may be used in a three-level scheme.**
  - **Hypergraph partitioning can partition for the UMA regions keeping the Schur complement small.**
  - **Multithreaded direct solvers need to scale for only a modest number of cores within the socket.**
  - **Flexible software framework allows any direct or incomplete solver on the subdomains.**
  - **Preliminary results are very promising.**
  - **Planning release in Trilinos.**

Backup

$$\begin{pmatrix} m_{11} & m_{12} & & & & \\ m_{21} & m_{22} & m_{23} & & & \\ & m_{32} & m_{33} & m_{34} & & \\ & & m_{43} & m_{44} & m_{45} & \\ & & & m_{54} & m_{55} & \ddots \\ & & & & \ddots & \ddots \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ \vdots & \vdots & \vdots \end{pmatrix} = \begin{pmatrix} m_{11} & m_{12} & 0 \\ m_{21} & m_{22} & m_{23} \\ m_{34} & m_{32} & m_{33} \\ m_{44} & m_{45} & m_{43} \\ m_{54} & m_{55} & m_{56} \\ \vdots & \vdots & \vdots \end{pmatrix}.$$