SAND2011-4733 C

SAND2011-4733C

Partitioning and Ordering for Sparse Linear Solvers

Erik Boman, Cédric Chevalier*, Siva Rajamanickam

Sandia National Laboratories, NM
*currently at CEA-DAM, France

ICIAM, July 21, 2011

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Outline

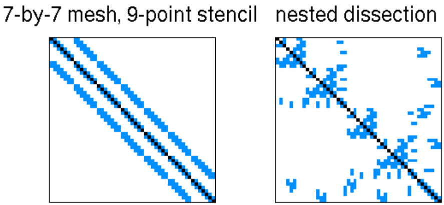
- Fill Reduction for Unsymmetric Problems
  - HUND
  - Parallel version.
- ShyLU Sparse hybrid solver
  - Schur Complement approach.
  - ShyLU implementation and Scalability results.
  - Combinatorial Issues.

Fill-reducing Ordering for Sparse Direct Solvers

- $Ax = b$, where A is large and sparse
- Find permutations P and Q such that PAQ reduces fill
- Classic, well studied problem
- NP-hard but good heuristics and software exist

SPD case

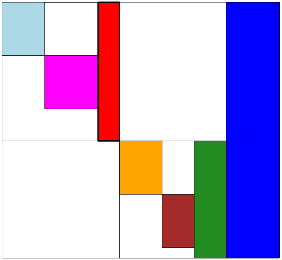
- Symmetric ordering: $Q = P^T$
- Usually no pivoting needed
- Two main categories of ordering methods:
  - Minimum degree etc. (best for small/medium problems)
  - Nested dissection (best for large problems)



Nonsymmetric case

- Minimum degree: COLAMD is a nonsymmetric version of AMD
- Nested dissection: Typically perform ND on symmetrized graph, either $A + A^T$ or $A^T A$.
- Q: Is there a better nested dissection method for nonsymmetric problems?

HUND

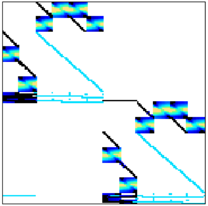


- HUND = Hypergraph Unsymmetric Nested Dissection (Grigori et al., 2010)
- Unsymmetric permutation into singly bordered block form
- Fill in LU is limited to colored blocks, even with pivoting
- Use any local column ordering inside the blocks
- Related work: MC66 (Hu et al.), MP48 (Duff & Scott)

Parallel HUND

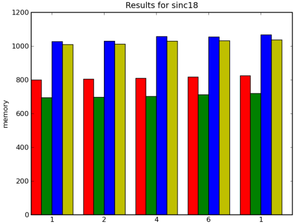
- HUND is a good fill-reducing ordering in serial
- HUND is extremely well suited for parallel computing
  - HUND gives divide-and-conquer parallelism (just like sym. ND)
  - HUND itself can be computed in parallel
  - Local ordering (e.g. COLAMD) can be performed in serial
- Parallel HUND is now in Zoltan, to be released in Trilinos
- Empirical study with parallel solvers still needs to be done

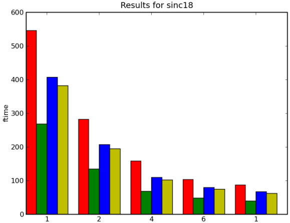
HUND Parallel Example: Sinc18



Compare four orderings:

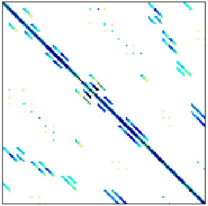
- COLAMD
- HUND
- Metis($A + A^T$)
- Scotch($A + A^T$)





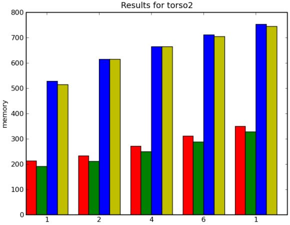
Sinc18 with up to 8 cores (threads) using SuperLU-MT.

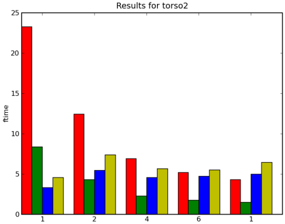
HUND Parallel Example: Torso2



Compare four orderings:

- COLAMD
- HUND
- Metis($A + A^T$)
- Scotch($A + A^T$)





Torso2 with up to 8 cores (threads) using SuperLU-MT.

ShyLU: Parallel Hybrid Sparse Solver

- Goal: Solve large-scale sparse linear systems on modern architectures
- Leverage existing parallel iterative solvers across nodes (Multigrid, Domain decomposition etc)
- Need better sparse solver on the node
  - Core counts increase rapidly
  - Use threaded or hybrid programming model
- Avoid the iteration creep by partitioning for the node.
- Should be hybrid in the mathematical sense as well.
- HyperLU is a new hybrid solver. It can be used
  - As a stand-alone iterative solver
  - As a preconditioner for subdomains

Schur Complement Framework

Solve $Ax = b$ where A has the form

$$A = \begin{pmatrix} D & C \\ R & G \end{pmatrix}, \tag{1}$$

where D and G are square and D is non-singular.

The Schur complement $S = G - R * D^{-1}C$.
Solving $Ax = b$ then consists of the three steps:

1. Solve $Dz = b_1$.
2. Solve $Sx_2 = b_2 - Rz$.
3. Solve $Dx_1 = b_1 - Cx_2$.

The exact S is almost dense, but may be applied as an operator.

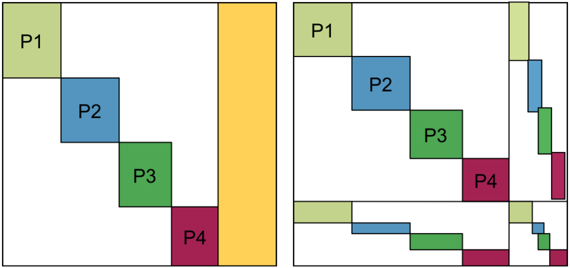
Hybrid Solvers: Related Work

Recently, several parallel hybrid (direct-iterative) solvers have been developed:

- HIPS (Gaidamour, Henon)
- PDSlin (Li, Ng, Yamazaki)
- MaPhys (Giraud, Haidar, et al.)

They differ in how they approximate Schur complements, and how they partition/reorder the matrix.

Schur Complement Framework: Partitioning



(a) (b)

- 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.
- Our implementation uses the symmetric permutation now.

Schur Complement Framework: Schur Approximation, Dropping

We can apply S implicitly as an operator, but need an explicit preconditioner.

$\bar{S} \approx S = G - R * D^{-1}C$.

Compute the Schur complement and drop the relatively smaller entries to find an approximate Schur complement. We have implemented two strategies: *dropping* and *probing*.

The Schur complement computation is fully parallel (or not) depending on whether we use a wide separator or a narrow separator.

ShyLU Scalability Results

Figure: Dropping

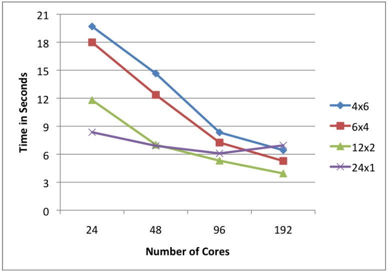
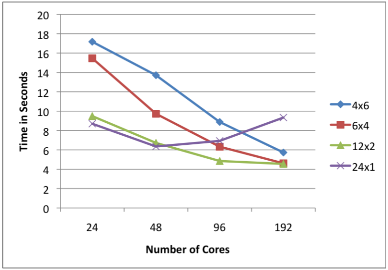


Figure: Probing



- ShyLU is implemented using the various components of the Trilinos framework. (Zoltan/Isorropia for partitioning, Epetra for basic foundation, AztecOO/Belos for itertive solves, Amesos for direct solves)
- Comparing various combination of MPIxTasks for Strong Scaling of a 2D finite element problem of size 360Kx360k.

ShyLU Scalability Results

Figure: Dropping

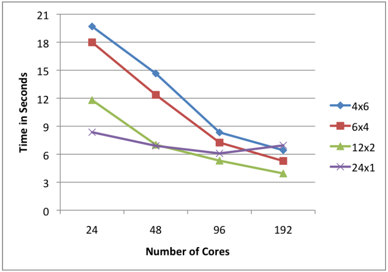
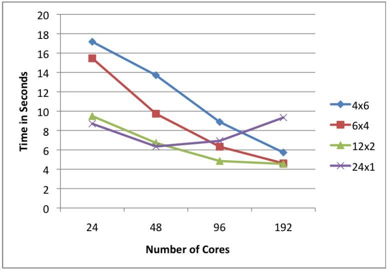


Figure: Probing



- Tests on the NERSC machine hopper with 24 cores (NUMA cores in a configuration of 4x6). The configuration uses hypergraph partitioning, Pardiso for the direct solve, inexact solve for the Schur complement, inner outer iteration using Belos.
- Hybrid methods perform better that flat MPI based methods as the problem size per core gets smaller.

ShyLU Robustness Results

Table: Comparison of number of iterations of ShyLU probing and dropping with ILU(1) and ILUT(2, 1e-8). A dash indicates no convergence

Matrix Name	N	Symm	Dropping	Probing	ILU	ILUT
Pres_Poisson	14.8K	Symm	74	53	-	-
bodyy5	18K	Symm	76	76	173	109
Lourakis_bundle1	10K	Symm	33	29	38	31
FIDAP_ex35	19K	Unsymm	5	8	-	-
igbt3	10.9K	Unsymm	29	18	-	-
FEM_3D_thermal2	147K	Unsymm	12	8	24	23
venkat50	62.4K	Unsymm	40	33	-	-
airfoil2d	14.2K	Unsymm	25	15	153	97
nmos3	18.5K	Unsymm	30	-	-	-
FEMLAB_waveguide3D	21K	Unsymm	130	-	-	-
TC_N_360K	360K	Symm	58	48	342	201
Tramanto1	6K	Unsymm	114	-	-	-

Narrow Separator vs Wide Separator

Let (V_1, V_2, S) be a partition of the vertices V in a graph $G(V, E)$.
Definitions:

Separator: S is a separator if there is no edge (v, w) such that $v \in V_1$ and $w \in V_2$.

Wide separator: Any path from V_1 to V_2 contains at least two vertices in S.

Narrow separator: A separator that is not wide.

A wide separator is never a minimal separator. A narrow separator may be minimal but this is not required.

Narrow Separator vs Wide Separator

Figure: Narrow Separator for a 96x96 matrix with 4 parts

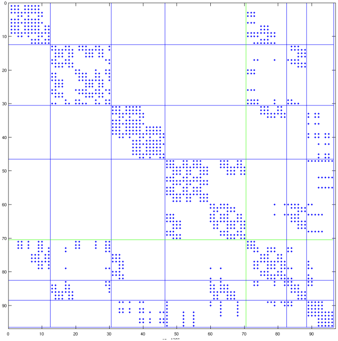
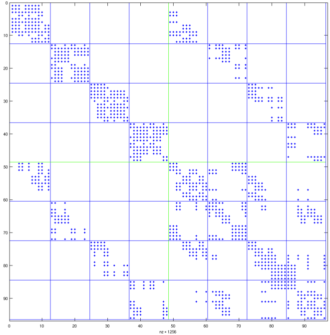


Figure: Wide Separator for a 96x96 matrix with 4 parts



The narrow separator can be as much as half the size of the wide separator. Wide separator results in a nice structure in the row and the column separators of the matrix.

Narrow Separator vs Wide Separator

The Schur complement itself has a block structure

$$S = \begin{pmatrix} S_{11} & S_{12} & \dots & S_{1k} \\ S_{21} & S_{22} & \dots & S_{2k} \\ \vdots & \vdots & & \vdots \\ S_{k1} & S_{k2} & \dots & S_{kk} \end{pmatrix} \qquad (2)$$

- When using the narrow separator, in a parallel setting, each processor can contribute to all the blocks of the Schur complement.
- When using the wide separator, each processor can contribute to only its diagonal blocks of the Schur complement. $S_{ij} = G_{ij}$ when $i \neq j$.
- ShyLU can use both types of separators for probing and dropping. All the following results use wide separators.

Narrow Separator vs Wide Separator

$$A_{narrow} = \begin{pmatrix} D_{11} & 0 & C_{11} & C_{12} \\ 0 & D_{22} & C_{21} & C_{22} \\ R_{11} & R_{12} & S_{11} & S_{12} \\ R_{21} & R_{22} & S_{21} & S_{22} \end{pmatrix} \quad (3)$$

$$A_{wide} = \begin{pmatrix} D_{11} & 0 & C_{11} & 0 \\ 0 & D_{22} & 0 & C_{22} \\ R_{11} & 0 & S_{11} & S_{12} \\ 0 & R_{22} & S_{21} & S_{22} \end{pmatrix} \quad (4)$$

The Schur complement is $S = G - R * D^{-1} * C$.

- For example, while using the narrow separator, $S_{11} = G_{11} - R_{11} * D_1^{-1} * C_{11} + R_{12} * D_2^{-1} * C_{21}$ which requires communication. This applies to all the blocks of the Schur complement.
- When using the wide separator, $S_{11} = G_{11} - R_{11} * D_1^{-1} * C_{11}$ can be computed locally. This applies to all the diagonal blocks of the Schur complement.
- When using the wide separator, the off-diagonal blocks of the Schur complement are equal to the off-diagonal blocks of G.

Choice of Partitioning Methods

What Partitioners do:

- Graph Partitioning: Minimize the number of non zeros in the off-diagonal blocks of G.
- Hypergraph Partitioning:
  - $\lambda - 1$ metric: Minimize the communication volume for the mat-vec in the outer iteration.
  - Cutnet metric: Minimize the number of cut hyperedges.

Choice of Partitioning Methods

What Partitioners do:

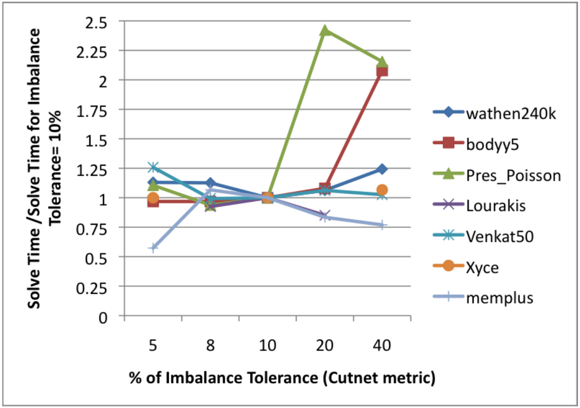
- Graph Partitioning: Minimize the number of non zeros in the off-diagonal blocks of G.
- Hypergraph Partitioning:
  - $\lambda - 1$ metric: Minimize the communication volume for the mat-vec in the outer iteration.
  - Cutnet metric: Minimize the number of cut hyperedges.

Solvers' Wishlist

- Balance the work in the outer iteration (direct solves on blocks)
- Minimize the communication volume for the mat-vec in the outer iteration.
- Reduce number of rows in G, with a good degree of parallelism.
- Minimize non-zeros in G (the work in the inner iteration)
- Well balanced G - minimizing the communication volume in inner iteration.

Effect of the Imbalance Tolerance

Figure: Run time vs. Imbalance tolerance.



We see that 10% imbalance is quite good for a most test matrices.

Conclusion

- Partitioning and ordering are important for performance in sparse linear solvers.
- HUND is promising ordering for parallel LU solvers.
- ShyLU is a flexible hybrid-hybrid solver that can scale well in modern architectures.
- Planning release in Trilinos (late 2011).
- Wide separators may result in more parallelism and faster solver than narrow separators.
- Many open partitioning problems.

Extra Slides

Schur Complement Framework: Fast Approximate Solve

Multiple options for the Solve:

- Can use the $\bar{S}$ as the preconditioner matrix and solve for the operator. (iteration on S is sufficient assuming D was solved exactly.)
- Can compute an inexact solve $\bar{S}$ resulting in an inner-outer iteration to solve for A.
- Can solve for D inexactly as well, in addition to using another preconditioner for the inexact solve of $\bar{S}$. This requires an inner-outer iteration as well.

The preconditioner for $\bar{S}$ can be a relatively cheap one. The later two approaches are the ones needed for subdomain solver.

Comparing Solver Performance with different Partitioning Methods

Matrix Name	Method Method	rows in G	nnz in G	Outer Iter	Solve Time	Inner LB
wathen60k	Graph	1736	20302	10	0.6	1.12
	Cutnet	1586	17964	12	0.71	1.21
	lambda	**1581**	17925	10	0.57	1.16
wathen240K	Graph	**3940**	46828	10	2.61	1.10
	Cutnet	5641	66117	9	**2.21**	1.23
	lambda	4201	49293	9	2.31	1.22
bodyy5	Graph	577	2865	59	0.68	1.08
	Cutnet	**523**	2595	55	0.64	1.05
	lambda	533	2639	55	0.644	1.05
Pres_Poisson	Graph	**1248**	42960	46	1.61	1.34
	Cutnet	1472	52984	40	1.63	1.42
	lambda	1816	65336	90	3.82	1.45
Lourakis	Graph	**3267**	334917	19	0.35	1.40
	Cutnet	3279	322563	19	0.68	**2.21**
	lambda	3300	325608	18	0.608	**1.94**

Comparing Solver Performance with different Partitioning Methods

Matrix Name	Method Method	rows in G	nnz in G	Outer Iter	Solve Time	Inner LB
FEM_3D_Thermal2	Graph	**5498**	95380	9	3.28	1.06
	Cutnet	5628	93630	11	3.88	1.11
	lambda	6226	105170	10	3.63	1.32
venkat50	Graph	**1468**	29200	129	13.12	1.32
	Cutnet	1608	32000	170	16.87	1.18
	lambda	1756	34960	143	15.93	1.52
Xyce_1	Graph	*				
	Cutnet	10330	305120	1	0.15	1.64
	lambda	**9712**	293132	1	0.15	1.17
ckt11752_dc_1	Graph	957	2459	214	9.88	1.46
	Cutnet	650	1393	*	*	2.70
	lambda	**638**	1529	*	*	2.52
memplus	Graph	4917	54267	357	7.75	1.33
	Cutnet	**4174**	49070	349	8.64	1.90
	lambda	4604	52230	299	**6.43**	1.38