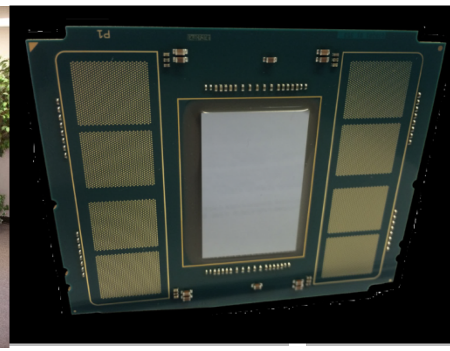
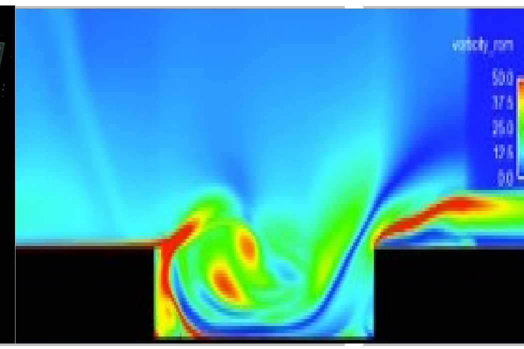
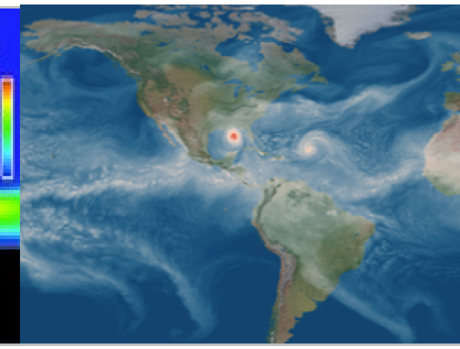
SAND2018-2195C

A Hierarchical Low-Rank Solver for Sparse Linear Systems and Its Variations

Erik Boman, Chao Chen, Eric Darve, Siva Rajamanickam, Ray Tuminaro,

SIAM PP18, Tokyo, Mar 2018

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. SAND NO. 2011-XXXXP

Hierarchical Low-Rank (HLR) Matrices and Solvers

- Hierarchical matrices and solvers currently popular
  - Alphabet soup: H, H2, HODLR, HSS, …
- Key insight: Many matrices have useful (rank) structure
  - Blocks far from diagonal can be approximated using low-rank
    - Holds for elliptic PDEs, some other (e.g., advection-diffusion problems)
    - Similar intuition as for Fast Multipole Methods (FMM)
    - May also apply to data science (e.g. covariance matrix)
- Stanford/Sandia collaboration. Our goals
  - Develop solver/preconditioner based on hierarchical matrices
  - Speed up sparse direct solvers (high accuracy)
  - Use as preconditioner (low accuracy)

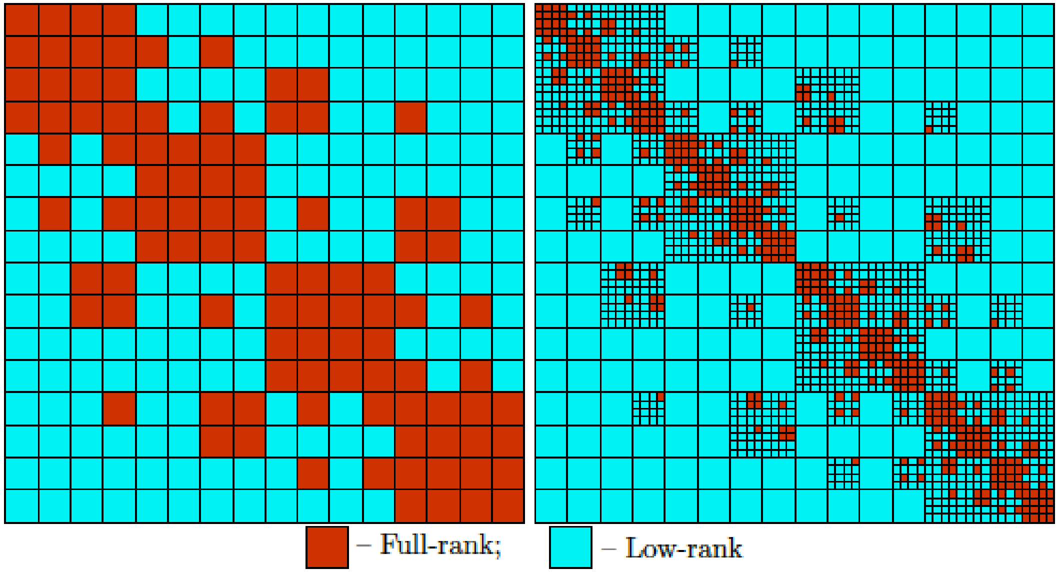
Low-Rank Structure

Low-rank structure often occurs in *off-diagonal blocks in*

- The inverse of A
- The LU factors of A

Hierarchical formats:

- HSS and HODLR may need high ranks in some blocks
- H and H2 need lower ranks due to more flexible partition



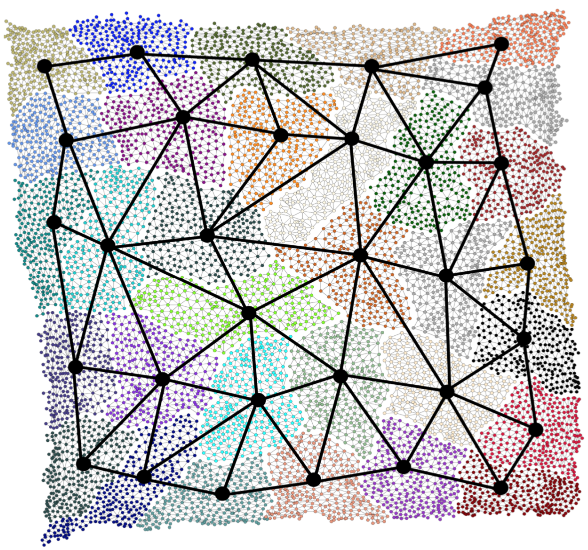
IFMM figure from Ambikasaran & Darve.

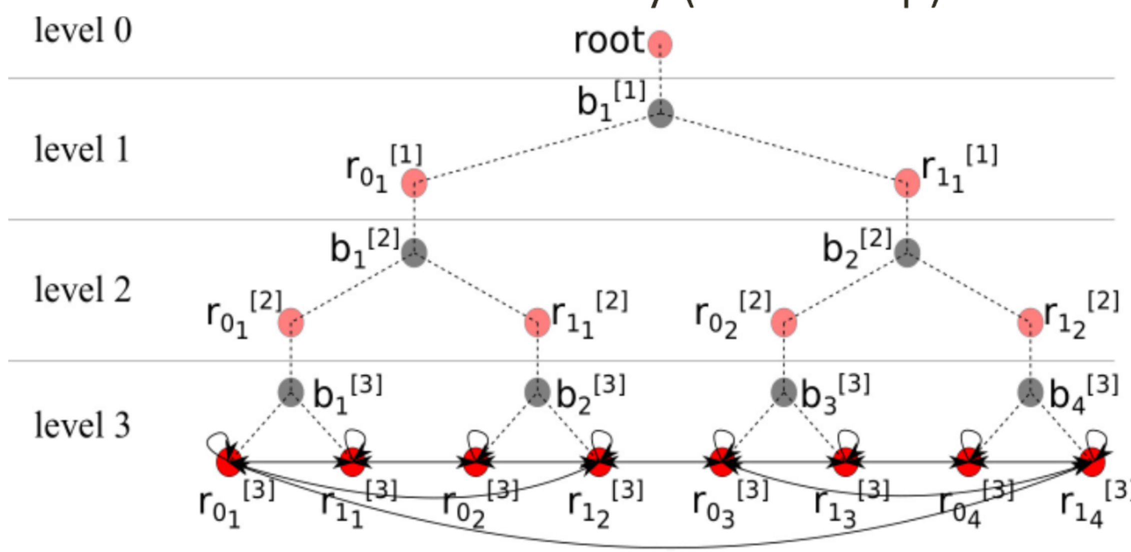
Fast Sparse Solver Approaches

- Early work was on *dense* matrices; we focus on *sparse*
- Approximate LU factors of A
  - Dense frontal matrices within sparse direct (multifrontal) method
    - Xia, Li, Ambisakaran/Darve, Ho/Ying, …
    - MUMPS, Pastix, Strumpack software
    - Rely on nested dissection ordering with separators
  - Work on entire sparse matrix (sparse triangular factors)
    - H-LU (Hackbusch, Grasedyck, Kriemann, LeBorne, …)
    - *LoRaSp* (Pouransari, Coulier, Darve)
      - Relies on graph partitioning (edge separators), as in domain decomp.

The LoRaSp/ParH2 Method

- LoRaSp: Pouransari, Coulier, Darve, SISC 2017
  - Parallel version by Chen et al, Parallel Computing, 2007.
- Partition graph via recursive bisection, gives a tree
- Eliminate “clusters” (matrix blocks) starting at leaf level
  - Approximate block LU factorization
  - New “coarse” dof via *extended sparsification*
- Merge neighbors, repeat for each level in the hierarchy (bottom up)





Figures courtesy Chen et al., and Pouransari et al.

5

Multilevel Block Incomplete Factorization

- Algebraic Interpretation:
  - LoRaSp/H2 solver can be viewed as a variation of Block ILU factorization
  - A= LU+E, where E has block structure
    - approximate E blockwise by low rank
  - We compensate for the dropped blocks by adding new rows/columns to the matrix (*extended sparsification*)
  - The Schur complement for these *coarse* vertices is a smaller matrix we can solve recursively
- Low-rank approximation is different from earlier ILU work
  - We use H2 structure only implicitly

6

Extended Sparsification

- For simplicity, assume symmetric A (can be extended)
- Suppose the off-diagonal blocks are (approx) low-rank:

$$\begin{pmatrix} A_1 & UV^T \\ VU^T & A_2 \end{pmatrix}$$

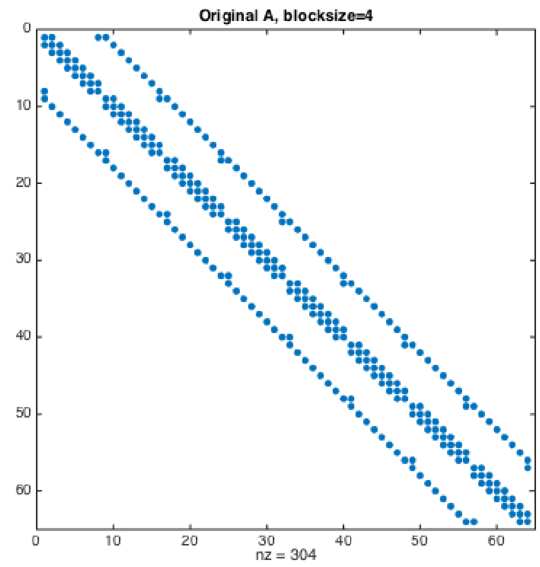
- We solve the equivalent extended system

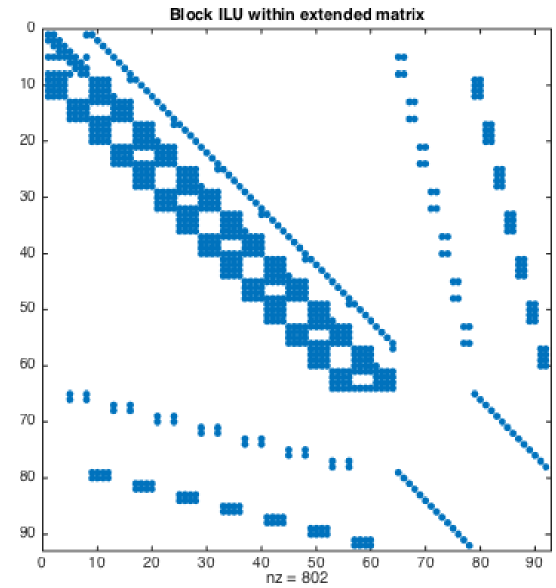
$$\begin{pmatrix} A_1 & 0 & U & 0 \\ 0 & A_2 & 0 & V \\ U^T & 0 & 0 & -I \\ 0 & V^T & -I & 0 \end{pmatrix}$$

- We sparsify the original matrix, but add extra rows/cols that also need to be factored.
- The lower the ranks of U,V, the smaller extended system

ILU and Extended Sparsification

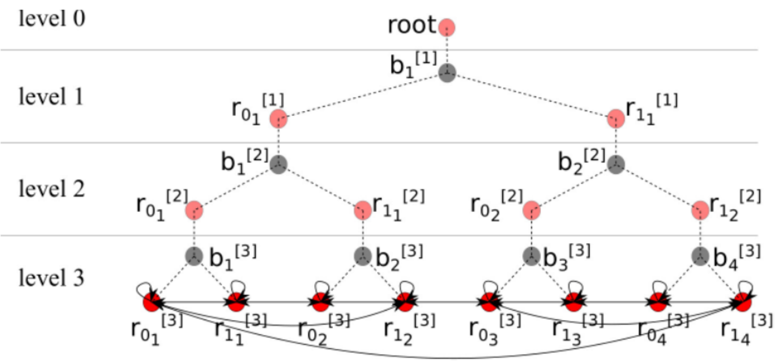
- Note: Fill blocks in the block LU factors often have low rank
  - Schur complements in the Gaussian elimination
- Approach: Compute blocks in ILU(0) exactly, and
  - Approximate blocks in ILU(1) (not in ILU(0)) using low-rank
  - Extend matrix with new rows/cols





Parallel Approaches

- Cluster tree gives dependencies
- Work on one level at a time
  - Bottom up, can switch near top of tree
  - Note: Each leaf not independent, cross edges typically exist!
- Similar to sparse direct solver at each level
  - Can use nested dissection on (each level of) cluster graph
  - Tree-based parallelism within each level – too complicated!
- Fill is limited to distance-2 vertices
  - Allows simpler parallel method

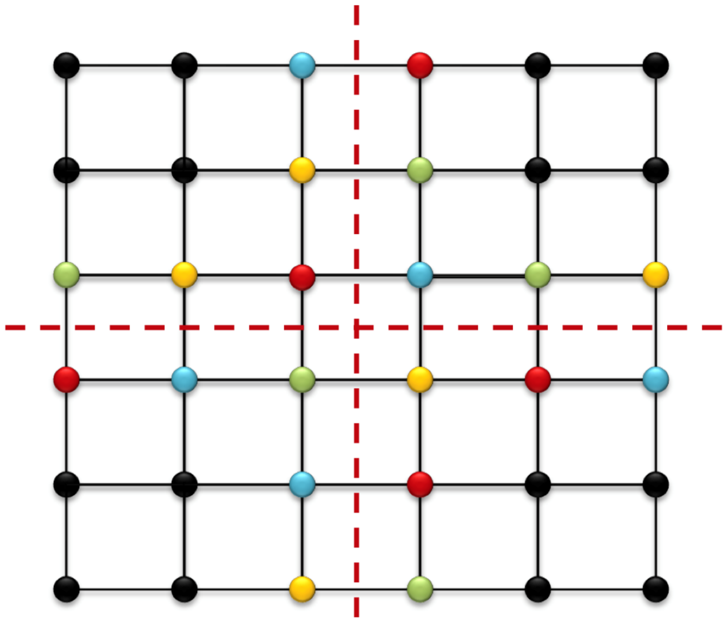


Tree level structure for algorithm. *Figure courtesy Pouransari.*

9

ParH2 Parallel Algorithm

- Work on one tree level at a time
  - Bottom up, can switch near top of tree
- Data parallel: Each processor works on a subgraph (subdomain).
- Consider the cluster graph:
  - Only boundary vertices need communication.
  - Interior vertices can be eliminated independently in parallel.
- Use graph coloring on the boundary to find concurrent work
  - #synchronization points = #colors
- Can overlap boundary and interior vertices
  - That is, overlap communication and computation



Example: 4 processors. Each vertex (node) corresponds to a cluster of variables.
Figure courtesy Chao Chen.

10

Experiments

- ParH2 solver (and results) by Chao Chen
  - Parallel extension of LoRaSp serial code
  - MPI everywhere
  - Eigen library for dense linear algebra (on node)

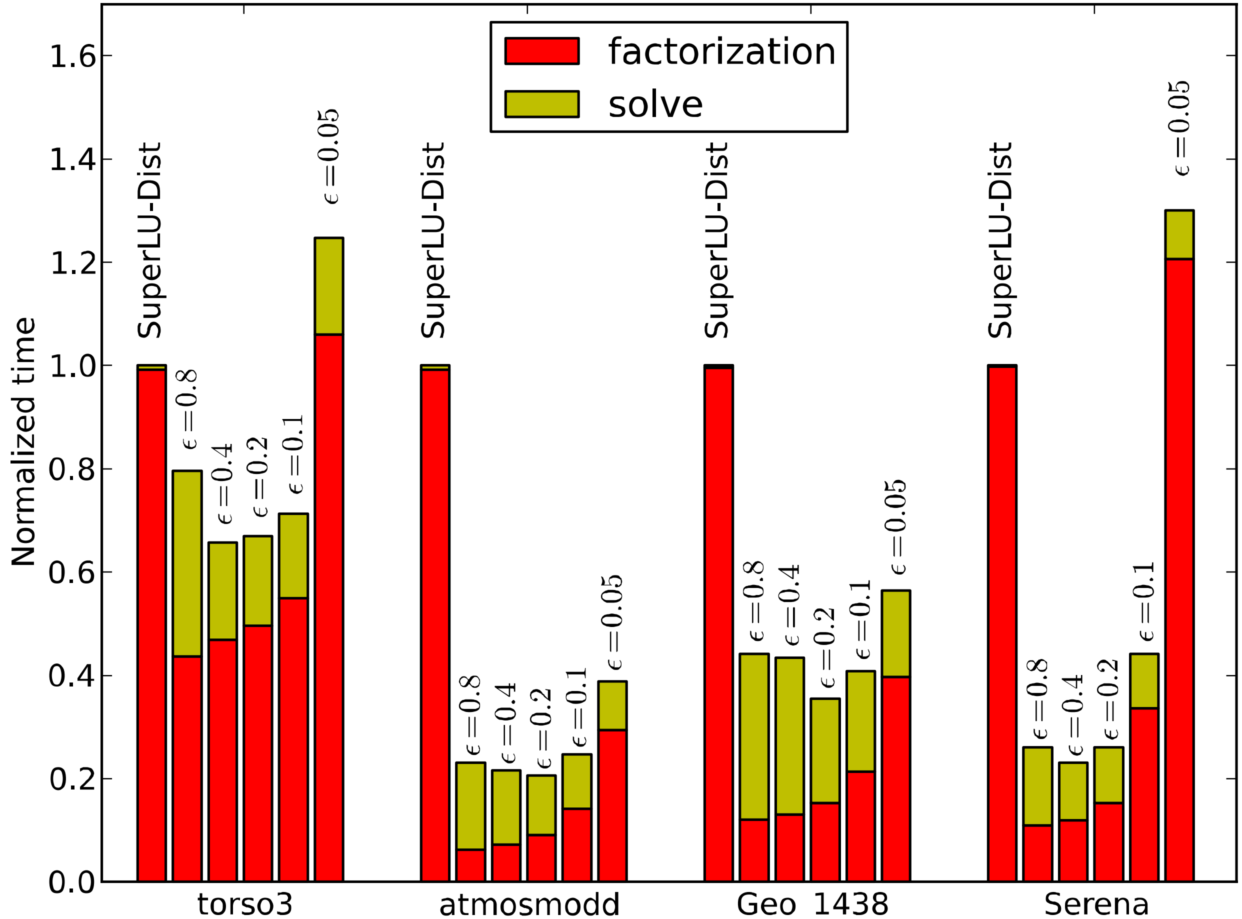
- SVD for low-rank compression
  - Fixed eps in matrix compression
  - Matrix ranks will vary

- Platform: Cray XC30 (Edison/NERSC)
  - Used 16 (out of 24) cores per node
  - Used up to 16 nodes (256 cores)

11

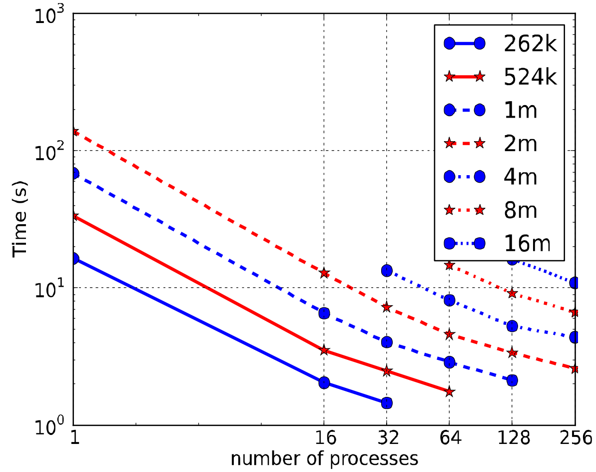
Results: Compare sparse direct.

Compare hierarchical solver as preconditioner vs. SuperLU-Dist direct solver on irregular problems from UF/SuiteSparse. Vary compression threshold epsilon. 16 processors (MPI ranks).

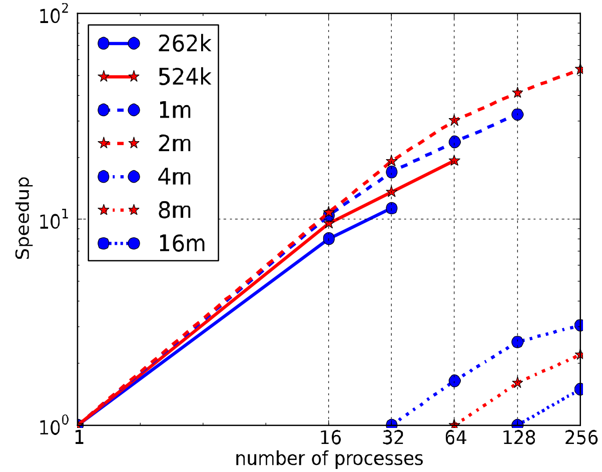


12

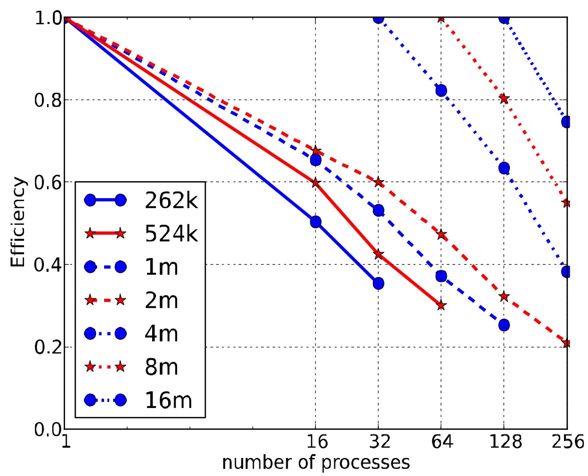
Results: 3D Poisson Eqn.



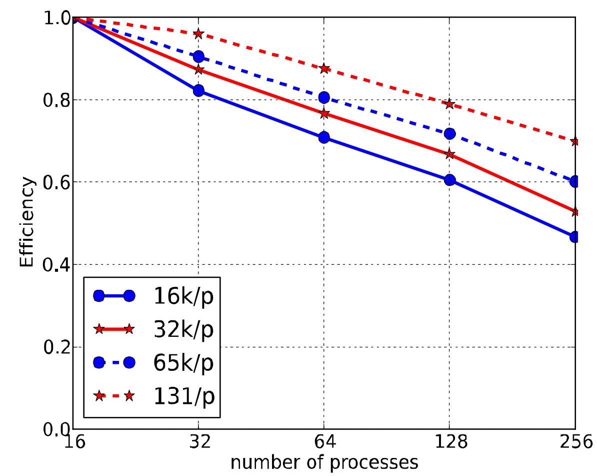
(a) Factorization time



(b) Factorization speedup



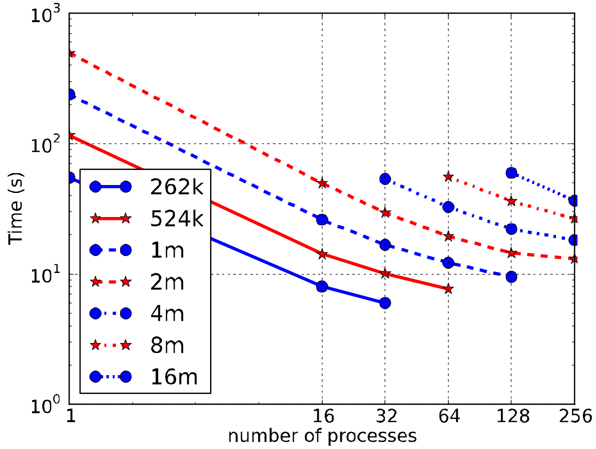
(c) Fixed total problem size



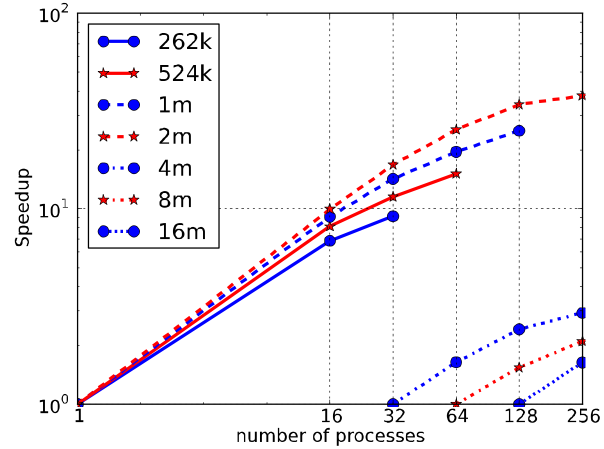
(d) Fixed problem size per processor

13

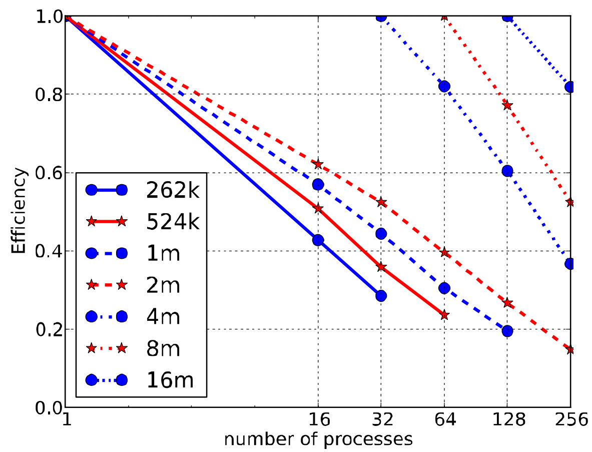
Results: Helmholtz eqn.



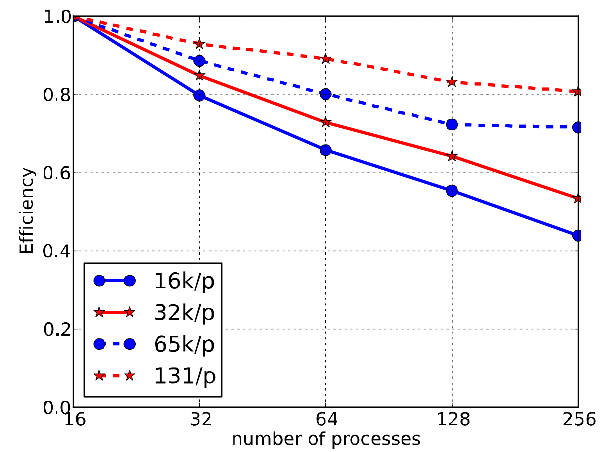
(a) Time



(b) Speedup



(c) Fixed total problem size



(d) Fixed problem size per processor

Improving Convergence

Idea: Improve convergence by preserving certain vectors on coarser levels (similar to AMG)

- Yang, Pouransari, Darve, *arXiv 1611.03189 (2016)*
- Preserve the near-null-space
  - often corresponds to translation & rotation

Test problem: Poisson on 2.5D box, Robin b.c.

- Motivated by ice sheet simulation
- Trade-off memory & cost per iteration vs #iterations

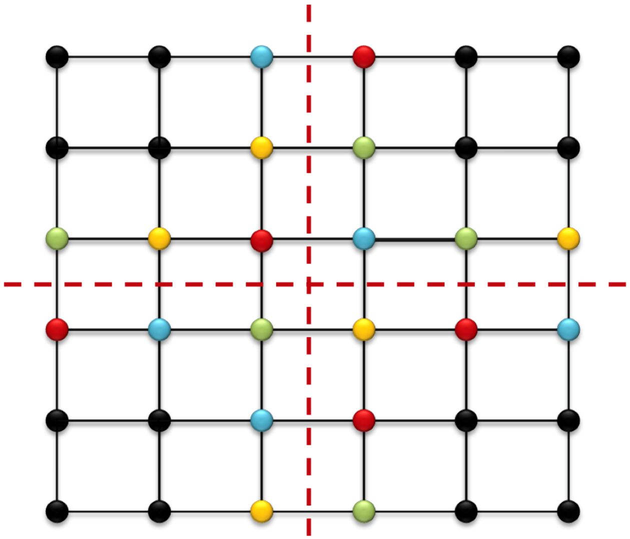
Method/ mesh	Orig. #iter	Const. #iter	Linear #iter	Orig. Nnz/row	Const. Nnz/row	Linear Nnz/row
16^2x8	8	8	5	135	176	255
32^2x8	12	10	6	169	226	357
64^2x8	18	12	7	180	243	406
128^2x8	25	15	7	188	250	439

Results due to Ray Tuminaro

15

Variations on Graph Coloring

- Processor/domain coloring
  - Color the processor graph
  - Dist-1 coloring
  - Most processors will be idle each phase
- Node/boundary coloring
  - Color the nodes along the boundaries
  - Dist-2 coloring
  - Most processors will have some work to do in each phase



Node coloring. Each node (vertex) corresponds to a cluster.
Figure courtesy Chao Chen.

Table 1: Node coloring (distance-2 coloring)

N	level	# nodes	# procs	# colors	makespan	coloring time	computation	communication
512^2	12	828	2^2	119	420	1.1e-2	1.71363	1.77387
1024^2	14	4844	4^2	145	534	7.5e-1	4.05081	10.4608
64^3	12	3432	2^3	288	811	3.9e-2	3.29297	3.17157

Table 2: Domain coloring

N	level	# nodes	# procs	# colors	makespan	coloring time	computation	communication
512^2	12	828	2^2	40	828	5.4e-4	1.60225	4.85464
1024^2	14	4844	4^2	60	2216	1.7e-3	4.23214	35.6914
64^3	12	3432	2^3	36	1716	7.1e-4	3.62744	16.9539

16

Graph Coloring: Work in Progress

- Distance-2 node coloring is too pessimistic
  - Clusters on the same processor are processed sequentially, so can have the same color
  - Only need different colors for paths across processor boundaries
  - Used it mainly because software was available (Zoltan)
- Better strategies
  - General greedy algorithm for new coloring problem
  - Custom heuristic based on faces/edges in the decomposition
- Both ways, we will reduce colors and improve performance!

Conclusions

- Hierarchical low-rank methods (HLR) augment the current solver/preconditioner ecosystem.
  - Faster than sparse direct
  - Most useful as preconditioner
- Setup phase can be expensive.
  - Can often amortize this cost over multiple solves
- Well suited for modern architectures
  - Most work is in dense linear algebra (even for sparse problems)
    - High arithmetic intensity
- Early days:
  - Several algorithm options, best choice unclear
  - Codes are still immature but rapidly improving

18

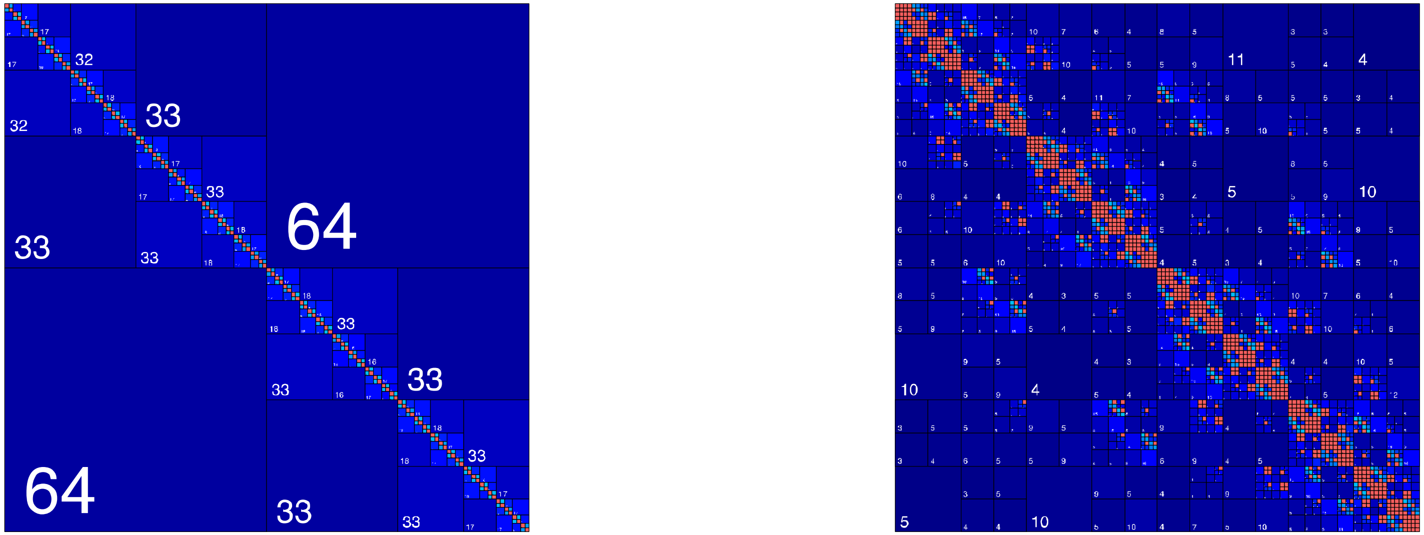
Backup Slides

19

Hierarchical Matrix Formats

	Simple basis	Nested basis
Weak admissibility	HODLR	HSS
Any admissibility	H	**H^2**

- HSS is perhaps most popular but has drawbacks
  - Weak admissibility may require high ranks (esp. 3D problems)
- Example: Inverse of 2D Poisson eqn. *(Courtesy G. Chavez et al.)*

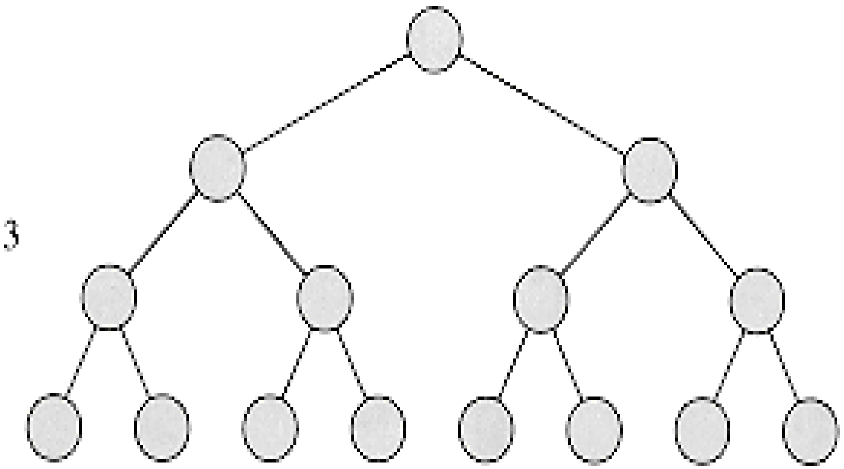


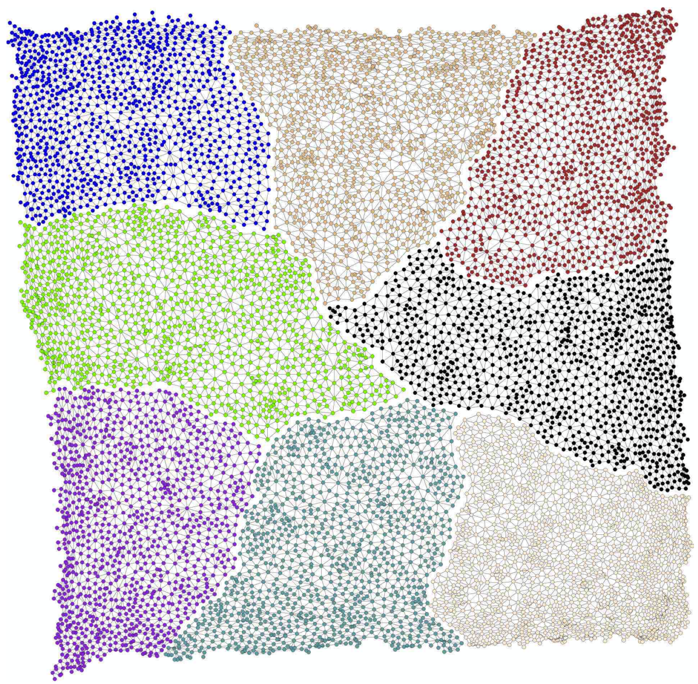
(a) Weak admissibility.

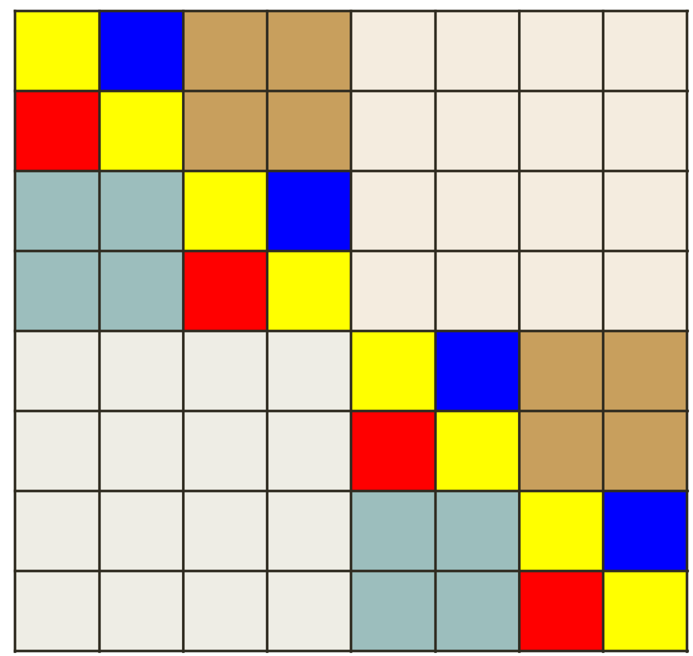
(b) Standard admissibility.

HLR: Tree and Matrix

- Recursively bisect the vertices (clustering)
- Corresponds to a binary tree
- Matrix: Low-rank approximation of off-diagonal blocks
  - Only if “well-separated”
  - How to choose ranks?
  - Use SVD, ACA, ULV, or RRQR?







Our Hierarchical Low-Rank Sparse Solver

- Collaboration Darve (Stanford) & Sandia
- Solver based on recent H^2 methods by Darve et al.
  - IFMM (dense), LoRaSp (sparse)
- Uses block approximate LU factorization with low-rank compression for "well separated" interactions
  - Partition matrix, build H-tree, factor approximately
  - Leaves are subdomains, internal vertices correspond to approximate Schur complements (low rank)
  - Tree implicitly gives approximate LU factorization
- New method, differs from multifrontal HSS
  - Simpler, no trees within trees

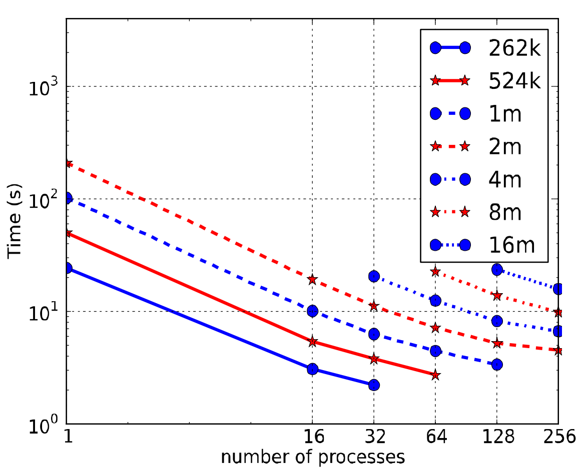
22

Variations

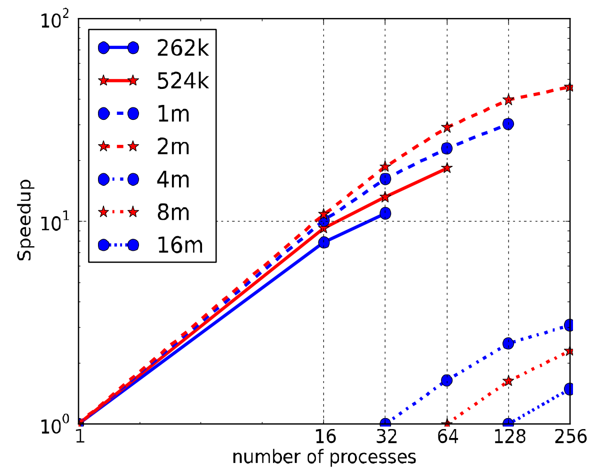
- Extended vs in-place sparsification
- Ordering of clusters
- Definition of “well separated” – could allow ILU(k)
- Aggressive “coarsening” – merge more clusters
- Compress-eliminate or eliminate-compress?
- Low-rank compression: SVD, RRQR, RRLU, ID, etc.

23

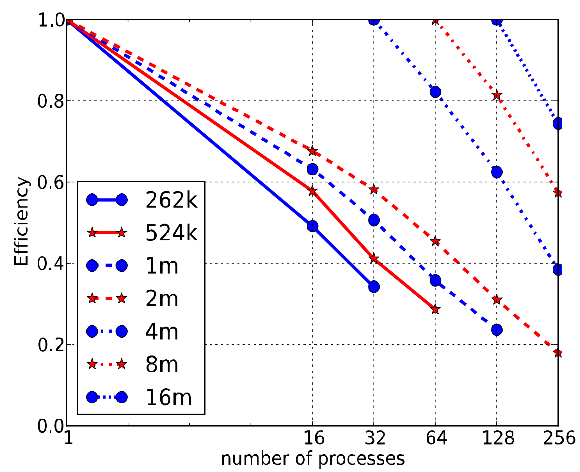
Results: Variable coeff. Poisson



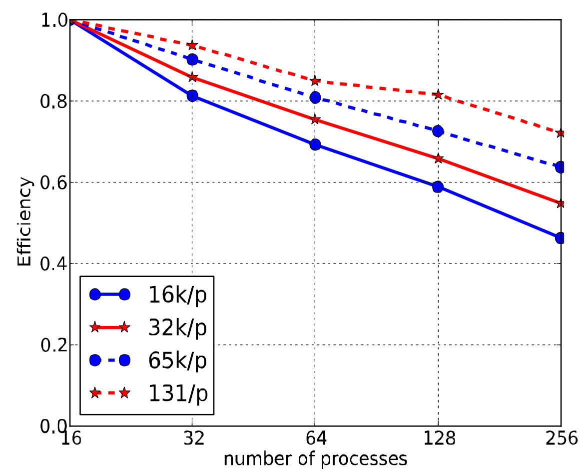
(a) Time



(b) Speedup



(c) Fixed total problem size



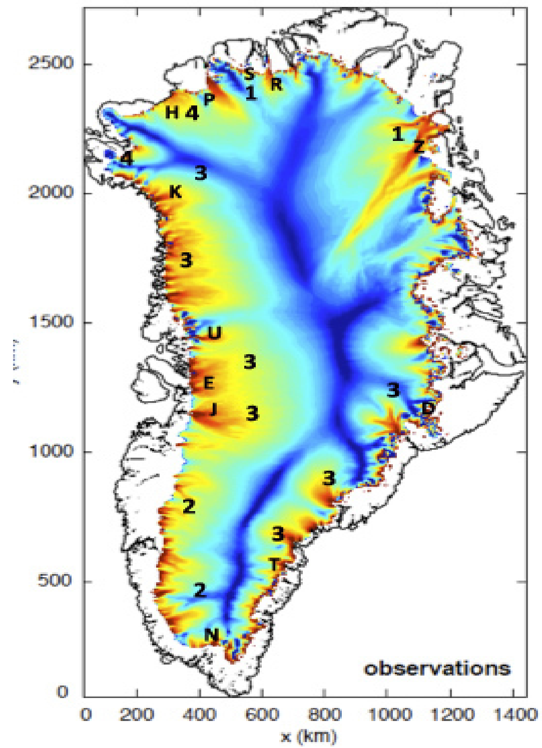
(d) Fixed problem size per processor

24

Ice Sheet Modeling: Greenland

We simulate ice sheet flow using Stokes' eqn. Use Albany/Felix software, Trilinos solvers for linear systems. 2.5D geometry is challenging as the z dimension is very different.

Precond.	8km mesh	4km mesh
ML	-	-
ML/custom	18	17
ILU (custom order)	12	21
H2(1e-1)	141	423
H2(1e-2)	36	153
H2(1e-1)*	19	21
H2(1e-2)*	14	13



* This version uses x-y mesh partitioning and treats "diagonal" grid points as neighbors (not well sep.)