Towards an Optimal Parallel Approximate Sparse Factorization Algorithm Using Hierarchically Semi-separable Structures

X. Sherry Li
Lawrence Berkeley National Laboratory

CACHE – Algorithms and Software for Communication Avoidance and Communication Hiding at the Extreme Scale (Math/CS Institute)

Collaborators: Shen Wang, Jianlin Xia, Maarten V. de Hoop (Purdue University)

DOE Applied Mathematics Program Meeting, Oct. 17-19, 2011

CACHE targets exascale computers & simulations

- Algorithm efficiency depends ...
  - less on FLOPS, more on data movements (on-node memory access, inter-node communication)
- Problems with 3D geometry
- Indefinite, ill-conditioned problems

The problem

- Solving sparse $Ax = b$ by Gaussian elimination: $A = LU$

↑ Deliver reliable solution, error bounds, condition estimation, efficient for many RHS, . . .

↓ **Complexity wall** ... far from linear

Serial [George '73, Hoffman/Martin/Rose, Eisenstat, Schultz and Sherman]

For model problems, Nested Dissection ordering gives optimal complexity in exact arithmetic

- 3D ($K^3 = N$ grids): $\mathcal{O}(N^{4/3})$ MEM, $\mathcal{O}(N^2)$ FLOPS

Parallel: [Gupta/Karypis/Kumar '97] (WSMP solver)

Subtree-subcube mapping, 2-dim. matrix partitioning

- 3D: $\mathcal{O}(N^{4/3}/\sqrt{P})$ COMM-Volume
- Flop-to-Byte ratio: $\mathcal{O}(\frac{N^{2/3}}{\sqrt{P}})$

  (c.f., ScaLAPACK dense LU: $\mathcal{O}(\frac{N}{\sqrt{P}})$)

Breaking the complexity wall

Exploit "data-sparseness" in dense submatrices

- data-sparse: matrix may be dense, but has a compressed representation smaller than N^2

What types of data-sparse representations?

- Fast multipole method [Greengard, Roklin, Starr, et al.]
- Hierarchical matrices: $\mathcal{H}$-matrix, $\mathcal{H}^2$-matrix [Bebendorf, Borm, Grasedyck, Hackbusch, Le Borne, Martinsson, Tygert, et al.]
- **Semi-separable matrices, Quasi-separable matrices** [Bini, Chandrasekaran, Dewilde, Eidelman, Gohberg, Gemignani, Gohberg, Gu, Kailath, Olshevsky, van der Veen, Van Barel, Vandebril, White, et al.]
- Others . . .

Breaking the complexity wall

Exploit "data-sparseness" in dense submatrices

- data-sparse: matrix may be dense, but has a compressed representation smaller than N^2

What types of data-sparse representations?

- Fast multipole method [Greengard, Roklin, Starr, et al.]
- Hierarchical matrices: $\mathcal{H}$-matrix, $\mathcal{H}^2$-matrix [Bebendorf, Borm, Grasedyck, Hackbusch, Le Borne, Martinsson, Tygert, et al.]
- **Semi-separable matrices, Quasi-separable matrices** [Bini, Chandrasekaran, Dewilde, Eidelman, Gohberg, Gemignani, Gohberg, Gu, Kailath, Olshevsky, van der Veen, Van Barel, Vandebril, White, et al.]
- Others . . .

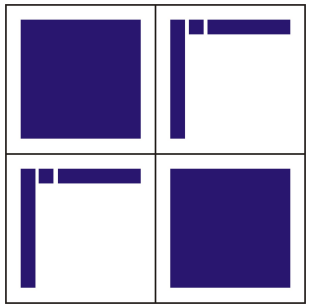
GOAL: sparse structured factorization = sparse factorization + internal rank structured factorization

(Hierarchically) Semi-Separable matrix

An HSS matrix is a dense matrix whose off-diagonal blocks are low-rank

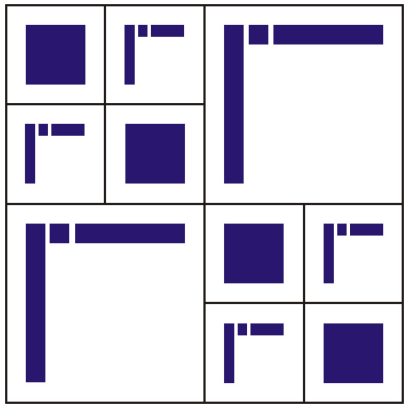
- Dense, 2x2 block $\longrightarrow$ SVD compression:

$$\begin{pmatrix} D_1 & U_1 B_1 V_2^T \\ U_2 B_2 V_1^T & D_2 \end{pmatrix}$$

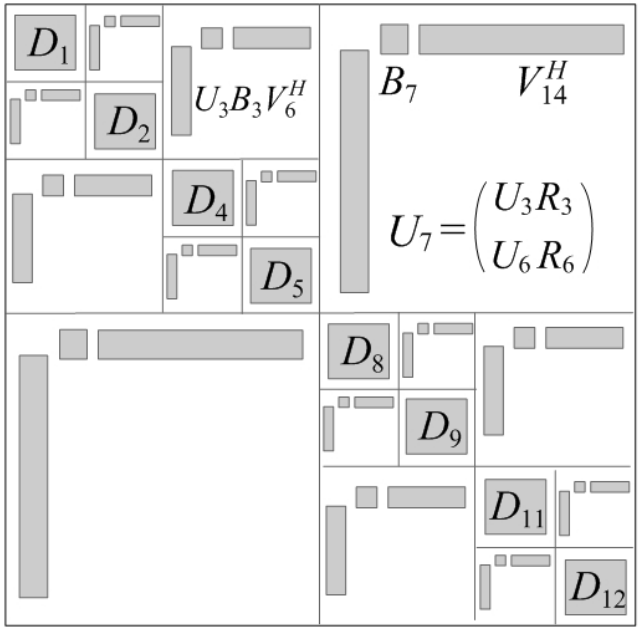


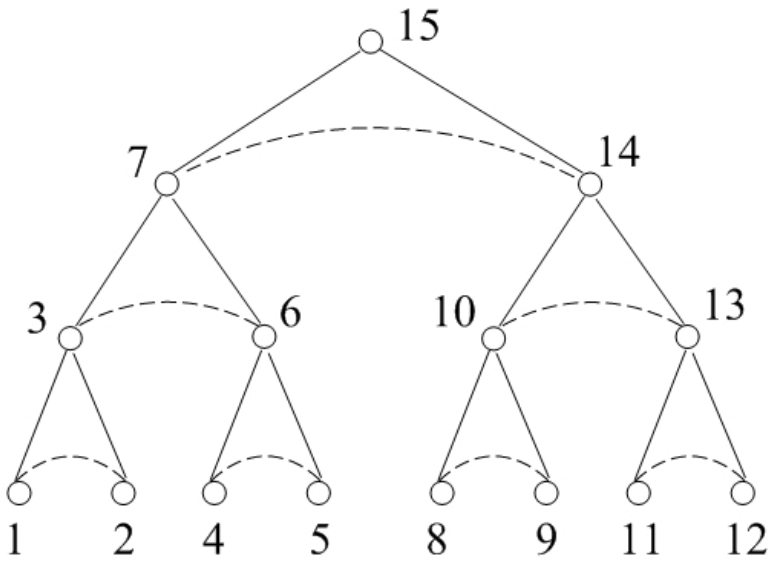
- Recursion $\longrightarrow$ Nested structure

$$A \approx \begin{pmatrix} \boxed{\begin{pmatrix} D_1 & U_1 B_1 V_2^T \\ U_2 B_2 V_1^T & D_2 \end{pmatrix}} & \begin{pmatrix} U_1 R_1 \\ U_2 R_2 \end{pmatrix} B_3 \begin{pmatrix} W_4^T V_4^T & W_5^T V_5^T \end{pmatrix} \\ \begin{pmatrix} U_4 R_4 \\ U_5 R_5 \end{pmatrix} B_6 \begin{pmatrix} W_1^T V_1^T & W_2^T V_2^T \end{pmatrix} & \begin{pmatrix} D_4 & U_4 B_4 V_5^T \\ U_5 B_5 V_4^T & D_5 \end{pmatrix} \end{pmatrix}.$$



Recursive relation, HSS tree





Assume k leaves, $2k-1$ nodes, $\log_2 k$ levels

$$D_{2k-1} = A, \quad U_{2k-1} = \emptyset, \quad V_{2k-1} = \emptyset,$$

$$D_i = A|_{t_i \times t_i} \approx \begin{pmatrix} D_{c_1} & U_{c_1} B_{c_1} V_{c_2}^T \\ U_{c_2} B_{c_2} V_{c_1}^T & D_{c_2} \end{pmatrix},$$

$$U_i = \begin{pmatrix} U_{c_1} & 0 \\ 0 & U_{c_2} \end{pmatrix} \begin{pmatrix} R_{c_1} \\ R_{c_2} \end{pmatrix}, \quad V_i = \begin{pmatrix} V_{c_1} & 0 \\ 0 & V_{c_2} \end{pmatrix} \begin{pmatrix} W_{c_1} \\ W_{c_2} \end{pmatrix},$$

Previous work (serial): solving linear systems in HSS form

- HSS construction: $\mathcal{O}(r\,N^2)$, r is the HSS rank
  - Rank Revealing QR (RRQR) with column pivoting: $AP = QR$
  - May use Modified Gram-Schmidt (MGS), or RR-TSQR [Hoemmen et al.], or random sampling
- HSS ULV factorization [Chandrasekaran-Dewilde-Gu]: $\mathcal{O}(r^2 N)$
  - $QL + LQ \rightarrow ULV$
- HSS solution: $\mathcal{O}(r\,N)$

Xia, *On the complexity of some Hierarchically structured matrix algorithms*, preprint, 2011.

Xia, Chandrasekaran, Gu, Li, *Fast algorithms for hierarchically semiseparable matrices*, pp. 953-976, 2010.

New in CACHE: parallel HSS for sparse linear systems

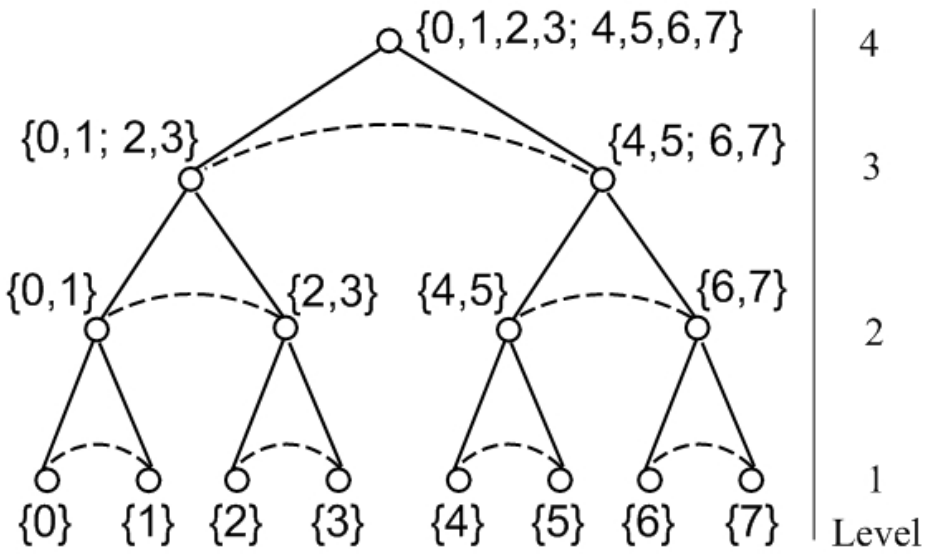
- **Parallel HSS construction, factorization, solution**
  - **Analysis of communication**

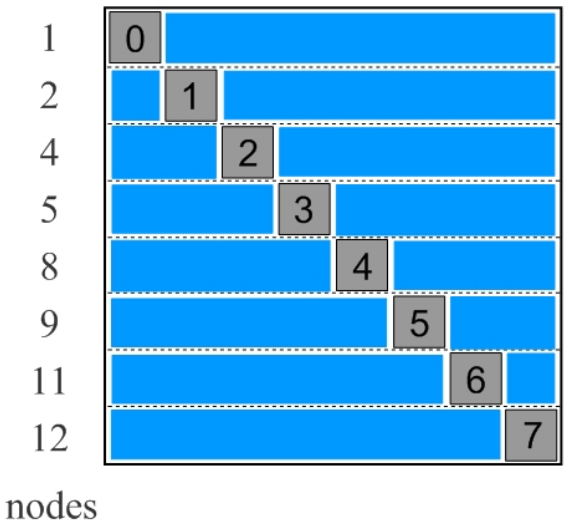
- **Embedding parallel HSS in parallel sparse multifrontal solver**

Wang, Li, Xia, de Hoop, *Efficient scalable algorithms for Hierarchically Semi-separable matrices*, submitted to SISC.

Parallelization strategy

- Work along the tree level by level, bottom up.
  - more parallelism than postorder, slightly more flops in lower order terms.
- 2D block-cyclic distribution at each tree node ($\#Levels = \log P$)
  - each P_i works on the bottom level leaf node i,
  - every 2 processors cooperate on a Level 2 node: 3, 6, 10 and 13,
  - every 4 processors cooperate on a Level 3 node: 7 and 14





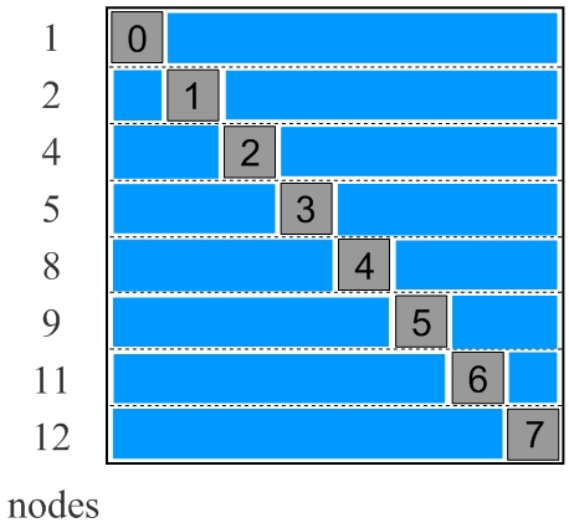
Parallel MGS-RRQR : $\sqrt{P} \times \sqrt{P}$ cores
$F_{M\times N} \rightarrow Q(:, 1:r)\, R(1:r, 1:r)$

```
for j = 1:r
    1.  IN PARALLEL, search for the column f_j with the maximum norm;
    2.  normalize f_j to q_j:  q_j = f_j/||f_j||,  r_jj = ||f_j||;
    3.  BROADCAST q_j within the context associated with the node i;
    4.  PBLAS2:  r_j = q_j^T F_i;
    5.  rank one update:  F_i = F_i - q_j r_j.
end
```

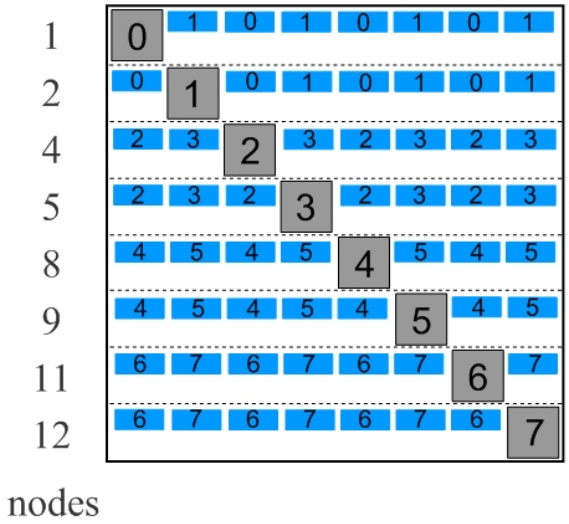
- Assume cost of broadcasting/reducing a message of n words: $comm = [\#msg, vol] = [\log P, n \log P]$
- Step 1: search of pivot column, $comm = [\log \sqrt{P}, \frac{N}{\sqrt{P}} \log \sqrt{P}]$
- Step 3: broadcast pivot column among process row, $[\log \sqrt{P},\ \frac{M}{\sqrt{P}} \log \sqrt{P}]$
- Total: $RRQR_{comm} = \left[\log \sqrt{P},\ \frac{M+N}{\sqrt{P}} \log \sqrt{P}\right] \cdot r$

Parallel row compression

- Level 1 (local): $T_{i,:} = U_i \tilde{T}_{i,:}$, P_i owns the ith stripe. Bounds on dimensions: $U_i : m \times r$, $\tilde{T}_{i,:} : r \times (N - m)$

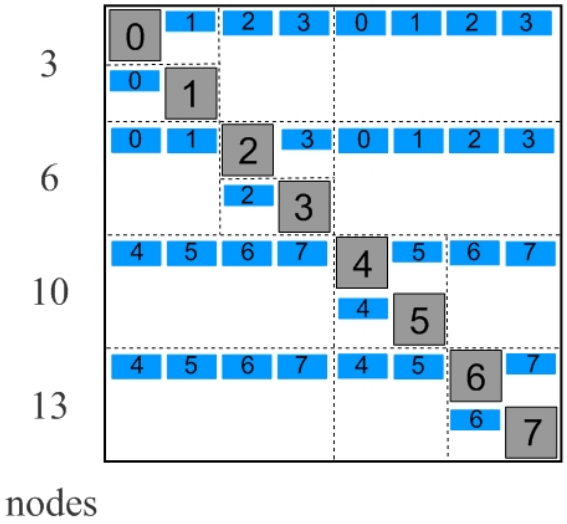


- Level 2 (subgroups of 2-cores)
  - Re-distribution: $\tilde{T}_{i,:}$ is re-distributed among 2 cores. pair-wise exchange : $0 \leftrightarrow 1$, $2 \leftrightarrow 3$, $4 \leftrightarrow 5$, $6 \leftrightarrow 7$ $comm = [2, \frac{rN}{2} 2]$
  - RRQR (MGS): $comm = [\log \sqrt{2}, \frac{2r+N}{\sqrt{2}} \log \sqrt{2}] \cdot r$



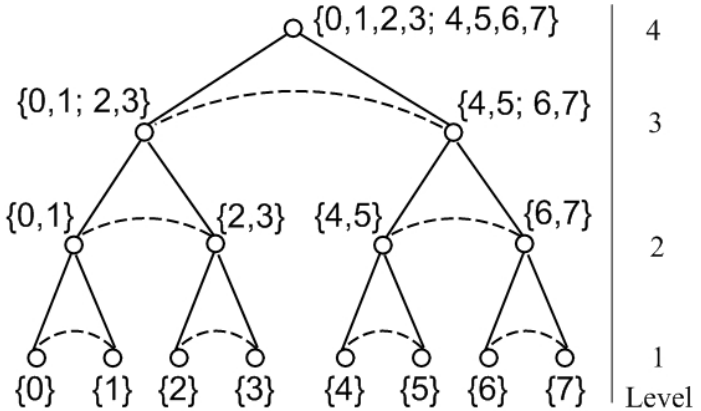
Parallel row compression (cont)

- Level 3 (subgroups of 4-cores)
  - Re-distribution: $\tilde{T}_{i,:}$ is re-distributed among 4 cores (2D block-cyclic).
    pair-wise exchange:
    $0 \leftrightarrow 2,\ 1 \leftrightarrow 3,\ 4 \leftrightarrow 6,\ 5 \leftrightarrow 7.$
    $comm = [2, \frac{r\,N}{4} 2]$
  - RRQR (MGS):
    $comm = [\log \sqrt{4}, \frac{2r+N}{\sqrt{4}} \log \sqrt{4}] \cdot r$



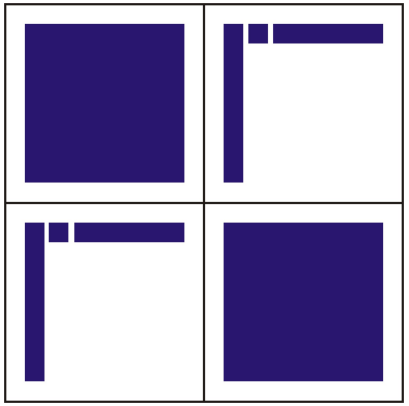
Communication in row compression:
$\#msg = \mathcal{O}(r \log^2 P)$
$\#words = \mathcal{O}(r\ N \log P)$

Flop-to-Byte ratio: $\mathcal{O}(\frac{N}{P \log P})$
(c.f. dense LU: $\mathcal{O}(\frac{N}{\sqrt{P}})$)



Parallel HSS ULV factorization

- Kernel operation: *ULV* factorization [Chandrasekaran et al.]
- $$\begin{pmatrix} D_1 & U_1 B_1 V_2^T \\ U_2 B_2 V_1^T & D_2 \end{pmatrix}$$



- P_0 assigned to first stripe, P_1 assigned to second stripe.

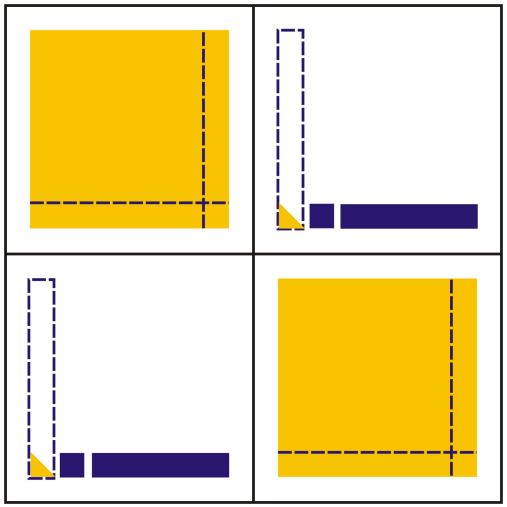
Two steps: $QL + LQ \rightarrow ULV$ factorization. (can handle unsymmetric matrix)

ULV (1): Parallel QL factorization of U_i

- QL factorization: $U_i = Q_i \begin{pmatrix} 0 \\ \tilde{U}_i \end{pmatrix}$
- Updating:

$$\begin{pmatrix} Q_1^T & 0 \\ 0 & Q_2^T \end{pmatrix} \begin{pmatrix} D_1 & U_1 B_1 V_2^T \\ U_2 B_2 V_1^T & D_2 \end{pmatrix}$$

$$= \begin{pmatrix} \hat{D}_1 & \begin{pmatrix} 0 \\ \tilde{U}_1 B_1 V_2^T \end{pmatrix} \\ \begin{pmatrix} 0 \\ \tilde{U}_2 B_2 V_1^T \end{pmatrix} & \hat{D}_2 \end{pmatrix}$$



- No communication.

ULV (2): Parallel *LQ* (transposed QR) of upper $\hat{D}_i$

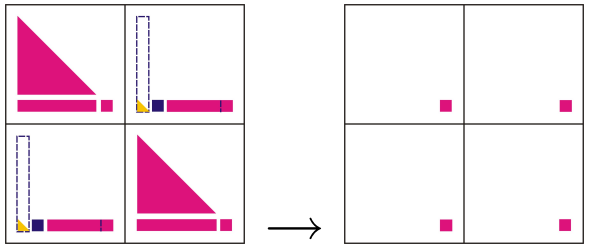
- (partial) *LQ* factorization:

$$\hat{D}_i = \begin{pmatrix} \hat{D}_i;1,1 & \hat{D}_i;1,2 \\ \hat{D}_i;2,1 & \hat{D}_i;2,2 \end{pmatrix} = \begin{pmatrix} L_i & 0 \\ L_i;2,1 & \widetilde{D}_i \end{pmatrix} q_i^T$$

- Updating $\hat{D}_i$ and V_i:

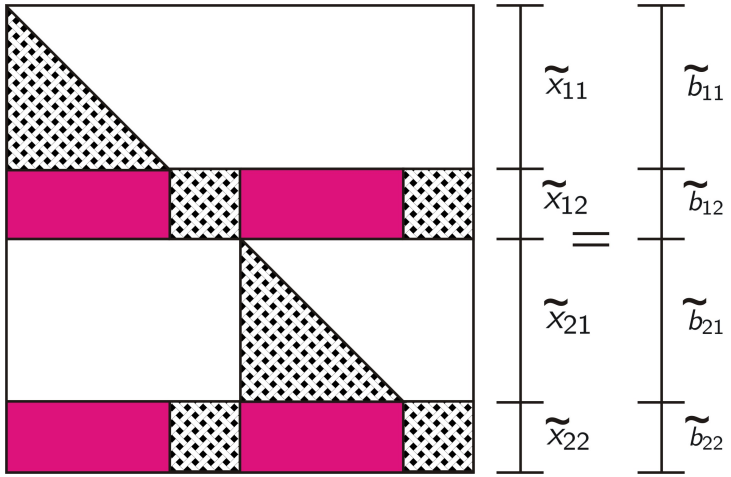
$$\begin{pmatrix} \hat{D}_1 & \begin{pmatrix} 0 \\ \tilde{U}_1 B_1 V_2^T \end{pmatrix} \\ \begin{pmatrix} 0 \\ \tilde{U}_2 B_2 V_1^T \end{pmatrix} & \hat{D}_2 \end{pmatrix} \begin{pmatrix} q_1 & 0 \\ 0 & q_2 \end{pmatrix} = \begin{pmatrix} \begin{pmatrix} L_1 & 0 \\ L_1;2,1 & \widetilde{D}_1 \end{pmatrix} & \begin{pmatrix} 0 \\ \widetilde{U}_1 B_1 (q_2^T V_2)^T \end{pmatrix} \\ \begin{pmatrix} 0 \\ \widetilde{U}_2 B_2 (q_1^T V_1)^T \end{pmatrix} & \begin{pmatrix} L_2 & 0 \\ L_2;2,1 & \widetilde{D}_2 \end{pmatrix} \end{pmatrix}$$

- One level up: merge 4 residual blocks to the parent node.
  two-one merge: $0 \leftrightarrow 1$



At the highest level: direct LU factorization of D_i

Parallel HSS solution



$$\begin{pmatrix} Q_1^T D_1 q_1 & \begin{pmatrix} 0 \\ \widetilde{U}_1 B_1 (q_2^T V_2)^T \end{pmatrix} \\ \begin{pmatrix} 0 \\ \widetilde{U}_2 B_2 (q_1^T V_1)^T \end{pmatrix} & Q_2^T D_2 q_2 \end{pmatrix} \begin{pmatrix} q_1^T x_{11} \\ q_1^T x_{12} \\ q_2^T x_{21} \\ q_2^T x_{22} \end{pmatrix} = \begin{pmatrix} Q_1^T b_{11} \\ Q_1^T b_{12} \\ Q_2^T b_{21} \\ Q_2^T b_{22} \end{pmatrix}$$

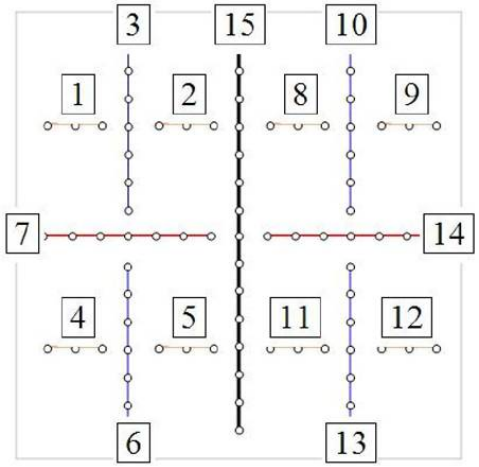
New in CACHE: parallel HSS for sparse linear systems

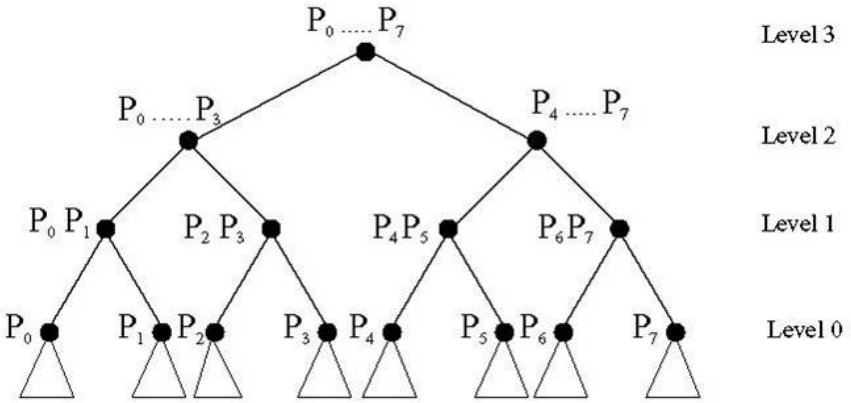
- Parallel HSS construction, factorization, solution
  - Analysis of communication

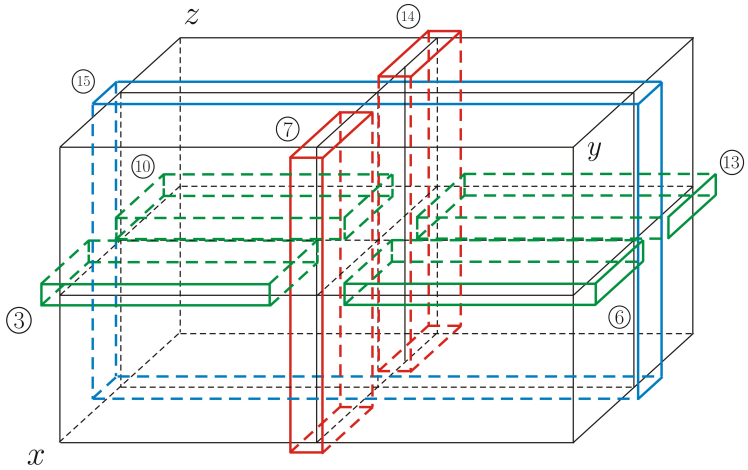
- **Embedding parallel HSS in parallel sparse multifrontal solver**

Parallel multifrontal sparse factorization

- Nested dissection ordering → Separator tree
- Top-down assignment of processors to subtrees
- Bottom-up elimination

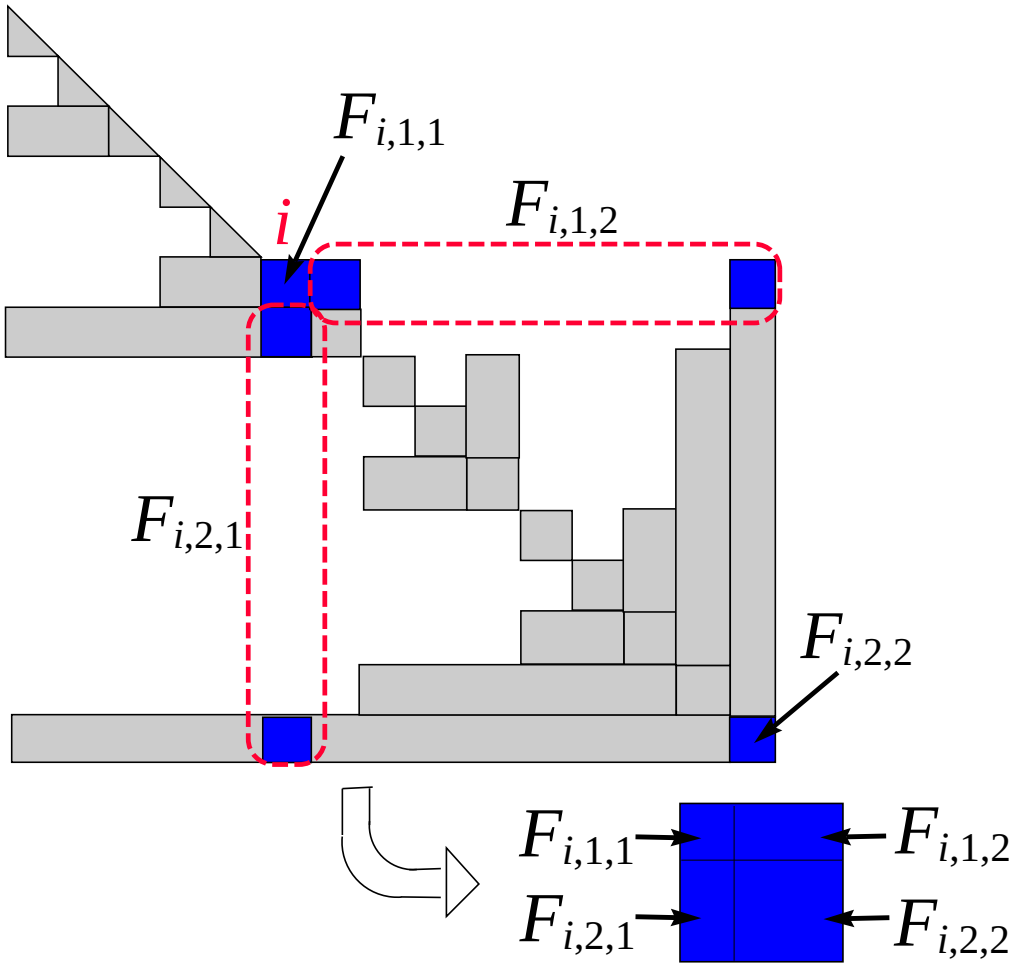






Frontal and Update matrices

- Each separator corresp. to a dense submatrix (frontal & update)
  - often, off-diagonal blocks are low-rank

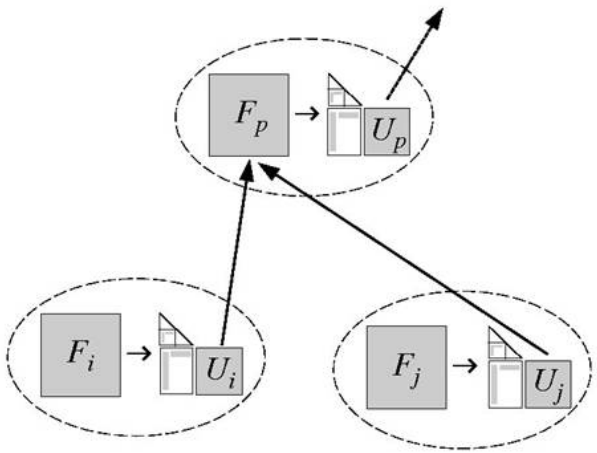


Embedding HSS in multifrontal

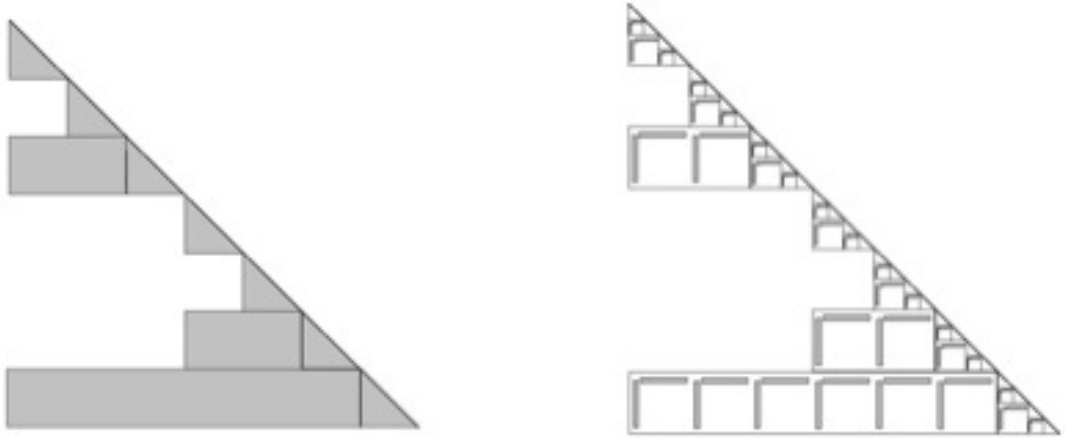
All **Frontal** & **Update** matrices are approximated by HSS

Need following operations:

- **frontal HSS factorization of** F_i
- **extend-add of two HSS update matrices** U_i **and** U_j



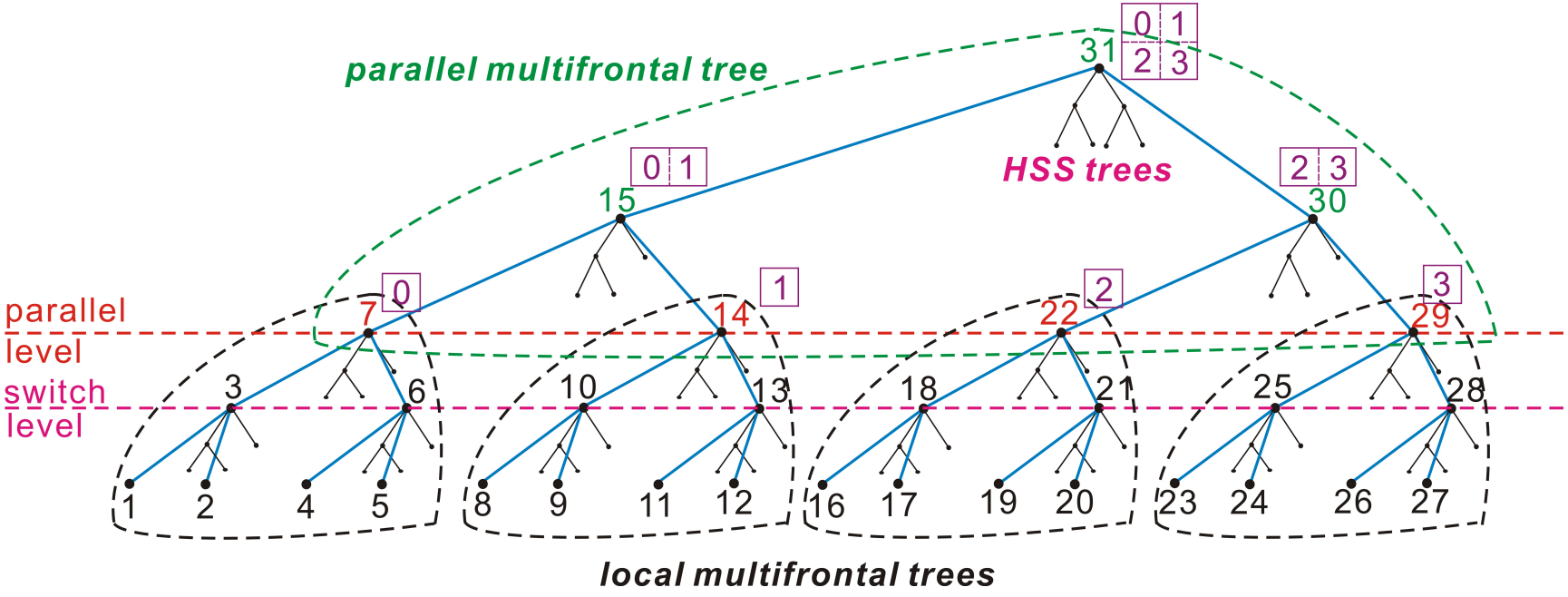
Final Cholesky factor: Classical vs HSS-embedded



MF + HSS: two types of tree-based parallelism

- **Outer tree**: separator tree for multifrontal factorization
- **Inner tree**: HSS tree at each internal separator node

Utilizing ScaLAPACK 2D block-cyclic distribution and sub-communicator



Rank-relaxed complexity

- Serial sparse Cholesky [Xia '11, Chandrasekaran et al. '11]
  - N = sparse matrix size
  - K = frontal matrix size: 2D: $K = N^{1/2}$, 3D: $K = N^{1/3}$

Problem	r = Max rank	Mem.	Fact. FLOPS	Classical Mem. & FLOPS	
2D Poisson	$\mathcal{O}(1)$	$\mathcal{O}(N \log \log N)$	$\mathcal{O}(N \log N)$	$(N \log N)$	$\mathcal{O}(N^{3/2})$
2D Helmholtz	$\mathcal{O}(\log K)$				
3D Poisson	$\mathcal{O}(K)$	$\mathcal{O}(N \log N)$	$\mathcal{O}(N^{4/3})$	$\mathcal{O}(N^{4/3})$	$\mathcal{O}(N^2)$
3D Helmholtz	$\mathcal{O}(K)$	$\mathcal{O}(N \log N)$	$\mathcal{O}(N^{4/3} \log N)$		

Rank-relaxed complexity

- Serial sparse Cholesky [Xia '11, Chandrasekaran et al. '11]
  - N = sparse matrix size
  - K = frontal matrix size: 2D: $K = N^{1/2}$, 3D: $K = N^{1/3}$

Problem	r = Max rank	Mem.	Fact. FLOPS	Classical Mem. & FLOPS	
2D Poisson	$\mathcal{O}(1)$	$\mathcal{O}(N \log\log N)$	$\mathcal{O}(N \log N)$	$(N \log N)$	$\mathcal{O}(N^{3/2})$
2D Helmholtz	$\mathcal{O}(\log K)$				
3D Poisson	$\mathcal{O}(K)$	$\mathcal{O}(N \log N)$	$\mathcal{O}(N^{4/3})$	$\mathcal{O}(N^{4/3})$	$\mathcal{O}(N^2)$
3D Helmholtz	$\mathcal{O}(K)$	$\mathcal{O}(N \log N)$	$\mathcal{O}(N^{4/3} \log N)$		

- Gaps in analysis of parallel algorithms
  - Classical sparse Cholesky [Gupta et al. '97]
    - 3D: $\mathcal{O}(N^{4/3}/\sqrt{P})$ COMM-Volume
  - HSS-embedded sparse factorization ??
  - Communication lower bound ??

Parallel performance

- Cray XE6 (hopper at NERSC)
- Example: Helmholtz equation with PML boundary

$$\left(-\Delta - \frac{\omega^2}{v(x)^2}\right) u(x,\omega) = s(x,\omega), \tag{1}$$

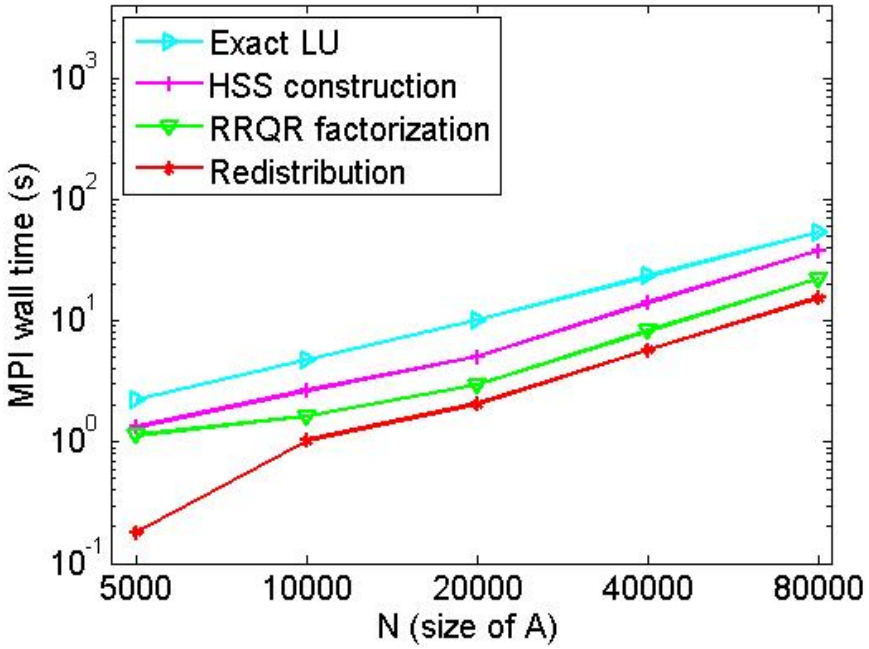
Δ: Laplacian
ω: angular frequency
$v(x)$: seismic velocity field
$u(x,\omega)$: time-harmonic wavefield solution

- FD discretized linear system:
  - Complex, pattern-symmetric, non-Hermitian,
  - Indefinite, ill-conditioned

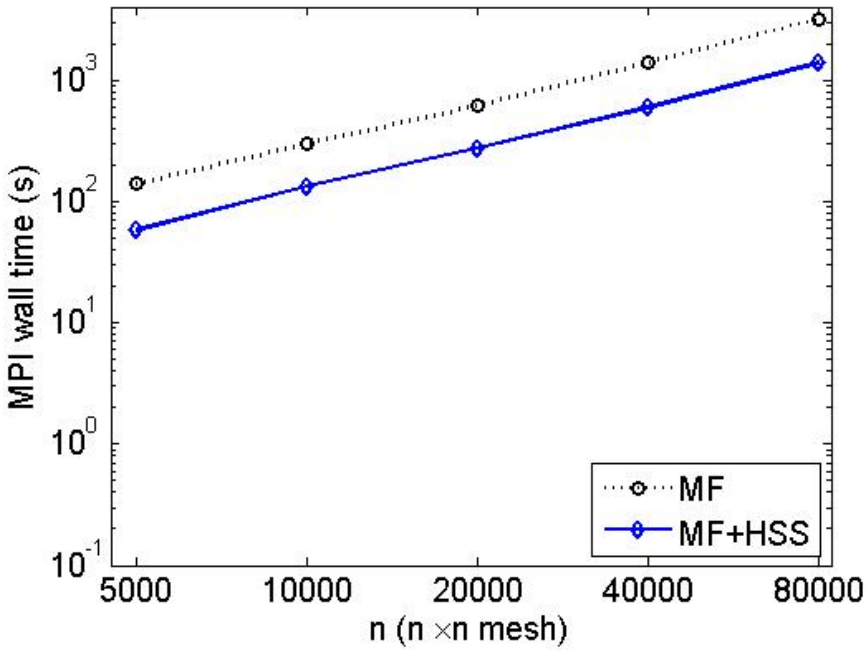
Weak scaling test

- 2D mesh $N \times N$: 5000, 10000, 20000, 40000, 80000
  Processor counts: 16, 64, 256, 1024, 4096
  Up to 6.4 billion unknowns
- $3x$ faster than classical multifrontal, needs $1/2$ memory

HSS on topmost separator



MF + HSS solver



3D test

- $N = 300^3 = 27M$
- Topmost frontal matrix size $K^2 = 90,000$,
  Max rank $= 1391$ ($\tau = 10^{-4}$)
- $P = 1024$

	Time (s)	Gflops/s (peak%)	Comm%	Mem (GB)
MF	4206.4	2385.2 (27.7%)	32.6%	3143.7
MF+HSS	2171.3	2511.3 (29.2%)	41.2%	1104.4
HSS-compr	1388.5		15.3%	

3D test

- $N = 300^3 = 27M$
- Topmost frontal matrix size $K^2 = 90,000$,
  Max rank $= 1391$ ($\tau = 10^{-4}$)
- $P = 1024$

	Time (s)	Gflops/s (peak%)	Comm%	Mem (GB)
MF	4206.4	2385.2 (27.7%)	32.6%	3143.7
MF+HSS	2171.3	2511.3 (29.2%)	41.2%	1104.4
HSS-compr	1388.5		15.3%	

- MF+HSS worked for 400^3, 500^3, but pure MF failed

Summary

STATUS

- First demonstration of parallel HSS-embedded sparse solver
- Removing "short-cut": dense extend-add $\rightarrow$ HSS extend-add ($\mathcal{O}(\log N)$ reduction in FLOPS; communication ??)
- Extending to generalized multifrontal code **[Artem Napov]**
  - Random sampling methods for HSS construction

Summary

STATUS

- First demonstration of parallel HSS-embedded sparse solver
- Removing "short-cut": dense extend-add $\rightarrow$ HSS extend-add ($\mathcal{O}(\log N)$ reduction in FLOPS; communication ??)
- Extending to generalized multifrontal code **[Artem Napov]**
  - Random sampling methods for HSS construction

LONG TERM RESEARCH

- Analyze communication bound, design communication avoiding version
- General purpose preconditioner? (different from ILU)
  - Apply to broader DOE simulation problems: acceleror, fusion, etc.
- Precondition the CA-Krylov algorithms [with Demmel's group]
- Analysis of wider problems that admit low rank