4-8-2021

Lecture 13: A low-rank factorization framework for building scalable algebraic solvers and preconditioners

X. Sherry Li
Lawrence Berkeley National Laboratory

A low-rank factorization framework for building scalable algebraic solvers and preconditioners

X. Sherry Li
Lawrence Berkeley National Laboratory

University of Arkansas
Department of Mathematical Sciences
46th Spring Lecture Series
April 8, 2021

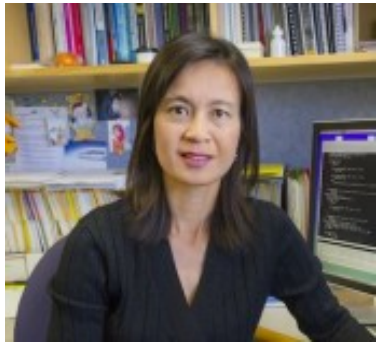
Acknowledgement

- Scientific Discovery through Advanced Computing (SciDAC) program through the FASTMath Institute under Contract No.DE-AC02-05CH11231.

- Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration.

Team

Sherry Li



Lisa Claus

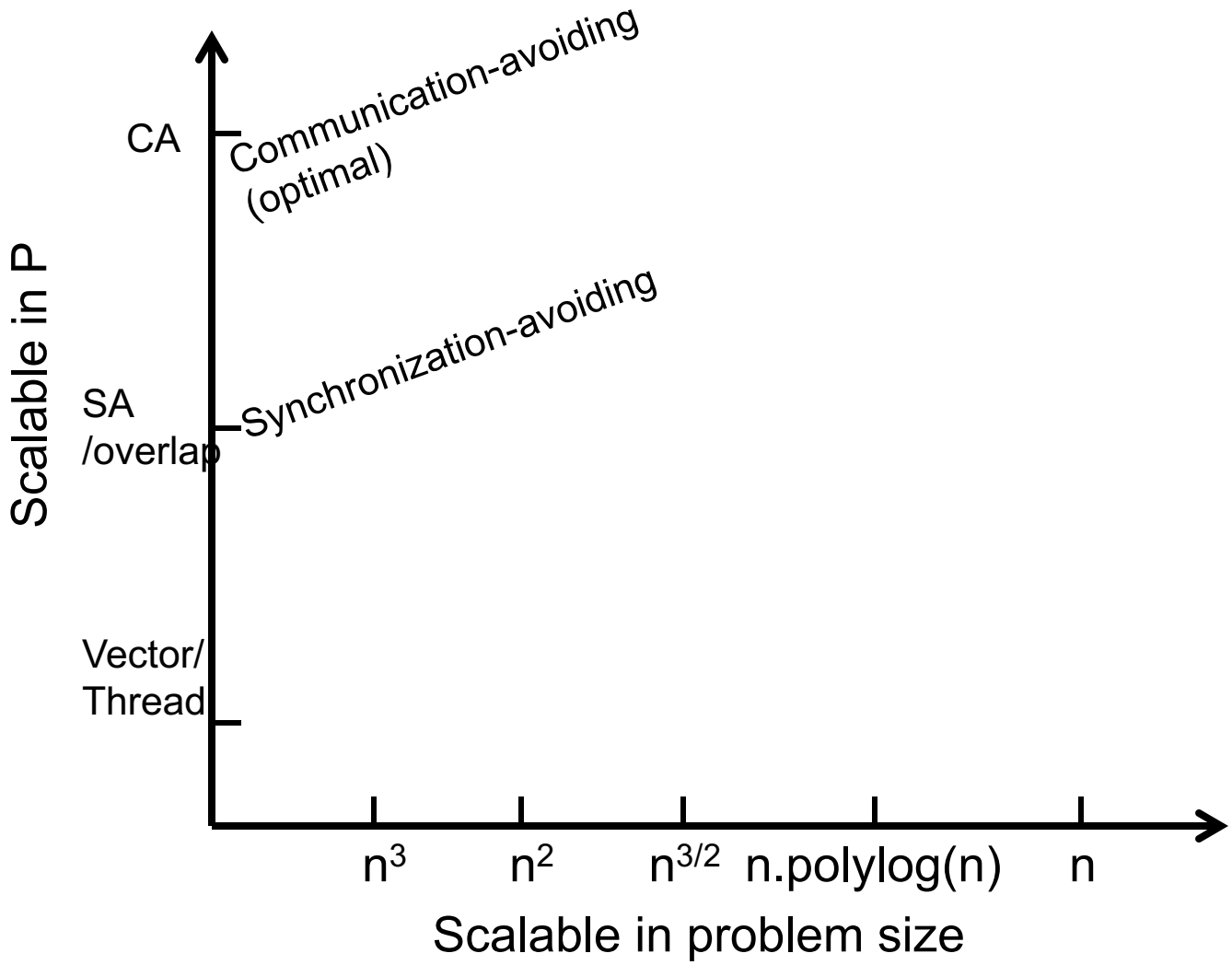
Pieter Ghysels

Yang Liu

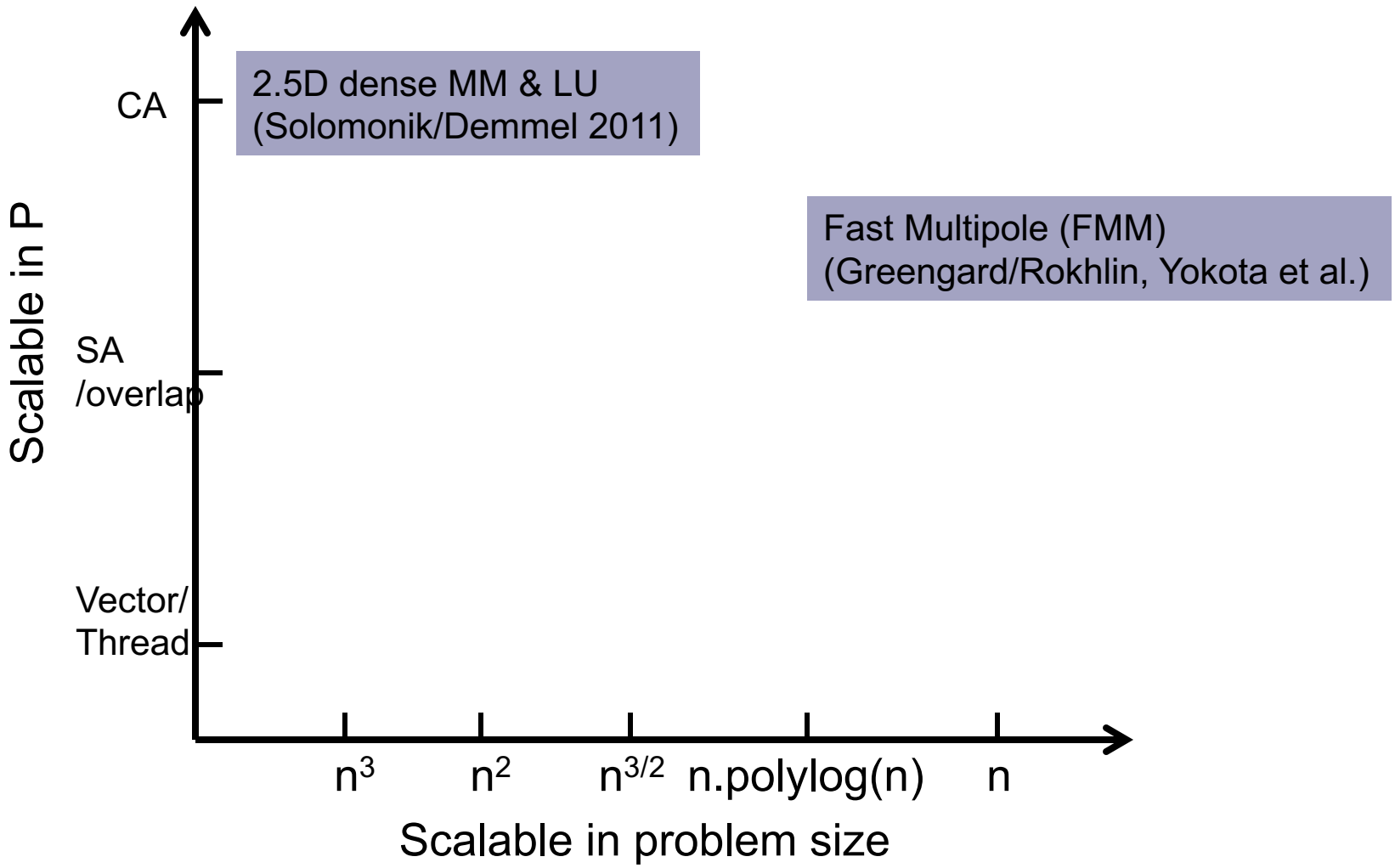
Past members:

- Gustavo Chavez, Machinify
- Chris Gorman, UC Santa Barbara
- Liza Rebrova, LBNL
- Francois-Henry Rouet, LSTC

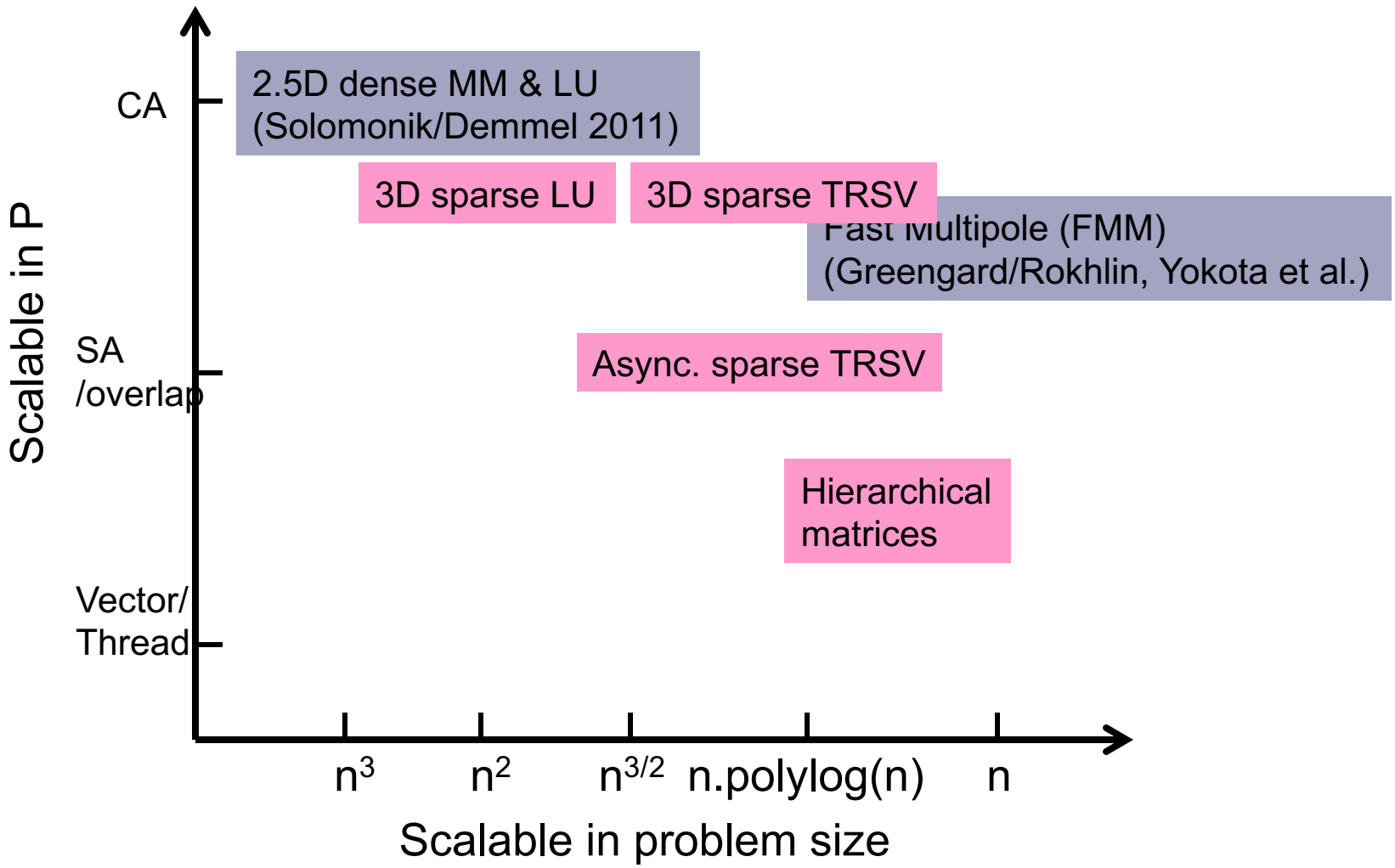
Algorithm scalability



Algorithm scalability



Algorithm scalability

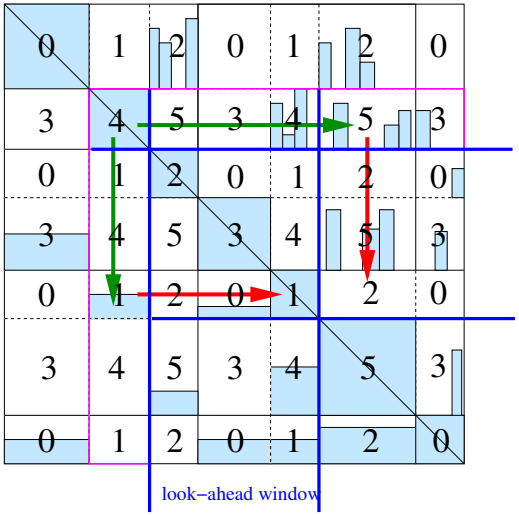


Factorization framework

- Cholesky, LU, QR, SVD, …
- Complexity wall
  - Dense: N^3 flops, N^2 memory
  - Sparse: N^2 flops, $N^{4/3}$ memory (3D discretized PDE)

- Distributed-memory algorithms usually use 2D block-cyclic layout
  - 2D matrix blocks maps to 2D process grid

$P = P_X \times P_Y$

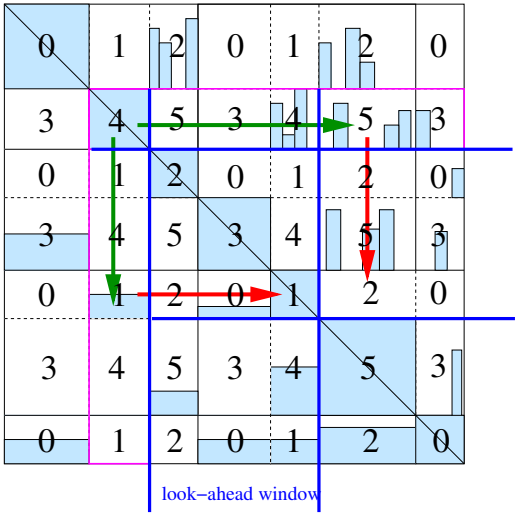


Factorization framework

- Cholesky, LU, QR, SVD, …
- Complexity wall
  - Dense: N^3 flops, N^2 memory
  - Sparse: N^2 flops, $N^{4/3}$ memory (3D discretized PDE)

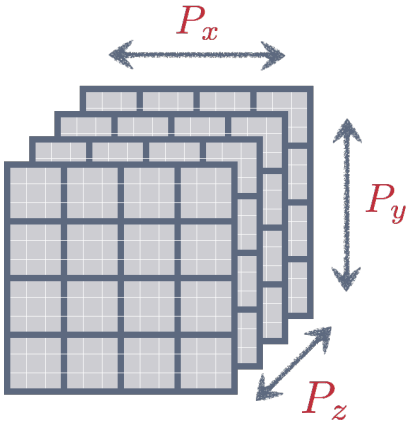
- Distributed-memory algorithms usually use 2D block-cyclic layout
  - 2D matrix blocks maps to 2D process grid

$P = P_X \times P_Y$



3D algorithm can reduce communication !

Third dimension for selective replication



3D sparse LU on Cray XC30 (edison @ NERSC)

Example: K2D5pt4096

hpcgarage.org/hookem

P_z	24	48	96	192	384	768
32	1.7	2	2	1.5		
16	1.1	1.3	1.5	1.7	1.3	0.81
8	0.67	0.93	0.91	1.1	0.97	0.94
4	0.28	0.42	0.49	0.48	0.49	0.53
2	0.093	0.12	0.17	0.21	0.23	0.24
1	0.036	0.044	0.055	0.065	0.064	0.073

MPI processes (2D process grid; 4 cores / process)

2D to 3D:
→ **23x speedup**

Teraflop/s
(32x procs → 2x speedup)

Sao, Vuduc, Li, JPDC 2019

hpcgarage.org/hookem

P_z	24	48	96	192	384	768
32	1.7	2	2	1.5		
16	1.1	1.3	1.5	1.7	1.3	0.81
8	0.67	0.93	0.91	1.1	0.97	0.94
4	0.28	0.42	0.49	0.48	0.49	0.53
2	0.093	0.12	0.17	0.21	0.23	0.24
1	0.036	0.044	0.055	0.065	0.064	0.073

MPI processes (2D process grid; 4 cores / process)

64x procs
→ **55x speedup**

Teraflop/s
(32x procs → 2x speedup)

Lower complexity: All about approximation …

Algebraic preconditioners

- Incomplete LU factorization: level-based ILU(k), threshold-based ILUTP (e.g., in sequential SuperLU package)
  Bollhofer, Chow, Meijerink/van der Vorst, Saad, …

- Approximate inverse
  AINV: Bollhofer, Meurant, …
  FSAI: Kolotilina/Yeremin 1993, …
  SPAI: Grote/Huckle, Chow/Saad, Benzi/Tuma, …

- These are strong preconditioners, can handle:
  - anisotropic, indefinite, unsymmetric, …

This talk:
Preconditioners using ideas from structured matrices

Two types of structured matrices (for the purpose of this talk …)

- "Dense" → data-sparse

  e.g., admits low-rank compressed representation

- "Sparse" → structurally-sparse

  combined with data-sparse becomes sparser

Software:

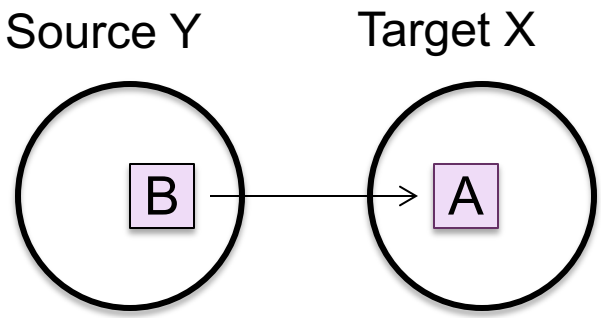
- ButterflyPACK – dense
- STRUMPACK – dense and sparse

Example: Applying integral operator (N-body computation)

$$(Kg)(x) = \int_Y K(x,y)g(y)\,dy, \quad Kernel\ K : X \times Y \to C$$

- Low rank property:

If $diam(A)diam(B) < D,\ K|_{AxB}$ has low rank



- Many kernels have this property:

e^{ixy}

Bessel function: $J_0(xy)$

Fourier-type kernele $e^{i\,\Phi(x,y)}$, where $\Phi(x,y)$ is smooth

Green's functions for Helmholtz equation:

$$\text{2D: } \mathrm{H}_0(k\,|x-y|), \qquad \text{3D: } \frac{e^{ik|x-y|}}{|x-y|}$$

Gaussian kernel

Approximation via low-rank compression

- Same mathematical foundation as FMM (Greengard-Rokhlin'87), put in matrix form:
  - Exact diagonal block ("near field") exact
  - Approximate off-diagonal block ("far field") approximate

FMM	Algebraic
separability of Green's function	low-rankness off-diagonal
$G(x,y) \approx \sum_{l=1}^{r} f_l(x) g_l(y), \quad x \in X, y \in Y$	$A \approx \begin{bmatrix} D_1 & U_1 B_1 V_2^T \\ U_2 B_2 V_1^T & D_2 \end{bmatrix}$

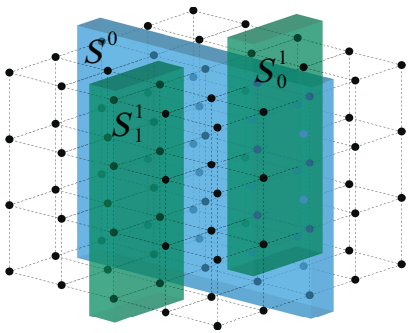
- Algebraic power: factorization, inversion, tensors, … in addition to matrix-vector multiplication

Example: Discretized PDE

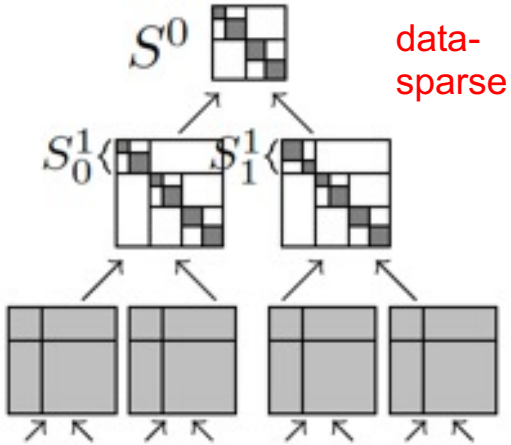
- Globally sparse, locally dense
  - Embed LR data-sparse in sparse multifrontal algorithm
- Baseline is a sparse multifrontal direct solver
- Nested Dissection ordering → separator tree
- In addition to structural sparsity, further apply LR data-sparsity to dense frontal matrices
- **Nested bases + randomized sampling** to achieve linear scaling in sparse case
  - **O(N logN) flops, O(N) memory for 3D elliptic PDEs**

    (as opposed to $O(N^2)$ flops with exact factorization)

Nested dissection ordering



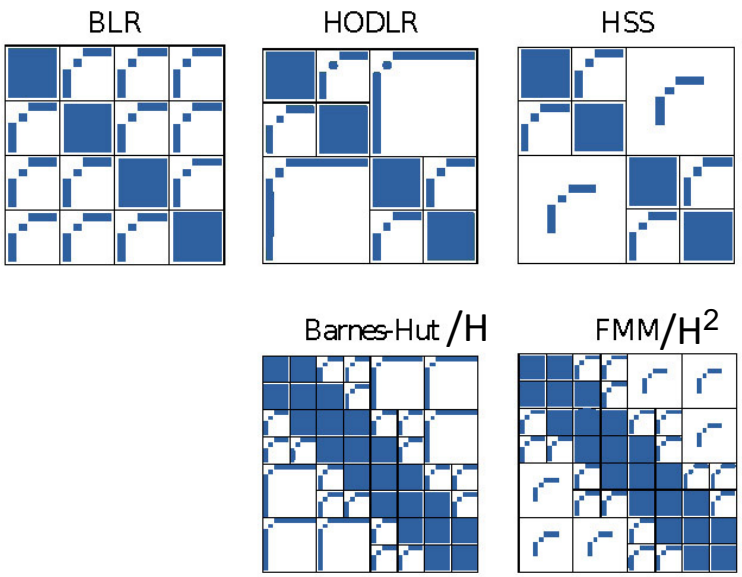
Multifrontal Separator tree



Classes of LR structured matrices

- Lower complexity is usually associated with multilevel / hierarchical metho

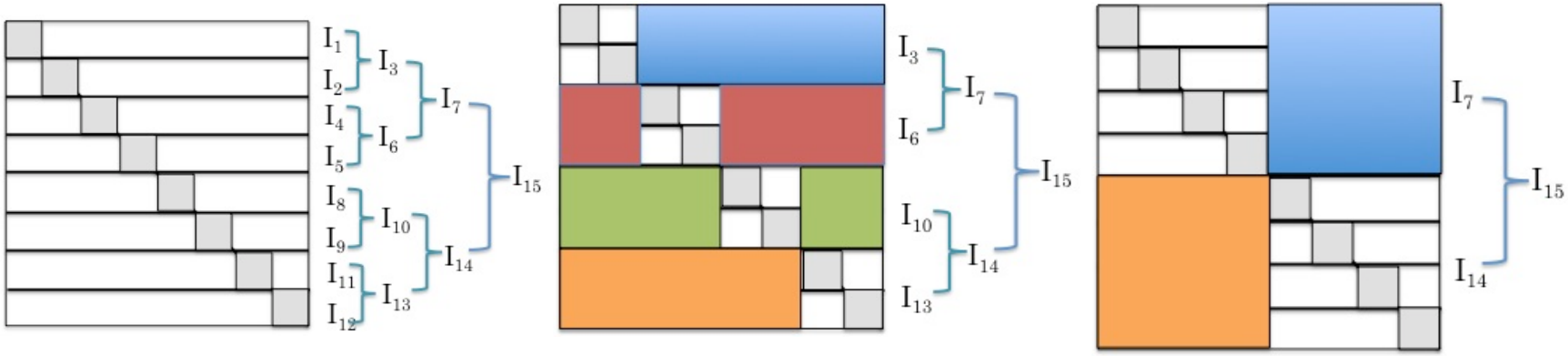
	Strong admissibility	Weak admissibility
Independent bases	H matrix	(Hierarchically off-diagonal low-rank) HODLR matrix
Nested bases	H^2 matrix Inverse FMM	HSS matrix Recursive Skeletonization HIF

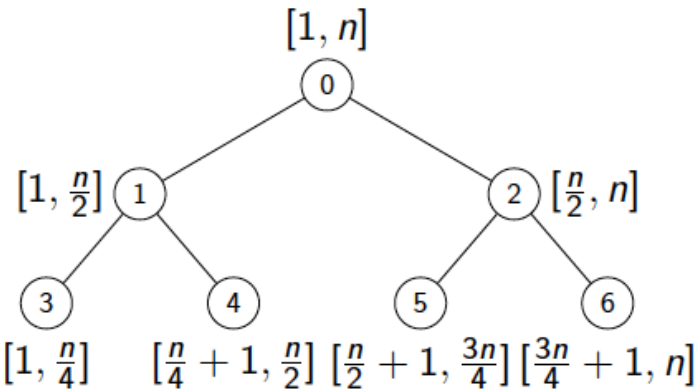


Weak admissibility

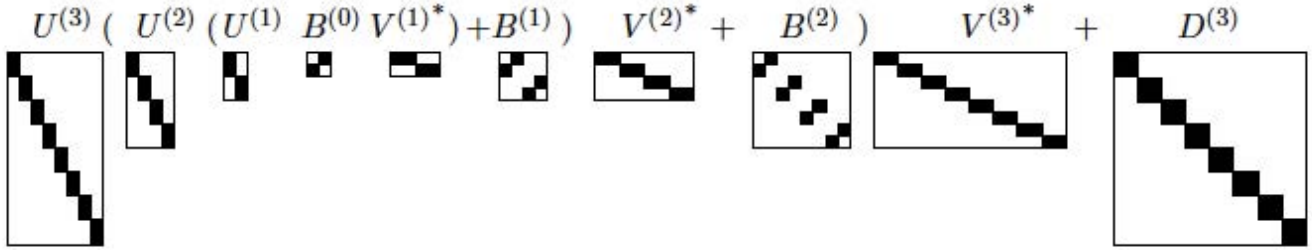
Strong admissibility

- Cluster tree





Martinsson's "telescope" view of HSS matrix



HSS matrix: ULV factorization

$$\Gamma_{1;b\leftrightarrow 2;t}\begin{bmatrix} I & & & \\ & \Omega_1 & & \\ & & I & \\ & & & \Omega_2 \end{bmatrix}\begin{bmatrix} \Gamma_{3;b\leftrightarrow 4;t} & \\ & \Gamma_{5;b\leftrightarrow 6;t} \end{bmatrix}\begin{bmatrix} \Omega_3 & & & \\ & \Omega_4 & & \\ & & \Omega_5 & \\ & & & \Omega_6 \end{bmatrix} A \begin{bmatrix} Q_3^* & & & \\ & Q_4^* & & \\ & & Q_5^* & \\ & & & Q_6^* \end{bmatrix}\begin{bmatrix} \Gamma_{3;b\leftrightarrow 4;t}^T & \\ & \Gamma_{5;b\leftrightarrow 6;t}^T \end{bmatrix}\begin{bmatrix} I & & & \\ & Q_1^* & & \\ & & I & \\ & & & Q_2^* \end{bmatrix}\Gamma_{1;b\leftrightarrow 2;t}^T$$

$$= \left[\begin{array}{cccccccc}
L_3 & & & & & & & \\
0 & L_4 & & & & & & \\
(\Omega_1 L_{4,3})_t & (\Omega_1 L_{3,4})_t & L_1 & & & & & \\
 & & & L_5 & & & & \\
 & 0 & & 0 & L_6 & & & \\
 & & & (\Omega_2 L_{6,5})_t & (\Omega_2 L_{5,6})_t & L_2 & & \\
(\Omega_1 L_{4,3})_b & (\Omega_1 L_{3,4})_b & W_{1;b} Q_{1;t}^* & B_{1,2} V_2^* \begin{bmatrix} V_5^* Q_{5;t}^* & V_5^* Q_{5;b}^* \\ V_6^* Q_{6;t}^* & V_6^* Q_{6;b}^* \end{bmatrix}\begin{bmatrix} I & \\ & Q_2^* \end{bmatrix} & & & & D_0 \\
B_{2,1} V_1^* \begin{bmatrix} V_3^* Q_{3;t}^* & V_3^* Q_{3;b}^* \\ V_4^* Q_{4;t}^* & V_4^* Q_{4;b}^* \end{bmatrix}\begin{bmatrix} I & \\ & Q_1^* \end{bmatrix} & & & (\Omega_2 L_{6,5})_b & (\Omega_2 L_{5,6})_b & W_{2;b} Q_{2;t}^* & &
\end{array}\right]$$

Software

- **ButterflyPACK** https://github.com/liuyangzhuan/ButterflyPACK
  - Distributed-memory, MPI + OpenMP, Fortran2008
  - Formats: H, HODLR with LR and Butterfly.

- **STRUMPACK** https://github.com/pghysels/STRUMPACK
  - Distributed-memory, MPI + OpenMP, C++
  - Formats: HSS, HODLR, BLR, interface with ButterflyPACK functionalities
  - Dense and sparse (multifrontal + data-sparse fronts)

	H (LR/BF)	HODLR (LR/BF)	HSS	BLR
ButterflyPACK	☺	☺		
STRUMPACK		☺	☺	☺

Rest of the talk

Complementary to David Keyes' lectures 3

Deep dive:

- Low-rank compression tool: adaptive randomized sketching
  - Understanding probabilistic error
- High-frequency wave equations: butterfly compression

Stages of operations

- **Data clustering, matrix reordering**
- **Compression** – usually dominating cost

  complexity depends on two situations:

  1. only black-box K*g and K'*g are available
  2. only black-box entry evaluation K(i,j) is available

  Goal: $O(N \log^{\alpha} N)$

- **Matrix-vector multiplication**
- **Factorization / solve, inversion**

Sweeping through **"trees"** upward / downward:

- HSS tree, butterfly tree
- Principal tools for parallelization

LR compression mechanisms

- SVD
- RRQR

- Randomized projection (sampling) : $O(n^2 r)$
- ACA, block ACA → hierarchical blocked ACA : $O(nr^2)$

  (Liu, Sid-Lakhdar, Rebrova, Ghysels, Li, 2020)
- Interpolative Decomposition (ID) (skeletonization)

$$B \approx B(:,J)\,X, \quad B(:,J) \text{ has } k \text{ columns, } X \text{ is called interpolation matrix}$$

- Nearest neighbors: relies on geometry

Low rank compression via randomized projection
(sketching, sampling)

… more flexible and faster than traditional rank-revealing QR

Approximate range of A:

1. Pick random matrix $\Omega_{nx(k+p)}$, k target rank, p small, e.g. 10
2. Sample matrix S = A Ω (tall-skinny)
3. Compute Q = ON-basis(S) via rank-revealing QR

Accuracy (in 2-norm) (Halko/Martinsson/Tropp ' 11)

- On average: $$\mathrm{E}[\| A - QQ^* A \|] = \left(1 + \frac{4\sqrt{k+p}}{p-1}\sqrt{\min\{m,n\}}\right)\sigma_{k+1}$$
- Probabilistic bound: with probability ≥ 1 – 3 x10^{-p},

$$\| A - QQ^* A \| \leq (1 + 9\sqrt{k+p}\sqrt{\min\{m,n\}})\sigma_{k+1}$$

Benefits:

- Matrix-free: only need matvec
- When embedded in sparse frontal solver, simplifies "extend-add"

Low rank compression via randomized projection
(sketching, sampling)

… more flexible and faster than traditional rank-revealing QR

Approximate range of A:

1. Pick random matrix $\Omega_{nx(k+p)}$, k target rank, p small, e.g. 10
2. Sample matrix S = A Ω (tall-skinny)
3. Compute Q = ON-basis(S) via rank-revealing QR

Accuracy (in 2-norm) (Halko/Martinsson/Tropp ' 11)

- On average: $$\mathrm{E}[\| A - QQ^* A \|] = \left(1 + \frac{4\sqrt{k+p}}{p-1}\sqrt{\min\{m,n\}}\right)\sigma_{k+1}$$
- Probabilistic bound: with probability ≥ 1 – 3 x10^{-p},

$$\| A - QQ^* A \| \leq (1 + 9\sqrt{k+p}\sqrt{\min\{m,n\}})\sigma_{k+1}$$

Main question: Do not know *k* !

Adaptive sampling for robustness and performance

Increase sample size d, build Q incrementally (block variant)
$[S_1 , S_2 , S_3 , \ldots]$

Need good error estimation

- Goal is to bound error for A: $||(I - QQ^{*})A||$
- . . . but A is not available, only have S

Need establish a stochastic relationship between ||A|| and ||S||

Stochastic norm estimation

Let $A \in \Re^{mxn}$, and $x \in \Re^{n}$ with $x_i \sim N(0,1)$. Consider SVD:

$$A = U\Sigma V^* = [U_1 \;\; U_2]\begin{bmatrix} \Sigma_r & 0 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} V_1^* \\ V_2^* \end{bmatrix}$$

Define $\zeta = V^* x$, ζ is also a Gaussian random vector.

$$\|Ax\|_2^2 = \|\Sigma\zeta\|_2^2 = \zeta_1^2\sigma_1^2 + \cdots + \zeta_r^2\sigma_r^2$$

here, $\sigma_1 \geq \sigma_2 \cdots \geq \sigma_r > 0$ are positive singular values. Then:

$$\mathrm{E}\left[\|Ax\|_2^2\right] = \sigma_1^2 + \cdots + \sigma_r^2 = \|A\|_F^2$$

For d sample vectors: $\mathrm{E}\left[\|S\|_F^2\right] = d\,\|A\|_F^2$

Gorman, Chavez, Ghysels, Mary, Rouet, Li, SISC 2019

Adaptive sampling: stopping criteria

Let $[S_1 \;\; S_2] = A\,[R_1 \; R_2\,]\,,\;\; Q = QR(S_1),$ block size d

Absolute stopping criterion:

$$\|(I - QQ^*)A\|_F \approx \frac{1}{\sqrt{d}}\,\|(I - QQ^*)S_2\|_F \leq \varepsilon_a$$

Relative stopping criterion:

$$\frac{\|(I - QQ^*)A\|_F}{\|A\|_F} \approx \frac{\|(I - QQ^*)S_2\|}{\|S_2\|} \leq \varepsilon_r$$

Cost: one reduction to compute norms of the sample vectors.

Result:
Have enough samples (robustness), but not too many (performance)

Adaptivity example 1: decay singular value (easy)

$A = \alpha I + UDV^*$, U, V rank $= 120$, $D_{k,k} = 2^{-24(k-1)/r}$

$\varepsilon_r \backslash \varepsilon_a$	1e-2	1e-4	1e-6	1e-8	1e-10	1e-12	1e-14
1e-1	24; 64	24; 64	24; 64	24; 64	24; 64	24; 64	24; 64
1e-2	42; 80	42; 80	42; 80	42; 80	42; 80	42; 80	42; 80
1e-3	59; 96	59; 96	59; 96	59; 96	59; 96	59; 96	59; 96
1e-4	77; 112	77; 112	77; 112	77; 112	77; 112	77; 112	77; 112
1e-5	94; 128	94; 128	94; 128	94; 128	94; 128	94; 128	94; 128
1e-6	111; 128	111; 128	111; 128	111; 128	111; 128	111; 128	111; 128
1e-7	120; 128	120; 128	120; 128	120; 128	120; 128	120; 128	120; 128
1e-8	120; 128	120; 128	120; 128	120; 128	120; 128	120; 128	120; 128

Table 10: Size: 100000; Alpha: 100000; Rank: 120; Decay-value: 24; d-start: 16; d-add: 16. These numbers are "(Computed HSS Rank); (Random Samples Used)". Here, we are using $\mathrm{fl}\left[(I - Q_1 Q_1^*) S_2\right] = (I - Q_1 Q_1^*)^2 S_2$, with the products computed intelligently.

Adaptivity example 2: constant singular value (worst case)

$$A = \alpha I + UDV^*,\ U,V\ rank = 120,\ D_{k,k} = 1$$

$\varepsilon_r \backslash \varepsilon_a$	1e-2	1e-4	1e-6	1e-8	1e-10	1e-12	1e-14
1e-1	120; 128	120; 128	120; 128	120; 128	120; 128	120; 128	120; 128
1e-2	120; 128	120; 128	120; 128	120; 128	120; 128	120; 128	120; 128
1e-3	120; 128	120; 128	120; 128	120; 128	120; 128	120; 128	120; 128
1e-4	120; 128	120; 128	120; 128	120; 128	120; 128	120; 128	120; 128
1e-5	120; 128	120; 128	120; 128	120; 128	120; 128	120; 128	120; 128
1e-6	120; 128	120; 128	120; 128	120; 128	120; 768	120; 768	120; 768
1e-7	120; 128	120; 128	120; 128	120; 128	120; 768	120; 768	120; 768
1e-8	120; 128	120; 128	120; 128	120; 128	120; 768	120; 768	120; 768

Table 8: Size: 100000; Alpha: 100000; Rank: 120; Decay-value: 0; d-start: 16; d-add: 16. These numbers are "(Computed HSS Rank); (Random Samples Used)". Here, we are using $\mathrm{fl}\left[(I - Q_1 Q_1^*)\, S_2\right] = (I - Q_1 Q_1^*)^2\, S_2$, with the products computed intelligently.

Approximation error bounds for HSS construction

Xi, Xia, Cauley, Balakrishnan, SIMAX 2014

Let $\tilde{A}$ be the HSS constructed via randomization, τ be the relative tolerance for truncating the singular values of the HSS blocks, then

$$\left\|A-\tilde{A}\right\|_F = O(\tau \cdot \max(r^{O(L)}, n))\left\|A\right\|_F$$

If $\tilde{A}$ is constructed via truncated SVDs applied to the off−diagonal blocks of A, then $\left\|A-\tilde{A}\right\|_F = O(\tau\, L\sqrt{r})\ \|A\|_F$

Rest of the talk

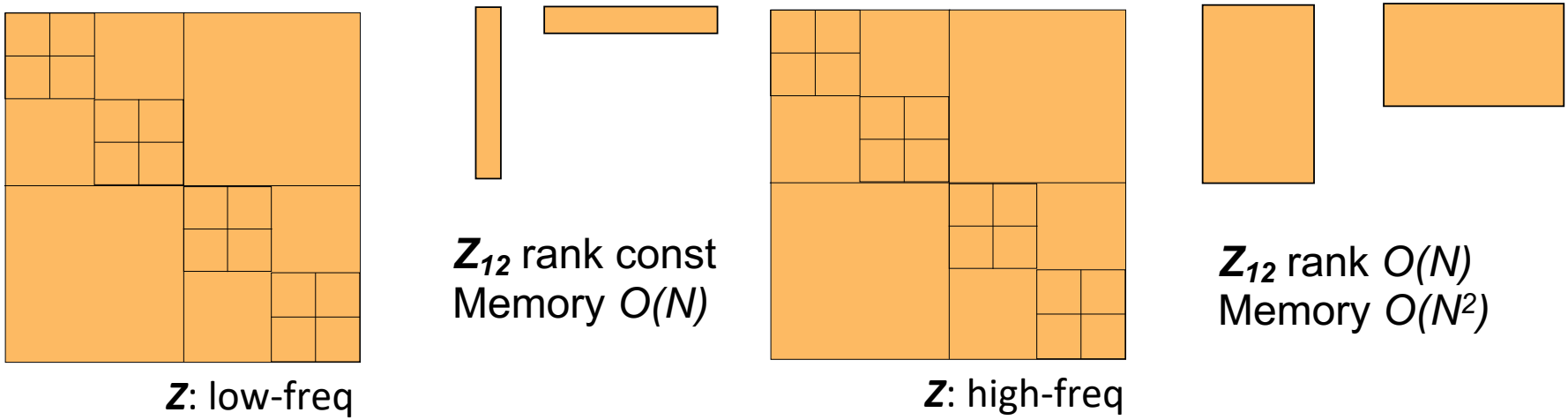
Complementary to David Keyes' lectures 3 & 4

Deep dives:

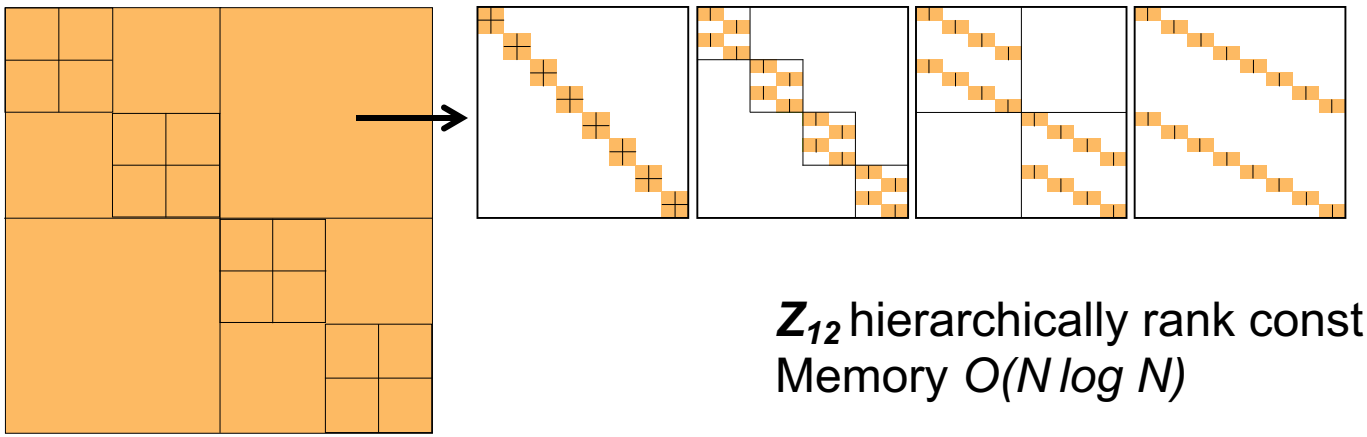
- Low-rank compression tool: adaptive randomized sketching
  - Understanding probabilistic error
- High-frequency wave equations: butterfly compression

What about wave equations ?

- Low/medium frequency: inversion cost $O(N \log^{\alpha} N)$
- High frequency: LR is not effective, inversion cost $O(N^3)$
- Remedies:
  - High-frequency FMM (Rokhlin 1993)
  - L-Sweeps (Taus), approximate-inverse (Ying), Xi, Vuik, etc.
  - Butterfly factorization (this talk)



Butterfly factorization



Z: high-freq

$\boldsymbol{Z_{12}}$ hierarchically rank const
Memory *O(N log N)*

- Butterfly is a FFT-inspired compression tool for highly oscillatory kernels
- Applications
  - Special function transforms (Radon, Fourier, spherical harmonic)
  - Fast direct and iterative integral (IE) equation solvers for high-frequency Helmholtz
  - Fast direct differential equation solvers for high-frequency Helmholtz
  - Scalable machine learning algorithms

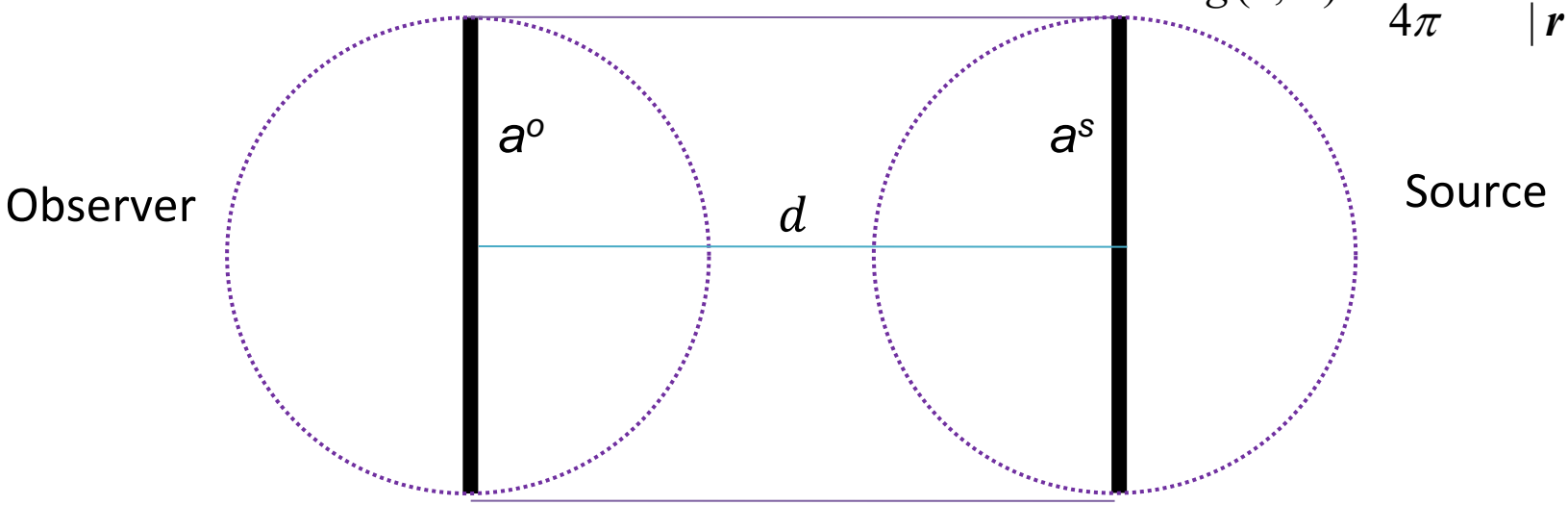
Physical interpretation of Butterfly

Degree of freedom (interaction rank): $2D \sim \dfrac{k\,a^s a^o}{d}, \quad 3D \sim \left(\dfrac{k\,a^s a^o}{d}\right)^2$

a^o : diameter of sub-observer. a^s : diameter of sub-source.

d : distance between source observer centers. k : wave number.

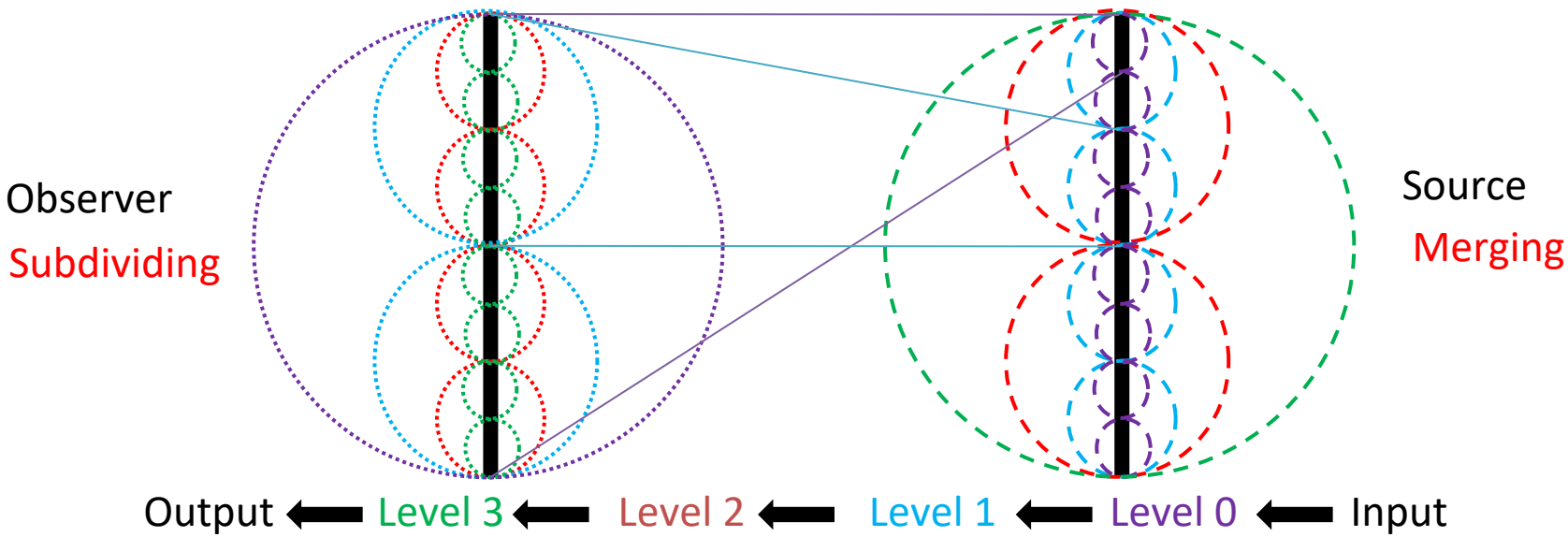
$$g(\boldsymbol{r},\boldsymbol{r}') = \frac{1}{4\pi}\frac{\exp(ik\,|\boldsymbol{r}-\boldsymbol{r}'|)}{|\boldsymbol{r}-\boldsymbol{r}'|}$$

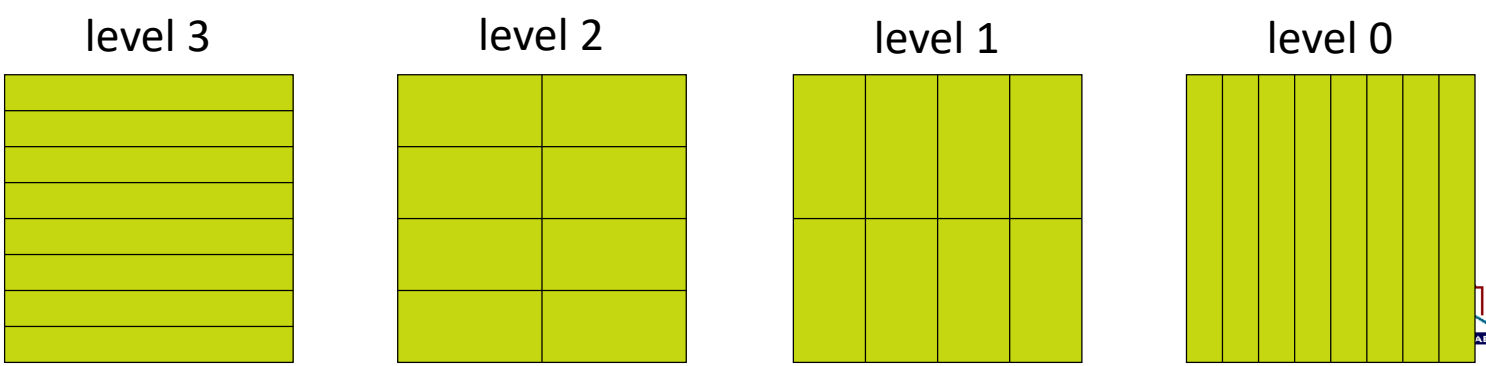


Physical interpretation of Butterfly

Degree of freedom (interaction rank): 2D $\sim \dfrac{k\,a^s a^o}{d}$, $\quad 3D \sim \left(\dfrac{k\,a^s a^o}{d}\right)^2$

The ranks of all submatrices are r = constant (butterfly rank)

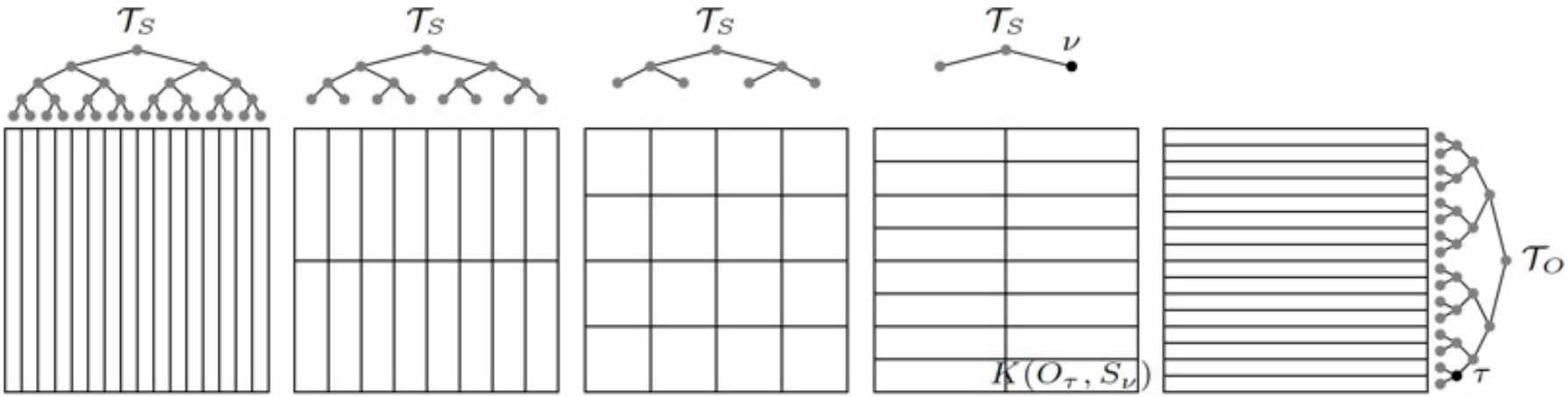




Hierarchical partitioning: dual trees

Complementary low-rank property

Example: 4-level butterfly decomposition



For any level l, can compress block $K(O_{\tau,} S_\nu)$

$$K(O_{\tau,} S_\nu) \approx U_{\tau,\nu} K(\bar{O}_{\tau,} \bar{S}_\nu) V_{\tau,\nu}^T = U_{\tau,\nu} B_{\tau,\nu} V_{\tau,\nu}^T$$

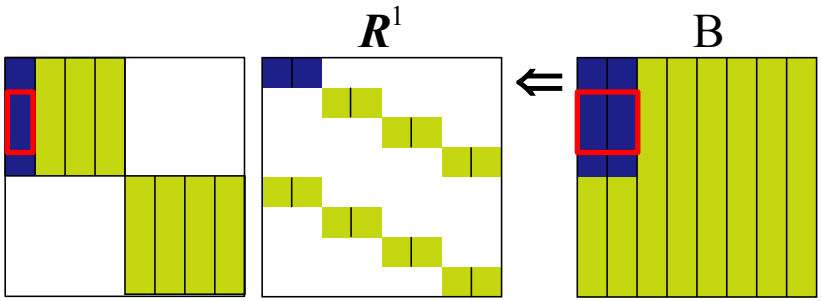
$\bar{O}_{\tau,}$ represents skeleton rows
$\bar{S}_\nu$ represents skeleton columns

Butterfly Construction: entry-evaluation based

Let ***B*** has butterfly rank *r*, its *L*-level butterfly factorization is $\boldsymbol{B} = \boldsymbol{R}^L \ldots \boldsymbol{R}^1$

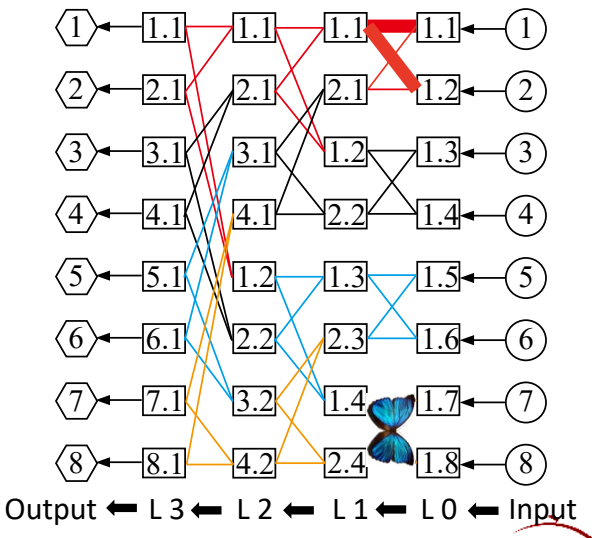
Compute LR compression for all judiciously selected submatrices

$R^{\ell}_{i,j}$ has size at most $r \times r$, Memory $O(n \log n)$, Flops $O(n \log n)$



$$\boldsymbol{R}^l = \mathrm{diag}(\boldsymbol{R}^l_1, \ldots, \boldsymbol{R}^l_{2^{l-1}})$$

$$\boldsymbol{R}^l_i = \begin{pmatrix} \boldsymbol{R}^l_{2i-1,1} \mid \boldsymbol{R}^l_{2i-1,2} & & \\ & \ddots & \\ & & \boldsymbol{R}^l_{2i-1,2^{L-l+1}-1} \mid \boldsymbol{R}^l_{2i-1,2^{L-l+1}} \\ \boldsymbol{R}^l_{2i,1} \mid \boldsymbol{R}^l_{2i,2} & & \\ & \ddots & \\ & & \boldsymbol{R}^l_{2i,2^{L-l+1}-1} \mid \boldsymbol{R}^l_{2i,2^{L-l+1}} \end{pmatrix}$$

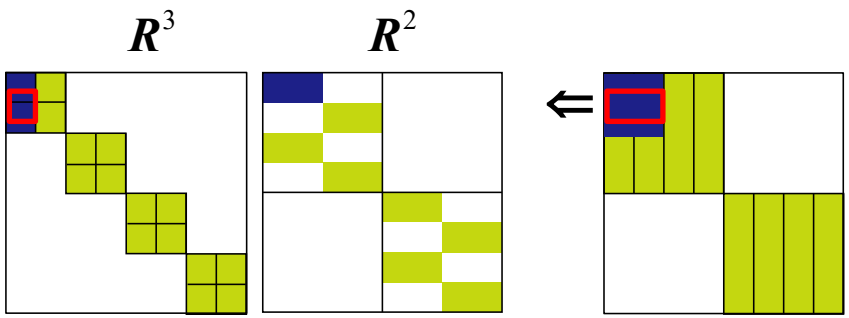


Butterfly Construction: entry-evaluation based

Let ***B*** has butterfly rank *r*, its *L*-level butterfly factorization is $\boldsymbol{B} = \boldsymbol{R}^L \ldots \boldsymbol{R}^1$

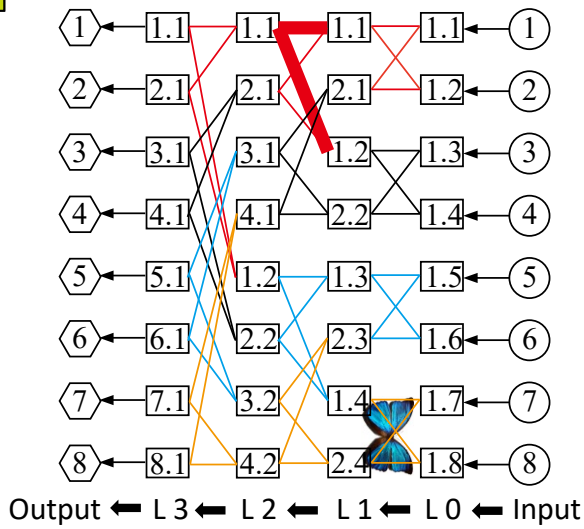
Compute LR compression for all judiciously selected submatrices

$R_{i,j}^{\ell}$ has size at most $r \times r$, Memory $O(n \log n)$, Flops $O(n \log n)$



$$\boldsymbol{R}^l = \mathrm{diag}(\boldsymbol{R}_1^l, \ldots, \boldsymbol{R}_{2^{l-1}}^l)$$

$$\boldsymbol{R}_i^l = \begin{pmatrix} \boldsymbol{R}_{2i-1,1}^l \mid \boldsymbol{R}_{2i-1,2}^l & & \\ & \ddots & \\ & & \boldsymbol{R}_{2i-1,2^{L-l+1}-1}^l \mid \boldsymbol{R}_{2i-1,2^{L-l+1}}^l \\ \boldsymbol{R}_{2i,1}^l \mid \boldsymbol{R}_{2i,2}^l & & \\ & \ddots & \\ & & \boldsymbol{R}_{2i,2^{L-l+1}-1}^l \mid \boldsymbol{R}_{2i,2^{L-l+1}}^l \end{pmatrix}$$

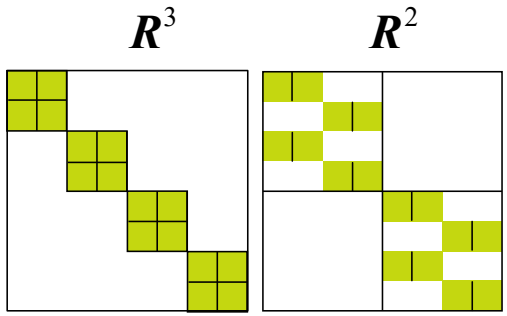


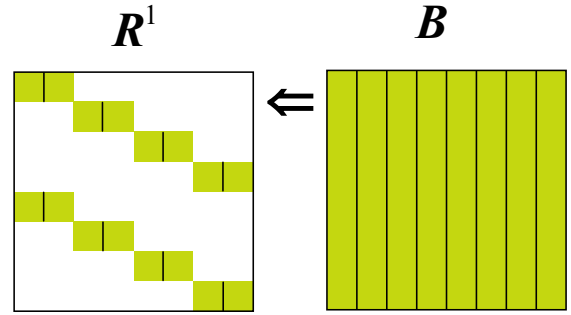
Butterfly Construction: entry-evaluation based

Let ***B*** has butterfly rank *r*, its *L*-level butterfly factorization is $\boldsymbol{B} = \boldsymbol{R}^L \ldots \boldsymbol{R}^1$

Compute LR compression for all judiciously selected submatrices

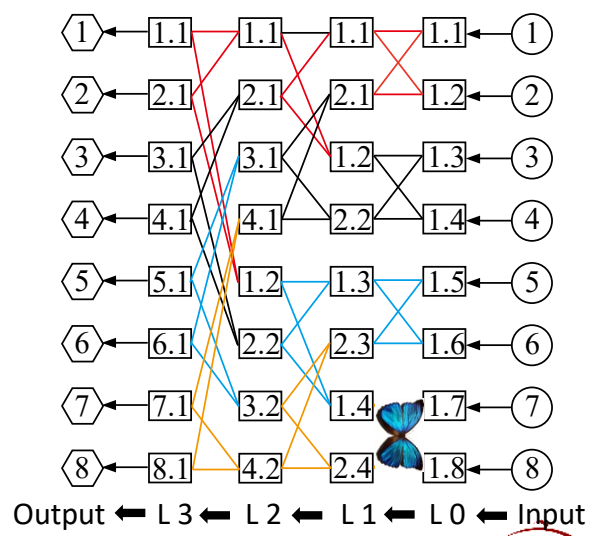
$R_{i,j}^{\ell}$ has size at most $r \times r$, Memory $O(n \log n)$, Flops $O(n \log n)$



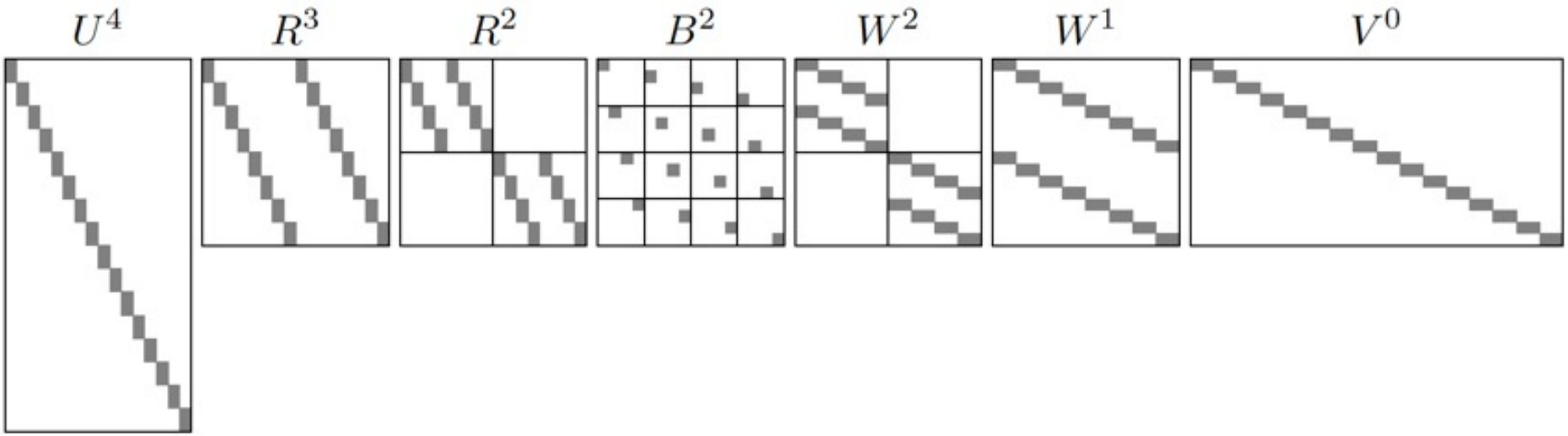


$$\mathrm{R}^l = \mathrm{diag}(\mathrm{R}_1^l, \ldots, \mathrm{R}_{2^{l-1}}^l)$$

$$\boldsymbol{R}_i^l = \begin{pmatrix} \boldsymbol{R}_{2i-1,1}^l \mid \boldsymbol{R}_{2i-1,2}^l & & \\ & \ddots & \\ & & \boldsymbol{R}_{2i-1,2^{L-l+1}-1}^l \mid \boldsymbol{R}_{2i-1,2^{L-l+1}}^l \\ \boldsymbol{R}_{2i,1}^l \mid \boldsymbol{R}_{2i,2}^l & & \\ & \ddots & \\ & & \boldsymbol{R}_{2i,2^{L-l+1}-1}^l \mid \boldsymbol{R}_{2i,2^{L-l+1}}^l \end{pmatrix}$$



4-level butterfly compressed representation



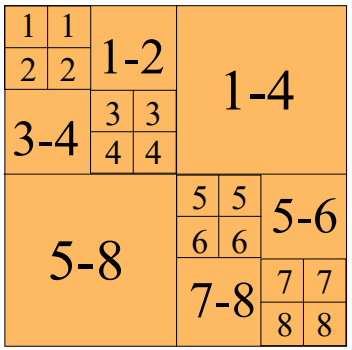
After construction, can do several operations

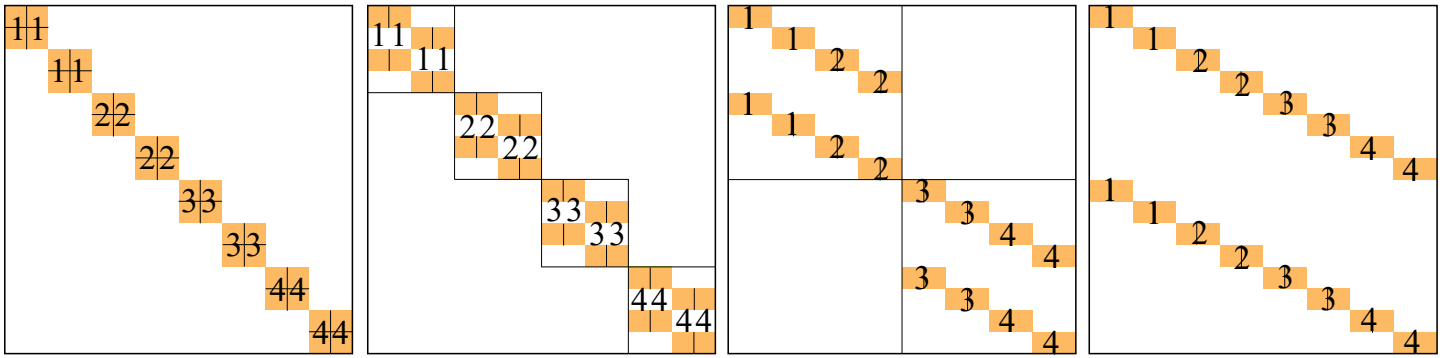
- Matrix-vector multiplication
  - Michielssen, Boag, 1996
  - Li, Yang, Martin, Ho, Ying, 2015
  - Poulson, Demanet , Maxwell, Ying, 2014 – parallelization

- Factorization / inversion
  - Liu, Guo, Michielssen, 2017
  - Liu, Xing, Guo, Michielssen, Ghysels, Li, 2020

Distributed-memory Parallelization: ButterflyPACK

- HODLR with Butterfly, symmetric bit-reversal ordering for each butterfly



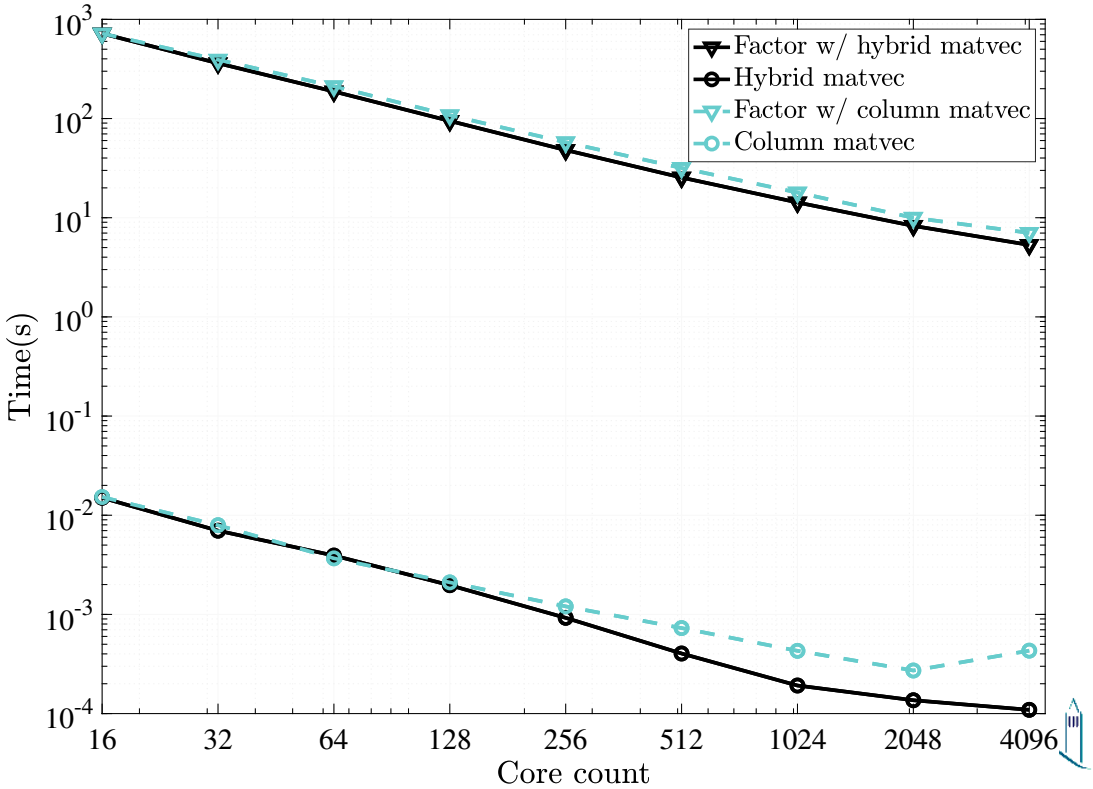


distributed

Distributed butterfly of Z_{12}

n = 2.56 x 10^6

core count: 16 - 4096

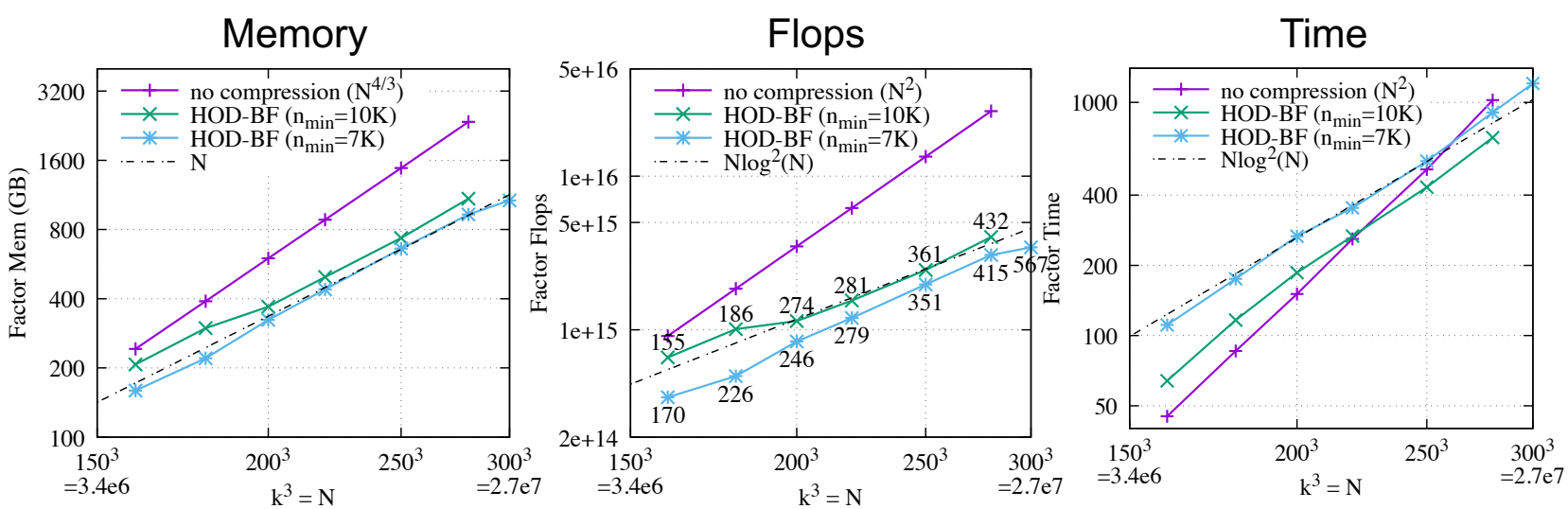


Example: 3D high-frequency Helmholtz equations

$$\left(\sum_i \rho(x)\frac{\partial}{\partial x_i}\frac{1}{\rho(x)}\frac{\partial}{\partial x_i}\right)p(x) + \frac{\omega^2}{\kappa^2(x)}p(x) = -f(x)$$

$\rho(x)$: mass density
$f(x)$: acoustic excitation
$p(x)$: pressure
ω: angular frequency

- Set $\omega = 8\pi Hz$, 15 grid points per wavelength
- Hibrid algorithm: HOD-BF = HOLDR + Butterfly
- 1024 CPU cores of the Cori machine at NERSC



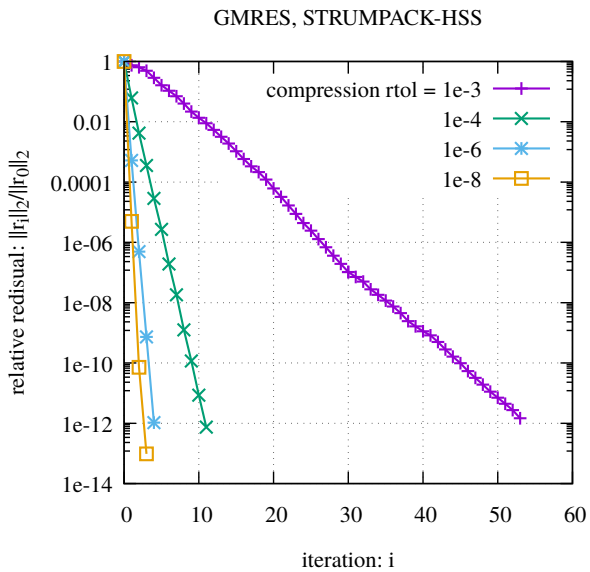
Example: precondition indefinite Maxwell equations

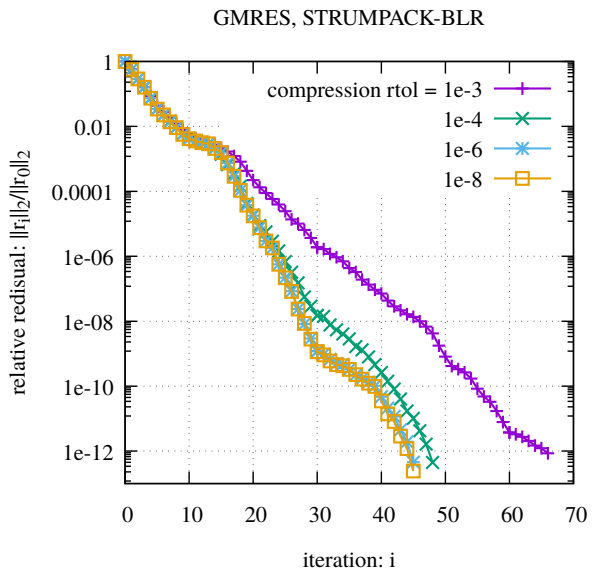
- Electromagnetic diffusion problem in MFEM (Mark Stowell, Tzanio Kolev)

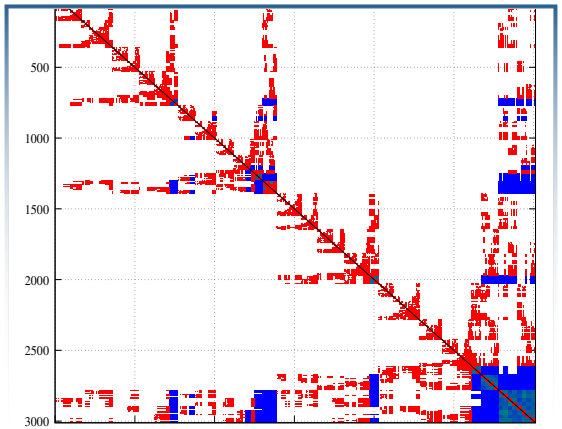
$$\nabla \times \nabla \times E - \Omega^2 E = (k^2 - \Omega^2)\sin(k\,x)$$

$$(\text{e.g., } \Omega = 2,\ \ k = \Omega / 2,\ \ \text{periodic-cube.mesh, } 331{,}776 \text{ DOFs})$$

- GMRES with AMS preconditioner (Auxiliary-space Multigrid Maxwell Solver) does not converge after 500 iterations
- GMRES with HSS and BLR work well







Blue blocks denote low-rank BLR tiles

Algorithm complexity (in bigO sense)

- Dense LU: $O(N^3)$
- Model PDEs with regular mesh, Nested Dissection ordering

	2D problems $N = k^2$			**3D problems $N = k^3$**		
	Factor flops	Solve flops	Memory	Factor flops	Solve flops	Memory
Exact sparse LU	$N^{3/2}$	N log(N)	N log(N)	N^2	$N^{4/3}$	$N^{4/3}$
STRUMPACK With LR compression	N	N	N	N^{α} polylog(N) ($\alpha < 2$)	N log(N)	N log(N)

Perspectives, future directions

- Techniques from structured matrices are very promising for inexact direct solvers and preconditioning
  - LA tools: ID, randomization
  - Parallelism: coarse-grain (trees), fine-grain (dense submatrices)

- Wide spectrum of algorithms, not (yet) possible to have a decision tree of algorithm choices (e.g., iterative solution template book)
  - Problem-specific (esp. clustering)
  - Implementations are still being worked on for larger scale problems and machines, esp. GPUs

- Randomized LA is a very useful tool, rigorous error analyses are needed to understand approximation quality

References

- J. Xia, S. Chandrasekaran, M. Gu, X. S. Li, ``Fast Algorithms for Hierarchically Semiseparable Matrices'', Numerical Linear Algebra with Applications, Vol 17, Issue 6, 953-976, 2010.
- A. Napov and X.S. Li, ``An algebraic multifrontal preconditioner that exploits the low-rank property'', Numerical Linear Algebra with Applications, 2016, 23:61-82.
- C. Gorman, G. Chavez, P. Ghysels, T. Mary, F.-H. Rouet, X.S. Li, ``Robust and Accurate Stopping Criteria for Adaptive Randomized Sampling in Matrix-free Hierarchically Semiseparable Construction", SIAM J. Sci. Comput., Vol. 41, No. 5, pp. S61-S85, 2019.
- Y. Liu, W. Sid-Lakhdar, E. Rebrova, P. Ghysels, and X.S. Li, ``A Parallel Hierarchical Blocked Adaptive Cross Approximation Algorithm", Int. Journal of High Performance Computing, September 2019.
- G. Chavez, Y. Liu, P. Ghysels, X.S. Li, ``Scalable and Memory-Efficient Kernel Ridge Regression", IPDPS 2020, May 18-22, New Orleans.
- Y. Liu, X. Xing, H. Guo, E. Michielssen, P. Ghysels, X.S. Li, ``Butterfly factorization via randomized matrix-vector multiplications'', SIAM J. Sci. Comput., 2021
- Y. Liu, P. Ghysels, L. Claus, X.S. Li, ``Sparse Approximate Multifrontal Factorization with Butterfly Compression for High Frequency Wave Equations", SIAM J. Sci. Comput., 2021.

- P. Sao, R. Vuduc, X.S. Li, ``A communication-avoiding 3D algorithm for sparse LU factorization on heterogeneous systems'', J. Parallel and Distributed Computing (JPDC), September 2019.
- P. Sao, R. Kannan, X.S. Li, R. Vuduc, ``A communication-avoiding 3D sparse triangular solver", ICS 2019, June 26-28, Phoenix, AZ. Proceedings, pp. 127-137.

THANK YOU