STARS-H: a High Performance Data-Sparse Matrix Market Library on Large-Scale Systems

A. Mikhalev [1] H. Ltaief [2] D. Keyes [2]

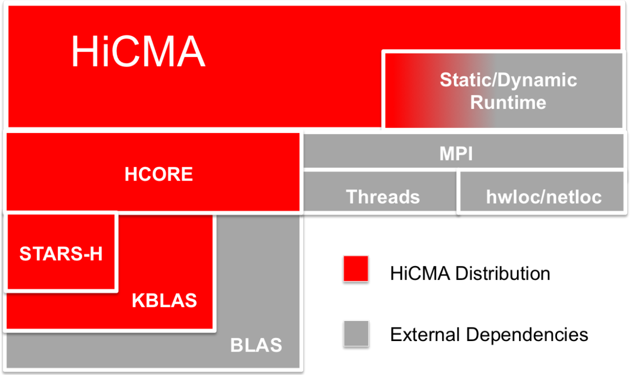
[1]RWTH Aachen, Germany

[2]ECRC, KAUST, Saudi Arabia

Inria Skoltech Moliére workshop, 6-8 July 2021 (Online)

What is the STARS-H?

The STARS-H is a part of a library called HiCMA which stands for Hierarchical Computations on Manycore Architectures.



Goal of the STARS-H is to feed the HiCMA with inputs.

What is the STARS-H?

The STARS-H stands for

Software for
Testing
Accuracy
Reliability and
Scalability of
-
Hierarchical computations.

Key reason

Every software for hierarchical matrices lives its own life: different programming languages, different input formats, different ways to store data and different target applications and hardware. This makes it difficult to compare results, reuse them in other software and present a "fair" list of best practices. **Salvation is a standardization**.

Why Hierarchical Matrices?

Such matrices are algebraic analogs or extension of the Fast Multipole Method (the FMM), that makes use of low-rank approximations in an analytical way. The FMM-able matrices are dense and fully-populated, but there are patterns exploiting low-rank property.

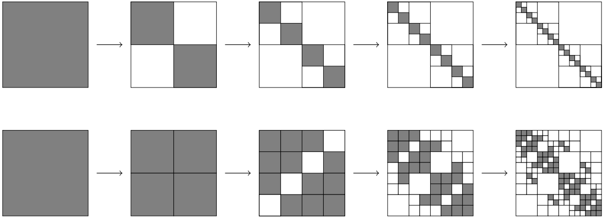


Figure: Examples of mosaic partitioning into $\mathcal{H}$-matrix. Gray submatrices are full-rank, while white submatrices are low-rank.

Why Hierarchical Matrices?

The hierarchical matrices are based on a simple geometric idea: interactions between groups of far-away objects can be described by a small amount of parameters up to any given precision.

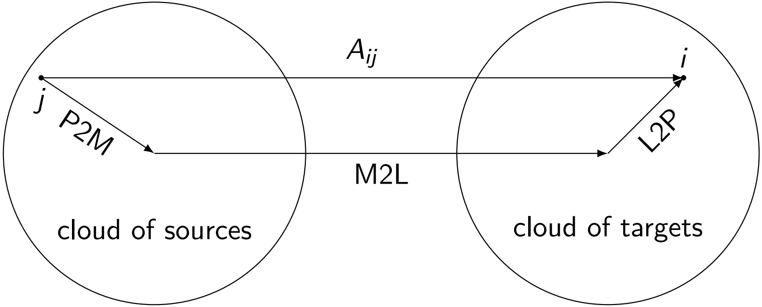


Figure: Approximation of a low-rank submatrix in a notation of the FMM.

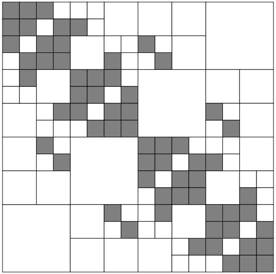
$$A \approx (\mathrm{L2P})_i \circ (\mathrm{M2L}) \circ (\mathrm{P2M})_j. \tag{1}$$

Why Hierarchical Matrices? N-body problems!

N-body problem (e.g. astrophysics, electrostatics): given "particles" $\{x_i\}$ with "charges" $\{q_i\}$, find "potentials" $\{p_i\}$, such that

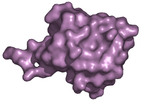
$$p_i = \sum_j f(x_i, x_j) q_j,$$

where $f(x_i, x_j)$ is a "potential" at x_i, created by a "particle" in x_j with a unit charge. If f is asymptotically smooth, then:



Why Hierarchical Matrices? Boundary integrals!

1. Given is a molecule Ω



Solve a boundary problem:

$$\begin{cases} Lu(x) = 0, & x \in \Omega, \\ Du(x) = \phi(x), & x \in \partial\Omega. \end{cases}$$

2. Let $G(x,y)$ be Green's function of operator L in a free space, then:

$$u(x) = \int_{\partial\Omega} G(x,y)w(y)dy,$$

3. Kernel $K(x,y)$, defined as:

$$K(x,y) = DG(x,y),$$

is, usually, asymptotically smooth.

4. Get **boundary integral equation**

$$\int_{\partial\Omega} K(x,y)w(y)dy = \phi(x), x \in \partial\Omega,$$

5. Discretize it into a system of equations

$$\tilde{K}\tilde{w} = \tilde{\phi}.$$

6. Use low-rank patterns of $\tilde{K}$ to solve the system fast

Why Hierarchical Matrices? Fractional derivatives!

Fractional derivatives appear in different non-stationary problems, where short-term system evolution does not follow integer PDE (e.g. non-classical diffusion).

One of many definitions of a fractional derivative

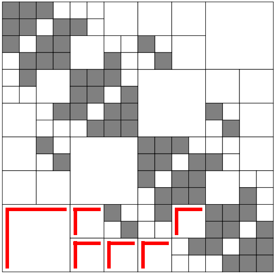
Fractional derivative of power α with n as nearest integer greater than α is following (Riemann-Liouville fractional derivative):

$$ {}_aD_t^{\alpha} f(t) = \frac{d^n}{dt^n} {}_aD_t^{-(n-\alpha)} f(t) = \frac{1}{\Gamma(n-\alpha)} \frac{d^n}{dt^n} \int_a^t \frac{f(\tau)}{(t-\tau)^{\alpha+1-n}} d\tau. \quad (2) $$

Fractional derivatives lead to dense matrices even for uniform discretizations. Example: oil droplet in water.

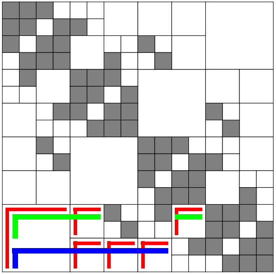
Why Hierarchical Matrices? Different formats!

- $\mathcal{H}$/HODLR.



Independent low-rank approximations.

- $\mathcal{H}^2$/HSS.

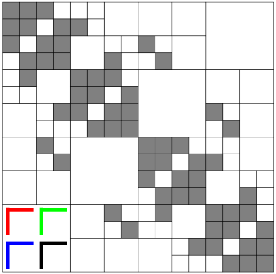


Submatrices share low-rank factors.

Why Hierarchical Matrices? Different formats!

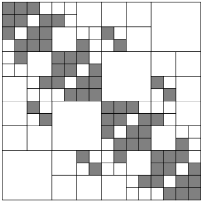
- $\mathcal{H}$/HODLR. Independent low-rank approximations.
- $\mathcal{H}^2$/HSS. Submatrices share low-rank factors.

- TLR/BLR. Independent per-tile storage.



Although this format seems "dumb" at the first sight, it is still useful in certain scenarios, e.g. due to a growing fill-in in a Cholesky factorization of an $\mathcal{H}$-matrix.

Why Hierarchical Matrix Market? Application-wise.



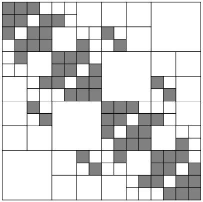
There are many formats of hierarchical matrices: e.g., HODLR, HSS, $\mathcal{H}$ and $\mathcal{H}^2$. The variety of software libraries is even greater: hlib, hlibpro, h2tools, strumpack, hicma, kifmm, pvfmm, hodlrlib, ...
Every library and every format have its own advantages and disadvantages.

Assume you have your own application. An opportunity and a challenge is to select such a software library, that fits a given hardware best.

Goal of the `STARS-H`

Provide comparison of a set of software libraries for a given application on a given hardware.

Why Hierarchical Matrix Market? Computation-wise.



There are many formats of hierarchical matrices: e.g., HODLR, HSS, $\mathcal{H}$ and $\mathcal{H}^2$. The variety of software libraries is even greater: hlib, hlibpro, h2tools, strumpack, hicma, kifmm, pvfmm, ...
Every library and every format have its own advantages and disadvantages.

Assume you have your own software library. An opportunity and a challenge is to compare against other software, since there is no standard set of tests.

Goal of the STARS-H

Provide a set of applications to standardize comparison of hierarchical libraries, just like **UF Sparse Matrix Collection** does for sparse libraries.

The STARS-H 0.3.0

Is an open-source project with a version 0.3.0 released on 21st of November 2020. Current features are:

- Data formats: Tile Low-Rank (TLR).
- Data types: double precision.
- Operations: approximation, TLR-by-dense product, CG solve.
- Synthetic applications: random TLR, Cauchy.
- Real applications: electrostatics, electrodynamics, spatial statistics, mesh deformations, acoustic wave scattering.
- Programming models: OpenMP, MPI and task-based via `StarPU`.
- Approximation techniques: SVD, Rank-Revealing QR, Randomized SVD.
- Hardware: CPUs and GPUs (CuBLAS/KBLAS).

Publicly available at `https://github.com/ecrc/stars-h`.

$\mathcal{H}$-Matrix Market: Synthetic Kernels

- Random Tile-Low Rank: each tile (i, j) is generated as $U_i S U_j$, where $\{U_i\}$ is a set of orthogonal square matrices, generated randomly, and S is a diagonal matrix of singular values, given by user.
- Cauchy: matrix element (i, j) is generated as $\frac{1}{r(x_i, x_j)}$ for some one-dimensional distribution of $\{x_i\}$.
- Real part of fundamental solution of Helmholtz equation: matrix kernel is

$$\frac{\cos(kr)}{r}$$

- Imaginary part of fundamental solution of Helmholtz equation: matrix kernel is

$$\frac{\sin(kr)}{r}$$

Distribution of Ranks for the TLR Format: Synthetic Kernels

8	8	8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	9	8	8
8	8	8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8	8	8
8	8	9	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8	8	8

(a) $\frac{\sin(kr)}{r}$, $k = 1$

26	27	28	28	27	27	28	28	27	27
27	27	28	28	28	27	28	28	28	27
28	28	28	28	28	27	29	29	28	28
28	28	28	27	27	27	28	29	28	28
27	28	28	27	25	26	27	27	27	27
27	27	27	27	26	25	27	28	28	27
28	28	29	28	27	27	27	28	28	28
28	28	29	29	27	28	28	28	28	28
27	28	28	28	27	28	28	28	27	27
27	27	28	28	27	27	28	28	27	26

(b) $\frac{\sin(kr)}{r}$, $k = 10$

107	111	108	87	78	91	75	79	70	61
111	113	114	91	88	106	96	98	82	70
108	114	120	112	108	98	87	104	98	79
87	91	112	112	109	74	66	87	96	75
78	88	108	109	106	82	74	98	106	91
91	106	98	74	82	106	109	108	88	78
75	96	87	66	74	109	112	112	91	87
79	98	104	87	98	108	112	120	114	108
70	82	98	96	106	88	91	114	113	111
61	70	79	75	91	78	87	108	111	107

(c) $\frac{\sin(kr)}{r}$, $k = 50$

250	250	250	114	104	250	103	111	97	82
250	250	250	115	115	250	250	250	115	97
250	250	250	250	250	250	111	250	250	111
114	115	250	250	250	90	79	111	250	103
104	115	250	250	250	101	90	250	250	250
250	250	250	90	101	250	250	250	115	104
103	250	111	79	90	250	250	250	115	114
111	250	250	111	250	250	250	250	250	250
97	115	250	250	250	115	115	250	250	250
82	97	111	103	250	104	114	250	250	250

(d) $\frac{\sin(kr)}{r}$, $k = 100$

250	115	69	35	27	33	23	24	21	18
115	250	108	43	36	79	36	33	26	21
69	108	250	104	69	69	37	48	33	24
35	43	104	250	111	29	23	37	36	23
27	36	69	111	250	38	29	69	79	33
33	79	69	29	38	250	111	69	36	27
23	36	37	23	29	111	250	104	43	35
24	33	48	37	69	69	104	250	108	69
21	26	33	36	79	36	43	108	250	115
18	21	24	23	33	27	35	69	115	250

(e) $\frac{\cos(kr)}{r}$, $k = 1$

250	120	76	43	36	40	31	33	30	26
120	250	115	51	44	86	44	42	34	30
76	115	250	111	76	77	46	56	42	33
43	51	111	250	116	37	33	46	44	31
36	44	76	116	250	46	37	77	86	40
40	86	77	37	46	250	116	76	44	36
31	44	46	33	37	116	250	111	51	43
33	42	56	46	77	76	111	250	115	76
30	34	42	44	86	44	51	115	250	120
26	30	33	31	40	36	43	76	120	250

(f) $\frac{\cos(kr)}{r}$, $k = 10$

250	15	6	5	4	4	4	4	4	3
15	250	15	6	5	4	4	4	4	4
6	15	250	15	6	5	4	4	4	4
5	6	15	250	15	6	5	4	4	4
4	5	6	15	250	15	6	5	4	4
4	4	5	6	15	250	15	6	5	4
4	4	4	5	6	15	250	15	6	5
4	4	4	4	5	6	15	250	15	6
4	4	4	4	4	5	6	15	250	15
3	4	4	4	4	4	5	6	15	250

(g) Cauchy

250	10	10	10	10	10	10	10	10	10
10	250	10	10	10	10	10	10	10	10
10	10	250	10	10	10	10	10	10	10
10	10	10	250	10	10	10	10	10	10
10	10	10	10	250	10	10	10	10	10
10	10	10	10	10	250	10	10	10	10
10	10	10	10	10	10	250	10	10	10
10	10	10	10	10	10	10	250	10	10
10	10	10	10	10	10	10	10	250	10
10	10	10	10	10	10	10	10	10	250

(h) Random TLR

$\mathcal{H}$-Matrix Market: Real Kernels

- Electrostatics (potential): matrix kernel is

$$\frac{1}{r}$$

- Gaussian processes, exponential kernel

$$e^{-\frac{r}{l}}$$

- Mesh deformations

- Gaussian processes, square exponential

$$e^{-\frac{r^2}{2l^2}}$$

- Gaussian processes, Matérn

$$\frac{2^{1-\nu}}{\Gamma(\nu)}\left(\sqrt{2\nu}\frac{r}{l}\right)^{\nu} K_{\nu}\left(\sqrt{2\nu}\frac{r}{l}\right)$$

- Acoustic wave scattering

Matérn($\nu = 0.5$) $\longrightarrow$ exponential
Matérn($\nu = \infty$) $\longrightarrow$ square exponential

Exponential kernel

Convariance length l of an exponential covariance function

$$f(x,y) = e^{\frac{-r(x,y)}{l}}$$

can be non-scalar:

$$f(x,y) = e^{-\sqrt{g(x,y)}}$$

where

$$g(x,y) = \left(\frac{x_1 - y_1}{l_1}\right)^2 + \left(\frac{x_2 - y_2}{l_2}\right)^2 + \left(\frac{x_3 - y_3}{l_3}\right)^2$$

If l_1, l_2 and l_3 differ a lot, then standard oct-tree clusterization of coordinates x and y will not partition matrix into low-rank blocks.

Distribution of Ranks for the TLR Format: Real kernels

250	115	68	34	27	32	22	23	20	16
115	250	107	43	36	79	35	33	25	20
68	107	250	104	69	69	36	48	33	23
34	43	104	250	111	28	23	36	35	22
27	36	69	111	250	37	28	69	79	32
32	79	69	28	37	250	111	69	36	27
22	35	36	23	28	111	250	104	43	34
23	33	48	36	69	69	104	250	107	68
20	25	33	35	79	36	43	107	250	115
16	20	23	22	32	27	34	68	115	250

(a) $\frac{1}{r}$, $\tau = 10^{-9}$

250	104	64	33	27	33	25	25	23	19
104	250	97	39	34	72	36	35	27	23
64	97	250	94	64	62	34	45	35	25
33	39	94	250	100	27	21	34	36	25
27	34	64	100	250	34	27	62	72	33
33	72	62	27	34	250	100	64	34	27
25	36	34	21	27	100	250	94	39	33
25	35	45	34	62	64	94	250	97	64
23	27	35	36	72	34	39	97	250	104
19	23	25	25	33	27	33	64	104	250

(b) Exp., $\tau = 10^{-9}$

250	77	69	48	44	53	42	38	34	33
77	250	77	47	48	68	56	57	46	34
69	77	250	75	68	55	43	55	57	38
48	47	75	250	75	33	27	43	56	42
44	48	68	75	250	40	33	55	68	53
53	68	55	33	40	250	75	68	48	44
42	56	43	27	33	75	250	75	47	48
38	57	55	43	55	68	75	250	77	69
34	46	57	56	68	48	47	77	250	77
33	34	38	42	53	44	48	69	77	250

(c) Sqr.exp., $\tau = 10^{-9}$

250	89	65	40	35	43	35	33	30	28
89	250	84	40	40	69	47	46	37	30
65	84	250	82	64	54	37	47	46	33
40	40	82	250	87	27	22	37	47	35
35	40	64	87	250	31	27	54	69	43
43	69	54	27	31	250	87	64	40	35
35	47	37	22	27	87	250	82	40	40
33	46	47	37	54	64	82	250	84	65
30	37	46	47	69	40	40	84	250	89
28	30	33	35	43	35	40	65	89	250

(d) Matérn, $\tau = 10^{-9}$

250	52	24	9	7	8	6	6	5	5
52	250	50	13	10	31	9	9	7	5
24	50	250	49	24	28	10	14	9	6
9	13	49	250	50	8	6	10	9	6
7	10	24	50	250	11	8	28	31	8
8	31	28	8	11	250	50	24	10	7
6	9	10	6	8	50	250	49	13	9
6	9	14	10	28	24	49	250	50	24
5	7	9	9	31	10	13	50	250	52
5	5	6	6	8	7	9	24	52	250

(e) $\frac{1}{r}$, $\tau = 10^{-4}$

250	44	21	9	8	10	8	8	7	6
44	250	42	12	9	26	10	10	8	7
21	42	250	41	21	23	9	13	10	8
9	12	41	250	41	7	6	9	10	8
8	9	21	41	250	10	7	23	26	10
10	26	23	7	10	250	41	21	9	8
8	10	9	6	7	41	250	41	12	9
8	10	13	9	23	21	41	250	42	21
7	8	10	10	26	9	12	42	250	44
6	7	8	8	10	8	9	21	44	250

(f) Exp., $\tau = 10^{-4}$

31	28	23	15	13	15	12	10	9	9
28	33	27	14	15	22	17	16	13	9
23	27	34	26	22	17	13	16	16	10
15	14	26	33	27	9	8	13	17	12
13	15	22	27	30	12	9	17	22	15
15	22	17	9	12	30	27	22	15	13
12	17	13	8	9	27	33	26	14	15
10	16	16	13	17	22	26	34	27	23
9	13	16	17	22	15	14	27	33	28
9	9	10	12	15	13	15	23	28	31

(g) Sqr.exp., $\tau = 10^{-4}$

39	29	21	13	11	14	11	9	9	9
29	41	28	13	13	21	15	15	13	9
21	28	43	27	21	18	11	14	15	9
13	13	27	40	28	8	7	11	15	11
11	13	21	28	39	10	8	18	21	14
14	21	18	8	10	39	28	21	13	11
11	15	11	7	8	28	40	27	13	13
9	15	14	11	18	21	27	43	28	21
9	13	15	15	21	13	13	28	41	29
9	9	9	11	14	11	13	21	29	39

(h) Matérn, $\tau = 10^{-4}$

Distribution of Ranks for TLR Format

Different applications expose different rank distributions, even 2D block cyclic distribution of tiles may not improve performance.

250	115	68	34	27	32	22	23	20	16
115	250	107	43	36	79	35	33	25	20
68	107	250	104	69	69	36	48	33	23
34	43	104	250	111	28	23	36	35	22
27	36	69	111	250	37	28	69	79	32
32	79	69	28	37	250	111	69	36	27
22	35	36	23	28	111	250	104	43	34
23	33	48	36	69	69	104	250	107	68
20	25	33	35	79	36	43	107	250	115
16	20	23	22	32	27	34	68	115	250

250	104	64	33	27	33	25	25	23	19
104	250	97	39	34	72	36	35	27	23
64	97	250	94	64	62	34	45	35	25
33	39	94	250	100	27	21	34	36	25
27	34	64	100	250	34	27	62	72	33
33	72	62	27	34	250	100	64	34	27
25	36	34	21	27	100	250	94	39	33
25	35	45	34	62	64	94	250	97	64
23	27	35	36	72	34	39	97	250	104
19	23	25	25	33	27	33	64	104	250

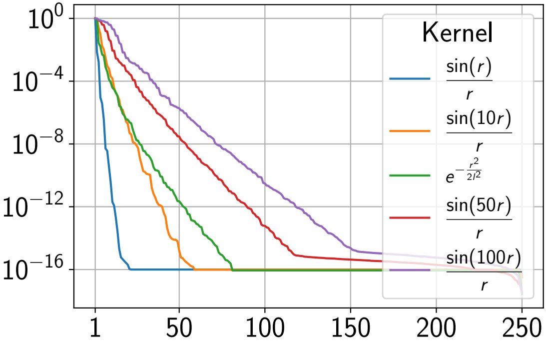
250	77	69	48	44	53	42	38	34	33
77	250	77	47	48	68	56	57	46	34
69	77	250	75	68	55	43	55	57	38
48	47	75	250	75	33	27	43	56	42
44	48	68	75	250	40	33	55	68	53
53	68	55	33	40	250	75	68	48	44
42	56	43	27	33	75	250	75	47	48
38	57	55	43	55	68	75	250	77	69
34	46	57	56	68	48	47	77	250	77
33	34	38	42	53	44	48	69	77	250

250	89	65	40	35	43	35	33	30	28
89	250	84	40	40	69	47	46	37	30
65	84	250	82	64	54	37	47	46	33
40	40	82	250	87	27	22	37	47	35
35	40	64	87	250	31	27	54	69	43
43	69	54	27	31	250	87	64	40	35
35	47	37	22	27	87	250	82	40	40
33	46	47	37	54	64	82	250	84	65
30	37	46	47	69	40	40	84	250	89
28	30	33	35	43	35	40	65	89	250

To avoid different load balance of computing cores/nodes one has to use asynchronous task-based programming models.

Decay of singular values

Normalized singular values of a left bottom tile of a 2500×2500 matrix with tile size 250 for different matrix kernels are following:



Performance impact on kernel dimensions

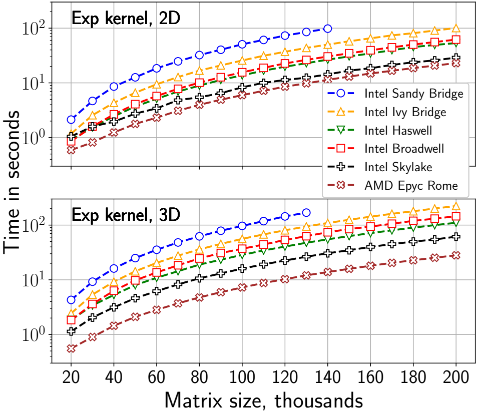


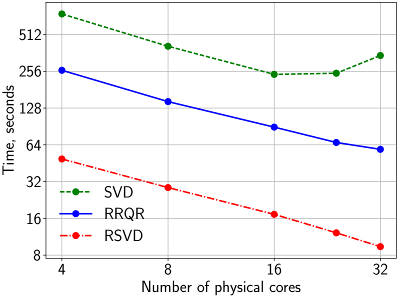
Figure: Randomized SVD compression time of 2D/3D exponential kernel ($\tau = 10^{-9}$) on different two-socket shared-memory systems.

Experimental Details

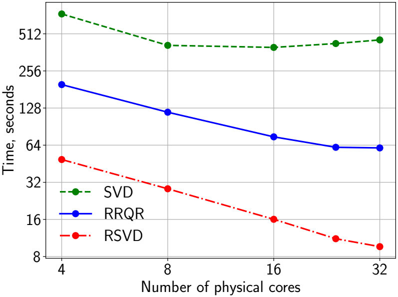
STARS-H was tested on a **Shaheen-II**, a CRAY XC40 system. The system has 6,174 dual sockets compute nodes based on 16 core Intel Haswell processors running at 2.3GHz. Each node has 128GB of DDR4 memory running at 2300MHz. Overall the system has a total of 197,568 processor cores and 790TB of aggregate memory.

Performance on a Single Node

TLR Approximation of **90k-by-90k** matrix of ones with **1k** tile size. Ordinary SVD in parallel for different matrices at a time saturates memory bandwidth, while Randomized SVDs show a decent scalability.



(a) Without HT.



(b) With HT.

Performance on a Single Node

Exponential and square exponential kernels get performance boost by adding "#pragma omp simd" directive and turning on hyperthreading.

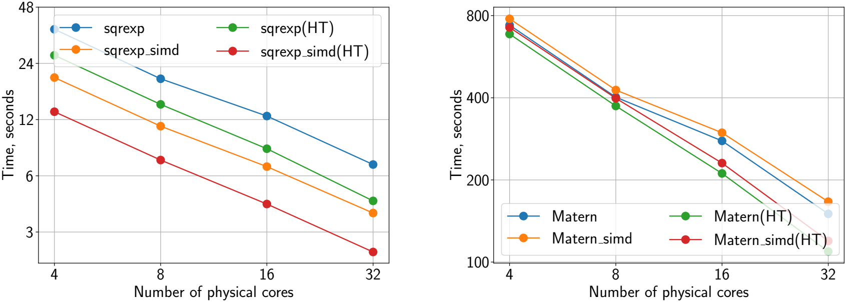
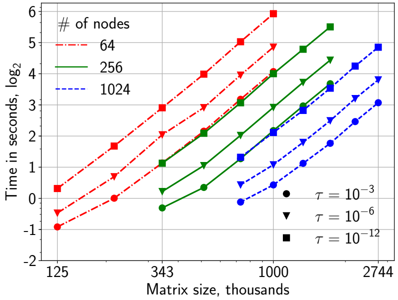
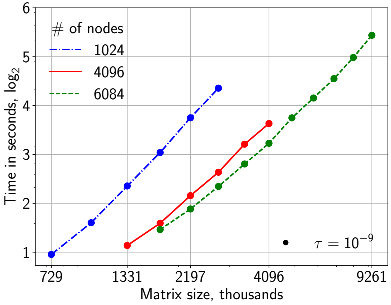


Figure: Generation of 90k-by-90k matrix of 2D spatial statistics problem

Performance on Multiple Nodes

Dependence of time-to-approximate on matrix size for spatial statistics problem with exponential covariation function in a 3D space. Up to **9 millions DOFs**, up to **6084** nodes ($\approx$ 200000 cores).





Performance on Multiple Nodes

Dependence of time-to-approximate on matrix size for spatial statistics problem with Matérn covariation function. Up to **6 millions DOFs**, up to **6084** nodes ($\approx$ 200000 cores).

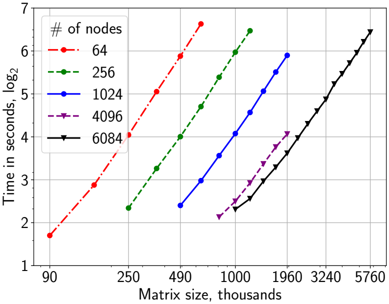


Figure: In a 2-dimensional space.

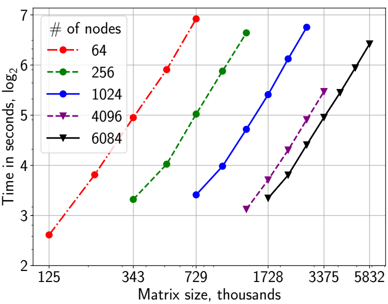
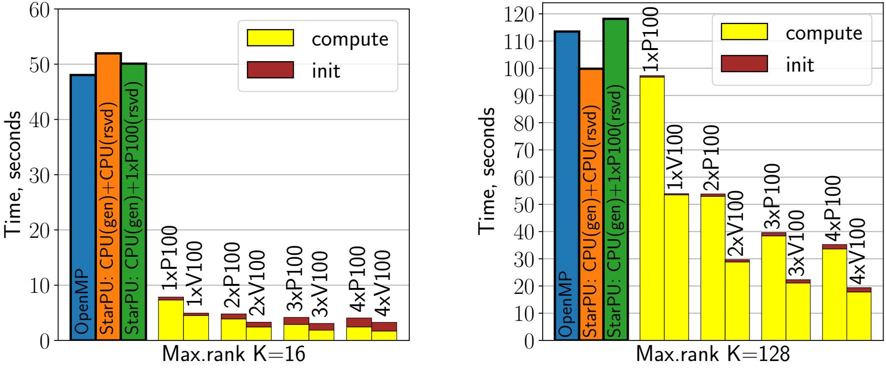


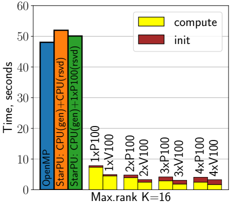
Figure: In a 3-dimensional space.

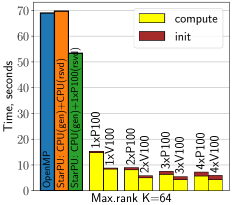
Performance on GPUs/KBLAS

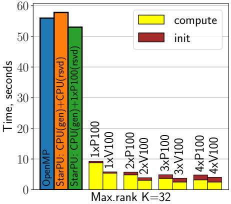


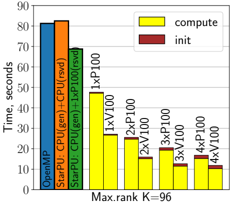
Tile Low-Rank approximation times of 153600-by-153600 matrix with different accuracies and random space sizes (for Randomized SVD). Randomized SVD on the GPUs is performed by the KBLAS library.

Performance on GPUs/KBLAS









TLR approximation of 153600-by-153600 matrix.

Conclusion

- Matrices generated by **STARS-H** have various rank distributions over tiles. It enables different inputs for further operations, like Cholesky factorization or matrix-matrix multiplication, making debugging and search for optimal implementation much easier.
- Although the TLR format itself is the simplest from a theoretical point of view, it scales nearly perfectly due to intrinsic load balance.
- The TLR could potentially be the best choice for emerging architectures, as a trend is to increase the number of computational cores, while sometimes decreasing power of a single core.
- As a popularity and number of compute nodes with several GPUs increases, TLR makes even more sense!

Future work

We are currently working on a C++ version of **STARS-H** to use C++ features to simplify development, improve user experience and bring following support:

- Boundary and volume integral equations.
- Different precisions: half, float, complex and mixed.
- Non-scalar kernels.
- Approximation techniques: adaptive randomized SVD, adaptive cross approximation schemes and interpolative decompositions.
- Hierarchical formats: $\mathcal{H}$ and $\mathcal{H}^2$.
- Support GPUs.

Acknowledgements

- Intel Parallel Computing Center
  KAUST/Cray Center of Excellence: Aniello Esposito and Adrian Tate
- For core-hours:
  Shaheen 2.0 from KAUST Supercomputer Lab
  Cray KNL internal system
- KAUST Supercomputer Lab
  B. Hadri and S. Feki
- INRIA/INP Bordeaux, France
  Runtime/HiePACS Teams
- R. Kriemann

References

1. L. Greengard and V. Rokhlin. 1987. A fast algorithm for particle simulations. *Journal of computational physics* 73, 2 (1987), 325-348.
2. E. Tyrtyshnikov. Mosaic-skeleton approximations. *Calcolo* 33, 1 (1996) 47-57.
3. W. Hackbusch. 1999. A Sparse Matrix Arithmetic Based on $\mathcal{H}$-Matrices. Part I: Introduction to $\mathcal{H}$-Matrices. *Computing* 62, 2 (1999), 89-108.
4. W. Hackbusch, B. Khoromskij and S. Sauter. 2000. On H2-Matrices. *Lectures on Applied Mathematics* (2000), 9-29.
5. SMart 2018. Structured Matrix Market. (Accessed January 2018). http://smart.math.purdue.edu.
6. S. Ambikasaran, D. Foreman-Mackey, L. Greengard, D. Hogg and M. O'Neil. 2016. Fast Direct Methods for Gaussian Processes. *IEEE Trans. Pattern Anal. Mach. Intell* 38, 2(2016), 252-265.
7. Y. Sun, B. Li and M. Genton. 2012. Geostatistics for Large Datasets. *In Space-Time Processes and Challenges Related to Environmental Problems*, 55-77.
8. C. Augonnet, S. Thibault, R. Namyst and P. Wacrenier. 2011. StarPU: a unified platform for task scheduling on heterogeneous multicore architectures. *Concurrency and Computation: Practice and Experience*, 23(2), pp.187-198.
9. B. Hadri, S. Kortas, S. Feki, R. Khurram and G. Newby. 2015. Overview of the KAUST's Cray X40 System: Shaheen II. *Cray User Group Conference*, Chicago, 2015.

Thank you for attention!
Questions?