Cholesky Factorization on Tile Low-Rank Matrices for Distributed-Memory Systems

Kadir Akbudak, Hatem Ltaief, Aleksandr Mikhalev, Ali Charara and David keyes
Extreme Computing Research Center
King Abdullah University of Science and Technology

IXPUG Workshop, KAUST

April 23-25, 2018

How does future of Dense Linear Algebra (DLA) look like?

How does future of Dense Linear Algebra (DLA) look like?

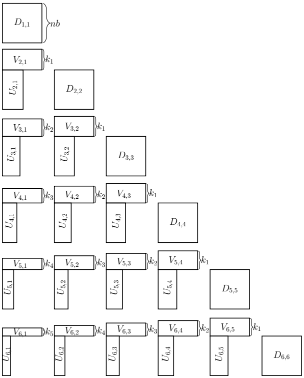
- Apparently dense matrices arising in scientific applications.

How does future of Dense Linear Algebra (DLA) look like?

- Apparently dense matrices arising in scientific applications.
- Dense matrices might be compressed.

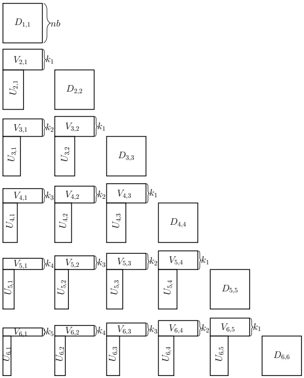
How does future of Dense Linear Algebra (DLA) look like?

- Apparently dense matrices arising in scientific applications.
- Dense matrices might be compressed.
  - Tile low rank (TLR) matrix format



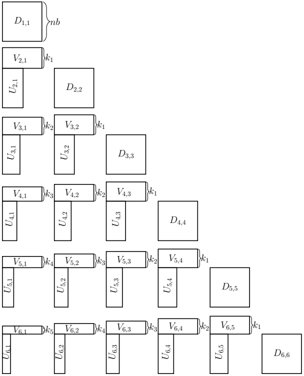
How does future of Dense Linear Algebra (DLA) look like?

- Apparently dense matrices arising in scientific applications.
- Dense matrices might be compressed.
  - Tile low rank (TLR) matrix format
  - Cholesky factorization (for distributed-memory architectures)



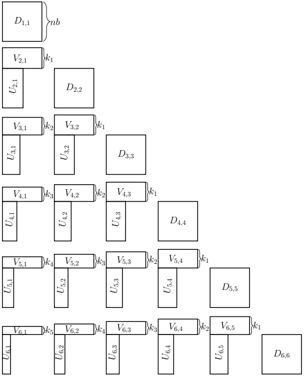
How does future of Dense Linear Algebra (DLA) look like?

- Apparently dense matrices arising in scientific applications.
- Dense matrices might be compressed.
    - Tile low rank (TLR) matrix format
    - Cholesky factorization (for distributed-memory architectures)
    - Huge performance improvement via cutting down flops



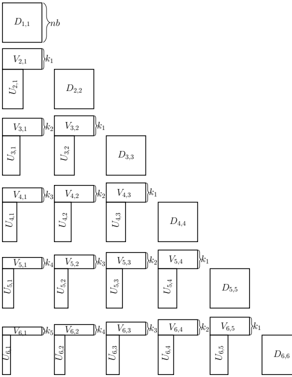
How does future of Dense Linear Algebra (DLA) look like?

- Apparently dense matrices arising in scientific applications.
- Dense matrices might be compressed.
  - Tile low rank (TLR) matrix format
  - Cholesky factorization (for distributed-memory architectures)
  - Huge performance improvement via cutting down flops
  - Significantly less memory via storing small $U_{i,j}$ and $V_{i,j}$ matrices instead of big $A_{i,j}$, where $A_{i,} = U_{i,j}V_{i,j}$



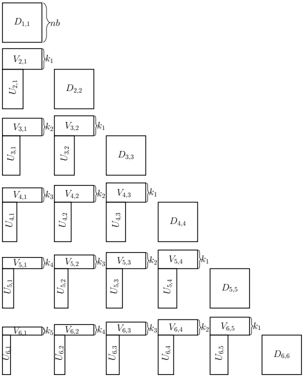
How does future of Dense Linear Algebra (DLA) look like?

- Apparently dense matrices arising in scientific applications.
- Dense matrices might be compressed.
  - Tile low rank (TLR) matrix format
  - Cholesky factorization (for distributed-memory architectures)
  - Huge performance improvement via cutting down flops
  - Significantly less memory via storing small $U_{i,j}$ and $V_{i,j}$ matrices instead of big $A_{i,j}$, where $A_{i,} = U_{i,j} V_{i,j}$
  - Preserving the accuracy requirements of the scientific application



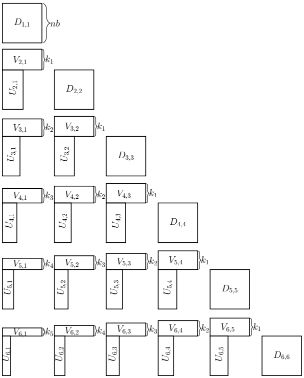
How does future of Dense Linear Algebra (DLA) look like?

- Apparently dense matrices arising in scientific applications.
- Dense matrices might be compressed.
  - Tile low rank (TLR) matrix format
  - Cholesky factorization (for distributed-memory architectures)
  - Huge performance improvement via cutting down flops
  - Significantly less memory via storing small $U_{i,j}$ and $V_{i,j}$ matrices instead of big $A_{i,j}$, where $A_{i,} = U_{i,j} V_{i,j}$
  - Preserving the accuracy requirements of the scientific application
- MKL ScaLAPACK



How does future of Dense Linear Algebra (DLA) look like?

- Apparently dense matrices arising in scientific applications.
- Dense matrices might be compressed.
    - Tile low rank (TLR) matrix format
    - Cholesky factorization (for distributed-memory architectures)
    - Huge performance improvement via cutting down flops
    - Significantly less memory via storing small $U_{i,j}$ and $V_{i,j}$ matrices instead of big $A_{i,j}$, where $A_{i,} = U_{i,j}V_{i,j}$
    - Preserving the accuracy requirements of the scientific application
- MKL ScaLAPACK
    - `pdpotrf` $\rightarrow$ `pdpotrf_tlr`?

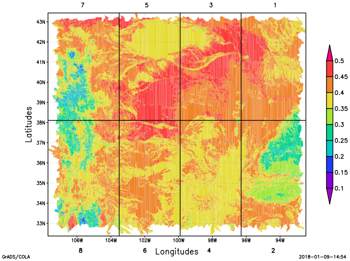


Climate/Weather Forecasting

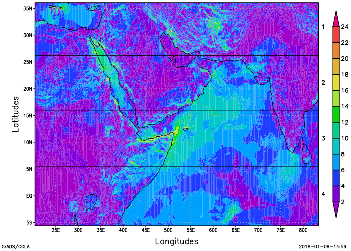
- Computational statistics: multivariate large spatial data sets in climate/weather modeling:

$$\ell(\boldsymbol{\theta}) = -\frac{1}{2}\mathbf{Z}^T\Sigma^{-1}(\boldsymbol{\theta})\mathbf{Z} - \frac{1}{2}\log|\Sigma(\boldsymbol{\theta})|$$

(a) Problem Definition.



(b) Soil moisture.



(c) Wind speed.

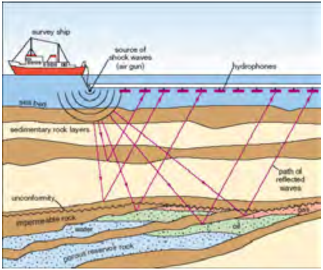
w/ Y. Sun and M. Genton

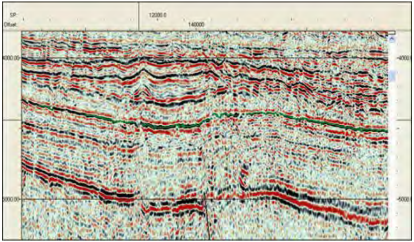
Earth Science

- Seismic imaging: Imaging the subsurface by solving the Helmholtz equation (acoustic wave equation):

$$(-\triangle - k^2)u(x, w) = s(x, w)$$

$k = \frac{w}{v(x)}$, w is the angular frequency, v(x) is the seismic velocity field, and u(x, w) is the time-harmonic wavefield solution to the forcing term s(x, w).





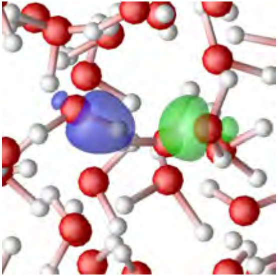
w/ S. Zampini

Materials Science

- Structural and vibrational analysis to problems in computational physics and chemistry like electronic and band structure calculations

$$(A - \lambda B)\, x = 0$$

(a) Problem Definition.



(b) Electronic structure.

Figure: Design of new materials.

w/ U. Schwingenschlogl

Common Properties

- Symmetric, positive-definite matrix

Common Properties

- Symmetric, positive-definite matrix
- (Apparently) dense matrices

Common Properties

- Symmetric, positive-definite matrix
- (Apparently) dense matrices
- Often data-sparse

Common Properties

- Symmetric, positive-definite matrix
- (Apparently) dense matrices
- Often data-sparse
    - Decay of parameter correlations with distance

Common Kernel: Cholesky Factorization (`potrf`)

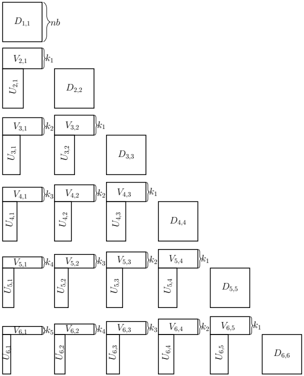
The Cholesky factorization of an $N \times N$ real symmetric, positive-definite matrix A has the form

$$A = LL^T,$$

where L is an $N \times N$ real lower triangular matrix with positive diagonal elements.

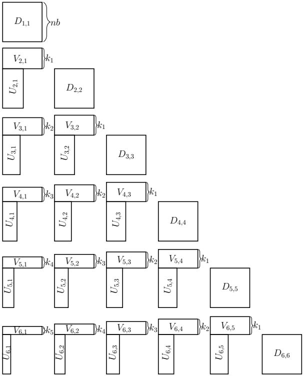
Ranks after Compression into Tile Low Rank (TLR) Format

- TLR format with varying ranks



Ranks after Compression into Tile Low Rank (TLR) Format

- TLR format with varying ranks

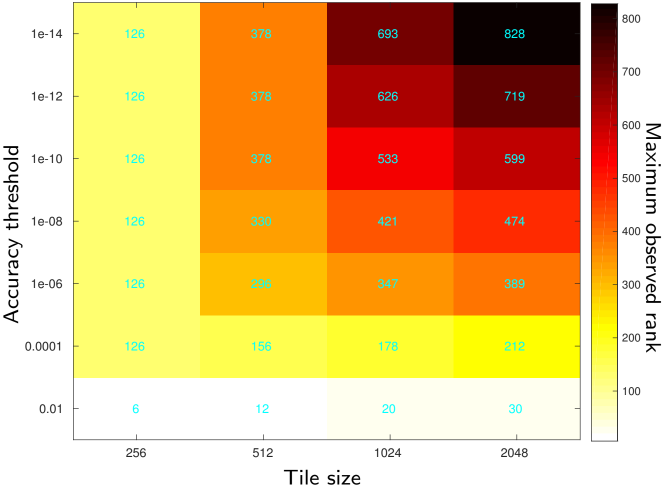


- Geospatial statistics, matrix size: 20000, block size: 500, accuracy threshold: 10^{-9}, 2D problem

	0					5					10					15				
0	500	133	47	35	154	83	44	33	59	49	38	30	40	37	33	29	33	32	30	27
	133	500	139	48	86	163	86	44	50	59	50	38	37	41	37	33	32	33	32	30
	47	139	500	137	44	86	153	83	38	49	59	49	33	37	41	37	30	32	33	32
	35	48	137	500	33	44	83	164	30	38	49	59	29	33	37	41	27	30	32	33
	154	86	44	33	500	134	48	34	165	84	44	33	59	50	38	30	41	37	33	30
5	83	163	86	44	134	500	139	48	86	163	85	44	49	59	50	38	37	40	37	33
	44	86	153	83	48	139	500	137	44	86	172	86	38	50	59	49	33	37	40	37
	33	44	83	164	34	48	137	500	33	44	86	166	30	39	50	59	29	33	37	41
	59	50	38	30	165	86	44	33	500	143	48	35	164	85	44	33	59	49	38	31
	49	59	49	38	84	163	86	44	143	500	143	48	84	159	87	44	49	59	49	38
10	38	50	59	49	44	85	172	86	48	143	500	134	44	86	156	81	38	49	58	49
	30	38	49	59	33	44	86	166	35	48	134	500	33	45	86	157	30	39	49	59
	40	37	33	29	59	49	38	30	164	84	44	33	500	138	48	35	162	86	44	33
	37	41	37	33	50	59	50	39	85	159	86	45	138	500	142	48	85	165	85	45
	33	37	41	37	38	50	59	50	44	87	156	86	48	142	500	133	44	84	159	85
15	29	33	37	41	30	38	49	59	33	44	81	157	35	48	133	500	33	44	81	157
	33	32	30	27	41	37	33	29	59	49	38	30	162	85	44	33	500	142	47	34
	32	33	32	30	37	40	37	33	49	59	49	39	86	165	84	44	142	500	136	48
	30	32	33	32	33	37	40	37	38	49	58	49	44	85	159	81	47	136	500	130
	27	30	32	33	30	33	37	41	31	38	49	59	33	45	85	157	34	48	130	500

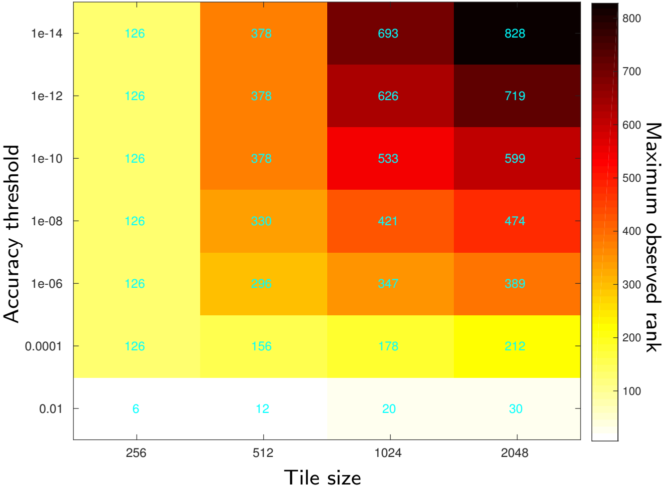
Maximum Ranks after Compression into Tile Low Rank (TLR) Format

- Seismic imaging: 3D Helmholtz with $N = 128$ and $k = N \times 0.625$ (10 grid points / wavelength) on $\Omega = [0, 1]^3$
- Heat map



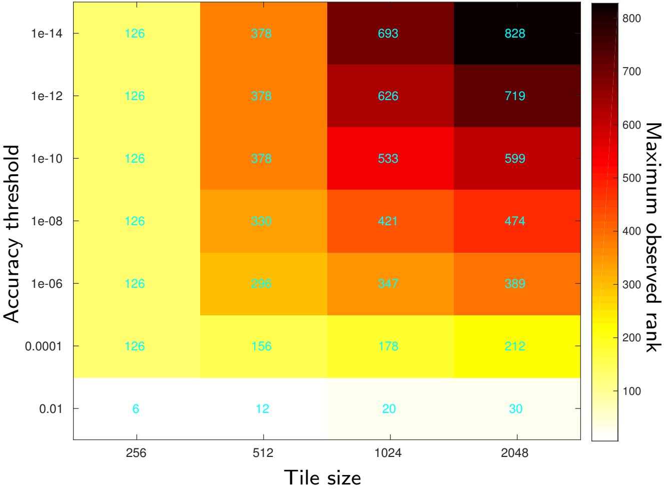
Maximum Ranks after Compression into Tile Low Rank (TLR) Format

- Seismic imaging: 3D Helmholtz with $N = 128$ and $k = N \times 0.625$ (10 grid points / wavelength) on $\Omega = [0, 1]^3$
- Heat map
- x axis: different tile sizes



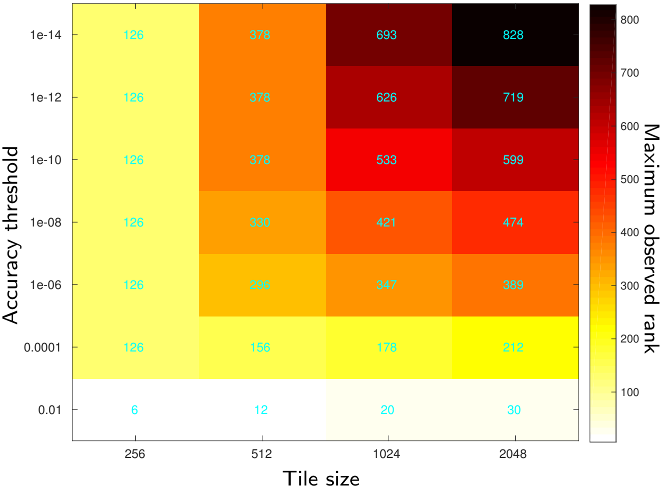
Maximum Ranks after Compression into Tile Low Rank (TLR) Format

- Seismic imaging: 3D Helmholtz with $N = 128$ and $k = N \times 0.625$ (10 grid points / wavelength) on $\Omega = [0, 1]^3$
- Heat map
- x axis: different tile sizes
- y axis: different accuracy thresholds



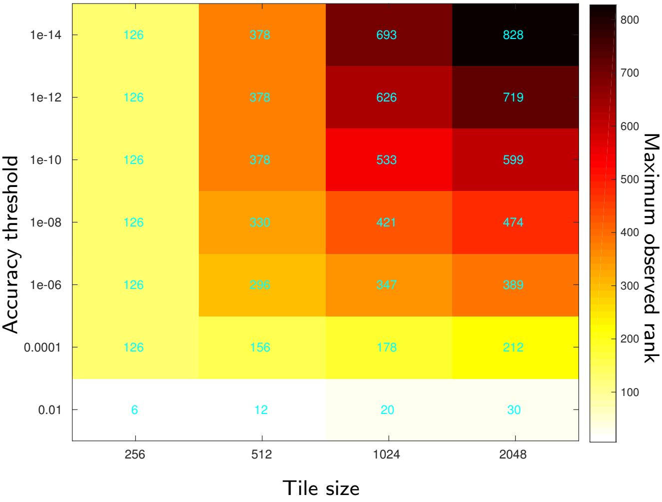
Maximum Ranks after Compression into Tile Low Rank (TLR) Format

- Seismic imaging: 3D Helmholtz with $N = 128$ and $k = N \times 0.625$ (10 grid points / wavelength) on $\Omega = [0, 1]^3$
- Heat map
- x axis: different tile sizes
- y axis: different accuracy thresholds
- value/color: maximum rank in TLR matrix for a specific tile size and accuracy



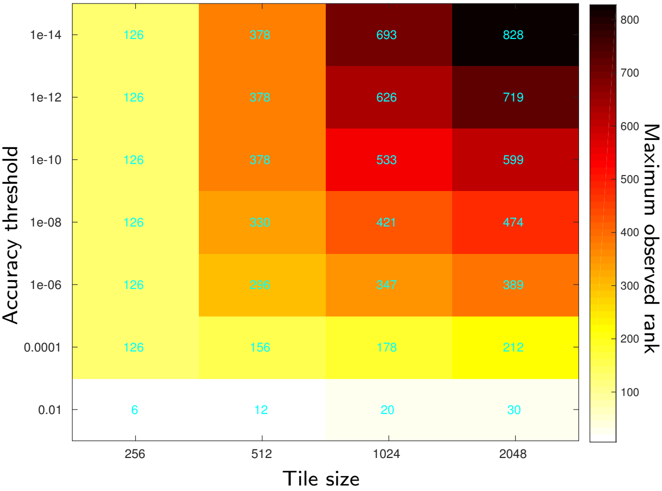
Maximum Ranks after Compression into Tile Low Rank (TLR) Format

- Seismic imaging: 3D Helmholtz with $N = 128$ and $k = N \times 0.625$ (10 grid points / wavelength) on $\Omega = [0, 1]^3$
- Heat map
- x axis: different tile sizes
- y axis: different accuracy thresholds
- value/color: maximum rank in TLR matrix for a specific tile size and accuracy
- lighter color: smaller maxrank



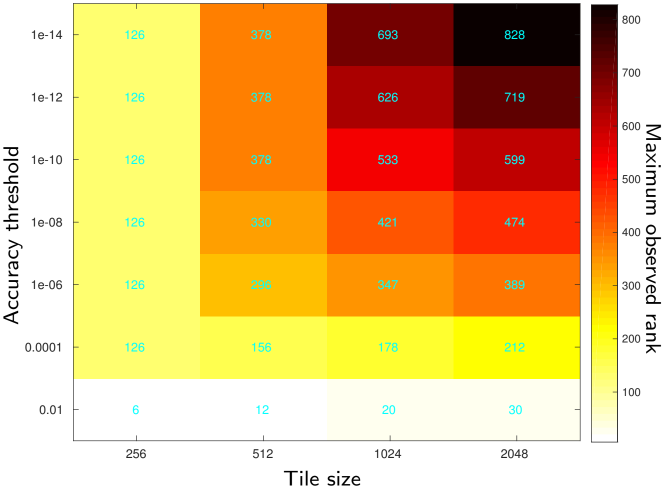
Maximum Ranks after Compression into Tile Low Rank (TLR) Format

- Seismic imaging: 3D Helmholtz with $N = 128$ and $k = N \times 0.625$ (10 grid points / wavelength) on $\Omega = [0, 1]^3$
- Heat map
- Even for accuracy of 1e-14, ranks are smaller than half of tile size.



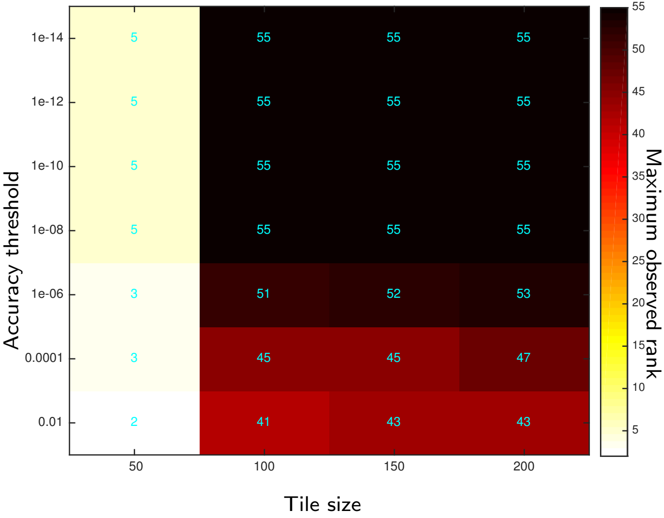
Maximum Ranks after Compression into Tile Low Rank (TLR) Format

- Seismic imaging: 3D Helmholtz with $N = 128$ and $k = N \times 0.625$ (10 grid points / wavelength) on $\Omega = [0, 1]^3$
- Heat map
- Even for accuracy of 1e-14, ranks are smaller than half of tile size.
- Even for tiles of size 2048, rank is smaller than 2048/2=1024.



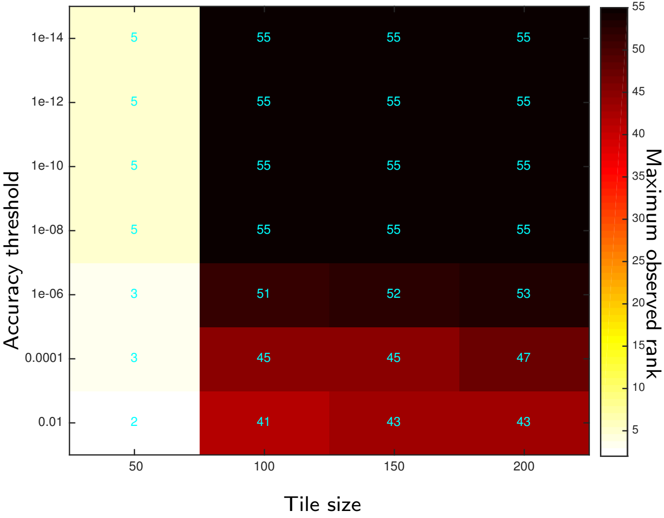
Maximum Ranks after Compression into Tile Low Rank (TLR) Format

- Electronic structure: Hamiltonian/Overlap matrix
- Heat map



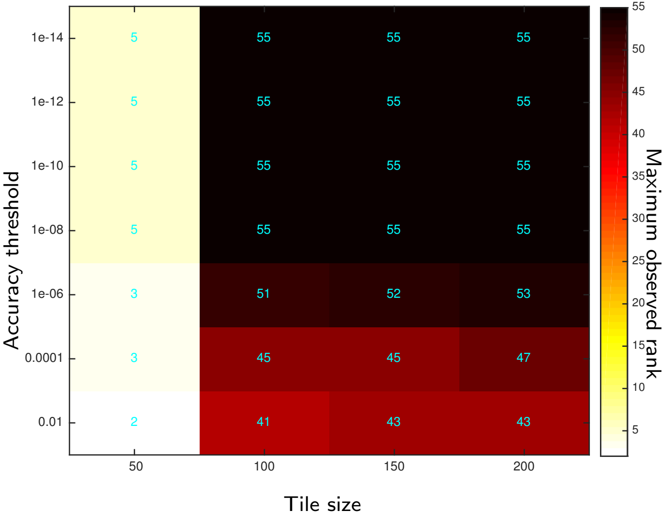
Maximum Ranks after Compression into Tile Low Rank (TLR) Format

- Electronic structure: Hamiltonian/Overlap matrix
- Heat map
- x axis: different tile sizes



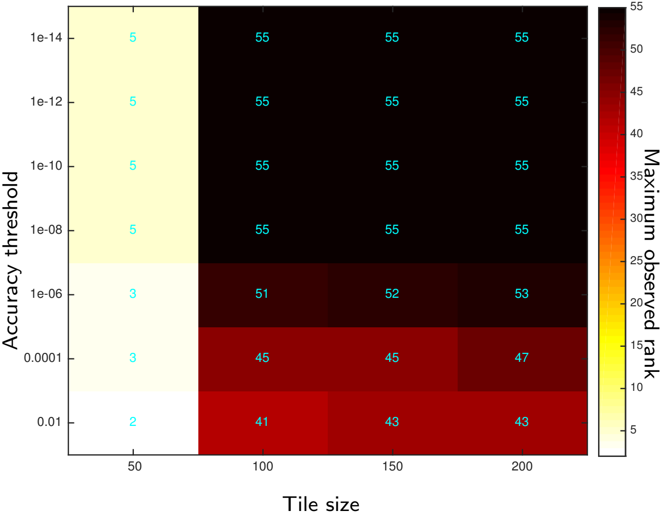
Maximum Ranks after Compression into Tile Low Rank (TLR) Format

- Electronic structure: Hamiltonian/Overlap matrix
- Heat map
- x axis: different tile sizes
- y axis: different accuracy thresholds



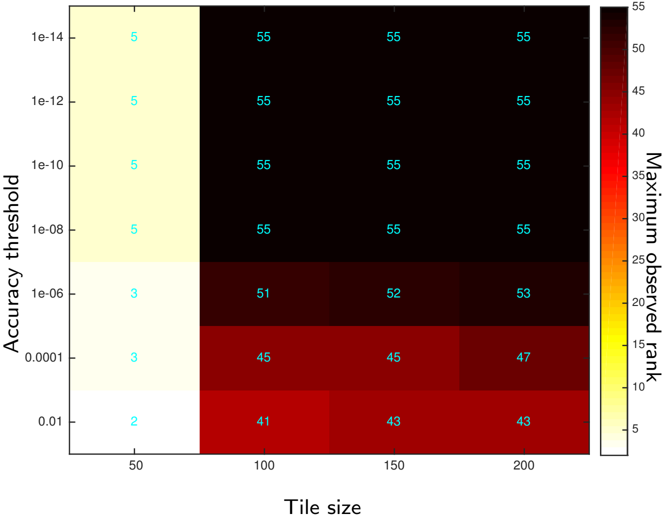
Maximum Ranks after Compression into Tile Low Rank (TLR) Format

- Electronic structure: Hamiltonian/Overlap matrix
- Heat map
- x axis: different tile sizes
- y axis: different accuracy thresholds
- value/color: maximum rank in TLR matrix for a specific tile size and accuracy



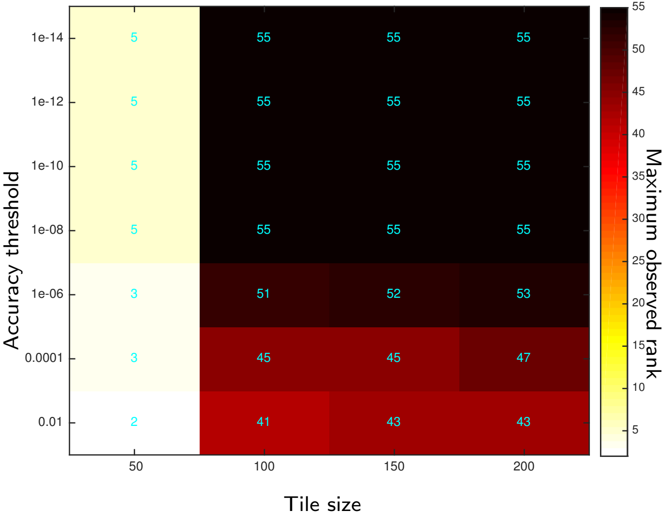
Maximum Ranks after Compression into Tile Low Rank (TLR) Format

- Electronic structure: Hamiltonian/Overlap matrix
- Heat map
- x axis: different tile sizes
- y axis: different accuracy thresholds
- value/color: maximum rank in TLR matrix for a specific tile size and accuracy
- lighter color: smaller maxrank



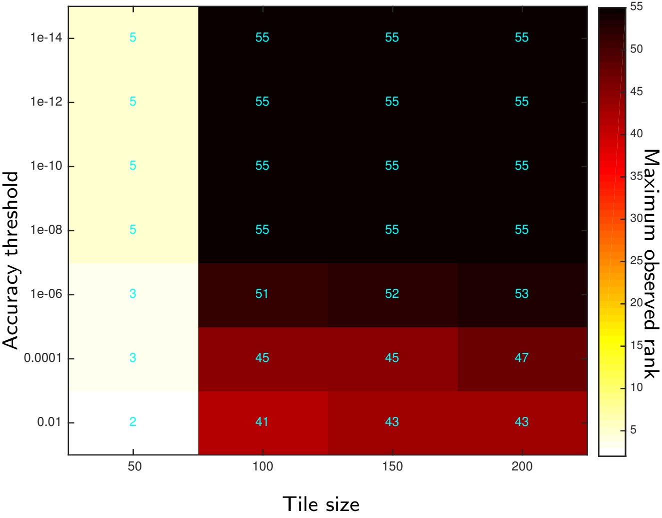
Maximum Ranks after Compression into Tile Low Rank (TLR) Format

- Electronic structure: Hamiltonian/Overlap matrix
- Heat map
- Even for accuracy of 1e-14, maximum rank is 55 which is considerably smaller than tile size.

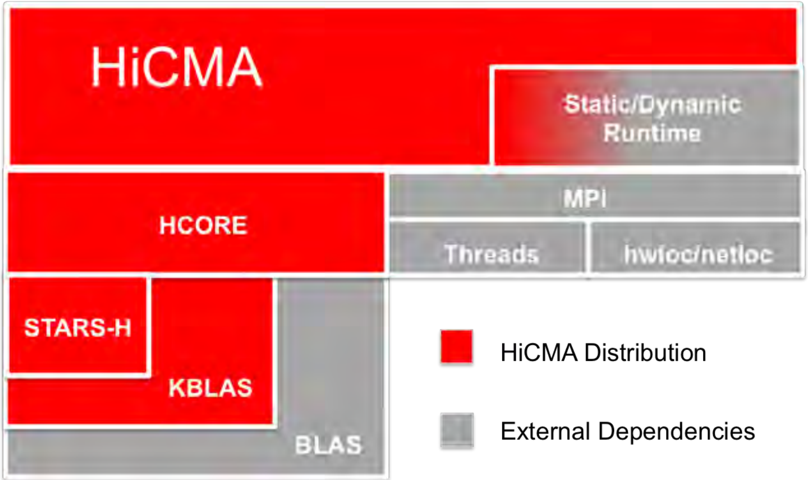


Maximum Ranks after Compression into Tile Low Rank (TLR) Format

- Electronic structure: Hamiltonian/Overlap matrix
- Heat map
- Even for accuracy of 1e-14, maximum rank is 55 which is considerably smaller than tile size.
- Even for larger tiles, maximum rank is 55.

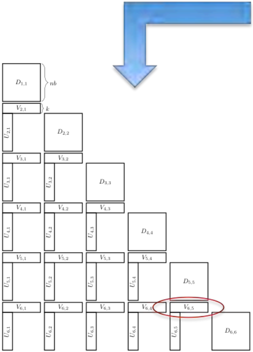


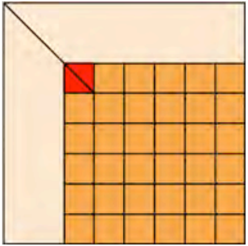
The HiCMA Library: Hierarchical Computations on Manycore Architectures



Available at http://github.com/ecrc/hicma

Dense Linear Algebra Renaissance

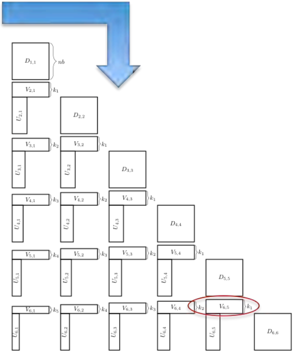




dense tiles
Cholesky: O(n³)

↓

tile low rank
Cholesky: O(kn²)



Fixed ranks
Preconditioners
Performance oriented

Fixed accuracy
Variable ranks
Dense/Sparse Direct Solvers

HiCMA dpotrf

The tile low-rank (TLR) Cholesky algorithm can be expressed with the following four computational kernels:

hcore_dpotrf: The kernel performs the Cholesky factorization of a diagonal (lower triangular) tile. It is similar to DPOTRF since the diagonal tiles are dense.

HiCMA dpotrf

The tile low-rank (TLR) Cholesky algorithm can be expressed with the following four computational kernels:

`hcore_dpotrf`: The kernel performs the Cholesky factorization of a diagonal (lower triangular) tile. It is similar to DPOTRF since the diagonal tiles are dense.

`hcore_dtrsm`: The operation applies an update to an off-diagonal low-rank tile of the input matrix, resulting from factorization of the diagonal tile above it and overrides it with the final elements of the output matrix: $V_{(i,k)} = V_{(i,k)} \times D^{-1}_{(k,k)}$. The operation is a triangular solve.

HiCMA dpotrf

The tile low-rank (TLR) Cholesky algorithm can be expressed with the following four computational kernels:

- `hcore_dpotrf`: The kernel performs the Cholesky factorization of a diagonal (lower triangular) tile. It is similar to DPOTRF since the diagonal tiles are dense.
- `hcore_dtrsm`: The operation applies an update to an off-diagonal low-rank tile of the input matrix, resulting from factorization of the diagonal tile above it and overrides it with the final elements of the output matrix: $V_{(i,k)} = V_{(i,k)} \times D^{-1}_{(k,k)}$. The operation is a triangular solve.
- `hcore_dsyrk`: The kernel applies updates to a diagonal (lower triangular) tile of the input matrix, resulting from factorization of the low-rank tiles to the left of it: $D_{(j,j)} = D_{(j,j)} - (U_{(j,k)} \times V^T_{(j,k)}) \times (U_{(j,k)} \times V^T_{(j,k)})^T$. The operation is a symmetric rank-k update.

HiCMA dpotrf

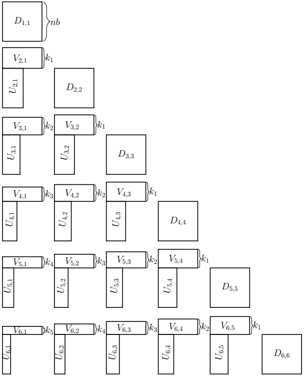
The tile low-rank (TLR) Cholesky algorithm can be expressed with the following four computational kernels:

- **hcore_dpotrf**: The kernel performs the Cholesky factorization of a diagonal (lower triangular) tile. It is similar to DPOTRF since the diagonal tiles are dense.
- **hcore_dtrsm**: The operation applies an update to an off-diagonal low-rank tile of the input matrix, resulting from factorization of the diagonal tile above it and overrides it with the final elements of the output matrix: $V_{(i,k)} = V_{(i,k)} \times D^{-1}_{(k,k)}$. The operation is a triangular solve.
- **hcore_dsyrk**: The kernel applies updates to a diagonal (lower triangular) tile of the input matrix, resulting from factorization of the low-rank tiles to the left of it: $D_{(j,j)} = D_{(j,j)} - (U_{(j,k)} \times V^T_{(j,k)}) \times (U_{(j,k)} \times V^T_{(j,k)})^T$. The operation is a symmetric rank-k update.
- **hcore_dgemm**: The operation applies updates to an off-diagonal low-rank tile of the input matrix, resulting from factorization of the low-rank tiles to the left of it. The operation involves two QR factorizations, one reduced SVD (depending on the rank and/or the accuracy parameter) and two matrix-matrix multiplications.

HiCMA dpotrf

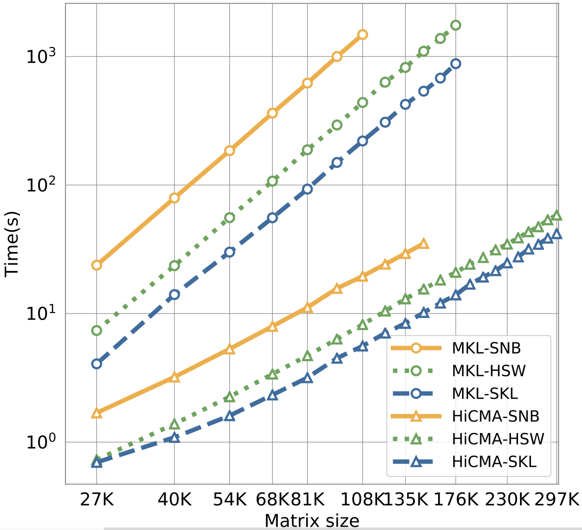
Algorithm 1 `hicma_dpotrf` (HicmaLower, D, U, V, N, nb, rank, acc)

```
p = N / nb
for k = 1 to p do
    #task access(read&write:D(k))
    hcore_dpotrf (HicmaLower, D(k))
    for i = k+1 to p do
        #task access(read:D(k), read&write:V(i,k))
        hcore_dtrsm (V(i,k), D(k,k))
    end for
    for j = k+1 to p do
        # task access(read:U(j,k), read:V(j,k), read&write:D(j))
        hcore_dsyrk (D(j), U(j,k), V(j,k))
        for i = j+1 to p do
            #task access(read:U(i,k),
                read:V(i,k)), read:U(j,k), read:V(j,k),
                read&write:U(i,j), read&write:V(i,j))
            hcore_dgemm (U(i,k), V(i,k), U(j,k), V(j,k), U(i,j), V(i,j), rank, acc)
        end for
    end for
end for
```



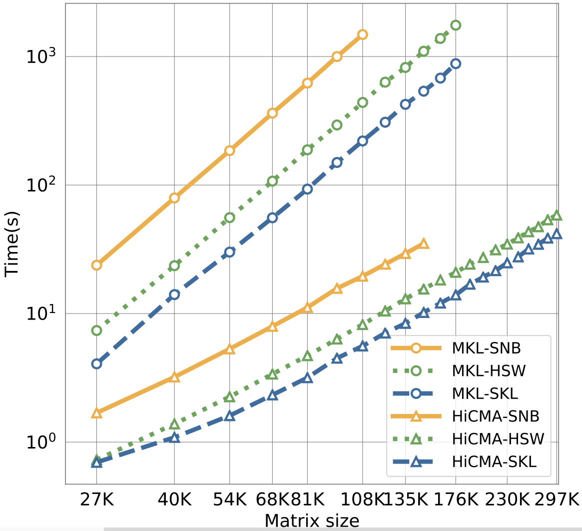
Dense vs TLR Cholesky on a Single Node

- Execution times for MKL dpotrf (dense) vs HiCMA dpotrf (TLR).
  Smaller the better.



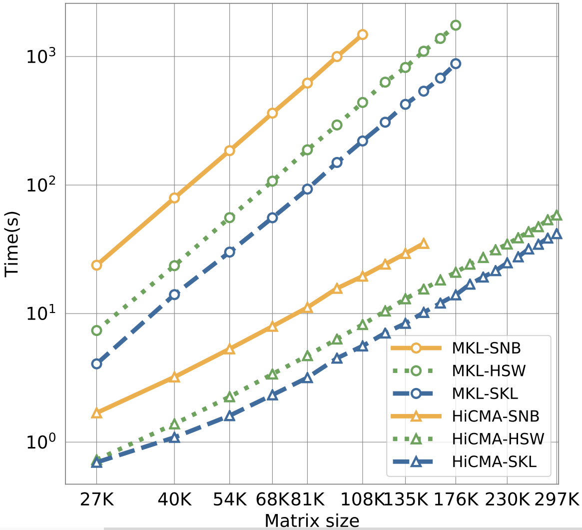
Dense vs TLR Cholesky on a Single Node

- Execution times for MKL dpotrf (dense) vs HiCMA dpotrf (TLR). Smaller the better.
- Application: Geospatial statistic, square exp. kernel



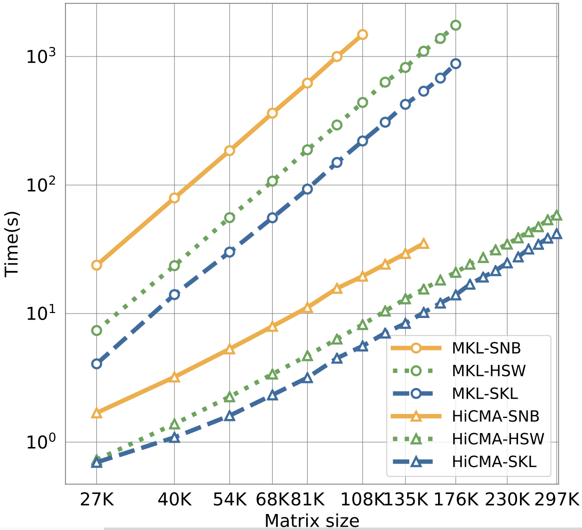
Dense vs TLR Cholesky on a Single Node

- Execution times for MKL dpotrf (dense) vs HiCMA dpotrf (TLR).
  Smaller the better.
- Application: Geospatial statistic, square exp. kernel
- Accuracy threshold: 10^{-9}. This accuracy level is enough for a large amount of cases encountered in the geospatial statistics application.



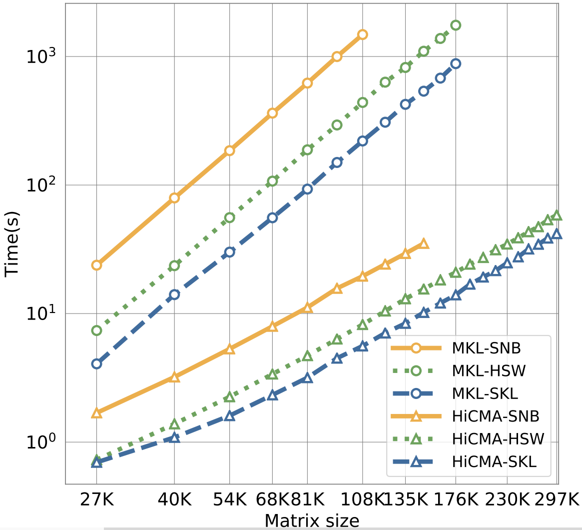
Dense vs TLR Cholesky on a Single Node

- Execution times for MKL dpotrf (dense) vs HiCMA dpotrf (TLR).
  Smaller the better.
- Application: Geospatial statistic, square exp. kernel
- Accuracy threshold: 10^{-9}. This accuracy level is enough for a large amount of cases encountered in the geospatial statistics application.
- StarPU (task-based runtime)



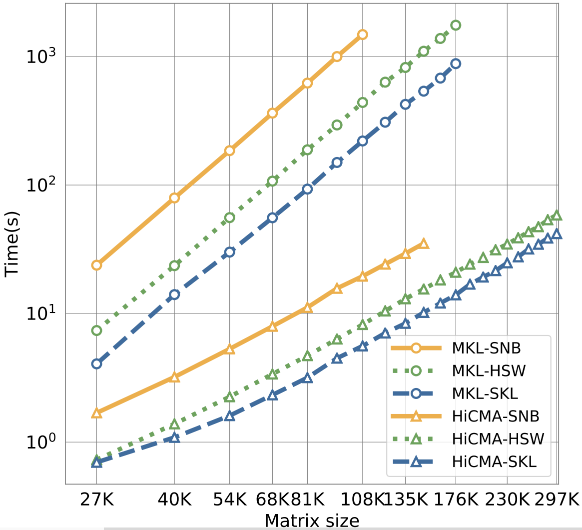
Dense vs TLR Cholesky on a Single Node

- Execution times for MKL dpotrf (dense) vs HiCMA dpotrf (TLR). Smaller the better.
- Application: Geospatial statistic, square exp. kernel
- Accuracy threshold: 10^{-9}. This accuracy level is enough for a large amount of cases encountered in the geospatial statistics application.
- StarPU (task-based runtime)
- Three architectures:
    - Sandy Bridge (SNB): AVX, 2.0GHz 2 sockets x 8 cores
    - Haswell (HSW): AVX2, 2.4GHz 2 sockets x 18 cores
    - Skylake (SKL): AVX512, 2.1GHz 2 sockets x 28 cores



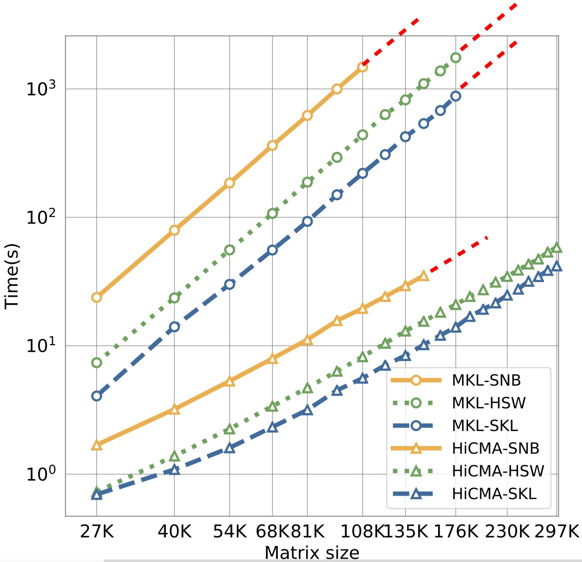
Dense vs TLR Cholesky on a Single Node

- Execution times for MKL dpotrf (dense) vs HiCMA dpotrf (TLR).
  Smaller the better.
- *x* axis (*log*-scale): different matrix sizes



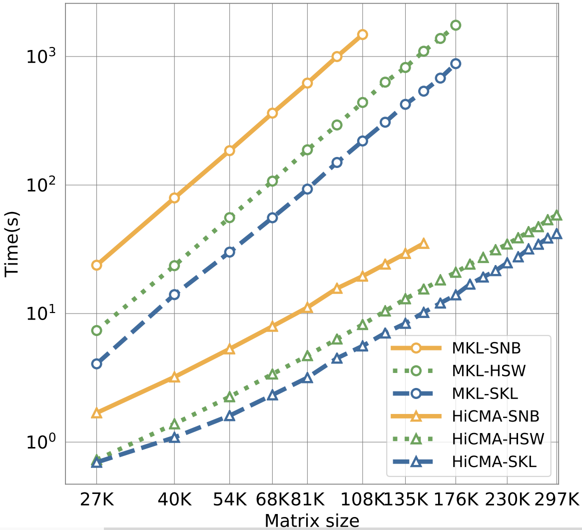
Dense vs TLR Cholesky on a Single Node

- Execution times for MKL dpotrf (dense) vs HiCMA dpotrf (TLR).
  Smaller the better.
- *x* axis (*log*-scale): different matrix sizes
  - Missing points due to not enough memory for larger matrices



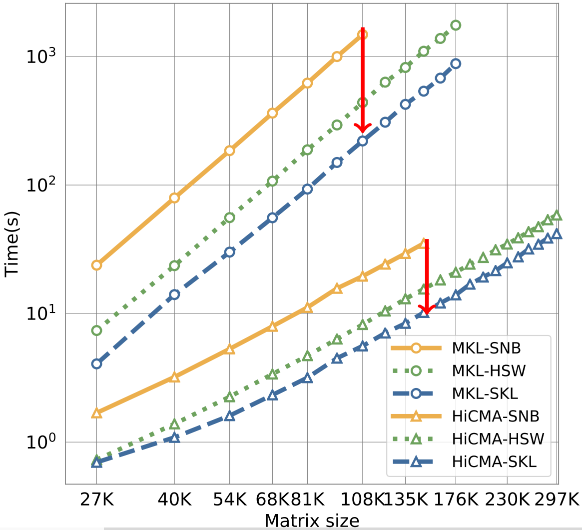
Dense vs TLR Cholesky on a Single Node

- Execution times for MKL dpotrf (dense) vs HiCMA dpotrf (TLR).
  Smaller the better.
- *x* axis (*log*-scale): different matrix sizes
  - Missing points due to not enough memory for larger matrices
- *y* axis (*log*-scale): time of `dpotrf` in seconds



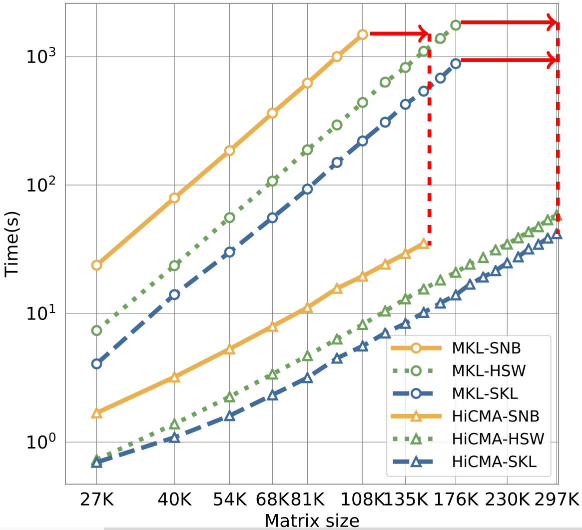
Dense vs TLR Cholesky on a Single Node

- Execution times for MKL dpotrf (dense) vs HiCMA dpotrf (TLR).
  Smaller the better.
- *x* axis (*log*-scale): different matrix sizes
  - Missing points due to not enough memory for larger matrices
- *y* axis (*log*-scale): time of `dpotrf` in seconds
- Smaller time as chips get better



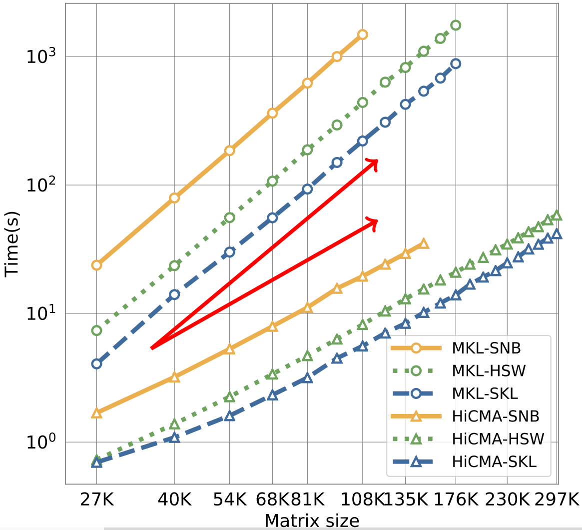
Dense vs TLR Cholesky on a Single Node

- Execution times for MKL dpotrf (dense) vs HiCMA dpotrf (TLR).
  Smaller the better.
- x axis (*log*-scale): different matrix sizes
  - Missing points due to not enough memory for larger matrices
- y axis (*log*-scale): time of `dpotrf` in seconds
- Smaller time as chips get better
- Solve larger problems



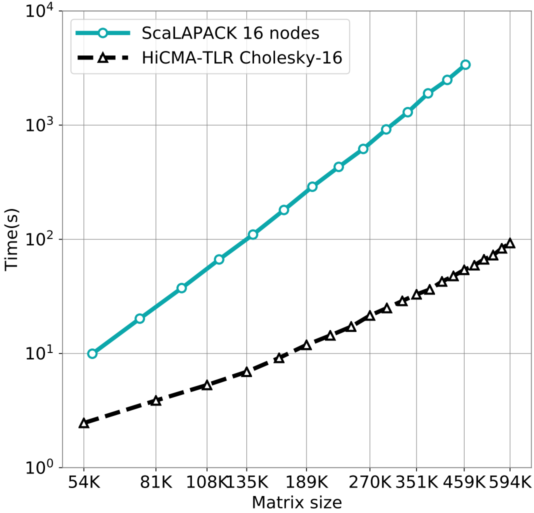
Dense vs TLR Cholesky on a Single Node

- Execution times for MKL dpotrf (dense) vs HiCMA dpotrf (TLR).
  Smaller the better.
- x axis (*log*-scale): different matrix sizes
  - Missing points due to not enough memory for larger matrices
- y axis (*log*-scale): time of `dpotrf` in seconds
- Smaller time as chips get better
- Solve larger problems
- Smaller slope: $O(n^3) \rightarrow O(kn^2)$



ScaLAPACK vs TLR Cholesky (*Shaheen-2*, HSW, acc$=10^{-9}$)

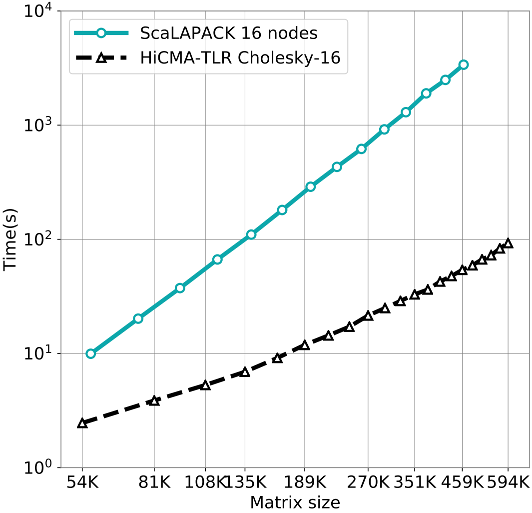
- Execution times for ScaLAPACK dpotrf (dense) vs HiCMA dpotrf (TLR). Smaller the better.



K. Akbudak, H. Ltaief, A. Mikhalev, A. Charara, and D. E. Keyes,
Exploiting Data Sparsity for Large-Scale Matrix Computations, Submitted to EuroPar, 2018.

ScaLAPACK vs TLR Cholesky (*Shaheen-2*, HSW, acc=10^{-9})

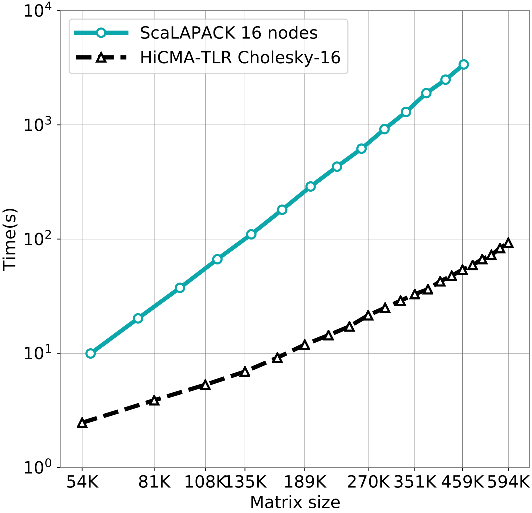
- Execution times for ScaLAPACK dpotrf (dense) vs HiCMA dpotrf (TLR). Smaller the better.
- Application: Geospatial statistic, square exp. kernel



K. Akbudak, H. Ltaief, A. Mikhalev, A. Charara, and D. E. Keyes,
Exploiting Data Sparsity for Large-Scale Matrix Computations, Submitted to EuroPar, 2018.

ScaLAPACK vs TLR Cholesky (*Shaheen-2*, HSW, acc=10^{-9})

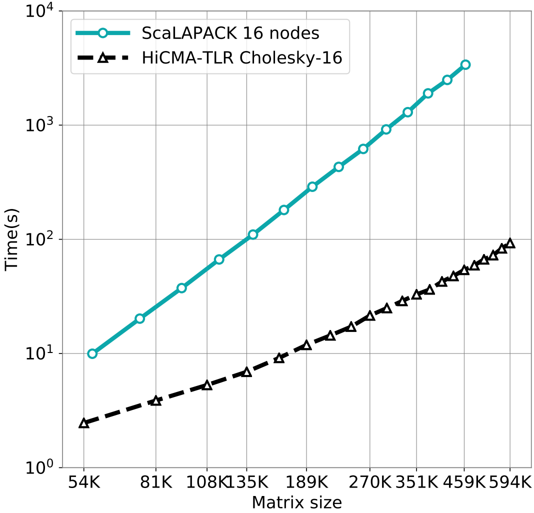
- Execution times for ScaLAPACK dpotrf (dense) vs HiCMA dpotrf (TLR). Smaller the better.
- Application: Geospatial statistic, square exp. kernel
- Accuracy threshold: 10^{-9}.



K. Akbudak, H. Ltaief, A. Mikhalev, A. Charara, and D. E. Keyes,
Exploiting Data Sparsity for Large-Scale Matrix Computations, Submitted to EuroPar, 2018.

ScaLAPACK vs TLR Cholesky (*Shaheen-2*, HSW, acc=10^{-9})

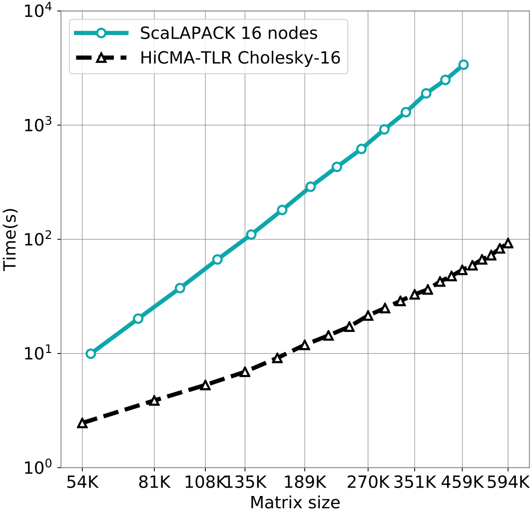
- Execution times for ScaLAPACK dpotrf (dense) vs HiCMA dpotrf (TLR). Smaller the better.
- Application: Geospatial statistic, square exp. kernel
- Accuracy threshold: 10^{-9}.
- StarPU (task-based runtime)



K. Akbudak, H. Ltaief, A. Mikhalev, A. Charara, and D. E. Keyes,
Exploiting Data Sparsity for Large-Scale Matrix Computations, Submitted to EuroPar, 2018.

ScaLAPACK vs TLR Cholesky (*Shaheen-2*, HSW, acc=10^{-9})

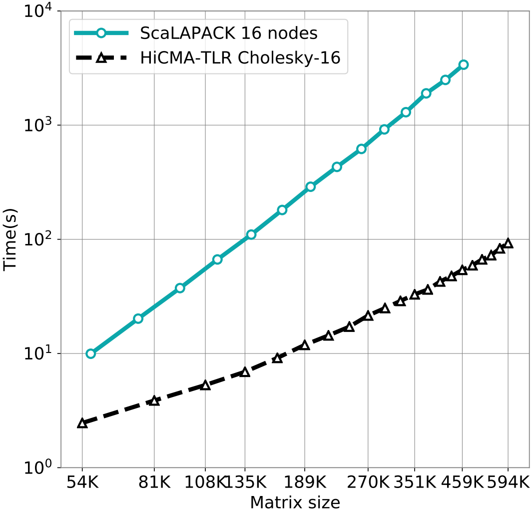
- Execution times for ScaLAPACK dpotrf (dense) vs HiCMA dpotrf (TLR). Smaller the better.
- Application: Geospatial statistic, square exp. kernel
- Accuracy threshold: 10^{-9}.
- StarPU (task-based runtime)
- x axis (*log*-scale): different matrix sizes



K. Akbudak, H. Ltaief, A. Mikhalev, A. Charara, and D. E. Keyes,
Exploiting Data Sparsity for Large-Scale Matrix Computations, Submitted to EuroPar, 2018.

ScaLAPACK vs TLR Cholesky (*Shaheen-2*, HSW, acc=10^{-9})

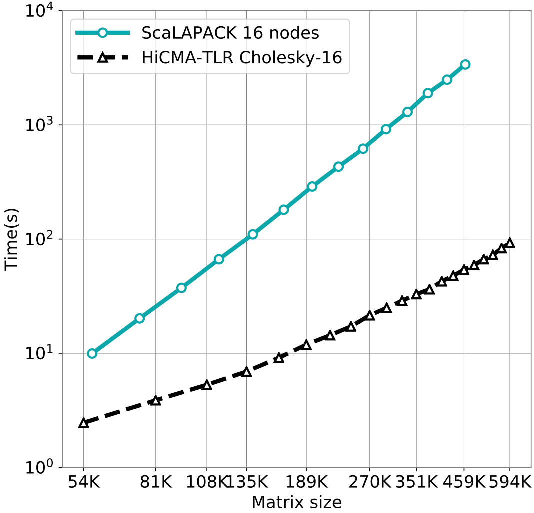
- Execution times for ScaLAPACK dpotrf (dense) vs HiCMA dpotrf (TLR). Smaller the better.
- Application: Geospatial statistic, square exp. kernel
- Accuracy threshold: 10^{-9}.
- StarPU (task-based runtime)
- x axis (*log*-scale): different matrix sizes
- y axis (*log*-scale): time of `dpotrf` in seconds



K. Akbudak, H. Ltaief, A. Mikhalev, A. Charara, and D. E. Keyes,
Exploiting Data Sparsity for Large-Scale Matrix Computations, Submitted to EuroPar, 2018.

ScaLAPACK vs TLR Cholesky (*Shaheen-2*, HSW, acc=10^{-9})

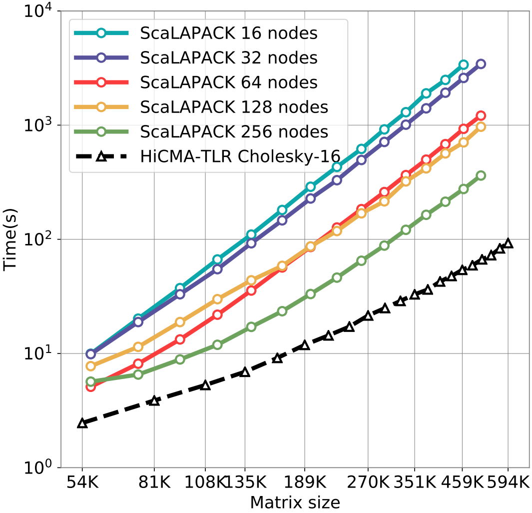
- Execution times for ScaLAPACK dpotrf (dense) vs HiCMA dpotrf (TLR). Smaller the better.
- Application: Geospatial statistic, square exp. kernel
- Accuracy threshold: 10^{-9}.
- StarPU (task-based runtime)
- x axis (*log*-scale): different matrix sizes
- y axis (*log*-scale): time of `dpotrf` in seconds
- 85x speedup



K. Akbudak, H. Ltaief, A. Mikhalev, A. Charara, and D. E. Keyes,
Exploiting Data Sparsity for Large-Scale Matrix Computations, Submitted to EuroPar, 2018.

ScaLAPACK vs TLR Cholesky (*Shaheen-2*, HSW, acc=10^{-9})

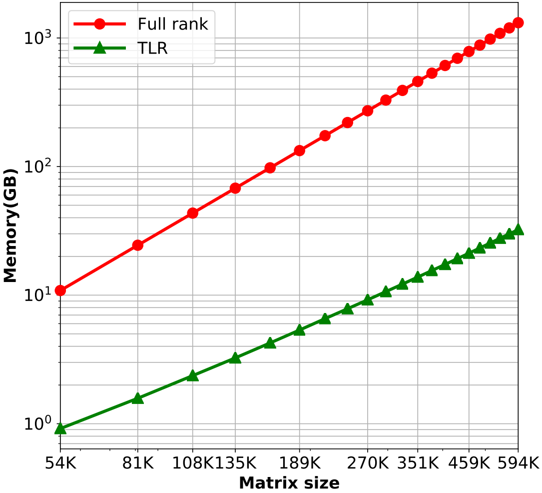
- Execution times for ScaLAPACK dpotrf (dense) vs HiCMA dpotrf (TLR). Smaller the better.
- Application: Geospatial statistic, square exp. kernel
- Accuracy threshold: 10^{-9}.
- StarPU (task-based runtime)
- x axis (*log*-scale): different matrix sizes
- y axis (*log*-scale): time of `dpotrf` in seconds



K. Akbudak, H. Ltaief, A. Mikhalev, A. Charara, and D. E. Keyes,
Exploiting Data Sparsity for Large-Scale Matrix Computations, Submitted to EuroPar, 2018.

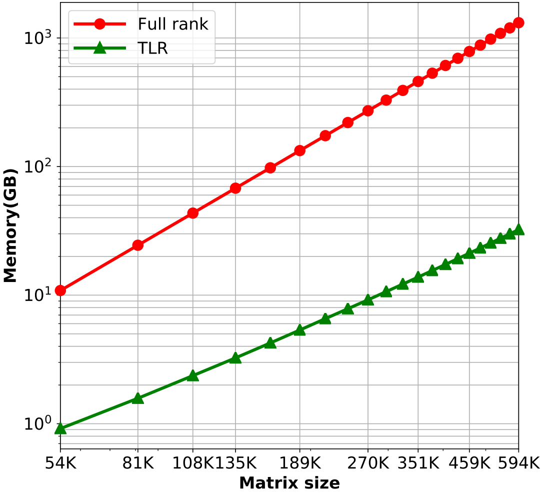
TLR Cholesky: Memory Footprint (acc$=10^{-9}$)

- Memory footprint for dense vs TLR matrices.
  Smaller the better.



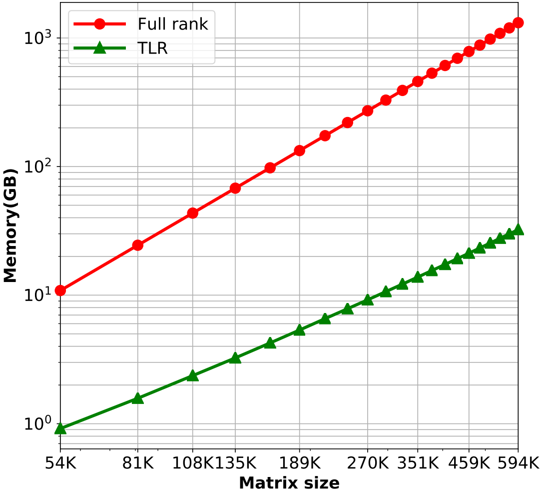
TLR Cholesky: Memory Footprint (acc$=10^{-9}$)

- Memory footprint for dense vs TLR matrices.
  Smaller the better.
- Application: Geospatial statistic, square exp. kernel



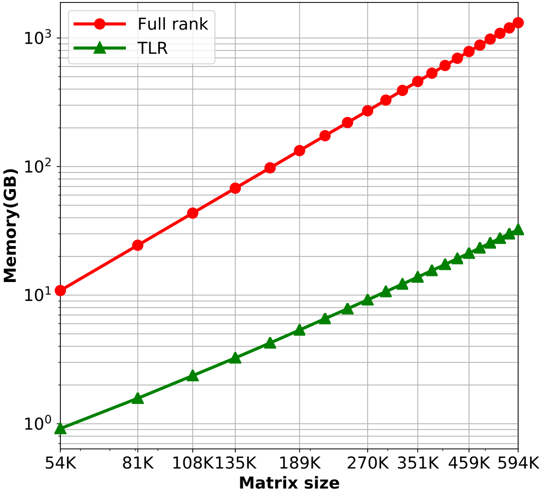
TLR Cholesky: Memory Footprint (acc=10^{-9})

- Memory footprint for dense vs TLR matrices.
  Smaller the better.
- Application: Geospatial statistic, square exp. kernel
- Accuracy threshold: 10^{-9}



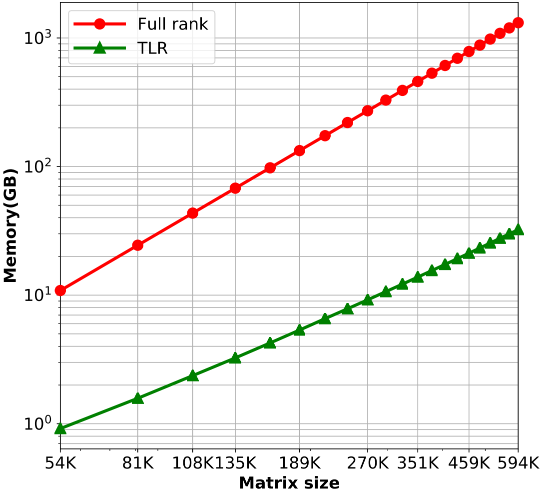
TLR Cholesky: Memory Footprint (acc=10^{-9})

- Memory footprint for dense vs TLR matrices.
  Smaller the better.
- Application: Geospatial statistic, square exp. kernel
- Accuracy threshold: 10^{-9}
- x axis (*log*-scale): different matrix sizes (N)



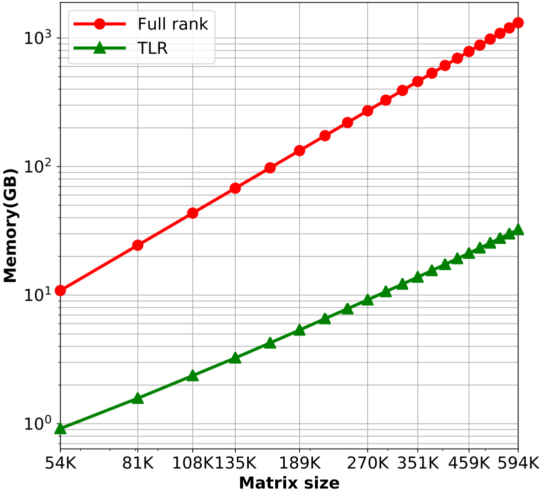
TLR Cholesky: Memory Footprint (acc=10^{-9})

- Memory footprint for dense vs TLR matrices. Smaller the better.
- Application: Geospatial statistic, square exp. kernel
- Accuracy threshold: 10^{-9}
- x axis (*log*-scale): different matrix sizes (N)
- y axis (*log*-scale): memory required for storing matrix in terms of giga bytes



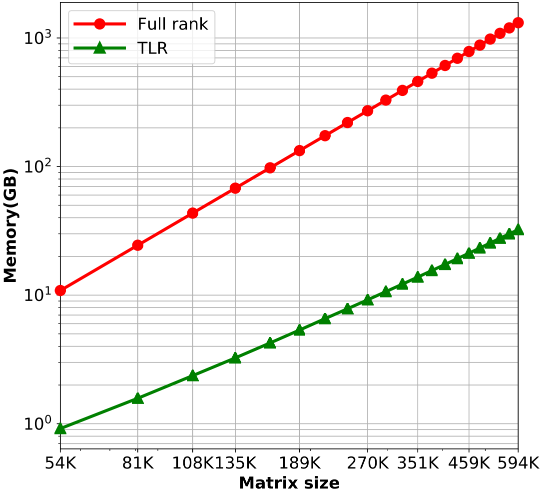
TLR Cholesky: Memory Footprint (acc=10^{-9})

- Memory footprint for dense vs TLR matrices.
  Smaller the better.
- Application: Geospatial statistic, square exp. kernel
- Accuracy threshold: 10^{-9}
- x axis (*log*-scale): different matrix sizes (N)
- y axis (*log*-scale): memory required for storing matrix in terms of giga bytes
- Full rank: N^2



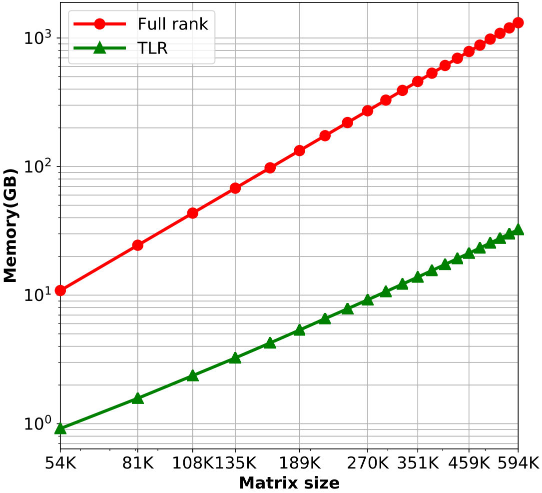
TLR Cholesky: Memory Footprint (acc=10^{-9})

- Memory footprint for dense vs TLR matrices.
  Smaller the better.
- Application: Geospatial statistic, square exp. kernel
- Accuracy threshold: 10^{-9}
- x axis (*log*-scale): different matrix sizes (N)
- y axis (*log*-scale): memory required for storing matrix in terms of giga bytes
- Full rank: N^2
- TLR: $\frac{N^2 k}{nb}$



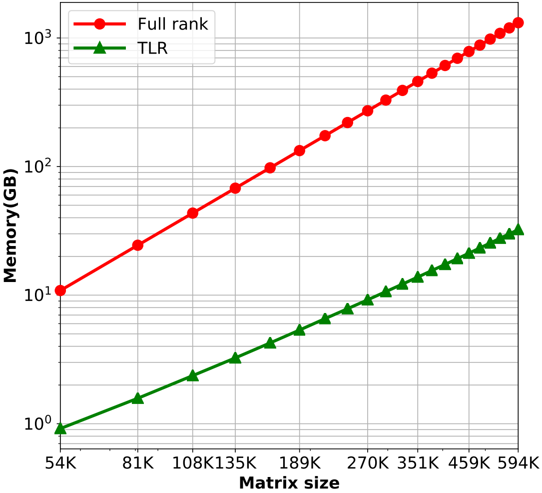
TLR Cholesky: Memory Footprint (acc=10^{-9})

- Memory footprint for dense vs TLR matrices. Smaller the better.
- Application: Geospatial statistic, square exp. kernel
- Accuracy threshold: 10^{-9}
- x axis (*log*-scale): different matrix sizes (N)
- y axis (*log*-scale): memory required for storing matrix in terms of giga bytes
- Full rank: N^2
- TLR: $\frac{N^2 k}{nb}$
    - k: average rank



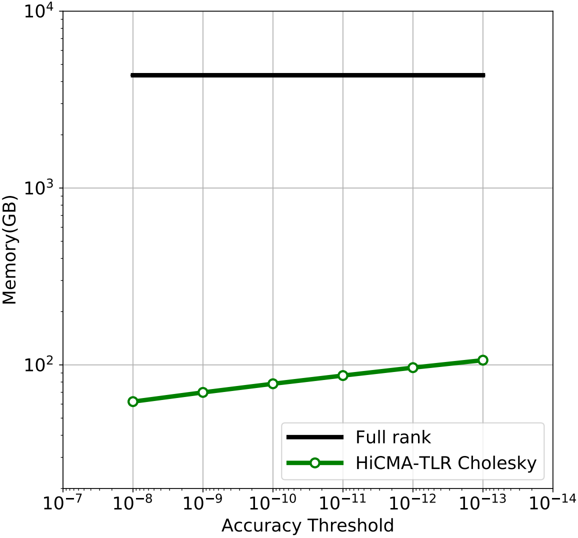
TLR Cholesky: Memory Footprint (acc$=10^{-9}$)

- Memory footprint for dense vs TLR matrices.
  Smaller the better.
- Application: Geospatial statistic, square exp. kernel
- Accuracy threshold: 10^{-9}
- x axis (*log*-scale): different matrix sizes (N)
- y axis (*log*-scale): memory required for storing matrix in terms of giga bytes
- Full rank: N^2
- TLR: $\frac{N^2 k}{nb}$
  - k: average rank
  - nb: tile size



Impact of Accuracy Thresholds, **1M** Matrix Size, nb=2700

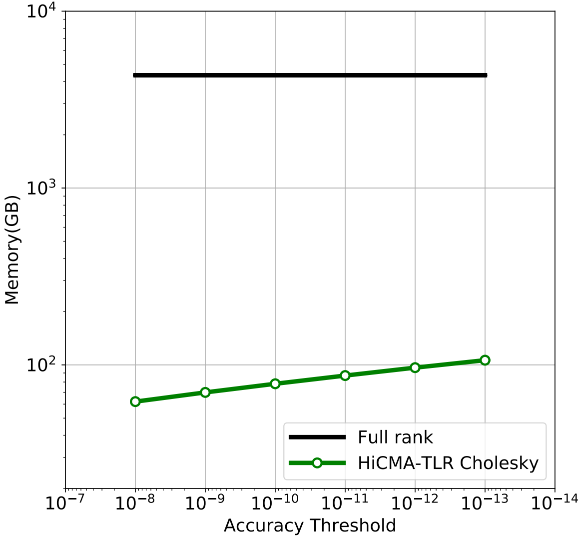
- Memory footprint for dense vs TLR matrices. Smaller the better.



K. Akbudak, H. Ltaief, A. Mikhalev, A. Charara, and D. E. Keyes, Exploiting Data Sparsity for Large-Scale Matrix Computations, Submitted to EuroPar, 2018.

Impact of Accuracy Thresholds, **1M** Matrix Size, nb=2700

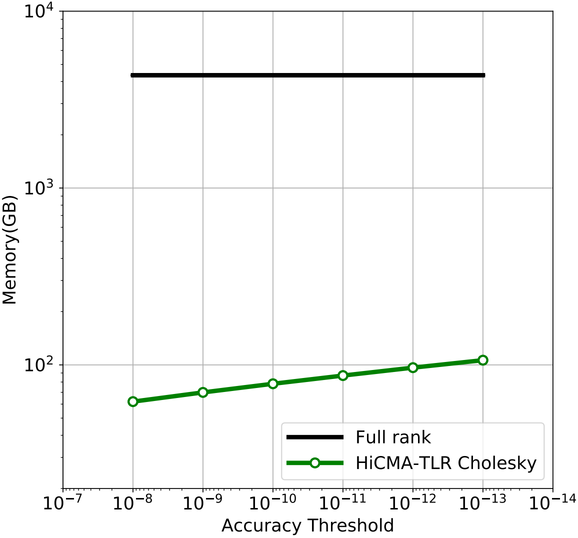
- Memory footprint for dense vs TLR matrices. Smaller the better.
- Application: Geospatial statistic, square exp. kernel



K. Akbudak, H. Ltaief, A. Mikhalev, A. Charara, and D. E. Keyes, Exploiting Data Sparsity for Large-Scale Matrix Computations, Submitted to EuroPar, 2018.

Impact of Accuracy Thresholds, **1M** Matrix Size, nb=2700

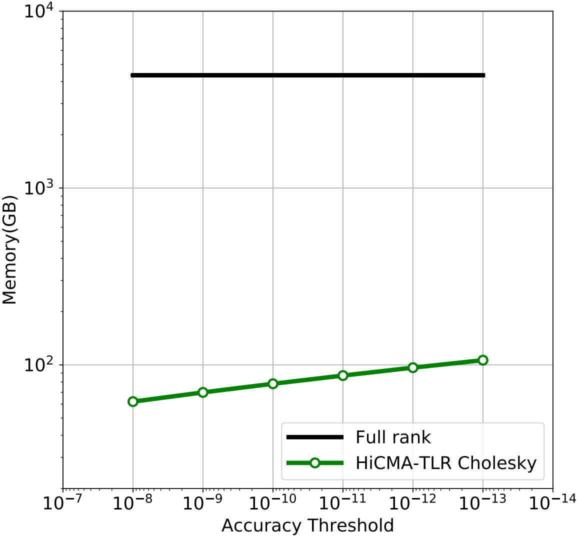
- Memory footprint for dense vs TLR matrices. Smaller the better.
- Application: Geospatial statistic, square exp. kernel
- *x* axis (*log*-scale): different accuracy thresholds



K. Akbudak, H. Ltaief, A. Mikhalev, A. Charara, and D. E. Keyes, Exploiting Data Sparsity for Large-Scale Matrix Computations, Submitted to EuroPar, 2018.

Impact of Accuracy Thresholds, **1M** Matrix Size, nb=2700

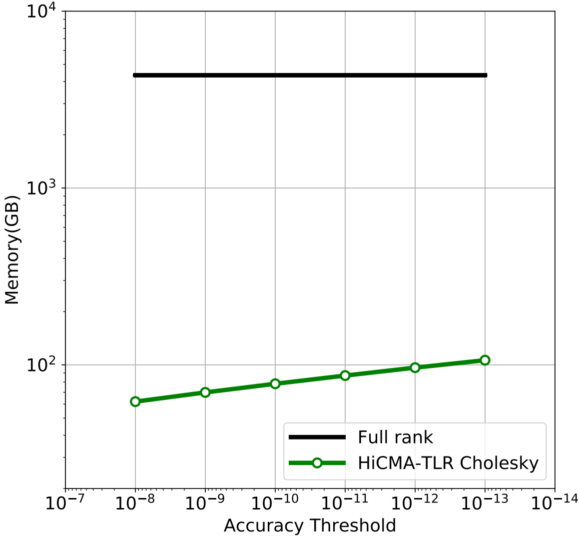
- Memory footprint for dense vs TLR matrices. Smaller the better.
- Application: Geospatial statistic, square exp. kernel
- *x* axis (*log*-scale): different accuracy thresholds
- *y* axis (*log*-scale): memory required for storing matrix in terms of giga bytes



K. Akbudak, H. Ltaief, A. Mikhalev, A. Charara, and D. E. Keyes, Exploiting Data Sparsity for Large-Scale Matrix Computations, Submitted to EuroPar, 2018.

Impact of Accuracy Thresholds, **1M** Matrix Size, nb=2700

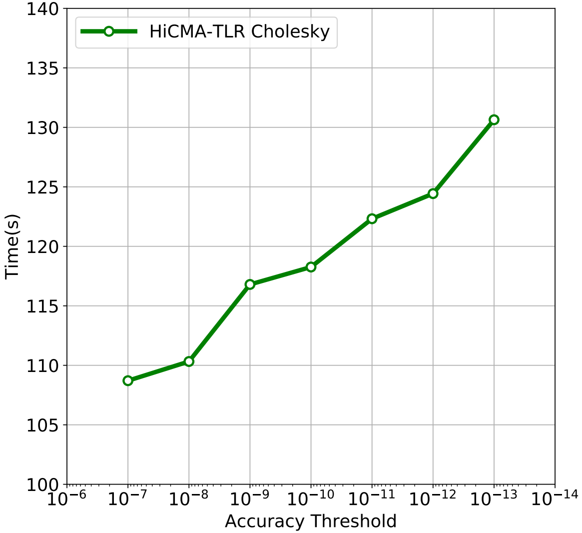
- Memory footprint for dense vs TLR matrices. Smaller the better.
- Application: Geospatial statistic, square exp. kernel
- *x* axis (*log*-scale): different accuracy thresholds
- *y* axis (*log*-scale): memory required for storing matrix in terms of giga bytes
- Being more accurate requires higher ranks so larger memory is used.



K. Akbudak, H. Ltaief, A. Mikhalev, A. Charara, and D. E. Keyes, Exploiting Data Sparsity for Large-Scale Matrix Computations, Submitted to EuroPar, 2018.

Impact of Accuracy Thresholds, **1M** Matrix Size, nb=2700, 64 nodes, *Shaheen-2*

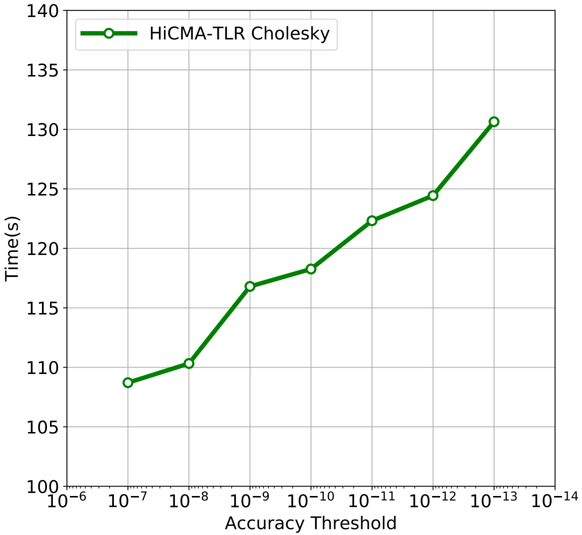
- Execution times for HiCMA dpotrf (TLR)



K. Akbudak, H. Ltaief, A. Mikhalev, A. Charara, and D. E. Keyes, Exploiting Data Sparsity for Large-Scale Matrix Computations, Submitted to EuroPar, 2018.

Impact of Accuracy Thresholds, **1M** Matrix Size, nb=2700, 64 nodes, *Shaheen-2*

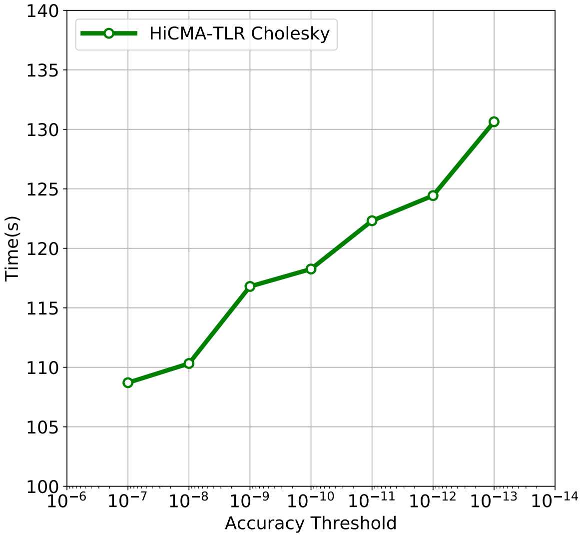
- Execution times for HiCMA dpotrf (TLR)
- Application: Geospatial statistic, square exp. kernel



K. Akbudak, H. Ltaief, A. Mikhalev, A. Charara, and D. E. Keyes, Exploiting Data Sparsity for Large-Scale Matrix Computations, Submitted to EuroPar, 2018.

Impact of Accuracy Thresholds, **1M** Matrix Size, nb=2700, 64 nodes, *Shaheen-2*

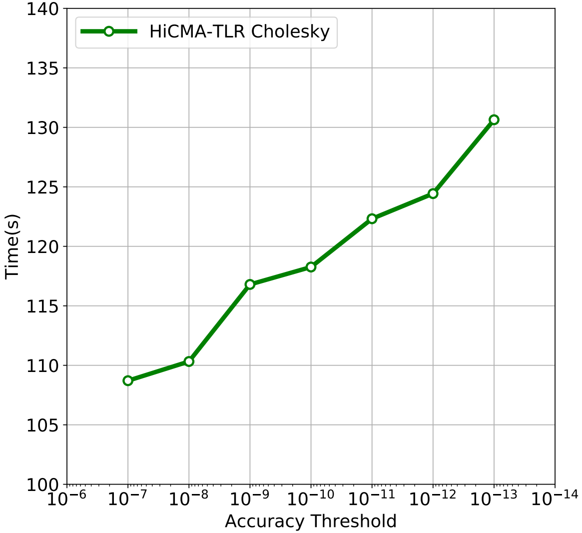
- Execution times for HiCMA dpotrf (TLR)
- Application: Geospatial statistic, square exp. kernel
- *x* axis (*log*-scale): different accuracy thresholds



K. Akbudak, H. Ltaief, A. Mikhalev, A. Charara, and D. E. Keyes, Exploiting Data Sparsity for Large-Scale Matrix Computations, Submitted to EuroPar, 2018.

Impact of Accuracy Thresholds, **1M** Matrix Size, nb=2700, 64 nodes, *Shaheen-2*

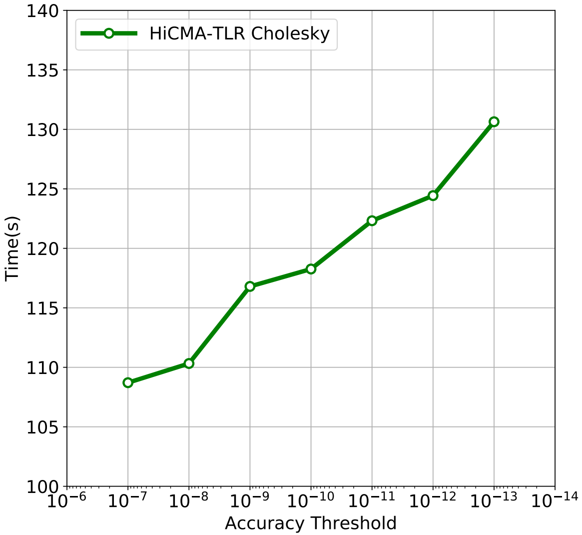
- Execution times for HiCMA dpotrf (TLR)
- Application: Geospatial statistic, square exp. kernel
- x axis (*log*-scale): different accuracy thresholds
- y axis (*log*-scale): time of `dpotrf` in seconds



K. Akbudak, H. Ltaief, A. Mikhalev, A. Charara, and D. E. Keyes, Exploiting Data Sparsity for Large-Scale Matrix Computations, Submitted to EuroPar, 2018.

Impact of Accuracy Thresholds, **1M** Matrix Size, nb=2700, 64 nodes, *Shaheen-2*

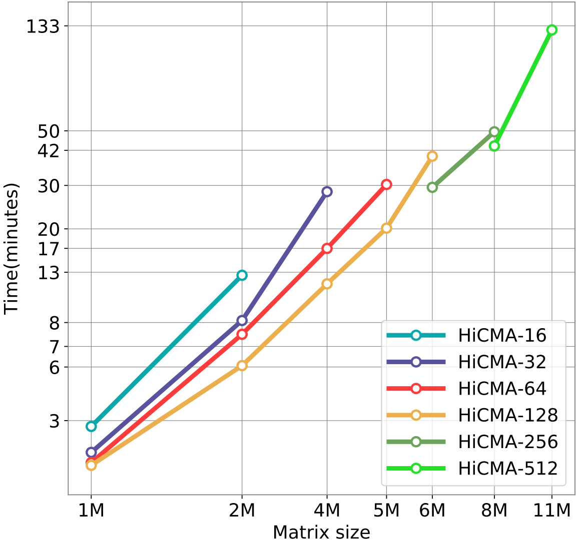
- Execution times for HiCMA dpotrf (TLR)
- Application: Geospatial statistic, square exp. kernel
- x axis (*log*-scale): different accuracy thresholds
- y axis (*log*-scale): time of `dpotrf` in seconds
- Being more accurate requires higher ranks so more flops are done.



K. Akbudak, H. Ltaief, A. Mikhalev, A. Charara, and D. E. Keyes, Exploiting Data Sparsity for Large-Scale Matrix Computations, Submitted to EuroPar, 2018.

TLR Cholesky up to 11M (*Shaheen-2*, HSW, Statistics - SqExp kernel, acc=10^{-9})

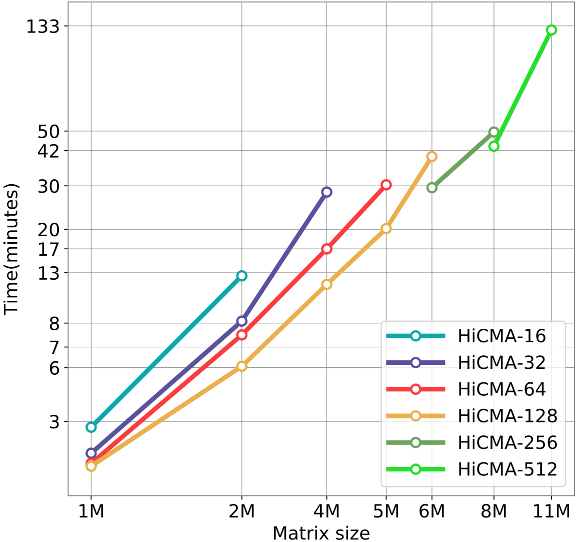
- Execution times for HiCMA dpotrf (TLR) for different number of nodes



K. Akbudak, H. Ltaief, A. Mikhalev, A. Charara, and D. E. Keyes, Exploiting Data Sparsity for Large-Scale Matrix Computations, Submitted to EuroPar, 2018.

TLR Cholesky up to 11M (*Shaheen-2*, HSW, Statistics - SqExp kernel, acc=10^{-9})

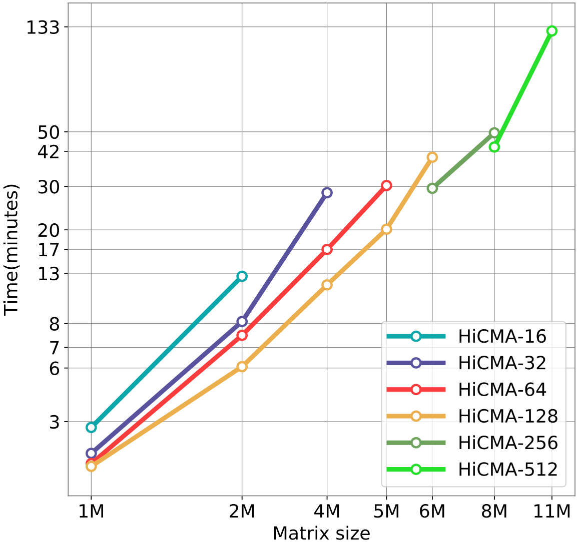
- Execution times for HiCMA dpotrf (TLR) for different number of nodes
- Application: Geospatial statistic, square exp. kernel



K. Akbudak, H. Ltaief, A. Mikhalev, A. Charara, and D. E. Keyes, Exploiting Data Sparsity for Large-Scale Matrix Computations, Submitted to EuroPar, 2018.

TLR Cholesky up to 11M (*Shaheen-2*, HSW, Statistics - SqExp kernel, acc=10^{-9})

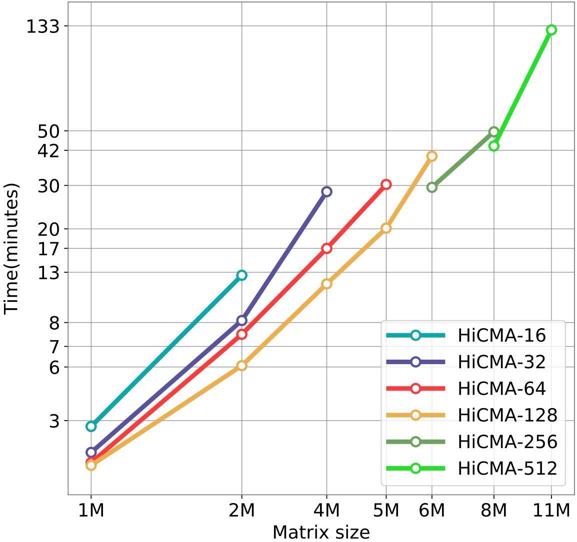
- Execution times for HiCMA dpotrf (TLR) for different number of nodes
- Application: Geospatial statistic, square exp. kernel
- *x* axis (*log*-scale): different matrix sizes



K. Akbudak, H. Ltaief, A. Mikhalev, A. Charara, and D. E. Keyes, Exploiting Data Sparsity for Large-Scale Matrix Computations, Submitted to EuroPar, 2018.

TLR Cholesky up to 11M (*Shaheen-2*, HSW, Statistics - SqExp kernel, acc=10^{-9})

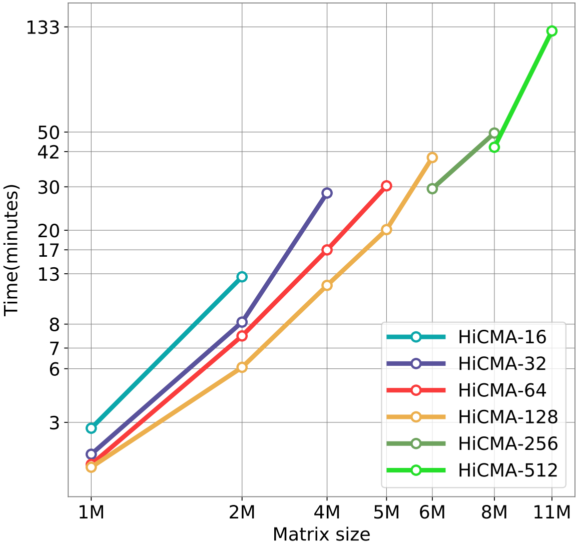
- Execution times for HiCMA dpotrf (TLR) for different number of nodes
- Application: Geospatial statistic, square exp. kernel
- *x* axis (*log*-scale): different matrix sizes
- *y* axis (*log*-scale): time of `dpotrf` in minutes



K. Akbudak, H. Ltaief, A. Mikhalev, A. Charara, and D. E. Keyes, Exploiting Data Sparsity for Large-Scale Matrix Computations, Submitted to EuroPar, 2018.

TLR Cholesky up to 11M (*Shaheen-2*, HSW, Statistics - SqExp kernel, acc=10^{-9})

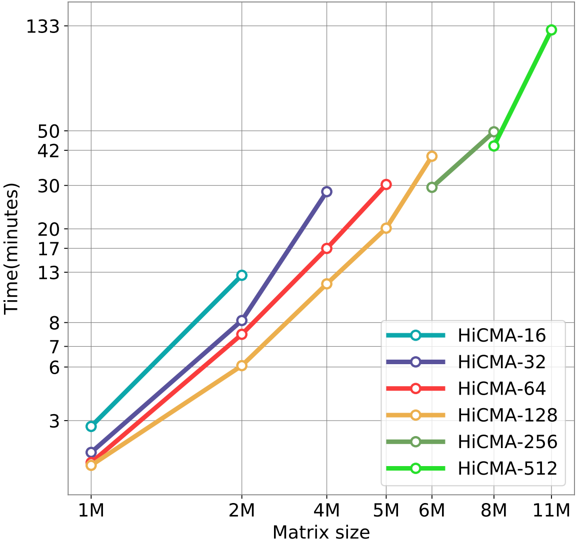
- Execution times for HiCMA dpotrf (TLR) for different number of nodes
- Application: Geospatial statistic, square exp. kernel
- x axis (*log*-scale): different matrix sizes
- y axis (*log*-scale): time of `dpotrf` in minutes
- Running ScaLAPACK for larger cases is not feasible because of time and memory.



K. Akbudak, H. Ltaief, A. Mikhalev, A. Charara, and D. E. Keyes, Exploiting Data Sparsity for Large-Scale Matrix Computations, Submitted to EuroPar, 2018.

TLR Cholesky up to 11M (*Shaheen-2*, HSW, Statistics - SqExp kernel, acc=10^{-9})

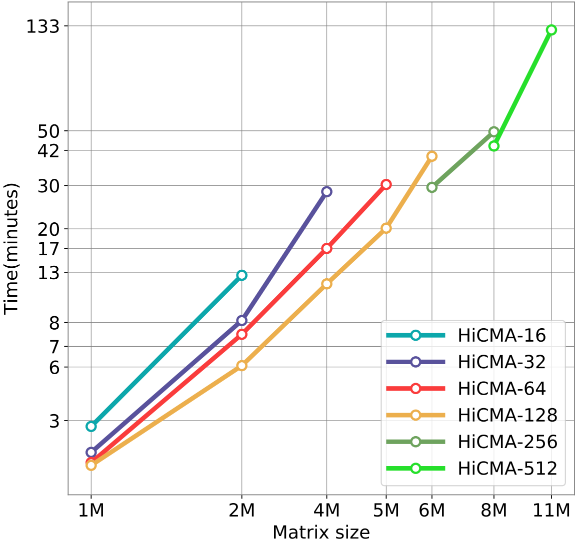
- Execution times for HiCMA dpotrf (TLR) for different number of nodes
- Application: Geospatial statistic, square exp. kernel
- x axis (*log*-scale): different matrix sizes
- y axis (*log*-scale): time of `dpotrf` in minutes
- Running ScaLAPACK for larger cases is not feasible because of time and memory.
- Missing points for larger matrices: not enough memory



K. Akbudak, H. Ltaief, A. Mikhalev, A. Charara, and D. E. Keyes, Exploiting Data Sparsity for Large-Scale Matrix Computations, Submitted to EuroPar, 2018.

TLR Cholesky up to 11M (*Shaheen-2*, HSW, Statistics - SqExp kernel, acc=10^{-9})

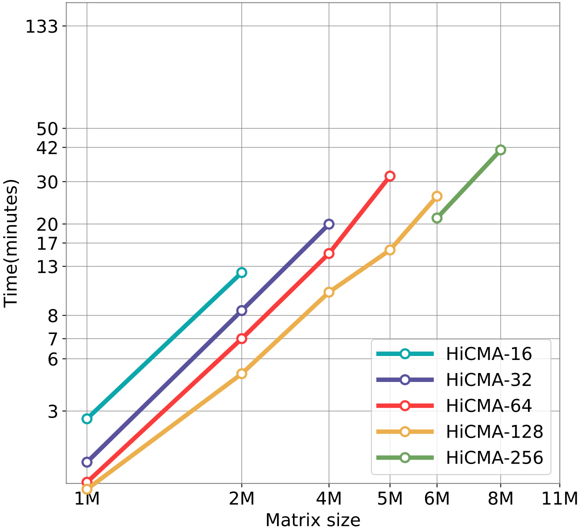
- Execution times for HiCMA dpotrf (TLR) for different number of nodes
- Application: Geospatial statistic, square exp. kernel
- x axis (*log*-scale): different matrix sizes
- y axis (*log*-scale): time of `dpotrf` in minutes
- Running ScaLAPACK for larger cases is not feasible because of time and memory.
- Missing points for larger matrices: not enough memory
- Missing points for smaller matrices: not enough work



K. Akbudak, H. Ltaief, A. Mikhalev, A. Charara, and D. E. Keyes, Exploiting Data Sparsity for Large-Scale Matrix Computations, Submitted to EuroPar, 2018.

TLR Cholesky up to 8M (SKL Cluster, Turbo On, Statistics - SqExp, acc=10^{-9})

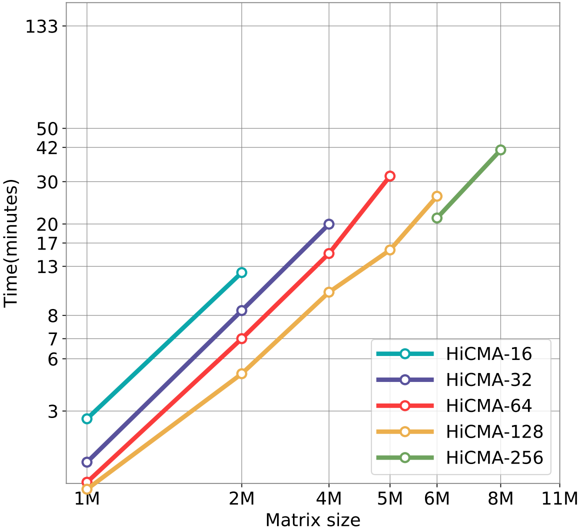
- Execution times for HiCMA dpotrf (TLR) for different number of nodes



K. Akbudak, H. Ltaief, A. Mikhalev, A. Charara, and D. E. Keyes, Exploiting Data Sparsity for Large-Scale Matrix Computations, Submitted to EuroPar, 2018.

TLR Cholesky up to 8M (SKL Cluster, Turbo On, Statistics - SqExp, acc=10^{-9})

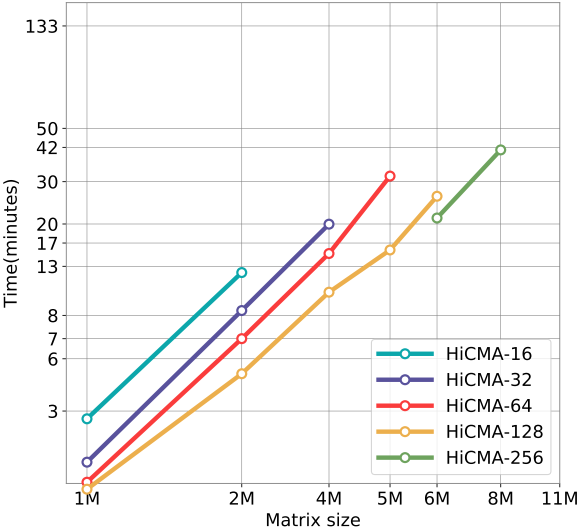
- Execution times for HiCMA dpotrf (TLR) for different number of nodes
- Application: Geospatial statistic, square exp. kernel



K. Akbudak, H. Ltaief, A. Mikhalev, A. Charara, and D. E. Keyes, Exploiting Data Sparsity for Large-Scale Matrix Computations, Submitted to EuroPar, 2018.

TLR Cholesky up to 8M (SKL Cluster, Turbo On, Statistics - SqExp, acc=10^{-9})

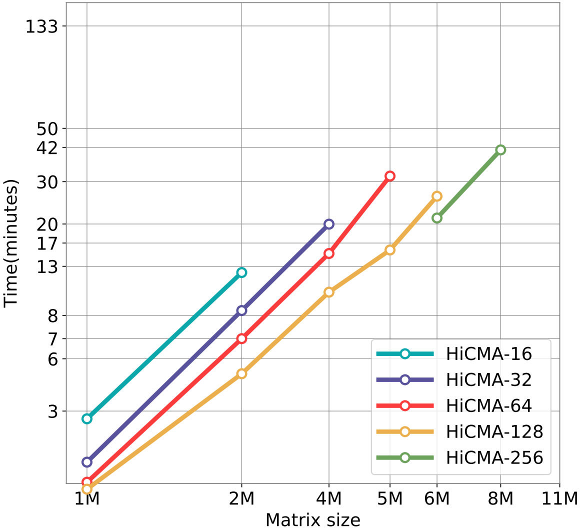
- Execution times for HiCMA dpotrf (TLR) for different number of nodes
- Application: Geospatial statistic, square exp. kernel
- *x* axis (*log*-scale): different matrix sizes



K. Akbudak, H. Ltaief, A. Mikhalev, A. Charara, and D. E. Keyes, Exploiting Data Sparsity for Large-Scale Matrix Computations, Submitted to EuroPar, 2018.

TLR Cholesky up to 8M (SKL Cluster, Turbo On, Statistics - SqExp, acc=10^{-9})

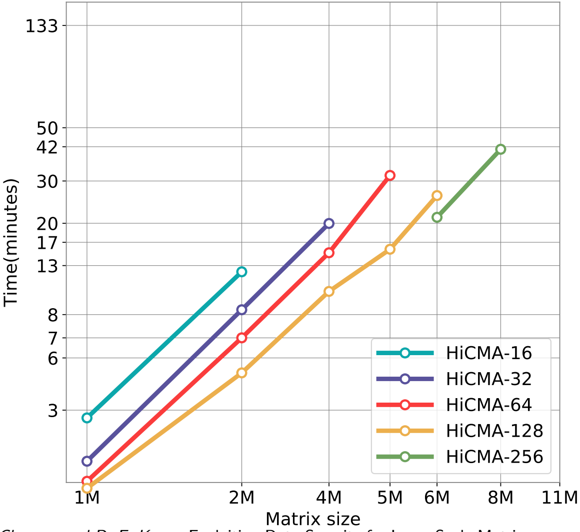
- Execution times for HiCMA dpotrf (TLR) for different number of nodes
- Application: Geospatial statistic, square exp. kernel
- x axis (*log*-scale): different matrix sizes
- y axis (*log*-scale): time of `dpotrf` in minutes



K. Akbudak, H. Ltaief, A. Mikhalev, A. Charara, and D. E. Keyes, Exploiting Data Sparsity for Large-Scale Matrix Computations, Submitted to EuroPar, 2018.

TLR Cholesky up to 8M (SKL Cluster, Turbo On, Statistics - SqExp, acc=10^{-9})

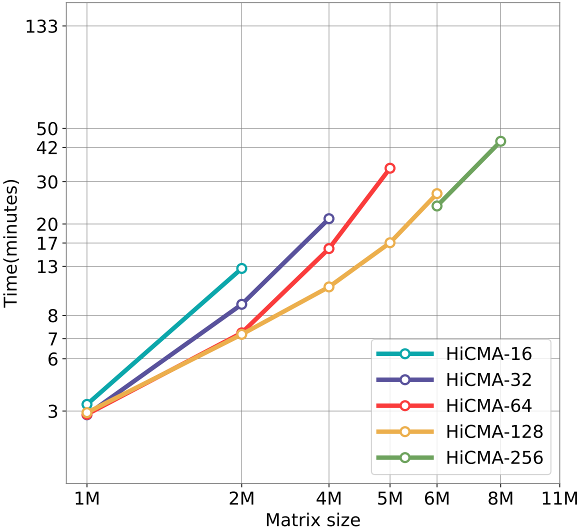
- Execution times for HiCMA dpotrf (TLR) for different number of nodes
- Application: Geospatial statistic, square exp. kernel
- *x* axis (*log*-scale): different matrix sizes
- *y* axis (*log*-scale): time of `dpotrf` in minutes
- We ran HiCMA on the latest Intel deployment.



K. Akbudak, H. Ltaief, A. Mikhalev, A. Charara, and D. E. Keyes, Exploiting Data Sparsity for Large-Scale Matrix Computations, Submitted to EuroPar, 2018.

TLR Cholesky up to 8M (SKL Cluster, Turbo Off, Statistics - SqExp, acc=10^{-9})

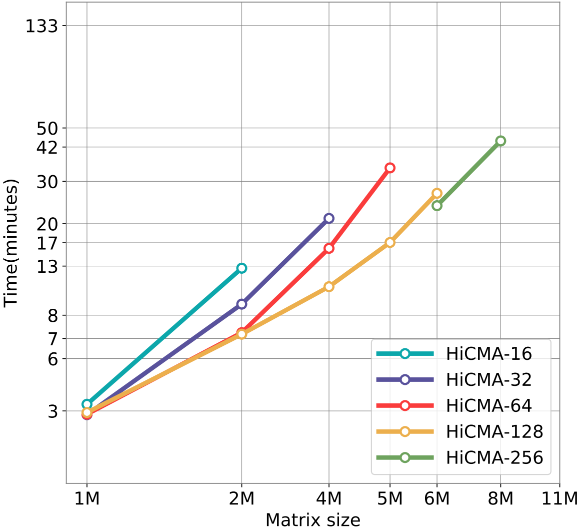
- Execution times for HiCMA dpotrf (TLR) for different number of nodes



K. Akbudak, H. Ltaief, A. Mikhalev, A. Charara, and D. E. Keyes, Exploiting Data Sparsity for Large-Scale Matrix Computations, Submitted to EuroPar, 2018.

TLR Cholesky up to 8M (SKL Cluster, Turbo Off, Statistics - SqExp, acc=10^{-9})

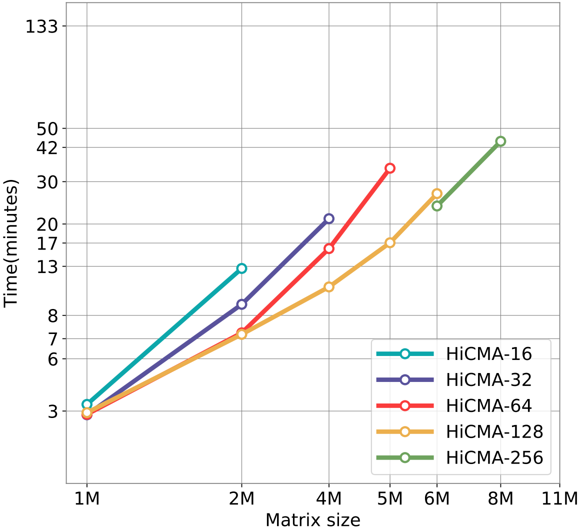
- Execution times for HiCMA dpotrf (TLR) for different number of nodes
- Application: Geospatial statistic, square exp. kernel



K. Akbudak, H. Ltaief, A. Mikhalev, A. Charara, and D. E. Keyes, Exploiting Data Sparsity for Large-Scale Matrix Computations, Submitted to EuroPar, 2018.

TLR Cholesky up to 8M (SKL Cluster, Turbo Off, Statistics - SqExp, acc=10^{-9})

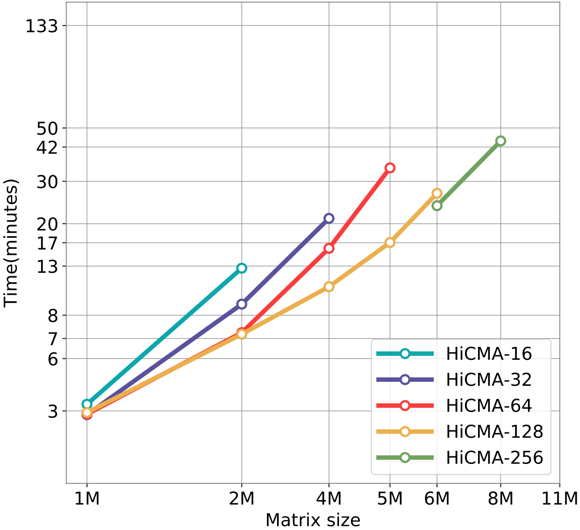
- Execution times for HiCMA dpotrf (TLR) for different number of nodes
- Application: Geospatial statistic, square exp. kernel
- *x* axis (*log*-scale): different matrix sizes



K. Akbudak, H. Ltaief, A. Mikhalev, A. Charara, and D. E. Keyes, Exploiting Data Sparsity for Large-Scale Matrix Computations, Submitted to EuroPar, 2018.

TLR Cholesky up to 8M (SKL Cluster, Turbo Off, Statistics - SqExp, acc=10^{-9})

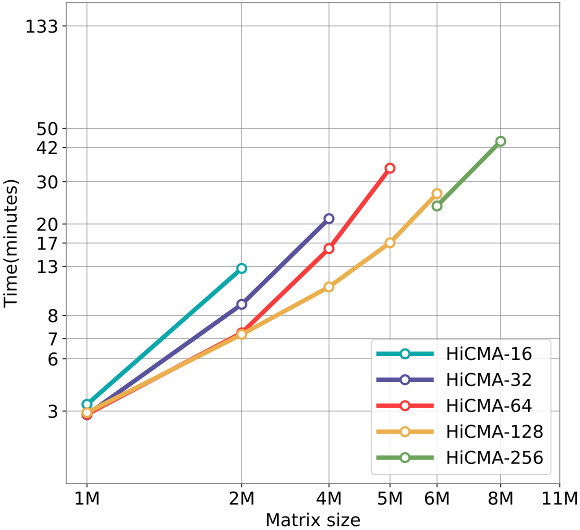
- Execution times for HiCMA dpotrf (TLR) for different number of nodes
- Application: Geospatial statistic, square exp. kernel
- *x* axis (*log*-scale): different matrix sizes
- *y* axis (*log*-scale): time of `dpotrf` in minutes



K. Akbudak, H. Ltaief, A. Mikhalev, A. Charara, and D. E. Keyes, Exploiting Data Sparsity for Large-Scale Matrix Computations, Submitted to EuroPar, 2018.

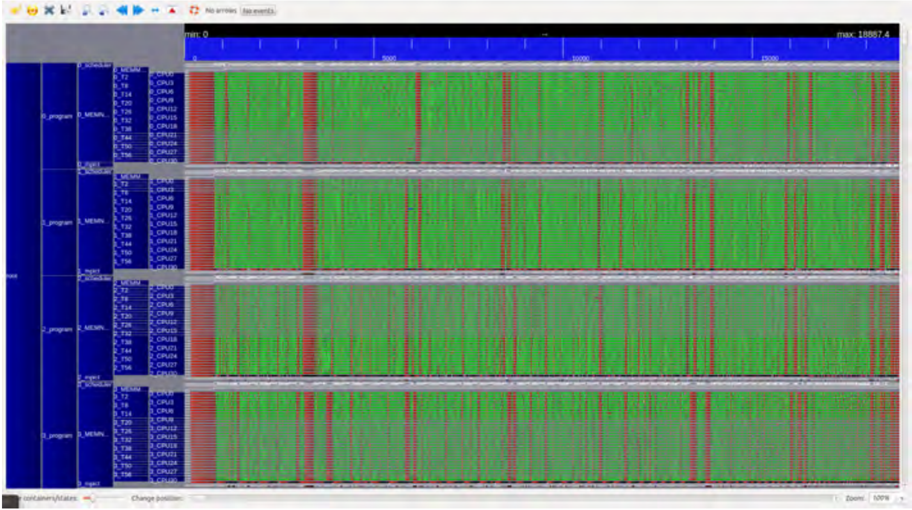
TLR Cholesky up to 8M (SKL Cluster, Turbo Off, Statistics - SqExp, acc=10^{-9})

- Execution times for HiCMA dpotrf (TLR) for different number of nodes
- Application: Geospatial statistic, square exp. kernel
- *x* axis (*log*-scale): different matrix sizes
- *y* axis (*log*-scale): time of `dpotrf` in minutes
- Not big difference between turbo on and off due low arithmetic intensity

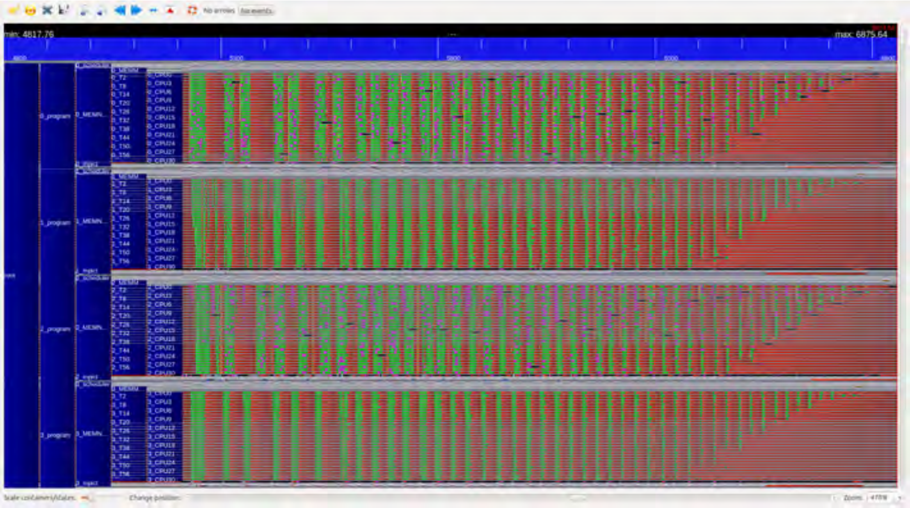


K. Akbudak, H. Ltaief, A. Mikhalev, A. Charara, and D. E. Keyes, Exploiting Data Sparsity for Large-Scale Matrix Computations, Submitted to EuroPar, 2018.

Traces Chameleon: Dense dpotrf time=18.1s on 4 nodes of *Shaheen-2* with a matrix size of 54K



Traces HiCMA: Data-sparse `dpotrf` time=1.8s on 4 nodes of *Shaheen-2* with a matrix size of 54K

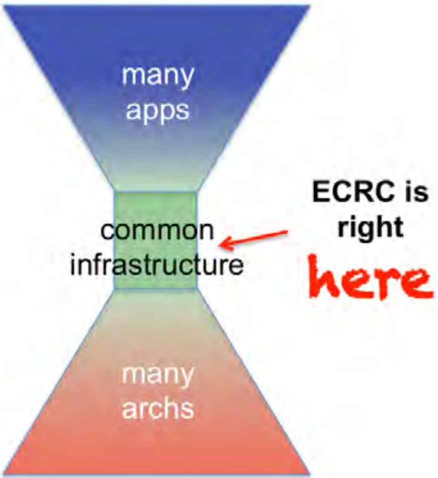


Conclusion & Future Work

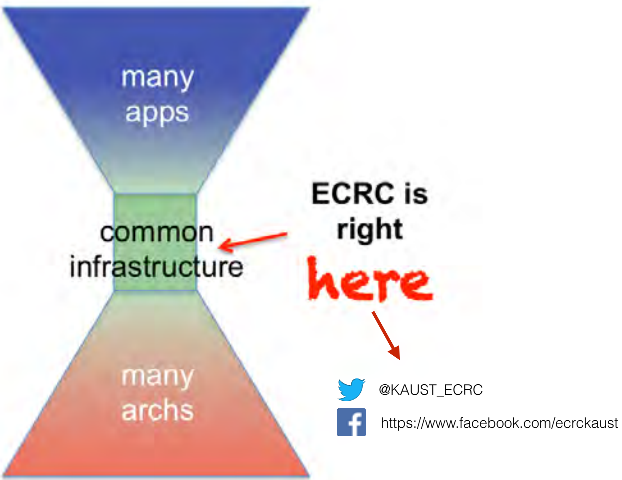
- We cut down flops through compression.
- We use less memory.
- Accuracy requirement is satisfied.
- HiCMA is for both shared and distributed memory systems.

- HODLR/H (non-nested bases) data compression format.
- More matrix computation algorithms (LU/QR, eigenvalue solvers/SVD, matrix inversion, etc.)
- Support for more large-scale scientific applications.

The Hourglass Revisited



The Hourglass Revisited



Students/Collaborators/Vendors

- Extreme Computing Research Center @ KAUST: **R. Abdelkhalek, S. Abdullah, K. Akbudak, R. AlOmairy, A. Alonazy, T. Alturkestani, W. Boukaram, A. Charara, N. Doucet, M. Genton, E. A. Gonzalez Fisher, D. Keyes, H. Ltaief, A. Mikhalev, D. Sukkari and Y. Sun**
- Tokyo Institute of Technology, Japan: **R. Yokota**
- Innovative Computing Laboratory @ UTK, USA: **PLASMA/MAGMA/PaRSEC Teams**
- INRIA/INP/LaBRI Bordeaux, France: **Runtime/HiePACS Teams**
- Max-Planck Institute@Leipzig, Germany: **R. Kriemann**
- American University of Beirut, Lebanon: **G. Turkiyyah**
- KAUST Supercomputing Lab
- Intel Parallel Computing Center
- Cray Center of Excellence

Thank You!

Questions?