Recent HPC developments for Multidimensional Scaling (MDS)

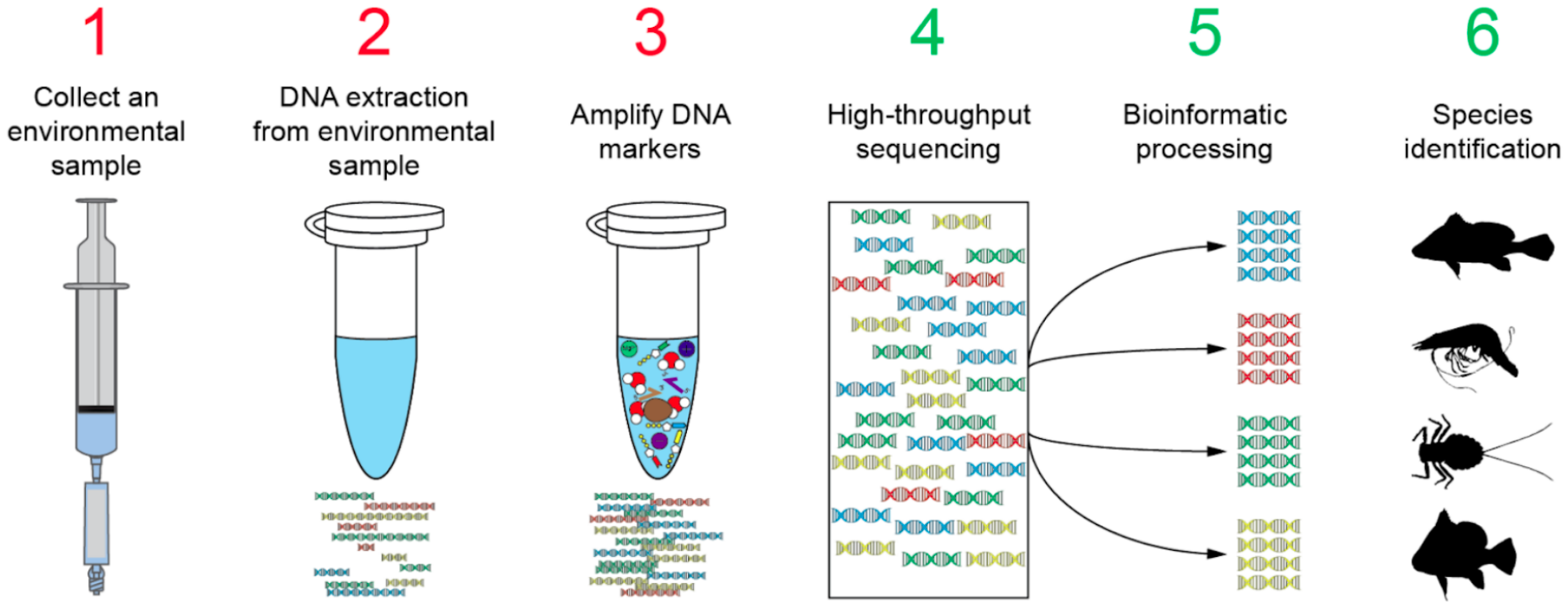
HPC Scalable ecosystem plenary meeting

Romain Peressoni

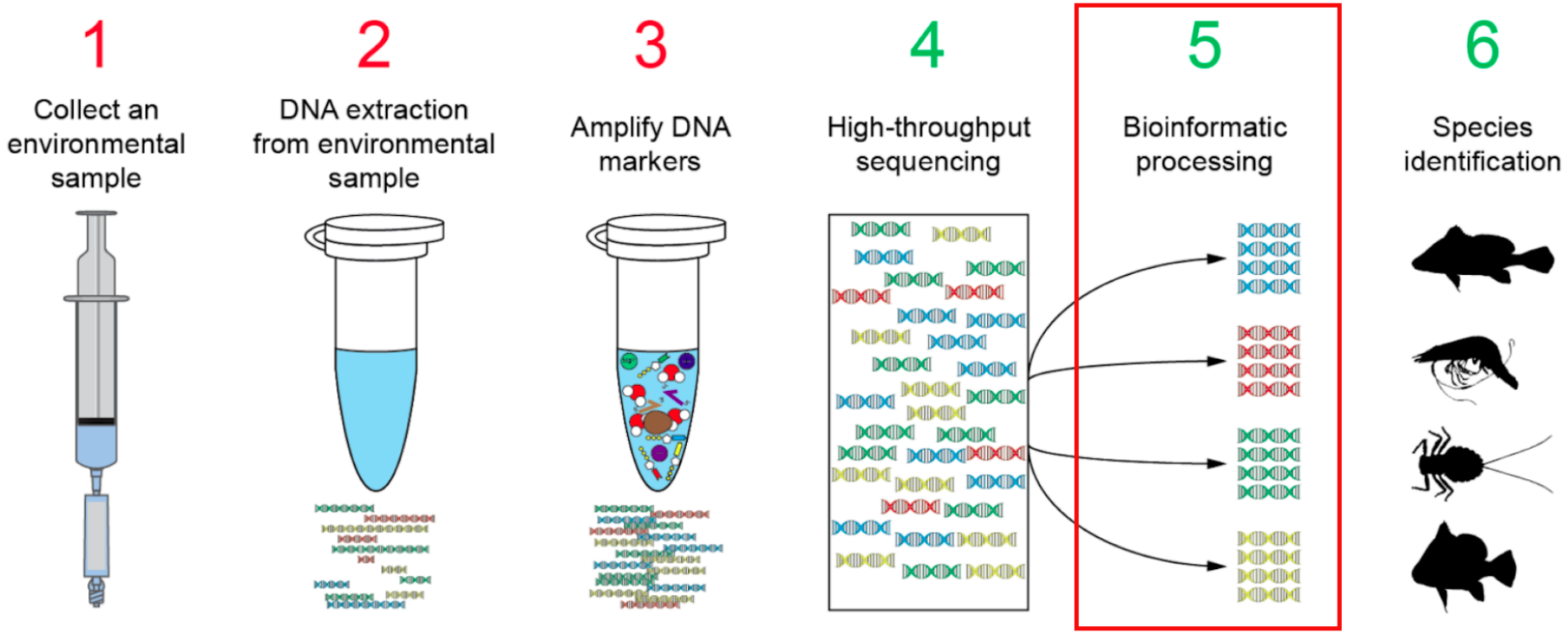
Centre Inria de l'Université de Bordeaux

31/03/2023

Metabarcoding



Metabarcoding



MDS (Euclidean distance matrix case)

MDS is a dimension reduction algorithm

Concept

Based on the link between

- Input: Distance (D)
- Inner Products (G)
- Output: Position (X)

Computation

- $G \leftarrow$ double centering of D
- // G $= XX^T$ by definition
- $Q, \Lambda, Q^T \leftarrow sEVD(G)$
- $X \leftarrow Q\Lambda^{\frac{1}{2}}$

MDS (Euclidean distance matrix case)

MDS is a dimension reduction algorithm

Concept

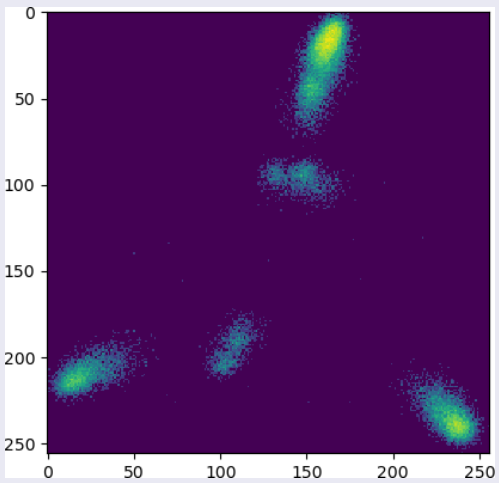
Based on the link between

- Input: Distance (D)
- Inner Products (G)
- Output: Position (X)

Computation

- $G \leftarrow$ double centering of D
- // G $= XX^T$ by definition
- $Q, \Lambda, Q^T \leftarrow sEVD(G)$
- $X \leftarrow Q\Lambda^{\frac{1}{2}}$

Output : Point Cloud



Limiting Factor

Cost of the sEVD

MDS (Euclidean distance matrix case)

MDS is a dimension reduction algorithm

Concept

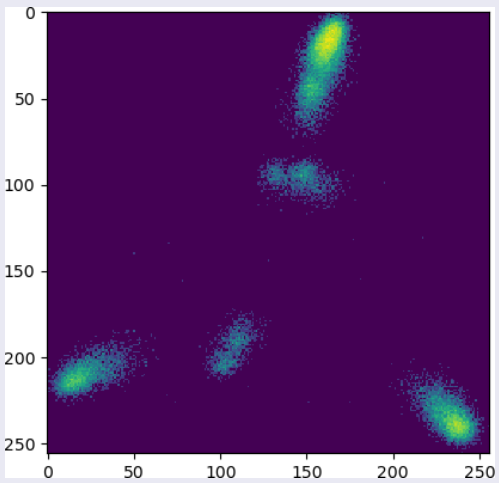
Based on the link between

- Input: Distance (D)
- Inner Products (G)
- Output: Position (X)

Computation

- $G \leftarrow$ double centering of D
- // G $= XX^T$ by definition
- $U, \Sigma, U^T \leftarrow SVD(G)$
- $X \leftarrow U\Sigma^{\frac{1}{2}}$

Output : Point Cloud

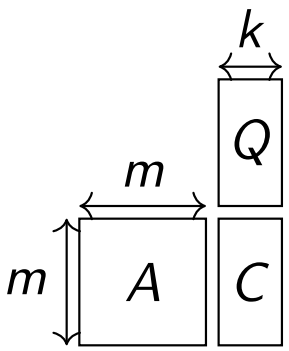


Limiting Factor

Cost of the sEVD or SVD

Solution : randomized algorithms

- Approximate $A \approx AQQ^T$ (Halko et al., 2011; Martinsson, 2016)
- Define $C = AQ$
- Compute $SVD(C) = U_C \Sigma V_C^T$

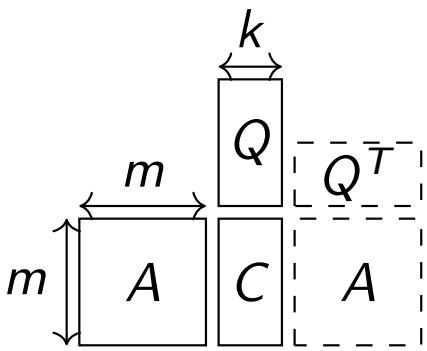


Implementation

- Random projection based MDS (Blanchard et al., 2018; Blanchard, 2017; Franc et al., 2020; Paradis, 2018)
- Task based HPC implementation of RSVD in the context of the Gordon Project (Agullo et al., 2022)

Solution : randomized algorithms

- Approximate $A \approx AQQ^T$ (Halko et al., 2011; Martinsson, 2016)
- Define $C = AQ$
- Compute $SVD(C) = U_C \Sigma V_C^T$
- Multiply by Q^T on the right to get $RSVD(A)$



Implementation

- Random projection based MDS (Blanchard et al., 2018; Blanchard, 2017; Franc et al., 2020; Paradis, 2018)
- Task based HPC implementation of RSVD in the context of the Gordon Project (Agullo et al., 2022)

Random Projection based SVD

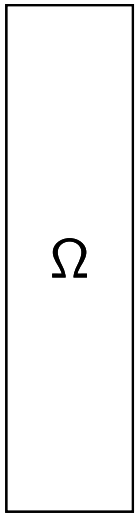
Algorithm: Random projection SVD algorithm: $(U, \Sigma, V) \simeq \text{RSVD(A)}$

Input: A: $m \times m$, k prescribed rank
Output: $A \simeq U\Sigma V^T$

1. Ω: $m \times k$ random matrix // RAND
2. $Y = A\Omega$ // MM1
3. Y: $QR = Y$ // QR
4. $C = AQ$ // MM2
5. C: $U_C \Sigma V_C^T = C$ // QR-SVD
6. $U = U_C$ and $V = QV_C$ // MM3
7. **return** U, Σ, V

Random Projection based SVD

RAND



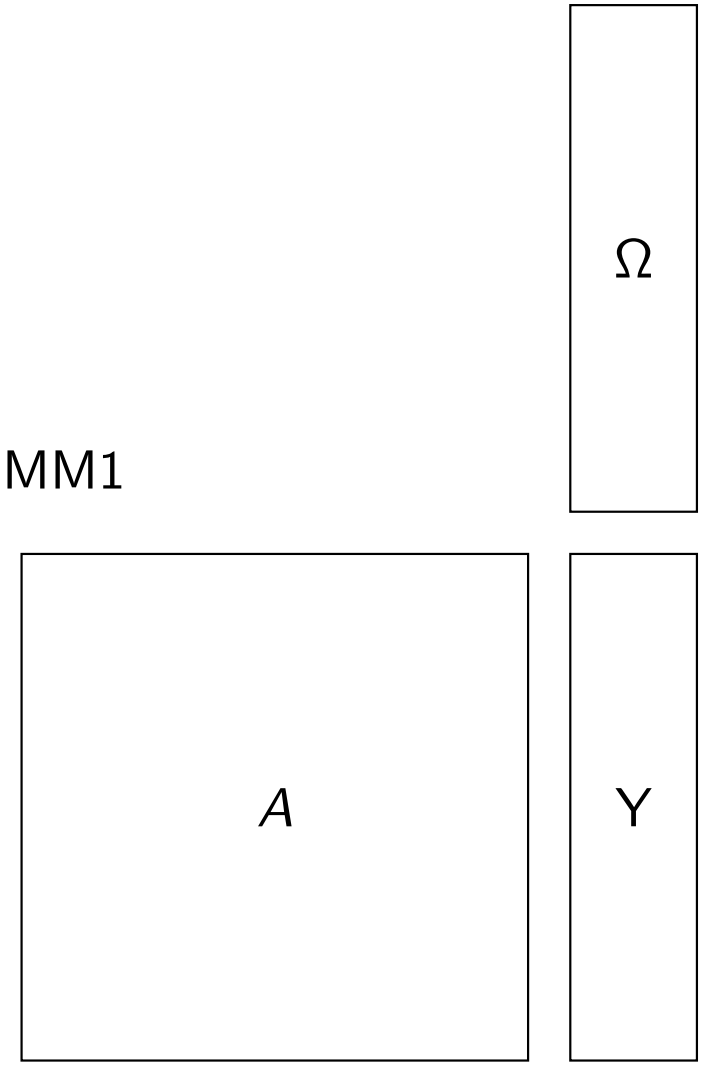
Algorithm: Random projection SVD algorithm: $(U, \Sigma, V) \simeq \text{RSVD(A)}$

Input: A: $m \times m$, k prescribed rank

Output: $A \simeq U \Sigma V^T$

1. Ω: $m \times k$ random matrix // RAND
2. $Y = A\Omega$ // MM1
3. Y: $QR = Y$ // QR
4. $C = AQ$ // MM2
5. C: $U_C \Sigma V_C^T = C$ // QR-SVD
6. $U = U_C$ and $V = QV_C$ // MM3
7. **return** U, Σ, V

Random Projection based SVD



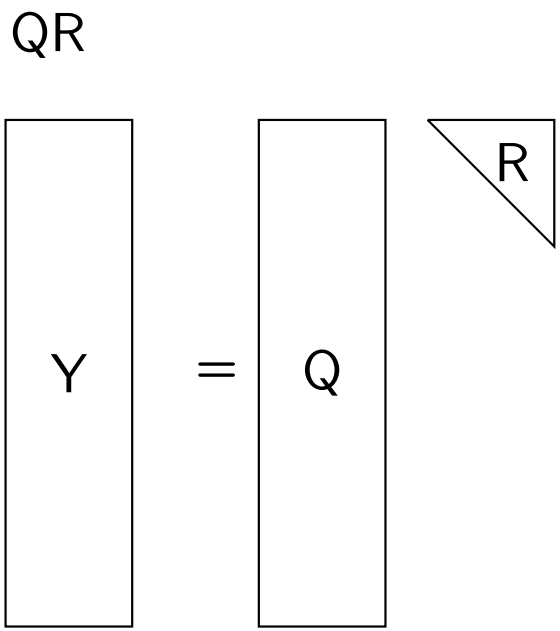
Algorithm: Random projection SVD algorithm: $(U, \Sigma, V) \simeq \text{RSVD(A)}$

Input: A: $m \times m$, k prescribed rank

Output: $A \simeq U \Sigma V^T$

1. Ω: $m \times k$ random matrix // RAND
2. $Y = A\Omega$ // MM1
3. Y: $QR = Y$ // QR
4. $C = AQ$ // MM2
5. C: $U_C \Sigma V_C^T = C$ // QR-SVD
6. $U = U_C$ and $V = QV_C$ // MM3
7. **return** U, Σ, V

Random Projection based SVD

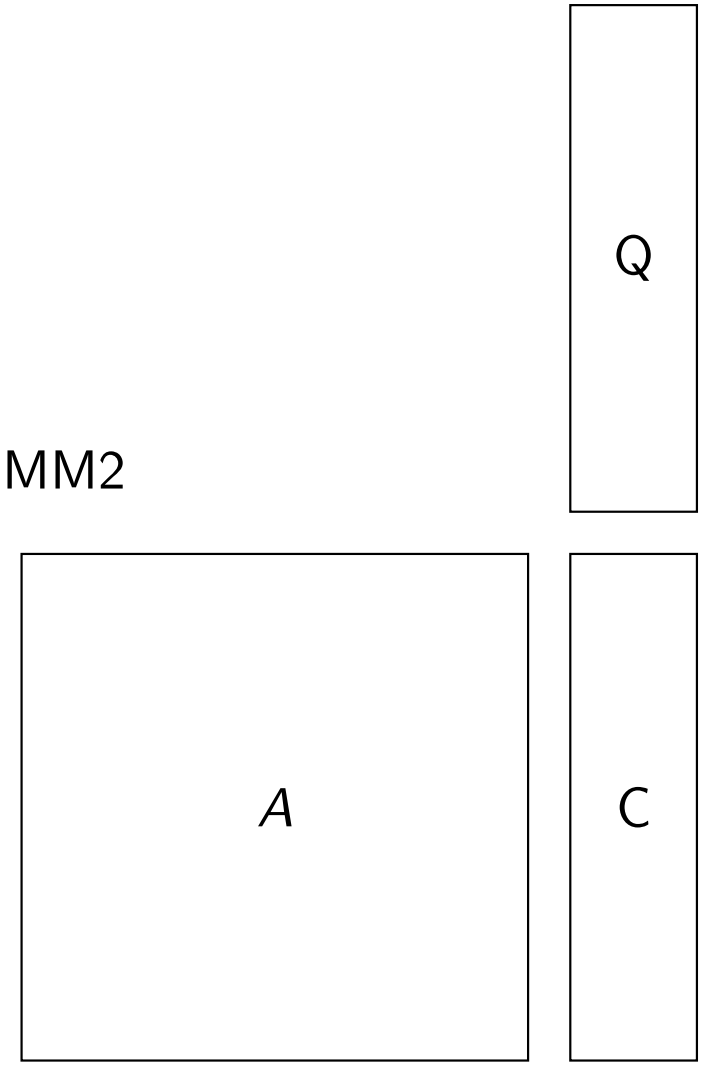


Algorithm: Random projection SVD algorithm: $(U, \Sigma, V) \simeq \text{RSVD(A)}$

Input: A: $m \times m$, k prescribed rank
Output: $A \simeq U\Sigma V^T$

1. Ω: $m \times k$ random matrix // RAND
2. $Y = A\Omega$ // MM1
3. Y: $QR = Y$ // QR
4. $C = AQ$ // MM2
5. C: $U_C \Sigma V_C^T = C$ // QR-SVD
6. $U = U_C$ and $V = QV_C$ // MM3
7. **return** U, Σ, V

Random Projection based SVD



Algorithm: Random projection SVD algorithm: $(U, \Sigma, V) \simeq \text{RSVD(A)}$

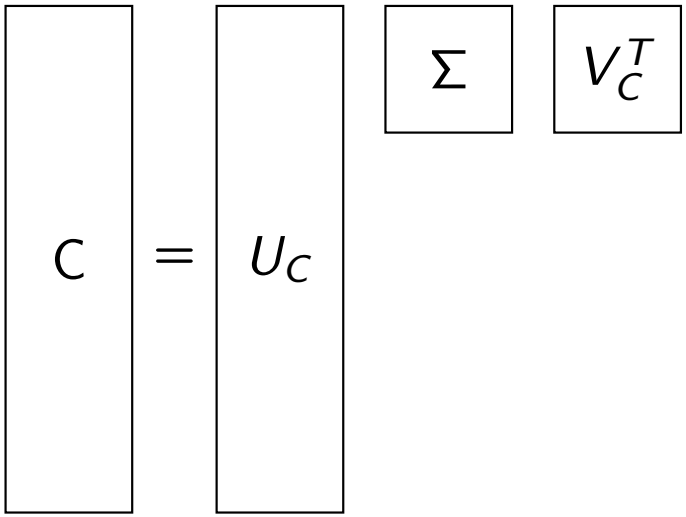
Input: A: $m \times m$, k prescribed rank

Output: $A \simeq U \Sigma V^T$

1. Ω: $m \times k$ random matrix // RAND
2. $Y = A\Omega$ // MM1
3. Y: $QR = Y$ // QR
4. $C = AQ$ // MM2
5. C: $U_C \Sigma V_C^T = C$ // QR-SVD
6. $U = U_C$ and $V = QV_C$ // MM3
7. **return** U, Σ, V

Random Projection based SVD

SVD

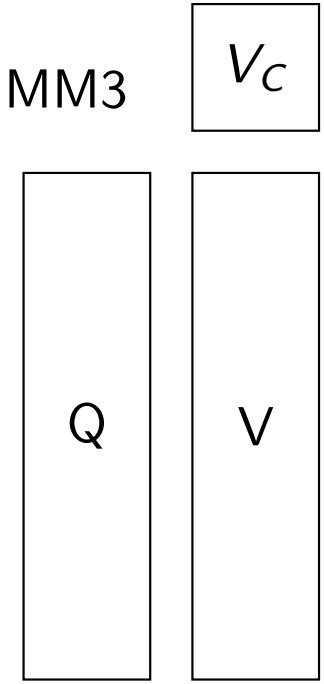


Algorithm: Random projection SVD algorithm: $(U, \Sigma, V) \simeq \text{RSVD(A)}$

Input: A: $m \times m$, k prescribed rank
Output: $A \simeq U \Sigma V^T$

1. Ω: $m \times k$ random matrix // RAND
2. $Y = A\Omega$ // MM1
3. Y: $QR = Y$ // QR
4. $C = AQ$ // MM2
5. C: $U_C \Sigma V_C^T = C$ // QR-SVD
6. $U = U_C$ and $V = QV_C$ // MM3
7. **return** U, Σ, V

Random Projection based SVD

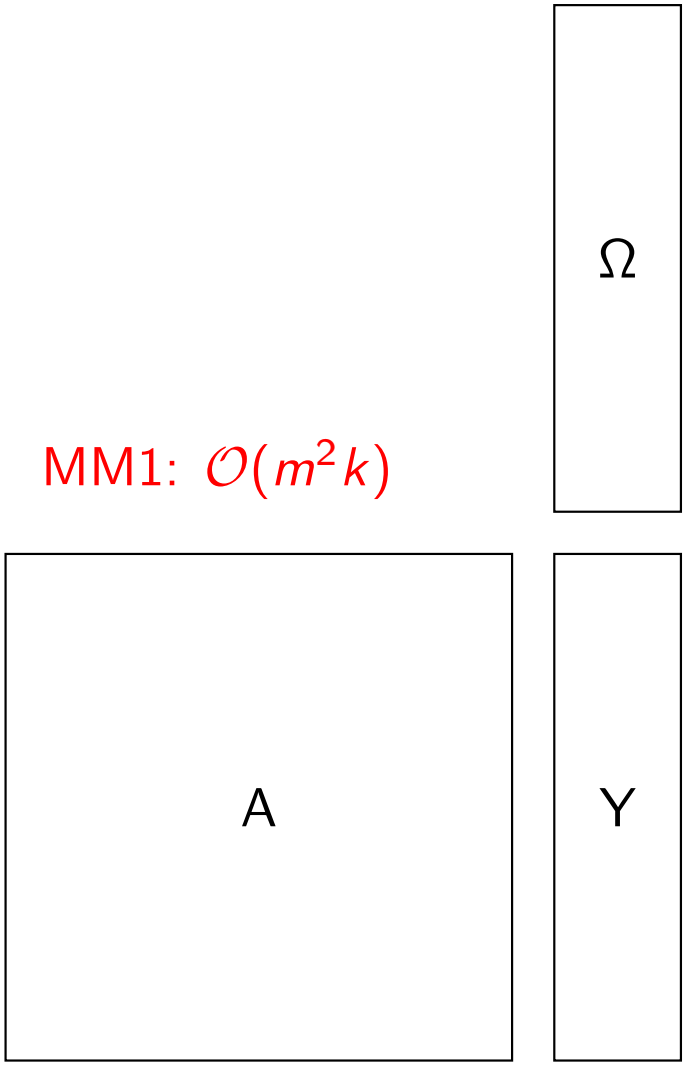


Algorithm: Random projection SVD algorithm: $(U, \Sigma, V) \simeq \text{RSVD(A)}$

Input: A: $m \times m$, k prescribed rank
Output: $A \simeq U \Sigma V^T$

1. Ω: $m \times k$ random matrix // RAND
2. $Y = A\Omega$ // MM1
3. Y: $QR = Y$ // QR
4. $C = AQ$ // MM2
5. C: $U_C \Sigma V_C^T = C$ // QR-SVD
6. $U = U_C$ and $V = QV_C$ // MM3
7. **return** U, Σ, V

Random Projection based SVD



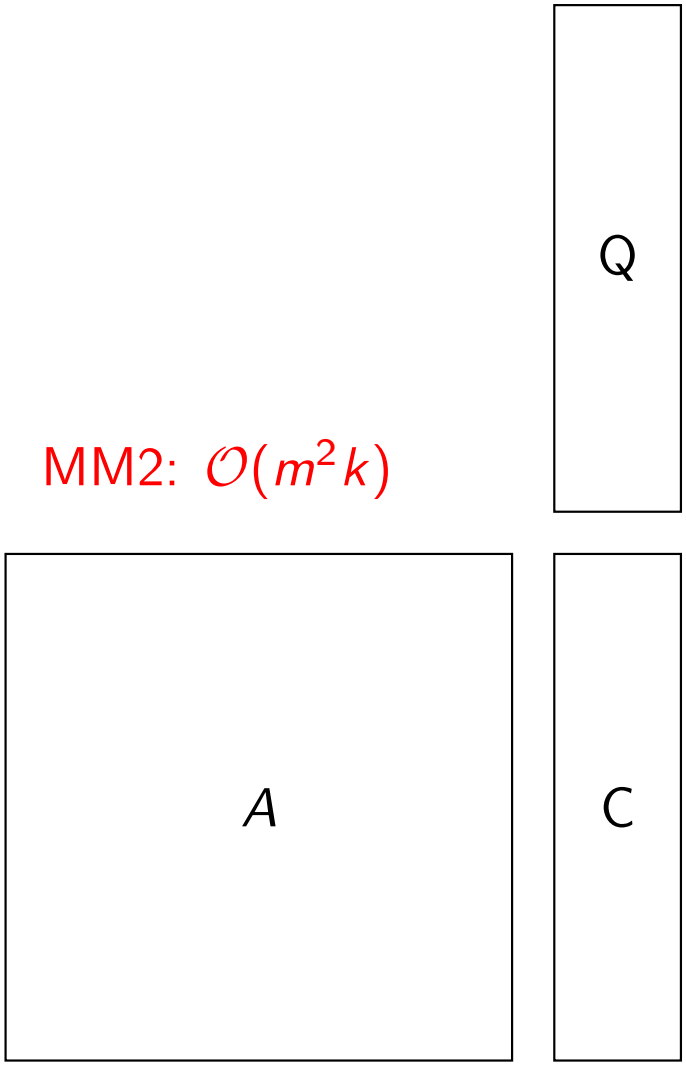
Algorithm: Random projection SVD algorithm: $(U, \Sigma, V) \simeq \text{RSVD(A)}$

Input: A: $m \times m$, k prescribed rank

Output: $A \simeq U \Sigma V^T$

1. Ω: $m \times k$ random matrix // RAND
2. $Y = A\Omega$ // MM1
3. Y: $QR = Y$ // QR
4. $C = AQ$ // MM2
5. C: $U_C \Sigma V_C^T = C$ // QR-SVD
6. $U = U_C$ and $V = QV_C$ // MM3
7. **return** U, Σ, V

Random Projection based SVD



Algorithm: Random projection SVD algorithm: $(U, \Sigma, V) \simeq \text{RSVD(A)}$

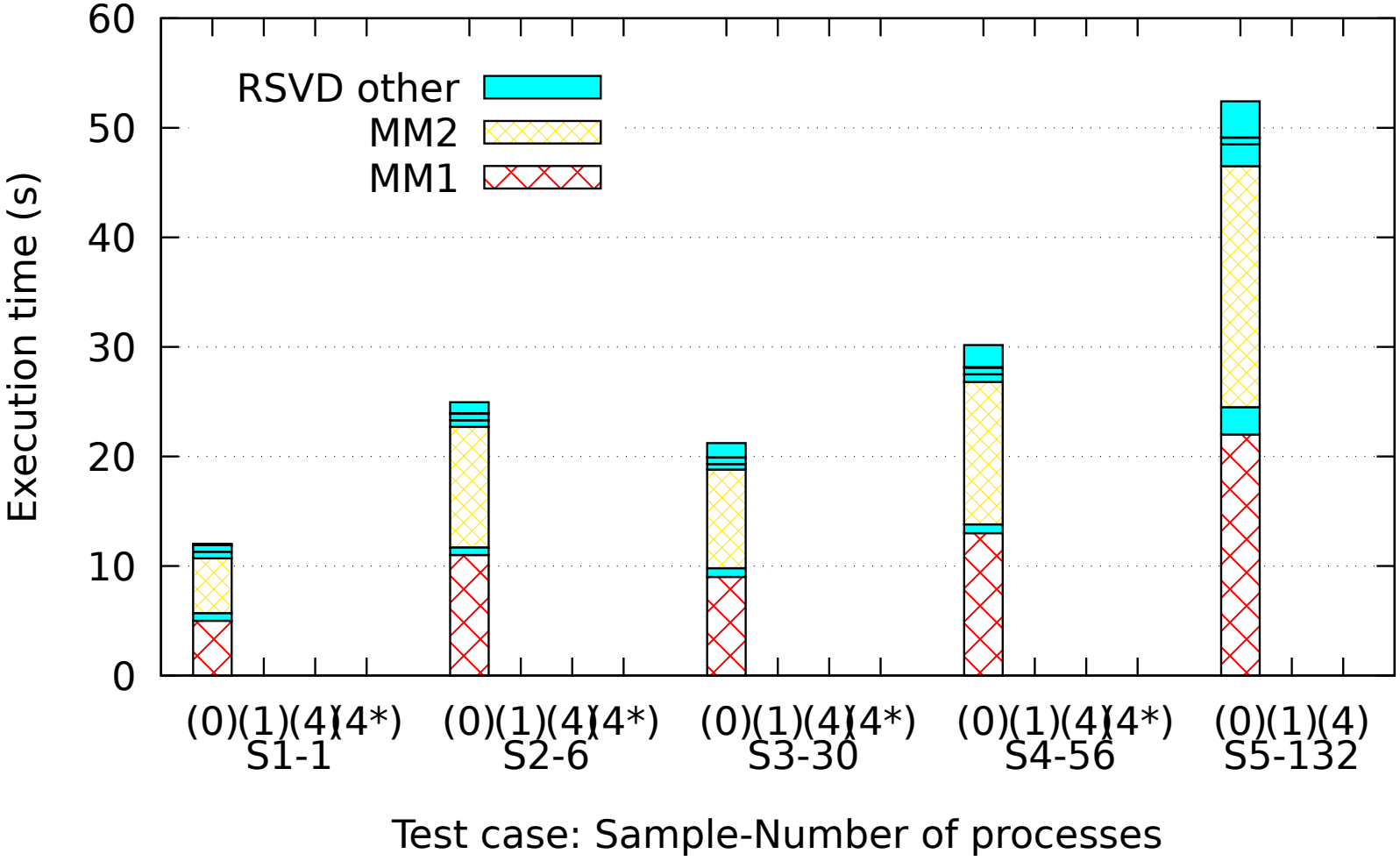
Input: A: $m \times m$, k prescribed rank

Output: $A \simeq U \Sigma V^T$

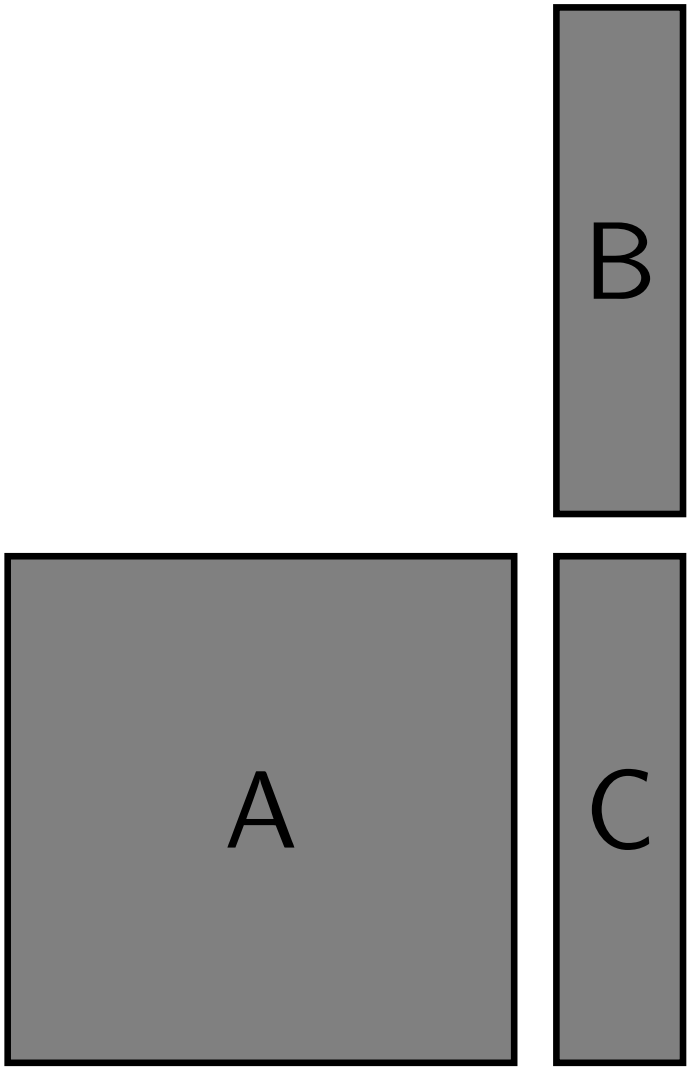
1. Ω: $m \times k$ random matrix // RAND
2. $Y = A\Omega$ // MM1
3. Y: $QR = Y$ // QR
4. $C = AQ$ // MM2
5. C: $U_C \Sigma V_C^T = C$ // QR-SVD
6. $U = U_C$ and $V = QV_C$ // MM3
7. **return** U, Σ, V

RSVD-MDS

Execution time for each step of the RSVD-MDS (Agullo et al., 2022)

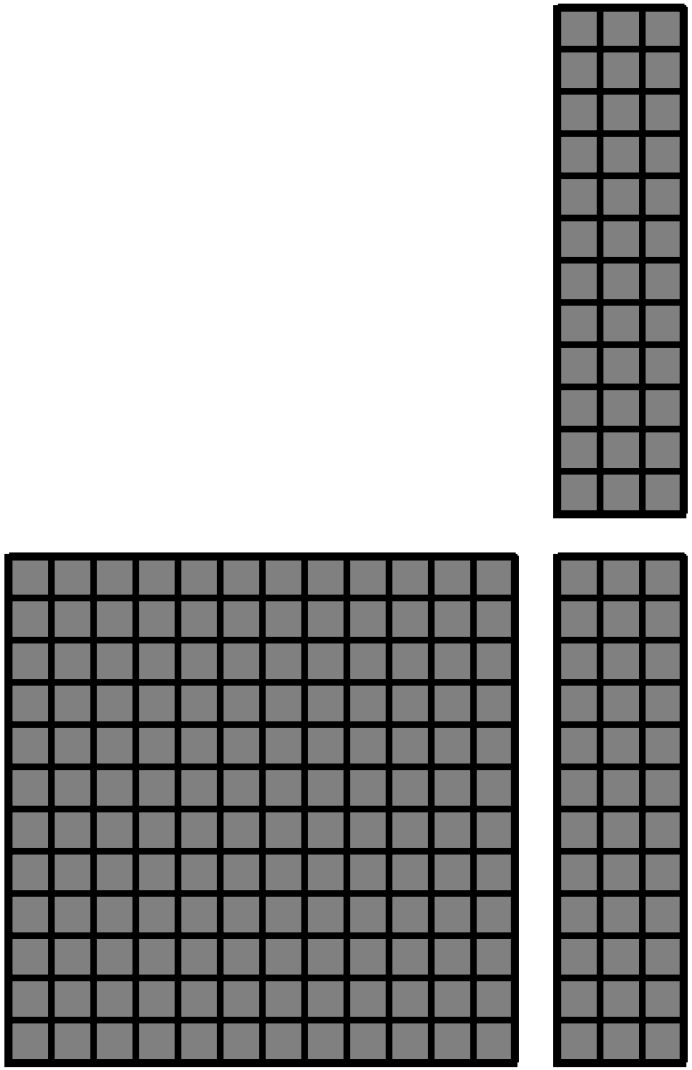


GEMM (GEneral Matrix-Matrix multiplication)



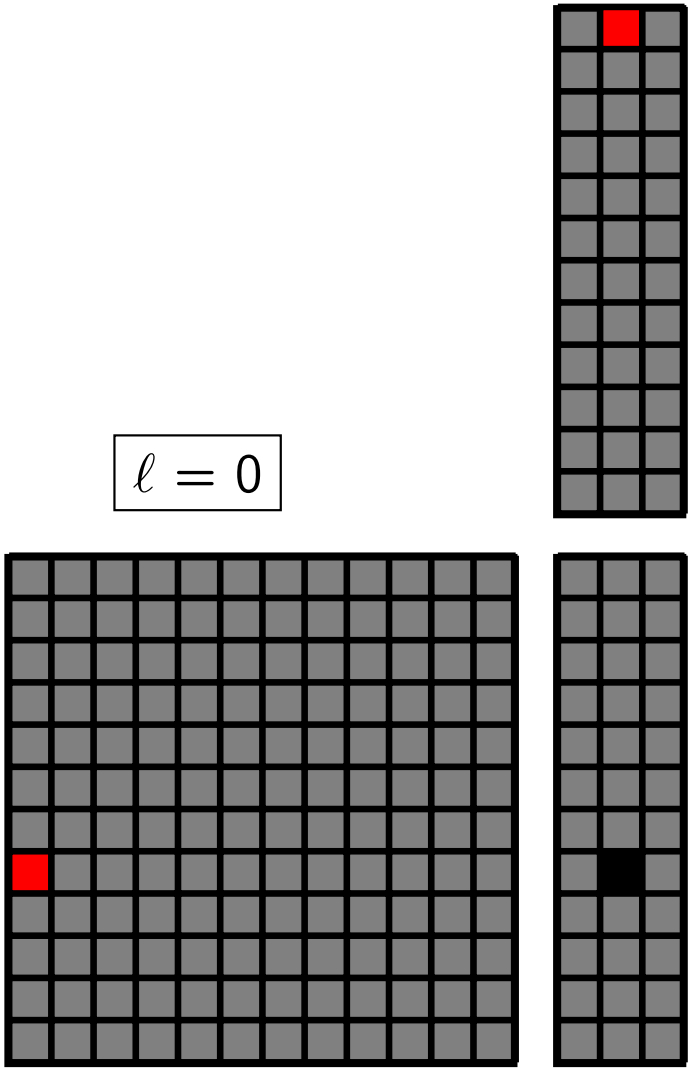
- Computing $C = AB$

GEMM (GEneral Matrix-Matrix multiplication)



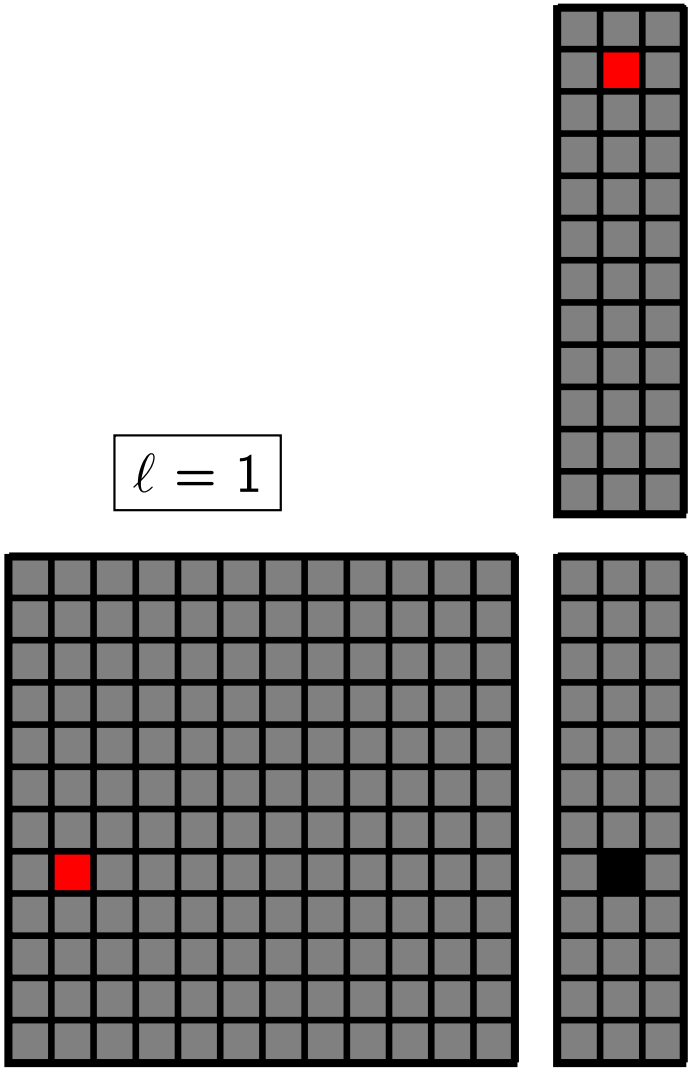
- Computing $C = AB$
- $C_{i,j} = \sum_{\ell=1}^{m} A_{i,\ell} B_{\ell,j}$

GEMM (GEneral Matrix-Matrix multiplication)



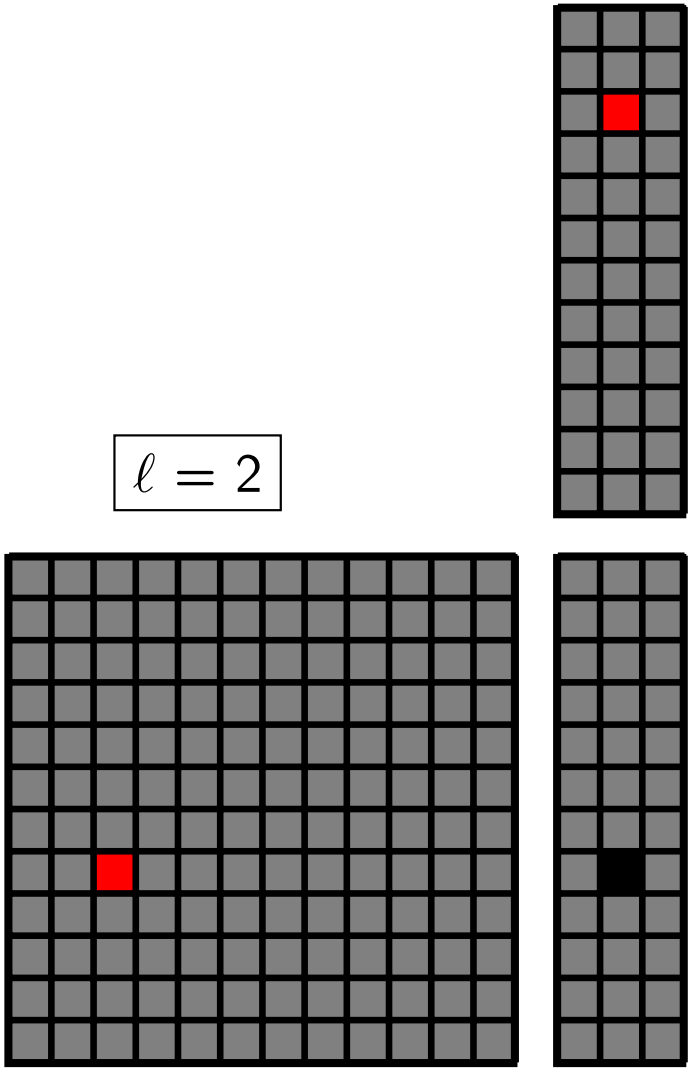
- Computing $C = AB$
- $C_{7,1} = \qquad A_{7,0}B_{0,1} + \dots$

GEMM (GEneral Matrix-Matrix multiplication)



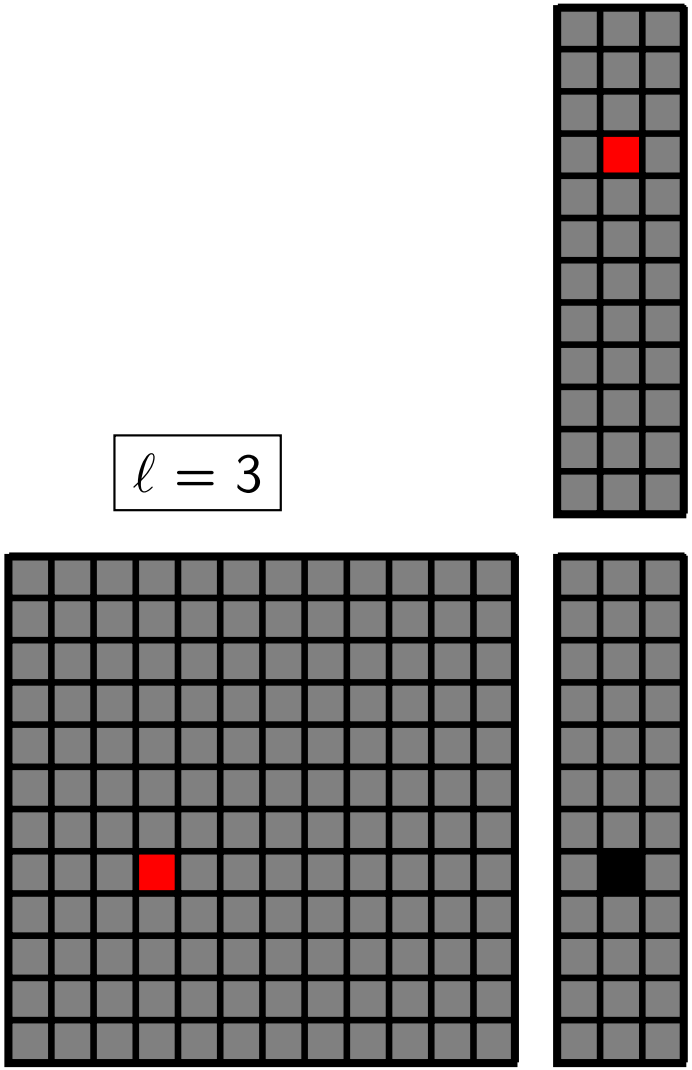
- Computing $C = AB$
- $C_{7,1} = \ldots + A_{7,1}B_{1,1} + \ldots$

GEMM (GEneral Matrix-Matrix multiplication)



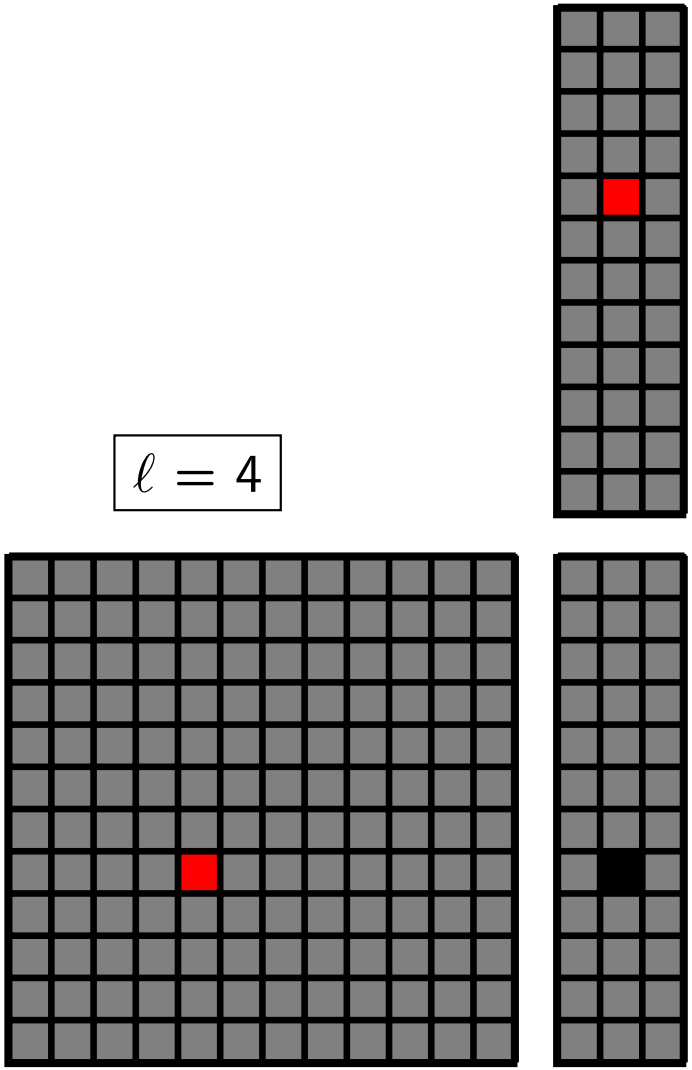
- Computing $C = AB$
- $C_{7,1} = \ldots + A_{7,2}B_{2,1} + \ldots$

GEMM (GEneral Matrix-Matrix multiplication)



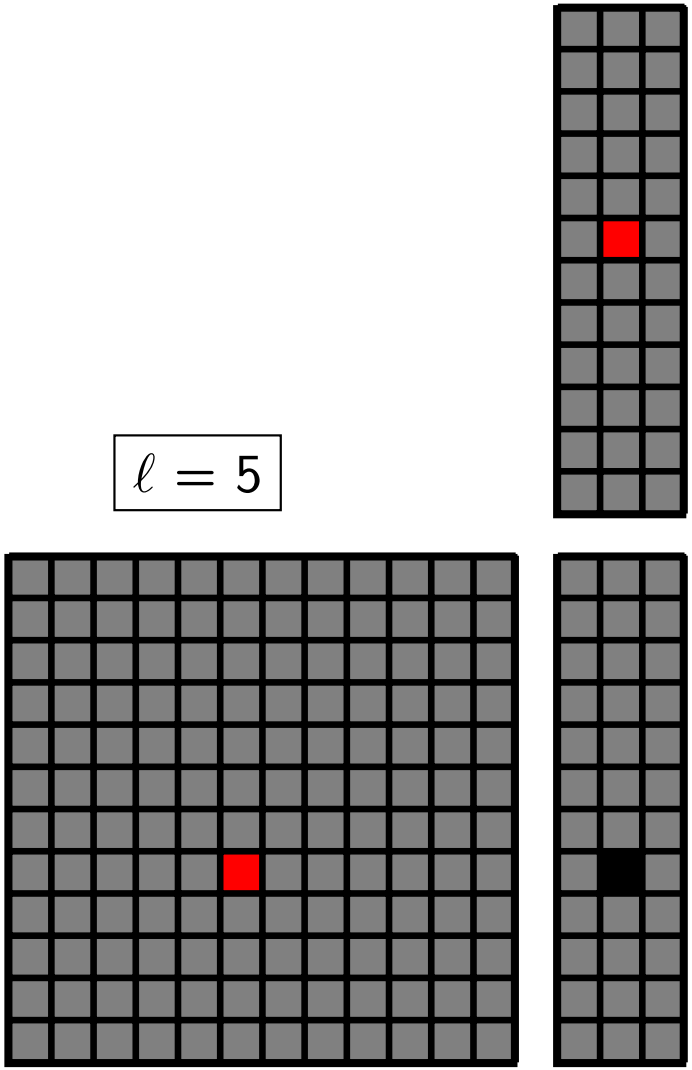
- Computing $C = AB$
- $C_{7,1} = \ldots + A_{7,3}B_{3,1} + \ldots$

GEMM (GEneral Matrix-Matrix multiplication)



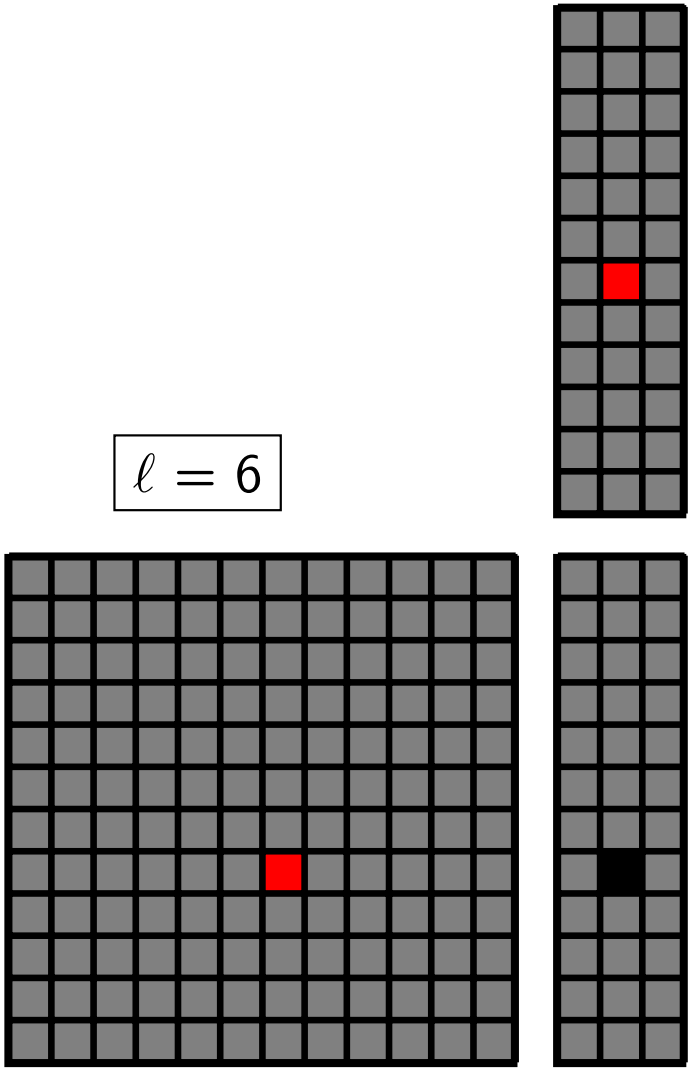
- Computing $C = AB$
- $C_{7,1} = \ldots + A_{7,4}B_{4,1} + \ldots$

GEMM (GEneral Matrix-Matrix multiplication)



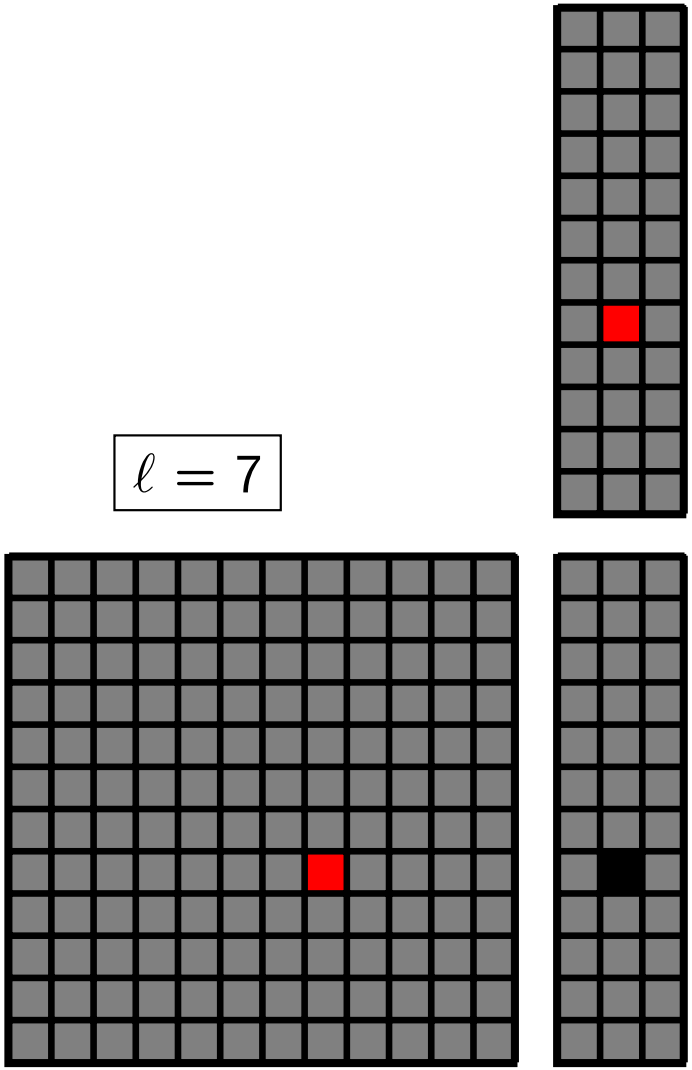
- Computing $C = AB$
- $C_{7,1} = \ldots + A_{7,5}B_{5,1} + \ldots$

GEMM (GEneral Matrix-Matrix multiplication)



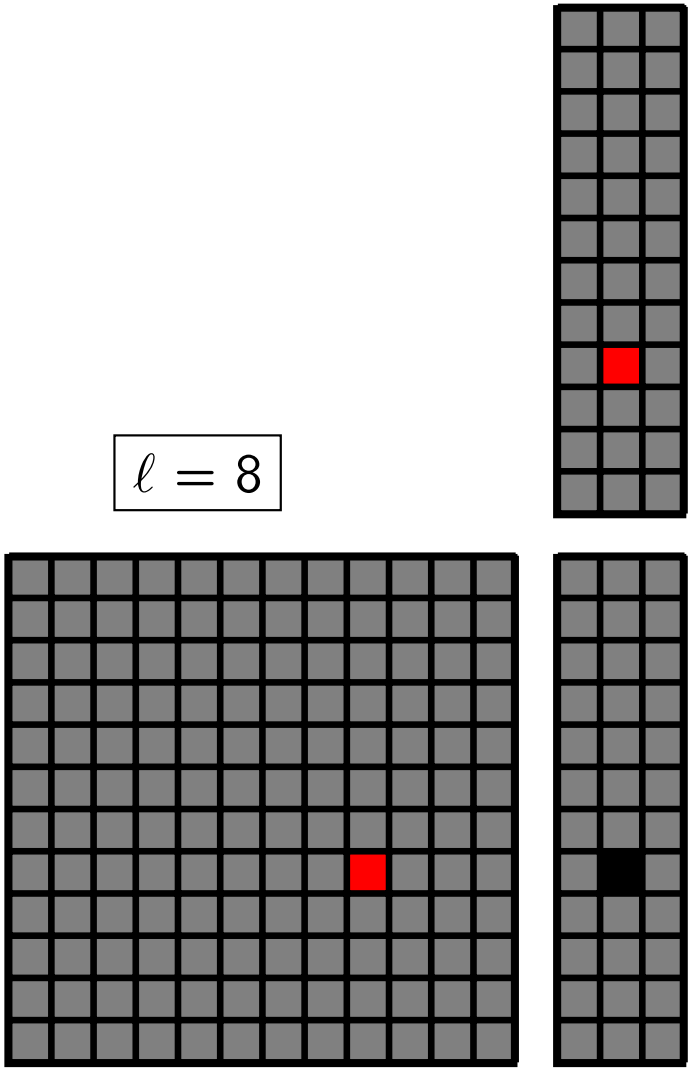
- Computing $C = AB$
- $C_{7,1} = \ldots + A_{7,6}B_{6,1} + \ldots$

GEMM (GEneral Matrix-Matrix multiplication)



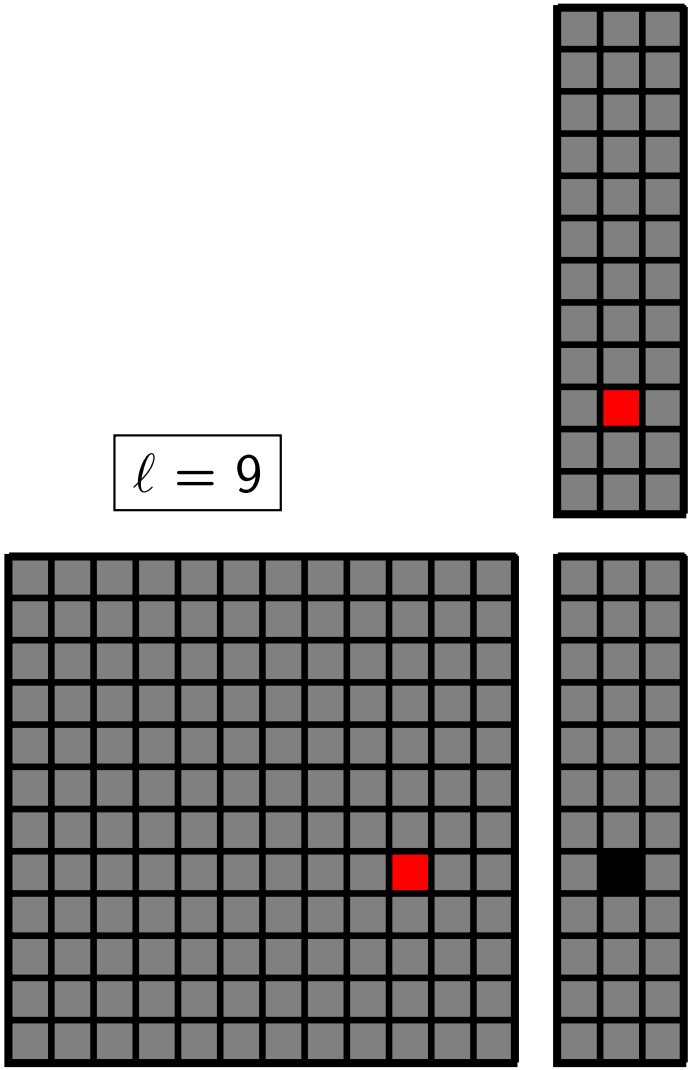
- Computing $C = AB$
- $C_{7,1} = \ldots + A_{7,7}B_{7,1} + \ldots$

GEMM (GEneral Matrix-Matrix multiplication)



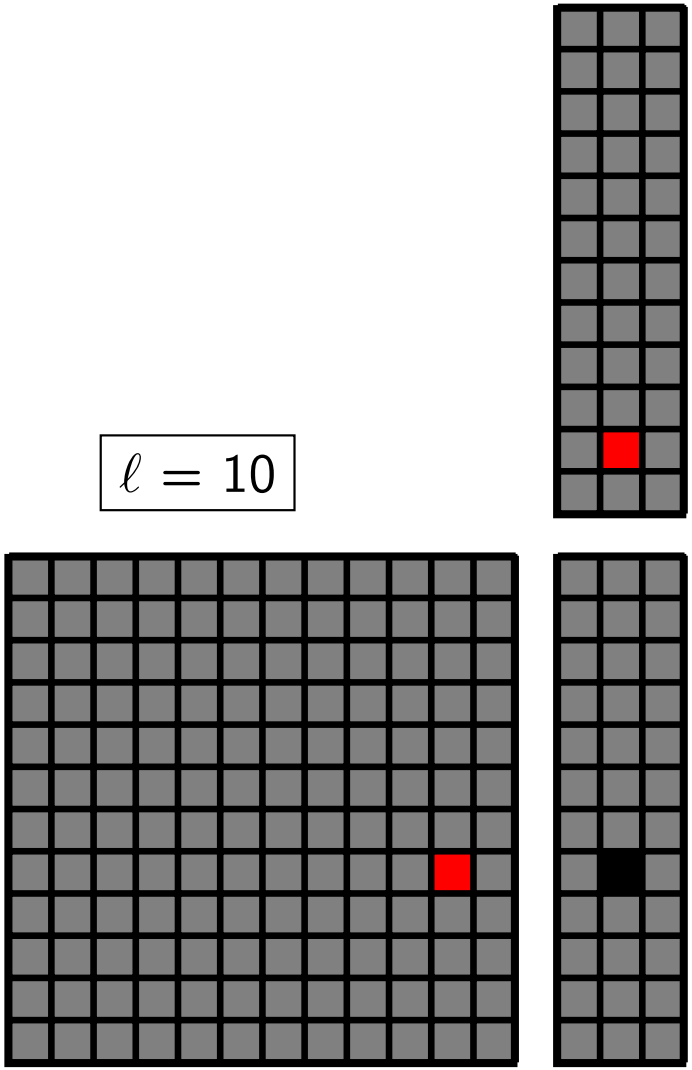
- Computing $C = AB$
- $C_{7,1} = \ldots + A_{7,8} B_{8,1} + \ldots$

GEMM (GEneral Matrix-Matrix multiplication)



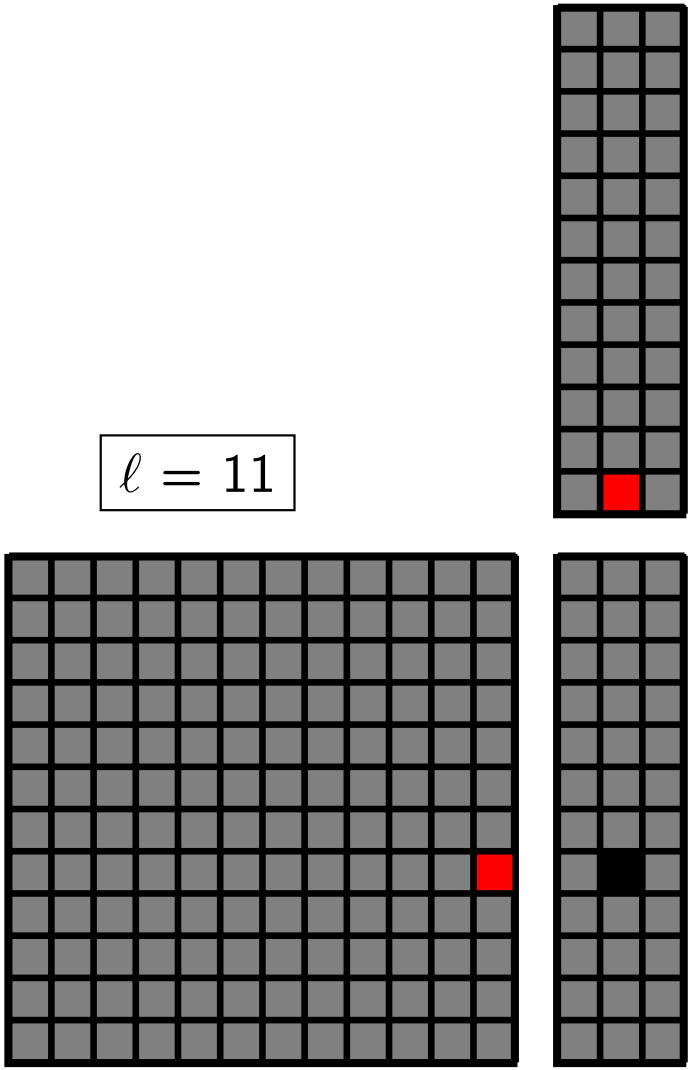
- Computing $C = AB$
- $C_{7,1} = \ldots + A_{7,9}B_{9,1} + \ldots$

GEMM (GEneral Matrix-Matrix multiplication)



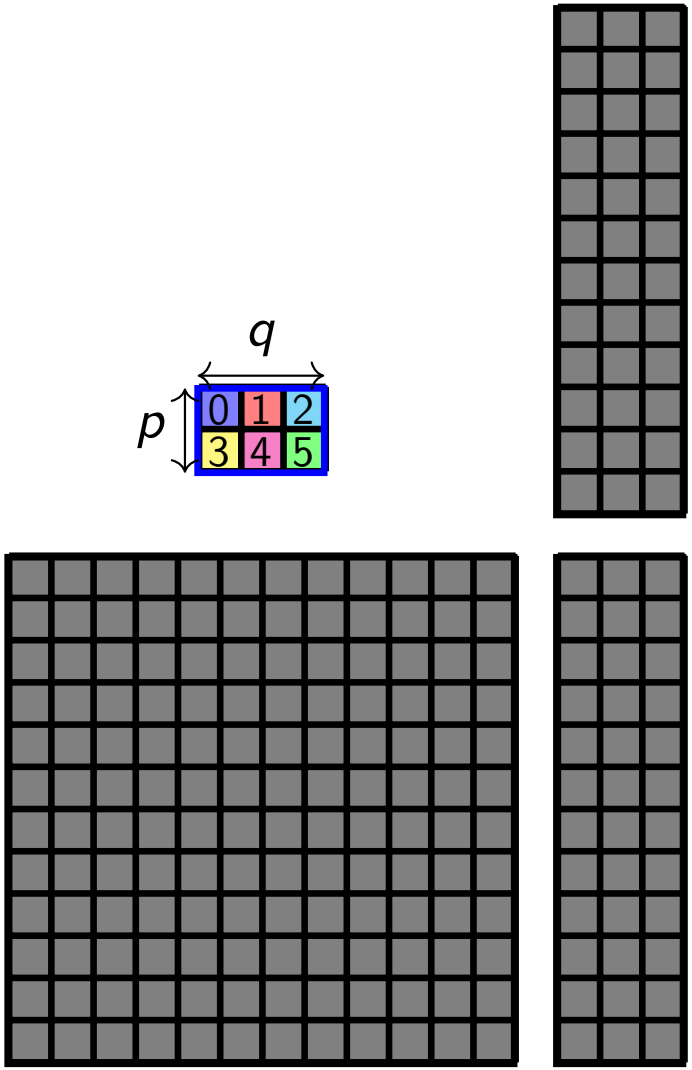
- Computing $C = AB$
- $C_{7,1} = \ldots + A_{7,10} B_{10,1} + \ldots$

GEMM (GEneral Matrix-Matrix multiplication)



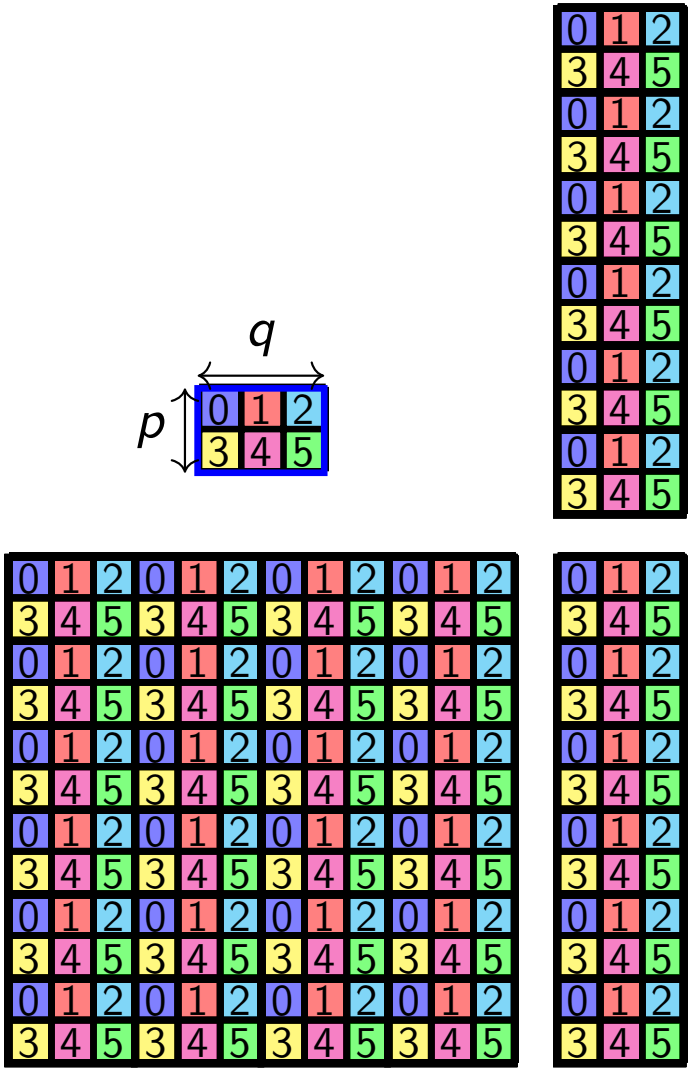
- Computing $C = AB$
- $C_{7,1} = \ldots + A_{7,11} B_{11,1}$

GEMM (GEneral Matrix-Matrix multiplication)



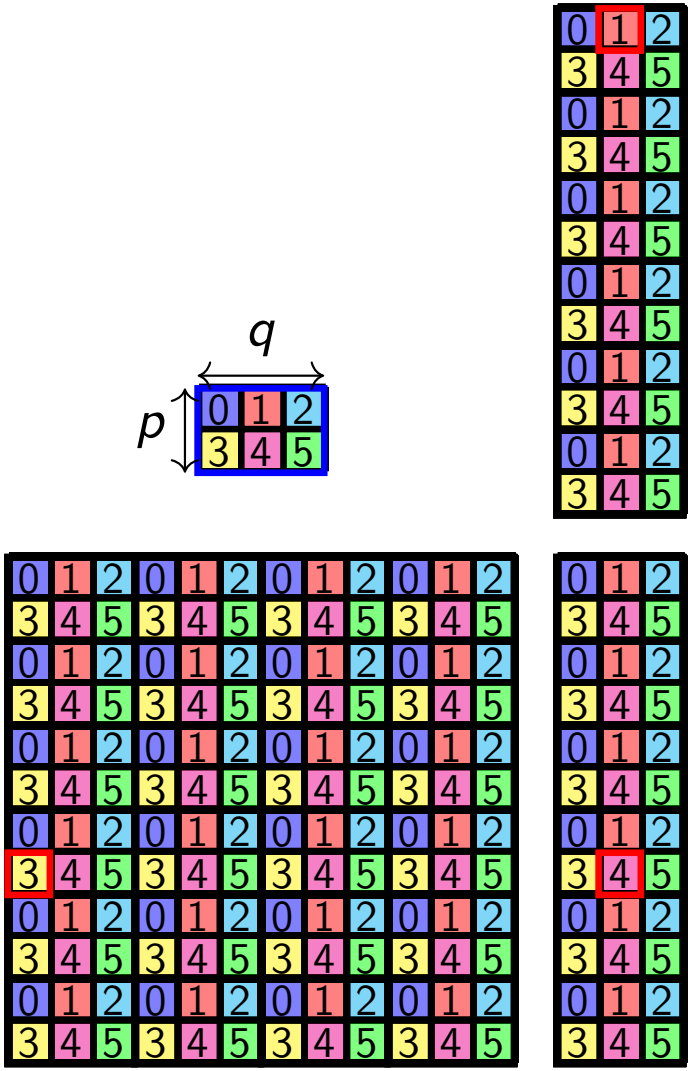
- Computing $C = AB$
- With P nodes : 2D Block-Cyclic scheme Blackford et al., 1997; van de Geijn and Watts, 1997

GEMM (GEneral Matrix-Matrix multiplication)



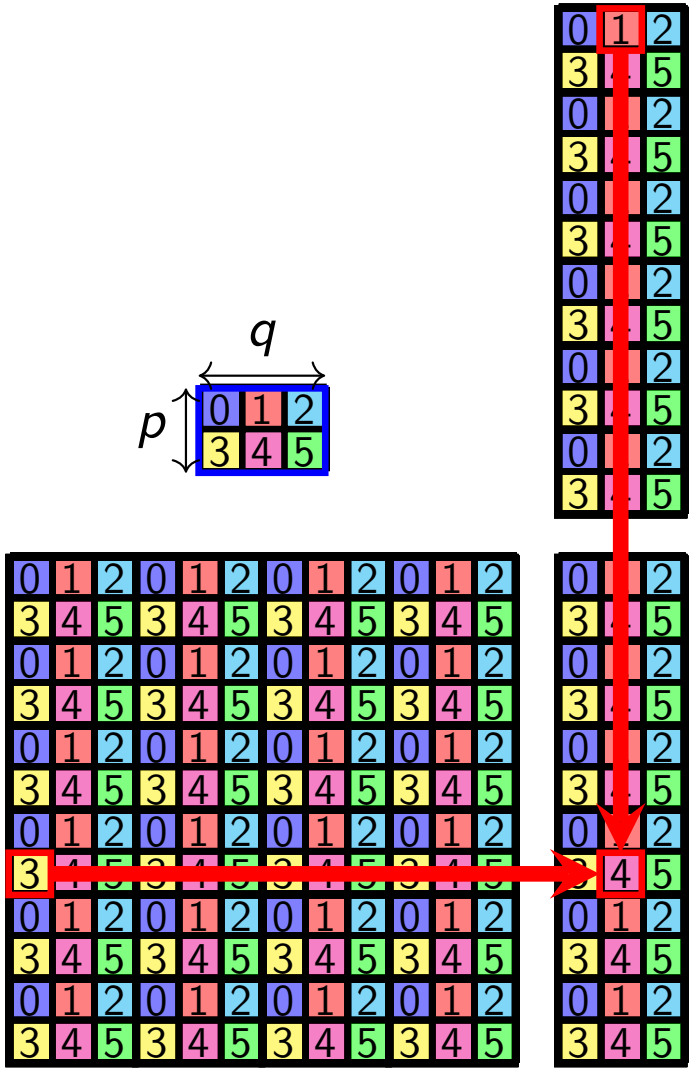
- Computing $C = AB$
- With P nodes : 2D Block-Cyclic scheme Blackford et al., 1997; van de Geijn and Watts, 1997

GEMM (GEneral Matrix-Matrix multiplication)



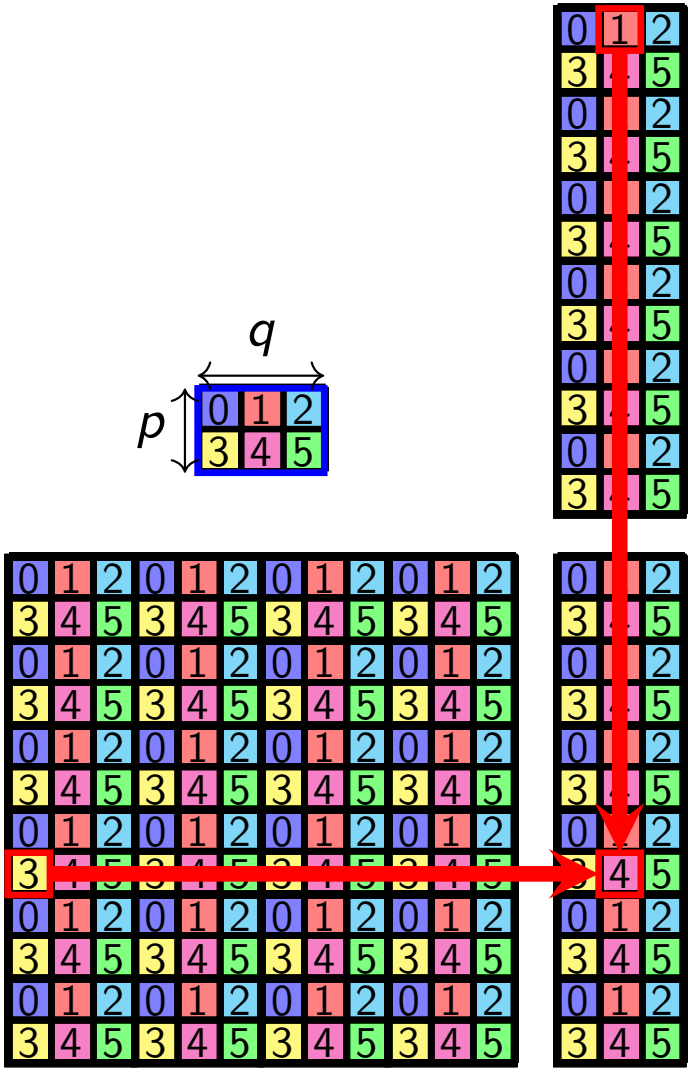
- Computing $C = AB$
- To compute one tile we need to transfer all the data to the corresponding node

GEMM (GEneral Matrix-Matrix multiplication)



- Computing $C = AB$
- To compute one tile we need to transfer all the data to the corresponding node
- Moving the data to the node that posseses the tile of C is called a C-Stationary scheme

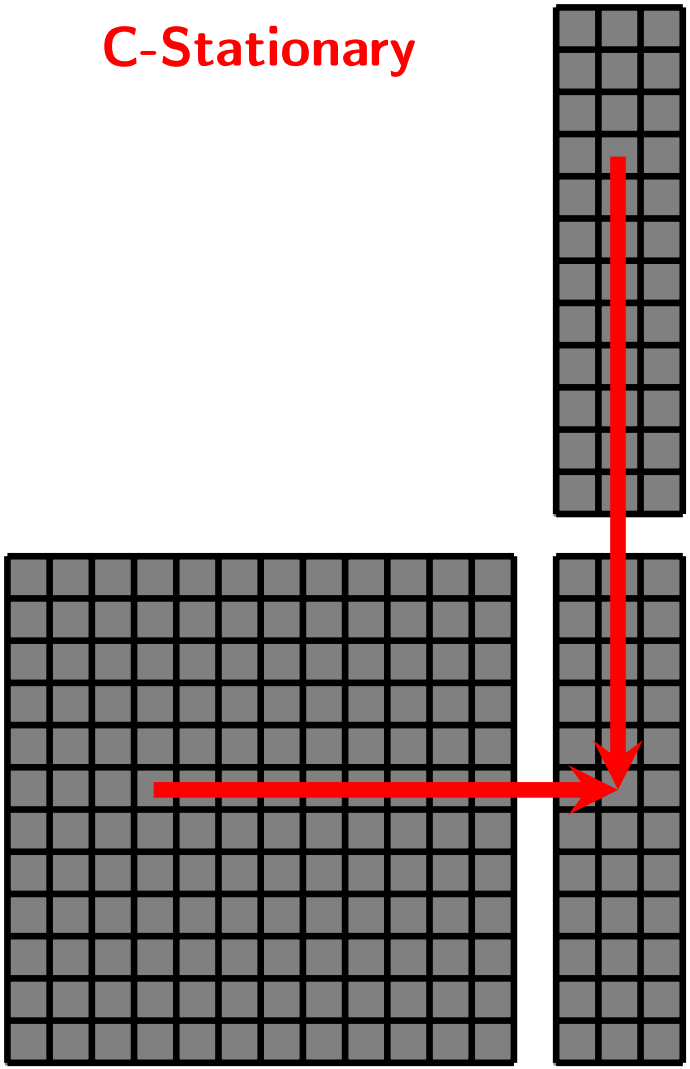
GEMM (GEneral Matrix-Matrix multiplication)



- Computing $C = AB$
- To compute one tile we need to transfer all the data to the corresponding node
- Moving the data to the node that posseses the tile of C is called a C-Stationary scheme

Method	Comm. Volume
GEMM C-Stat	$qm^2 + pmk$

A-stationary GEMM



Standard : C-Stationary

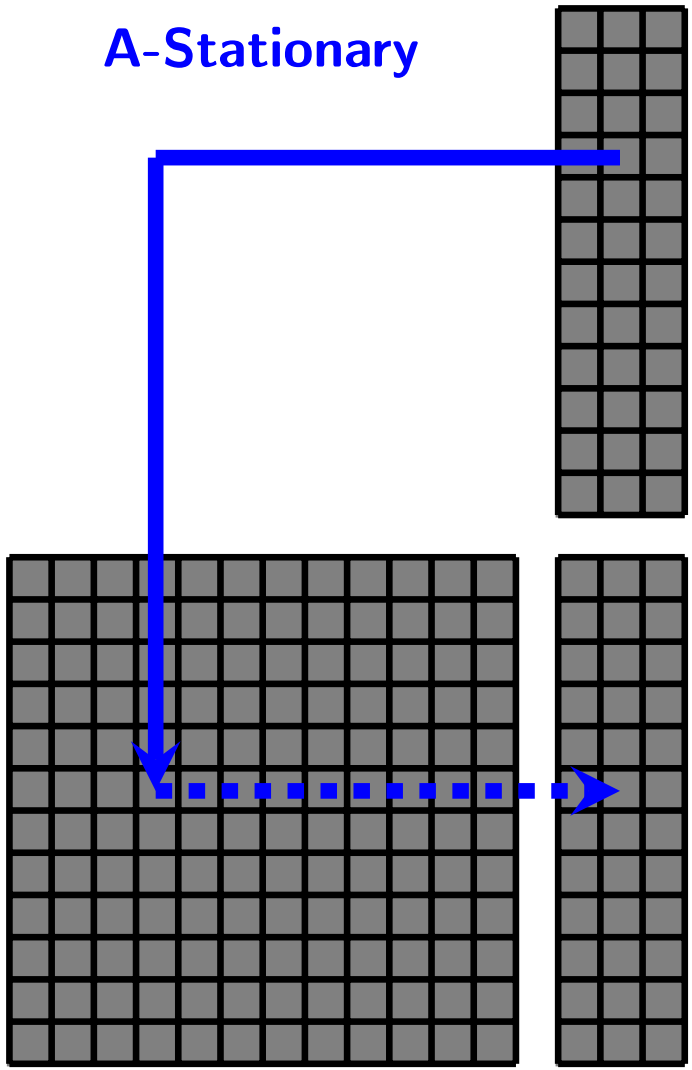
We move the blocks of A and B to be computed on C

Aim : A-Stationary

We move the blocks of B and C to be computed on A (Agullo et al., 2023)

Method	Comm. Volume
GEMM C-Stat	$qm^2 + pmk$

A-stationary GEMM



Standard : C-Stationary

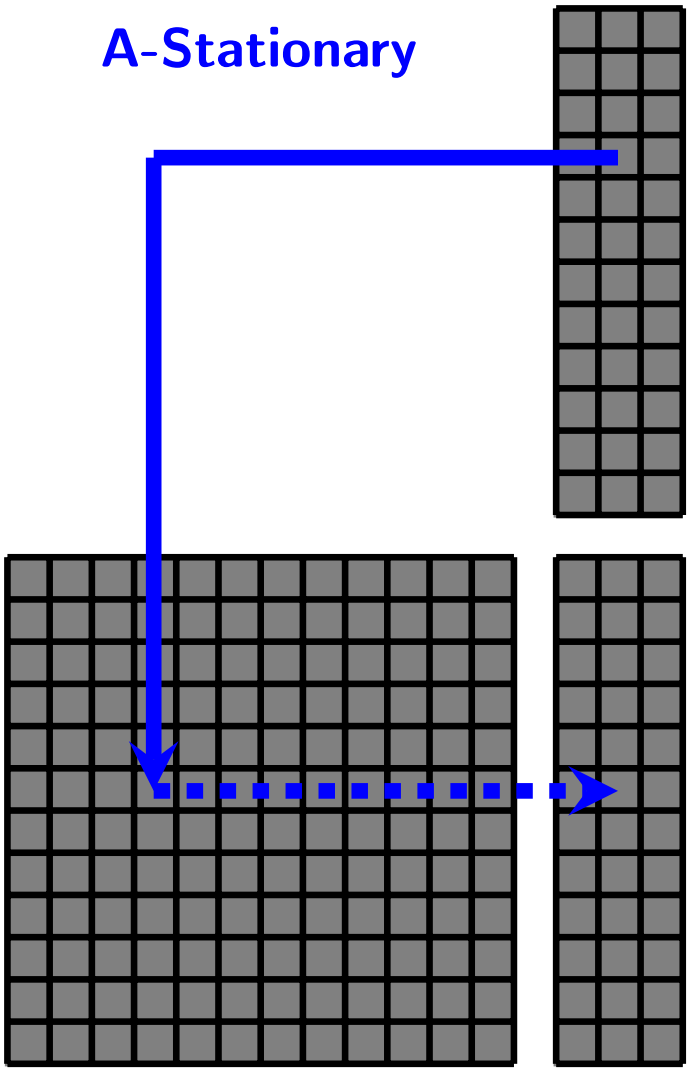
We move the blocks of A and B to be computed on C

Aim : A-Stationary

We move the blocks of B and C to be computed on A (Agullo et al., 2023)

Method	Comm. Volume
GEMM C-Stat	$qm^2 + pmk$

A-stationary GEMM



Standard : C-Stationary

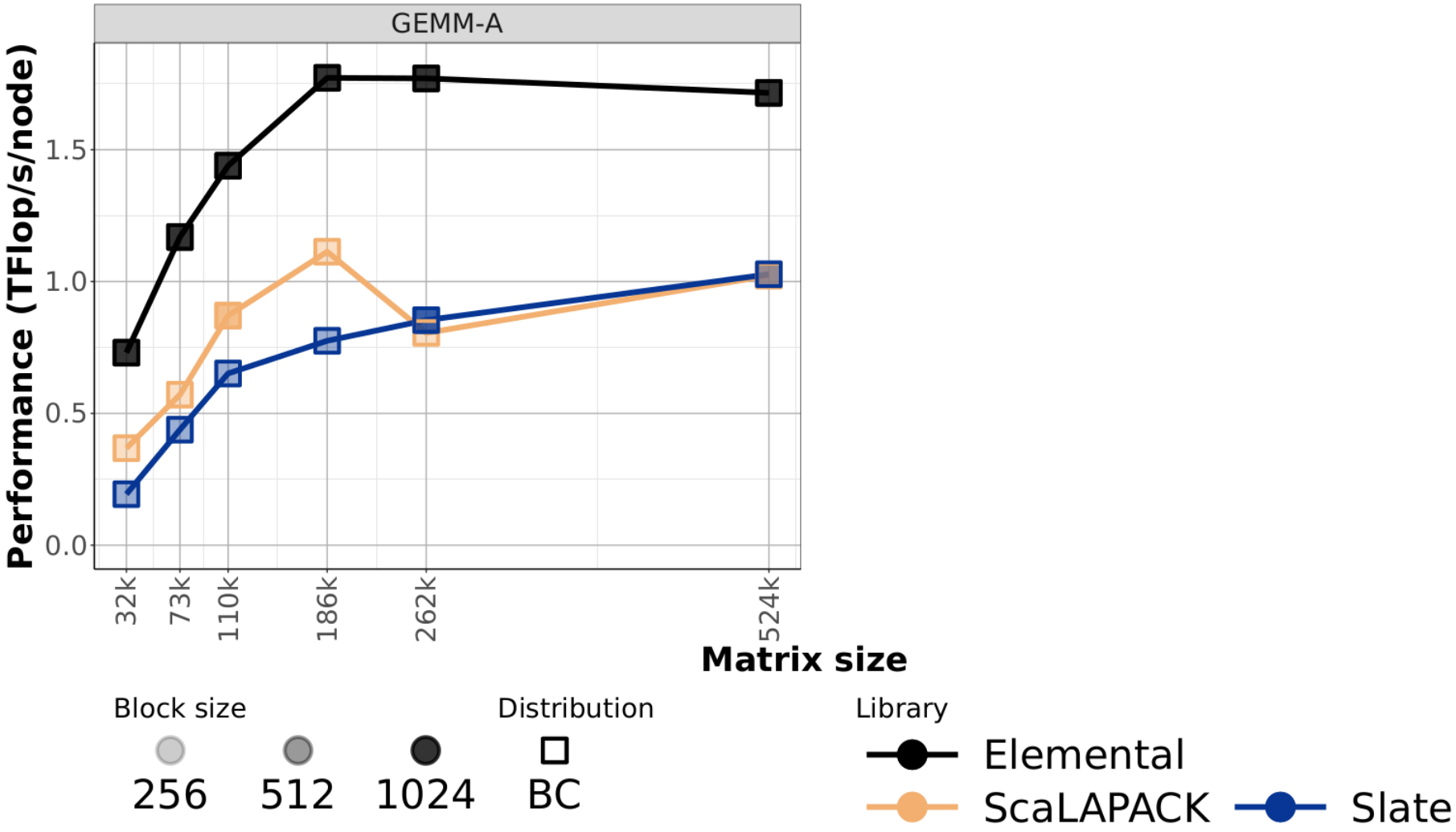
We move the blocks of A and B to be computed on C

Aim : A-Stationary

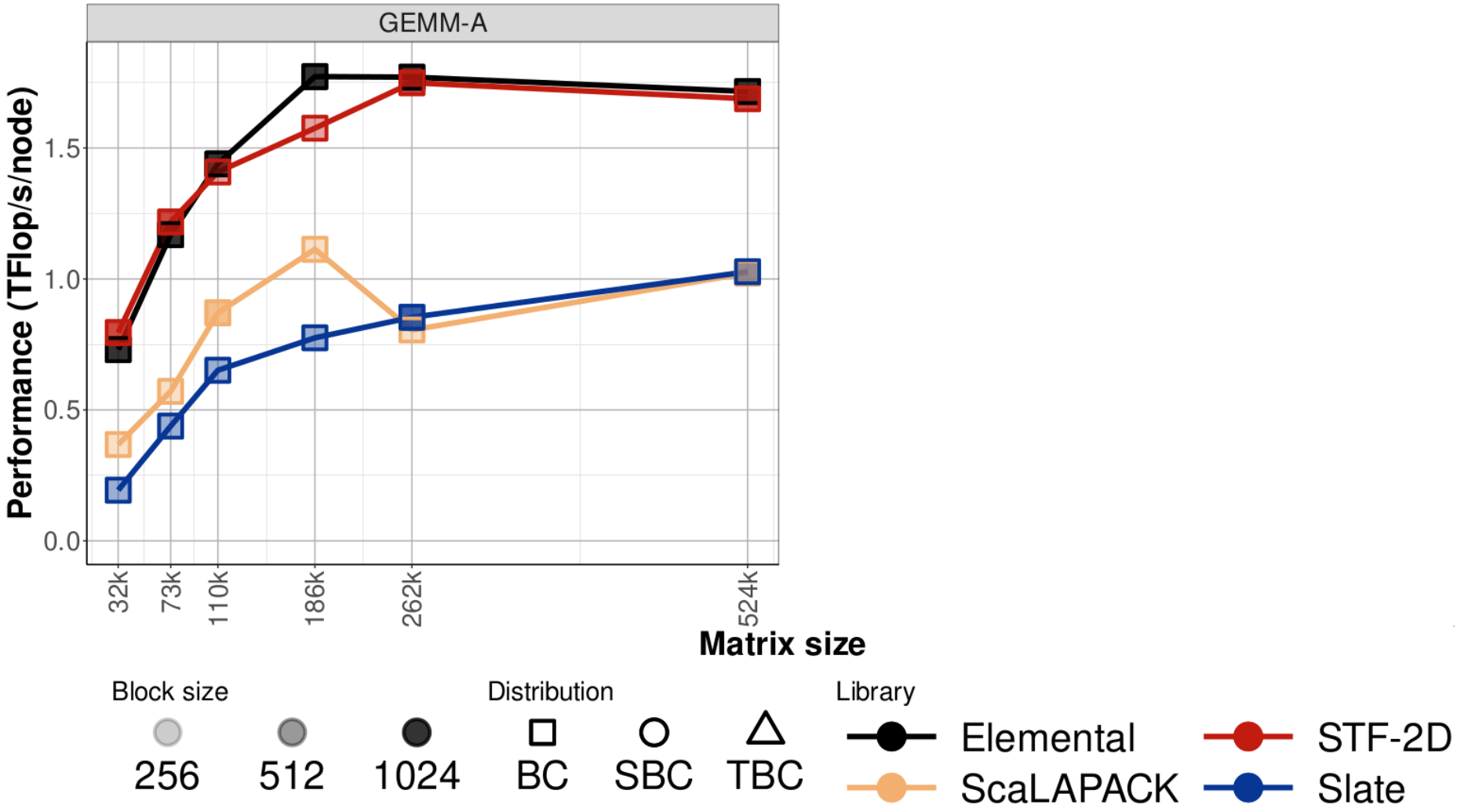
We move the blocks of B and C to be computed on A (Agullo et al., 2023)

Method	Comm. Volume
GEMM C-Stat	$qm^2 + pmk$
GEMM A-Stat	$\approx 2\sqrt{P}mk$

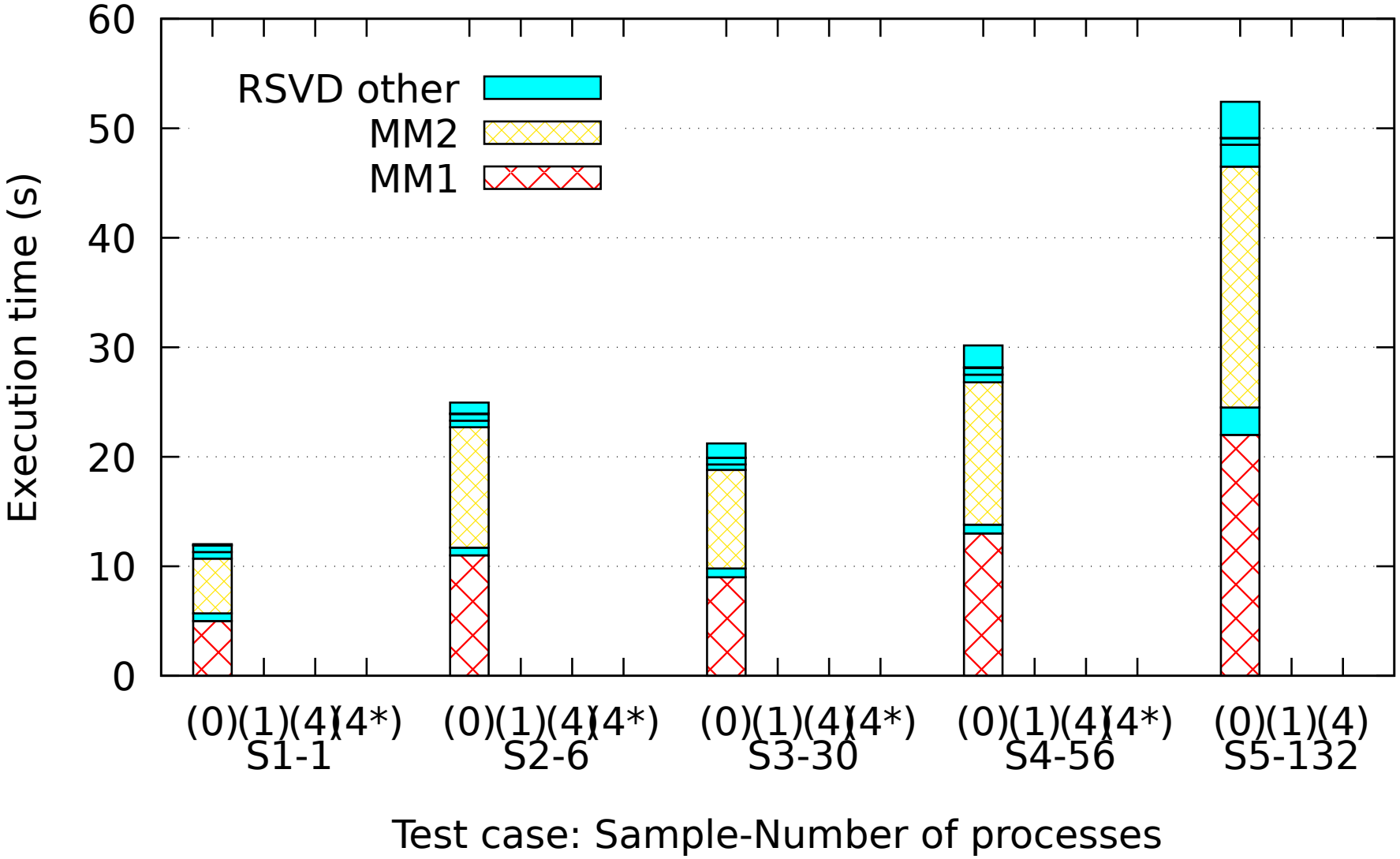
Performance of A-Stationary GEMM



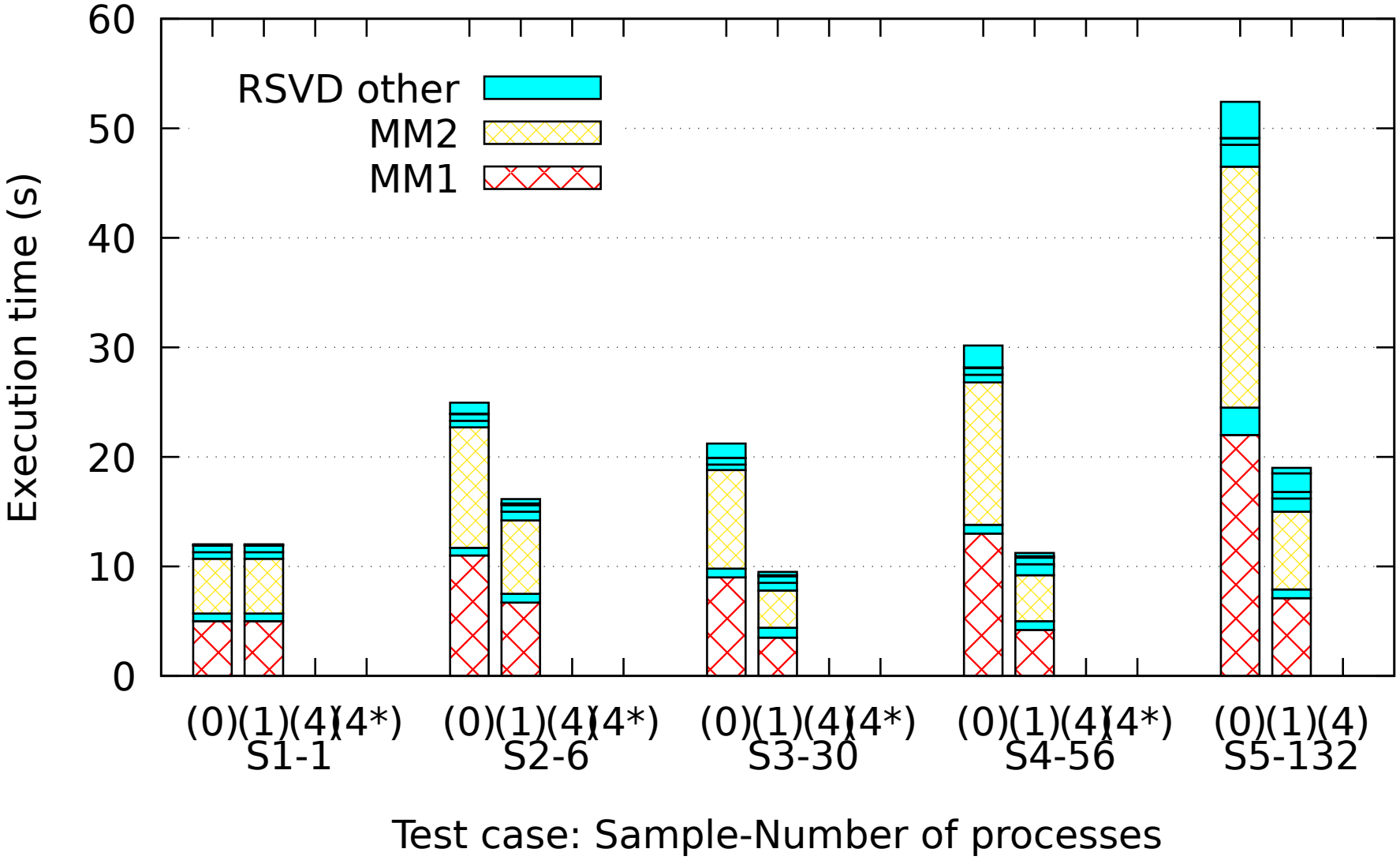
Performance of A-Stationary GEMM



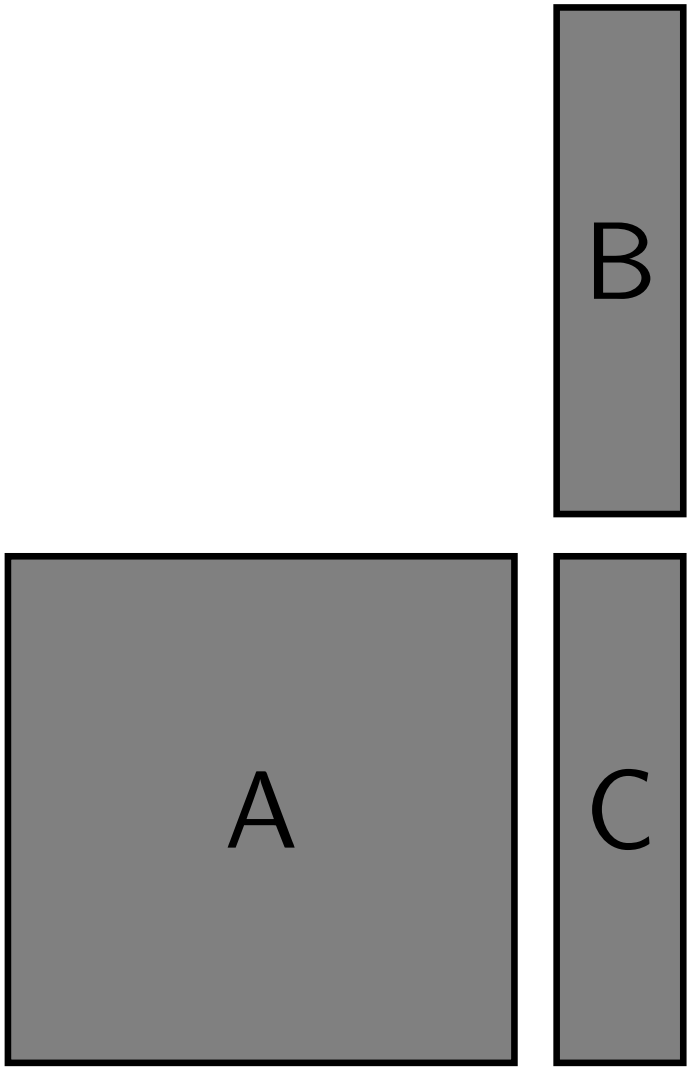
Impact of A-Stationary GEMM for RSVD-MDS



Impact of A-Stationary GEMM for RSVD-MDS

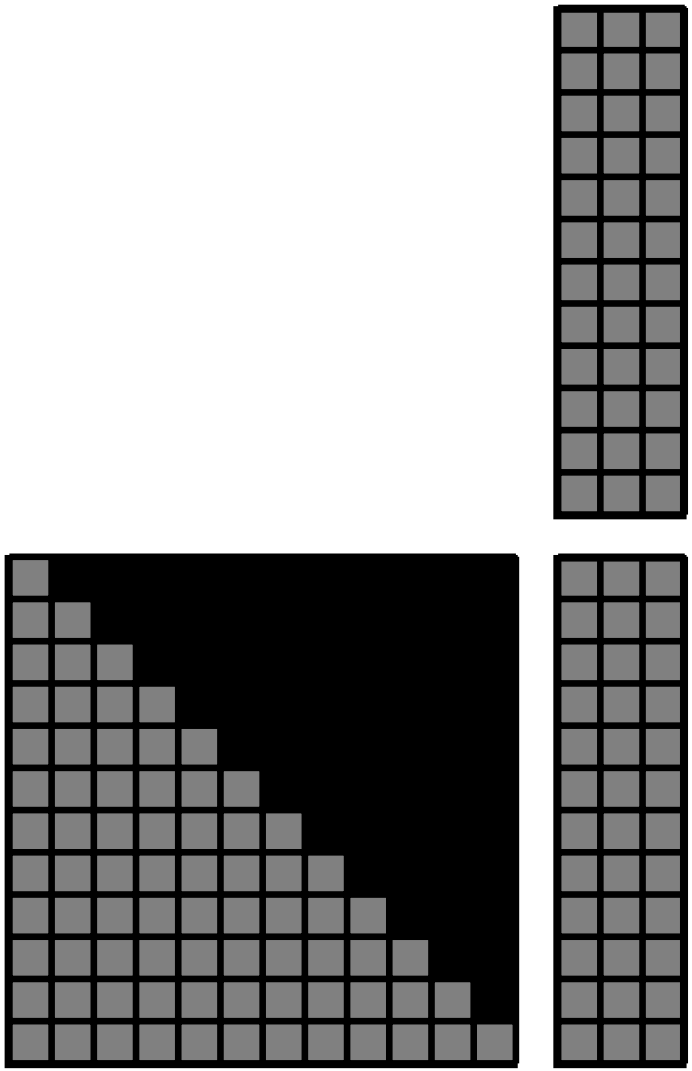


SYMM (SYmmetric Matrix-Matrix multiplication)



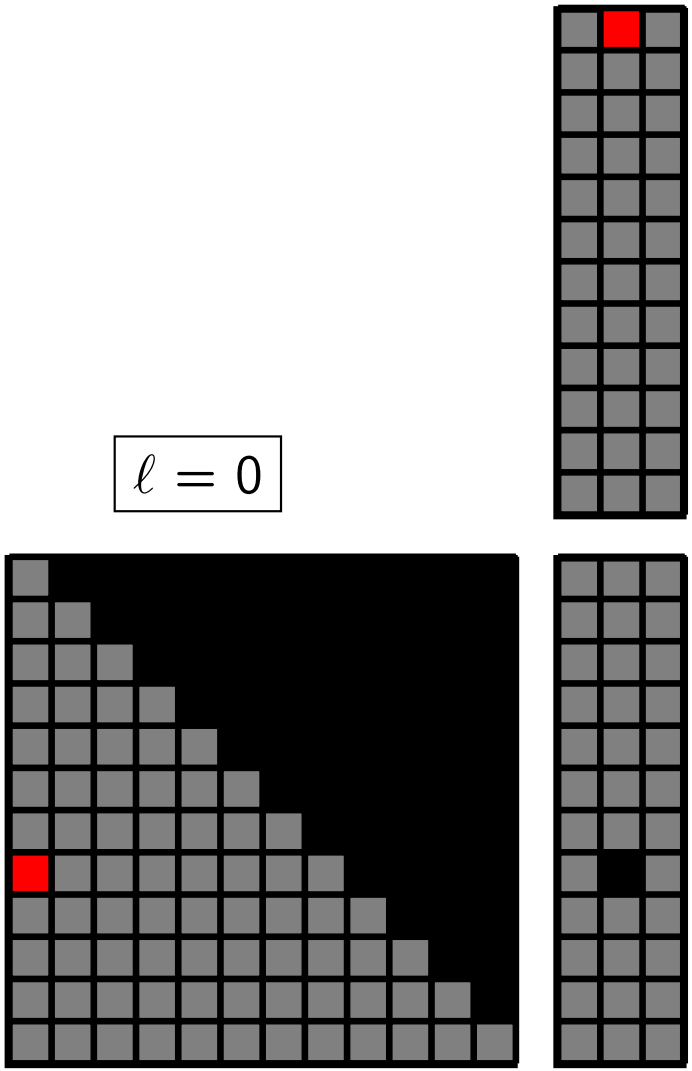
- Computing $C = AB$

SYMM (SYmmetric Matrix-Matrix multiplication)



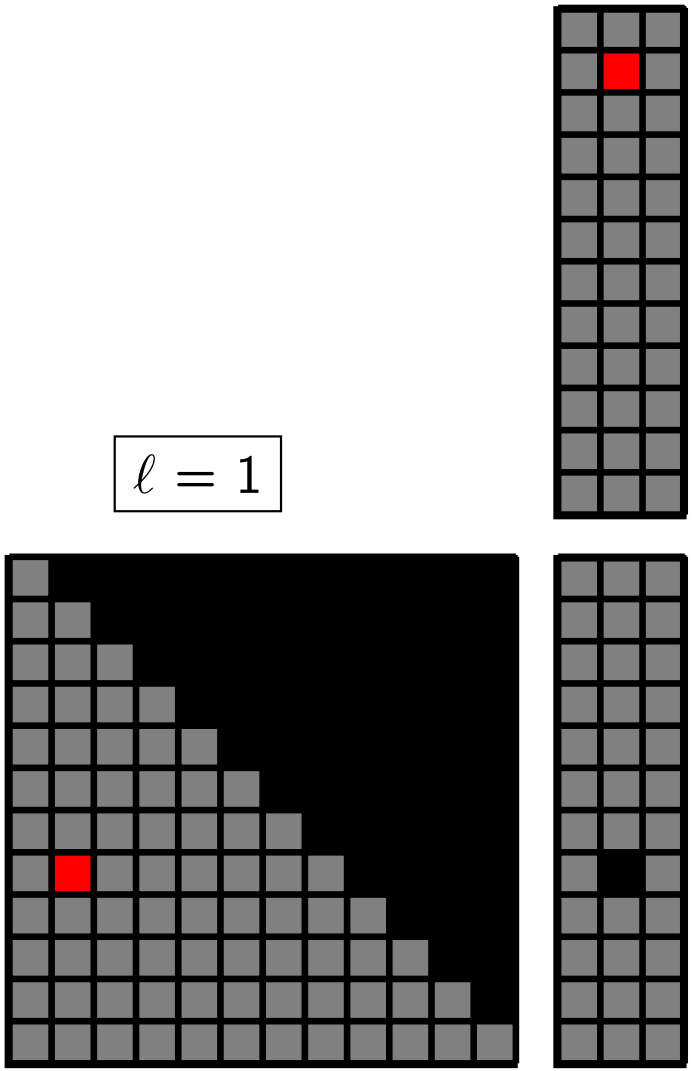
- Computing $C = AB$
- $C_{i,j} = \sum_{\ell=1}^{i} A_{i,\ell} B_{\ell,j} + \sum_{\ell=i+1}^{m} A_{\ell,i}^{T} B_{\ell,j}$

SYMM (SYmmetric Matrix-Matrix multiplication)



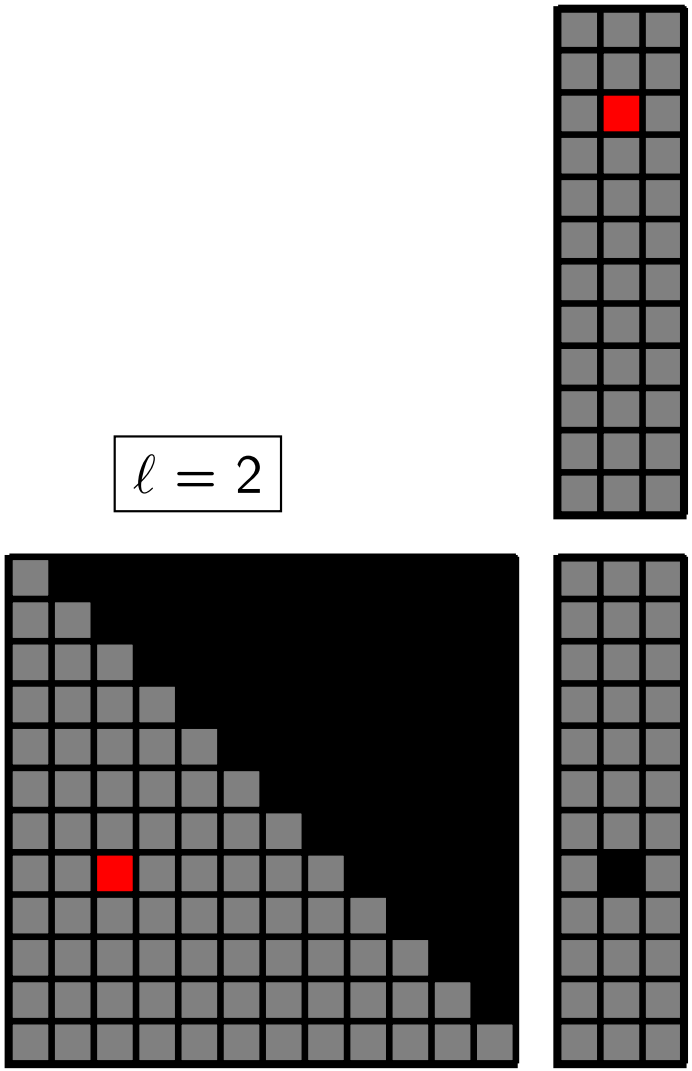
- Computing $C = AB$
- $C_{i,j} = \sum_{\ell=1}^{i} A_{i,\ell} B_{\ell,j} + \sum_{\ell=i+1}^{m} A^T_{\ell,i} B_{\ell,j}$
- $C_{7,1} = \qquad A_{7,0} B_{0,1} + \ldots$

SYMM (SYmmetric Matrix-Matrix multiplication)



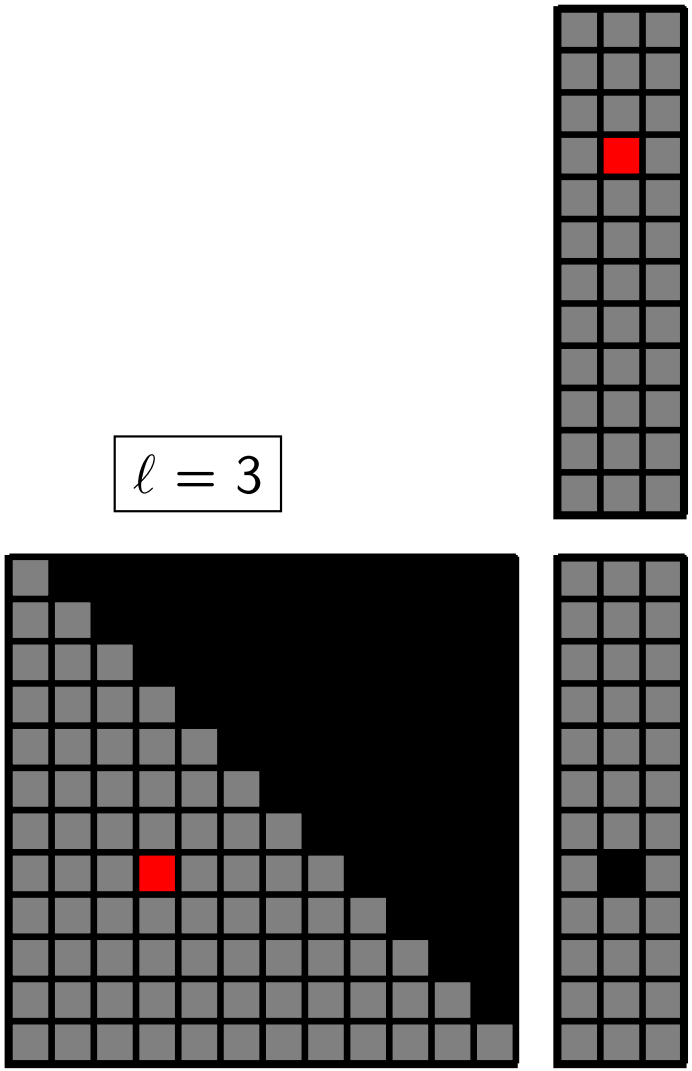
- Computing $C = AB$
- $C_{i,j} = \sum_{\ell=1}^{i} A_{i,\ell} B_{\ell,j} + \sum_{\ell=i+1}^{m} A^T_{\ell,i} B_{\ell,j}$
- $C_{7,1} = \ldots + A_{7,1} B_{1,1} + \ldots$

SYMM (SYmmetric Matrix-Matrix multiplication)



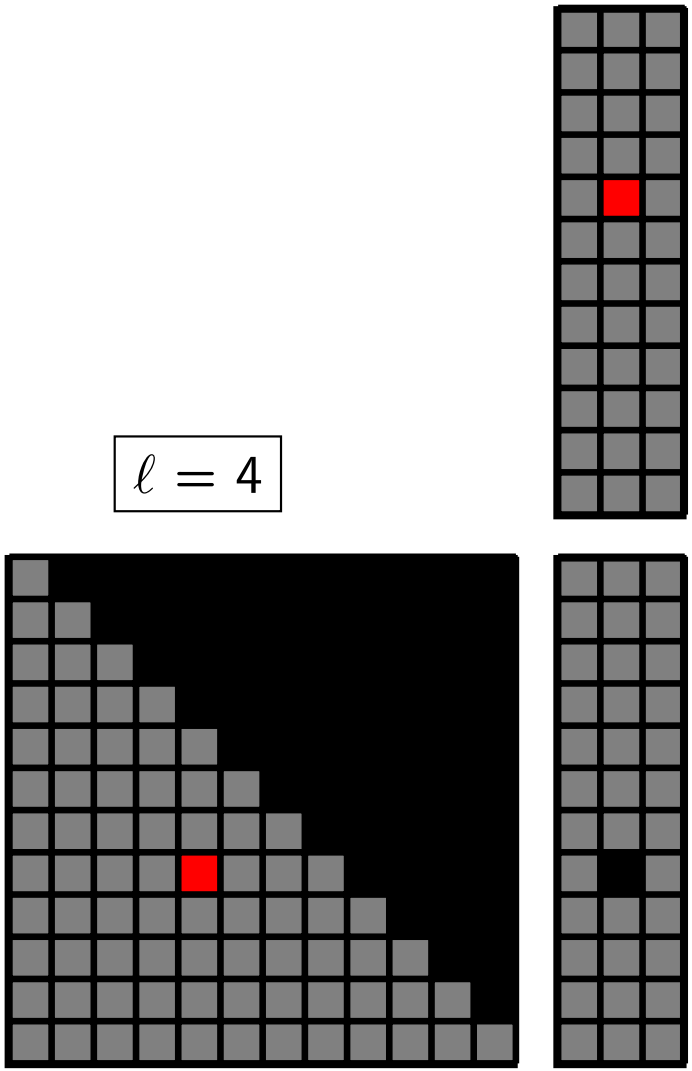
- Computing $C = AB$
- $C_{i,j} = \sum_{\ell=1}^{i} A_{i,\ell} B_{\ell,j} + \sum_{\ell=i+1}^{m} A^T_{\ell,i} B_{\ell,j}$
- $C_{7,1} = \ldots + A_{7,2} B_{2,1} + \ldots$

SYMM (SYmmetric Matrix-Matrix multiplication)



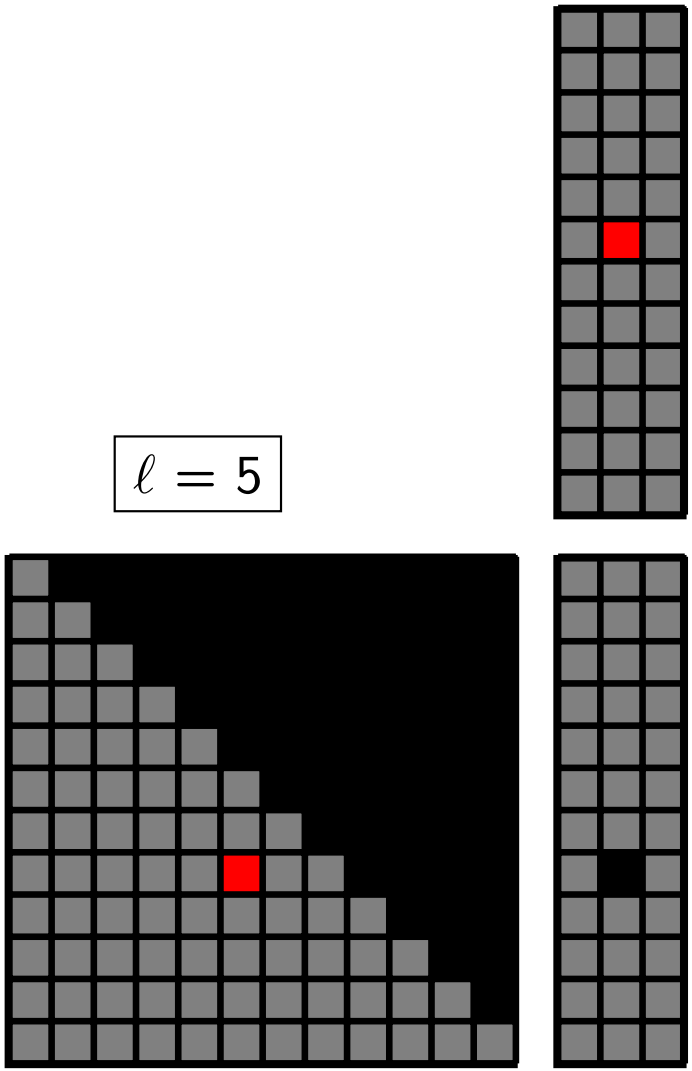
- Computing $C = AB$
- $C_{i,j} = \sum_{\ell=1}^{i} A_{i,\ell} B_{\ell,j} + \sum_{\ell=i+1}^{m} A^T_{\ell,i} B_{\ell,j}$
- $C_{7,1} = \ldots + A_{7,3} B_{3,1} + \ldots$

SYMM (SYmmetric Matrix-Matrix multiplication)



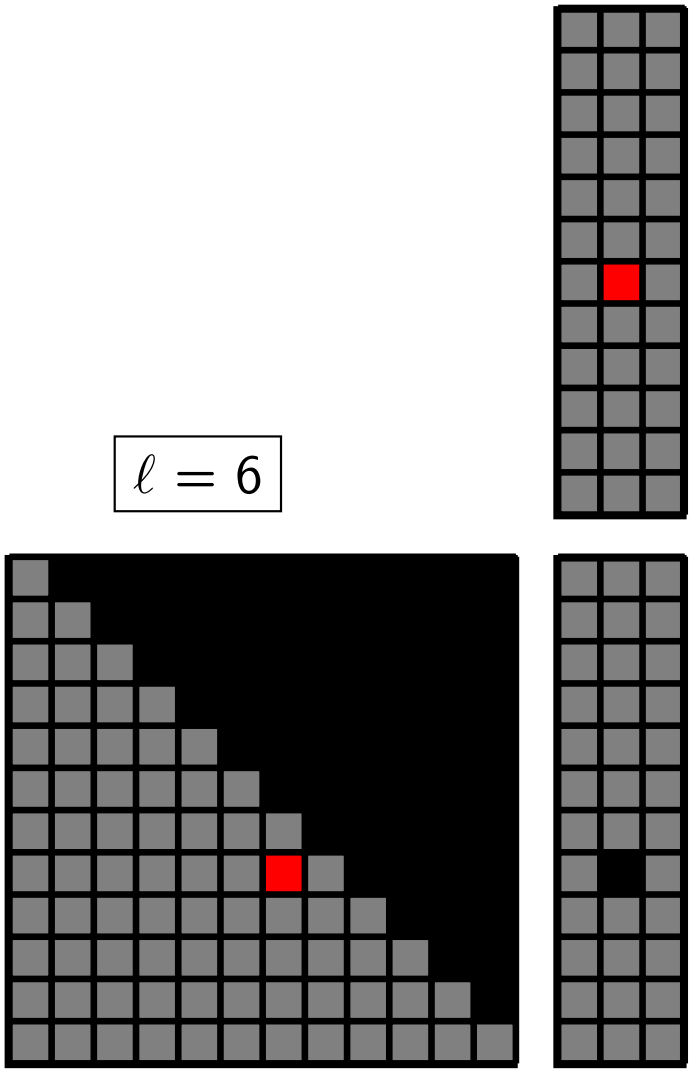
- Computing $C = AB$
- $C_{i,j} = \sum_{\ell=1}^{i} A_{i,\ell} B_{\ell,j} + \sum_{\ell=i+1}^{m} A^T_{\ell,i} B_{\ell,j}$
- $C_{7,1} = \ldots + A_{7,4} B_{4,1} + \ldots$

SYMM (SYmmetric Matrix-Matrix multiplication)



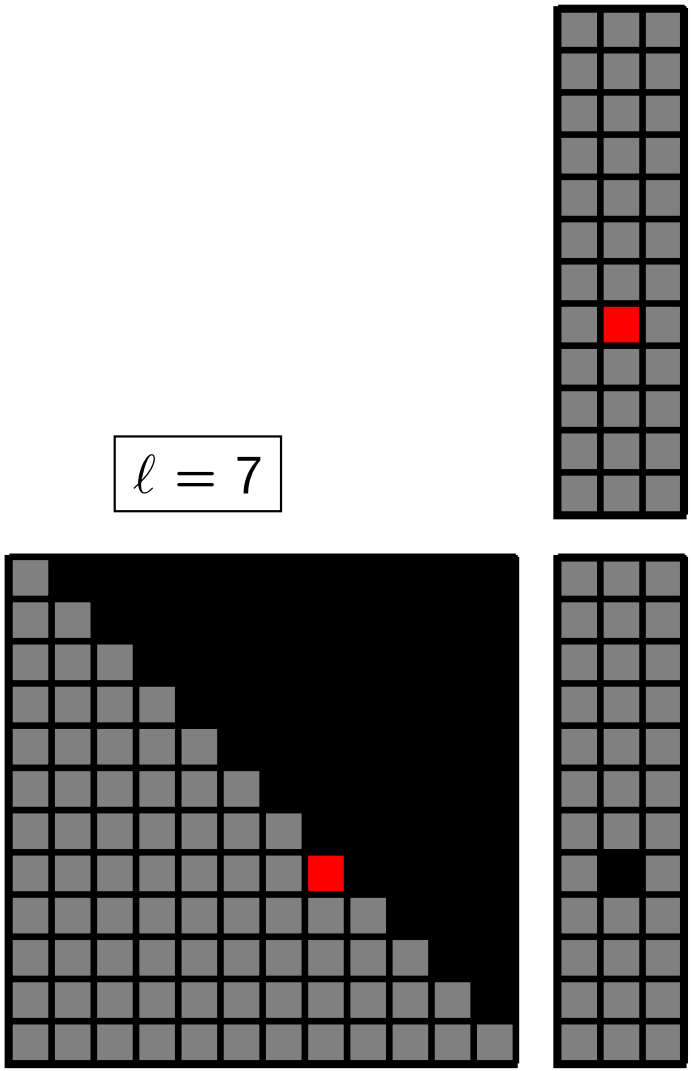
- Computing $C = AB$
- $C_{i,j} = \sum_{\ell=1}^{i} A_{i,\ell} B_{\ell,j} + \sum_{\ell=i+1}^{m} A^T_{\ell,i} B_{\ell,j}$
- $C_{7,1} = \ldots + A_{7,5} B_{5,1} + \ldots$

SYMM (SYmmetric Matrix-Matrix multiplication)



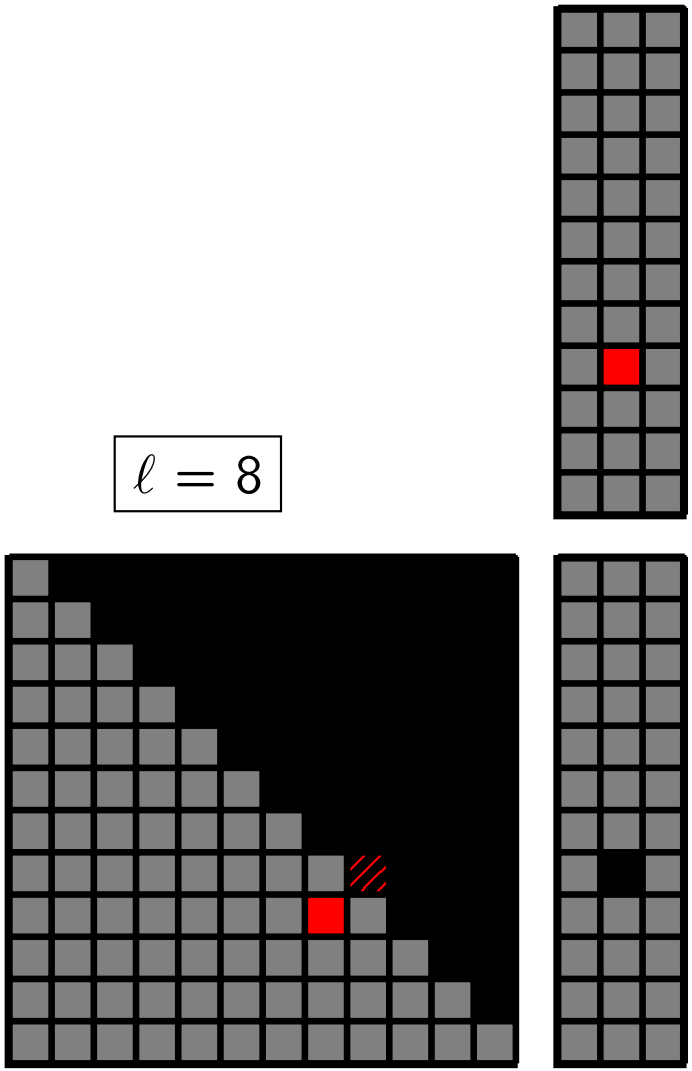
- Computing $C = AB$
- $C_{i,j} = \sum_{\ell=1}^{i} A_{i,\ell} B_{\ell,j} + \sum_{\ell=i+1}^{m} A^T_{\ell,i} B_{\ell,j}$
- $C_{7,1} = \ldots + A_{7,6} B_{6,1} + \ldots$

SYMM (SYmmetric Matrix-Matrix multiplication)



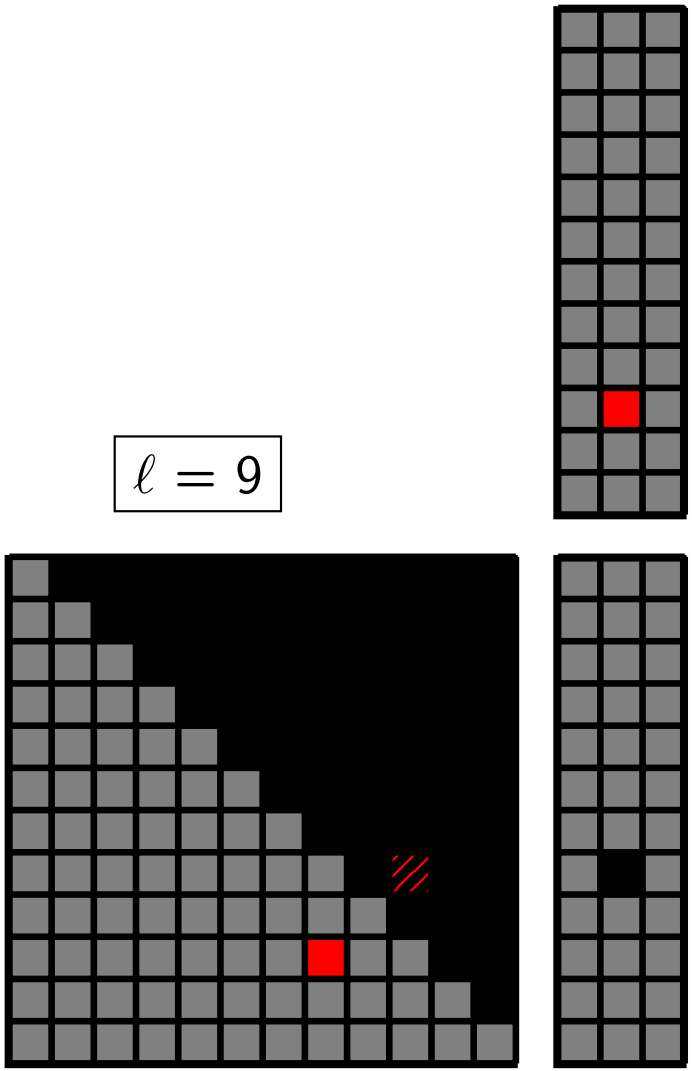
- Computing $C = AB$
- $C_{i,j} = \sum_{\ell=1}^{i} A_{i,\ell} B_{\ell,j} + \sum_{\ell=i+1}^{m} A^T_{\ell,i} B_{\ell,j}$
- $C_{7,1} = \ldots + A_{7,7} B_{7,1} + \ldots$

SYMM (SYmmetric Matrix-Matrix multiplication)



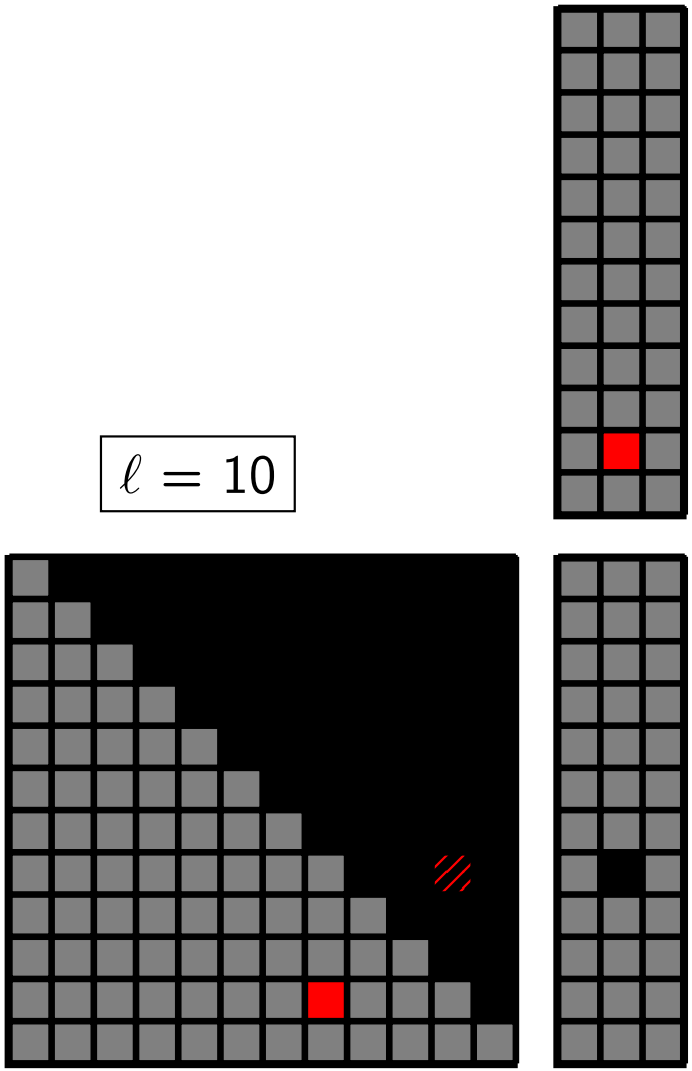
- Computing $C = AB$
- $C_{i,j} = \sum_{\ell=1}^{i} A_{i,\ell} B_{\ell,j} + \sum_{\ell=i+1}^{m} A^T_{\ell,i} B_{\ell,j}$
- $C_{7,1} = \ldots + A^T_{8,7} B_{8,1} + \ldots$

SYMM (SYmmetric Matrix-Matrix multiplication)



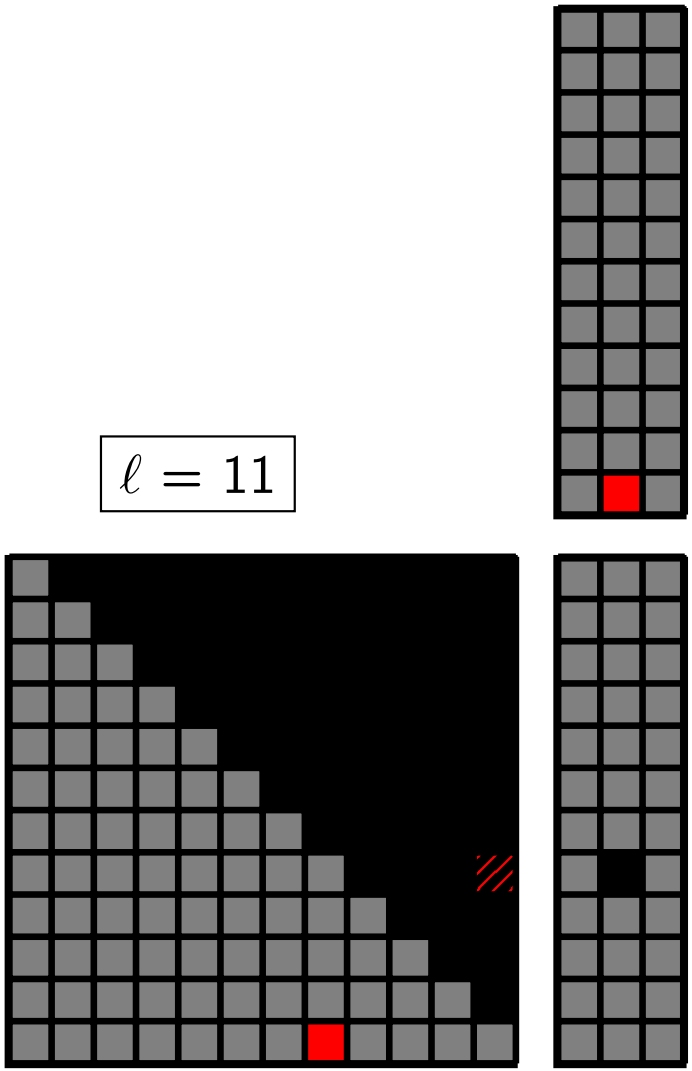
- Computing $C = AB$
- $C_{i,j} = \sum_{\ell=1}^{i} A_{i,\ell} B_{\ell,j} + \sum_{\ell=i+1}^{m} A^T_{\ell,i} B_{\ell,j}$
- $C_{7,1} = \ldots + A^T_{9,7} B_{9,1} + \ldots$

SYMM (SYmmetric Matrix-Matrix multiplication)



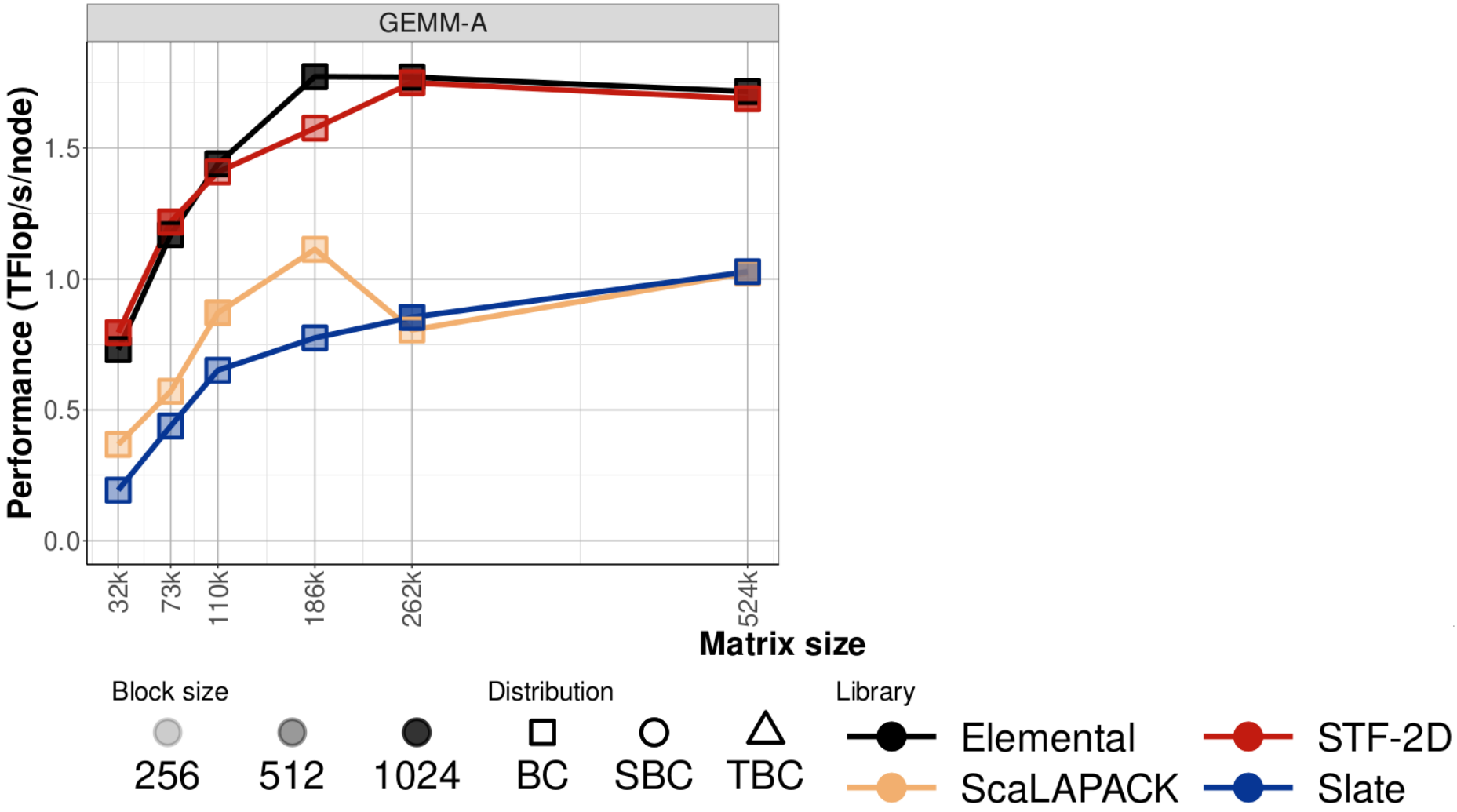
- Computing $C = AB$
- $C_{i,j} = \sum_{\ell=1}^{i} A_{i,\ell} B_{\ell,j} + \sum_{\ell=i+1}^{m} A^T_{\ell,i} B_{\ell,j}$
- $C_{7,1} = \ldots + A^T_{10,7} B_{10,1} + \ldots$

SYMM (SYmmetric Matrix-Matrix multiplication)

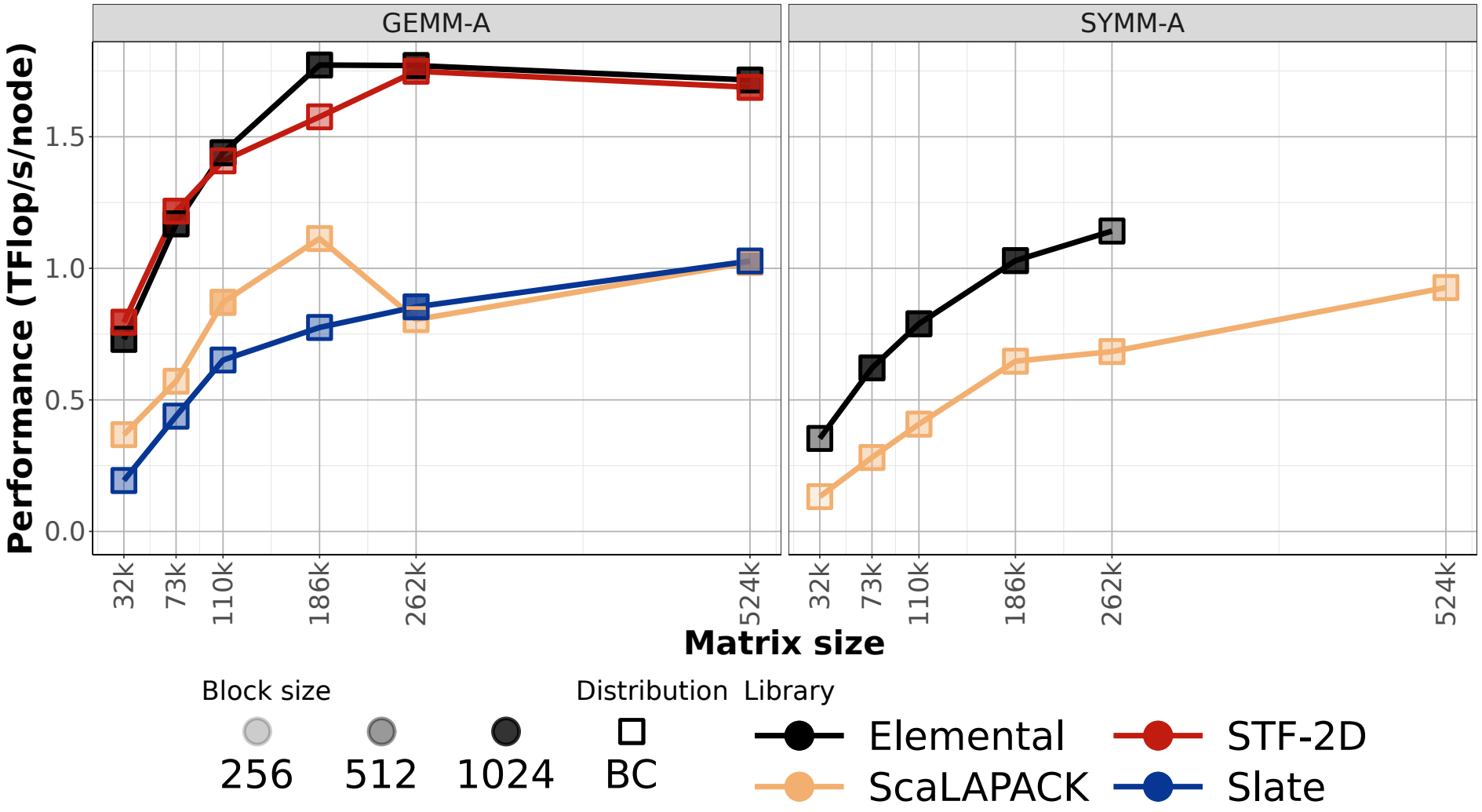


- Computing $C = AB$
- $C_{i,j} = \sum_{\ell=1}^{i} A_{i,\ell} B_{\ell,j} + \sum_{\ell=i+1}^{m} A^T_{\ell,i} B_{\ell,j}$
- $C_{7,1} = \ldots + A^T_{11,7} B_{11,1}$

Baseline performance of SYMM



Baseline performance of SYMM



Question

	Arithmetic Intensity ($\frac{Computation}{Communication}$)	Memory footprint
GEMM	☺	☹
SYMM	☹	☺

Joint Work

Emmanuel Agullo, Alfredo Buttari, Olivier Coulaud, Lionel Eyraud-Dubois, Mathieu Faverge, Alain Franc, Abdou Guermouche, Antoine Jego, Romain Peressoni and Florent Pruvost

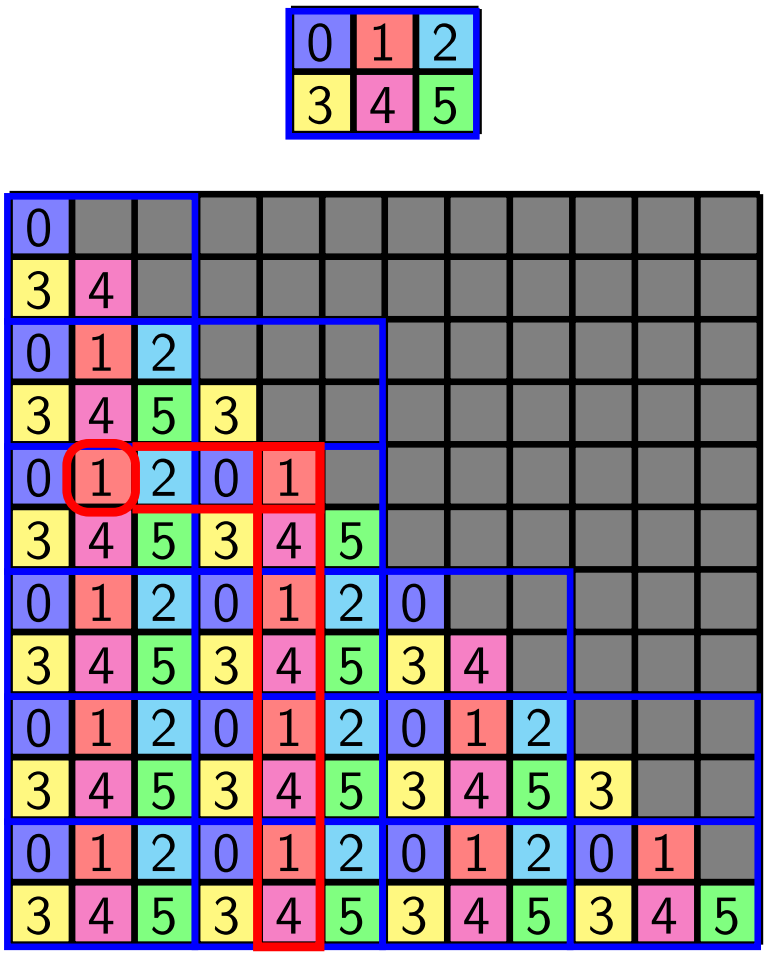
Question

	Arithmetic Intensity ($\frac{Computation}{Communication}$)	Memory footprint
GEMM-2DBC	☺	☹
SYMM-2DBC	☹	☺

Joint Work

Emmanuel Agullo, Alfredo Buttari, Olivier Coulaud, Lionel Eyraud-Dubois, Mathieu Faverge, Alain Franc, Abdou Guermouche, Antoine Jego, Romain Peressoni and Florent Pruvost

2D Block-Cyclic Distribution

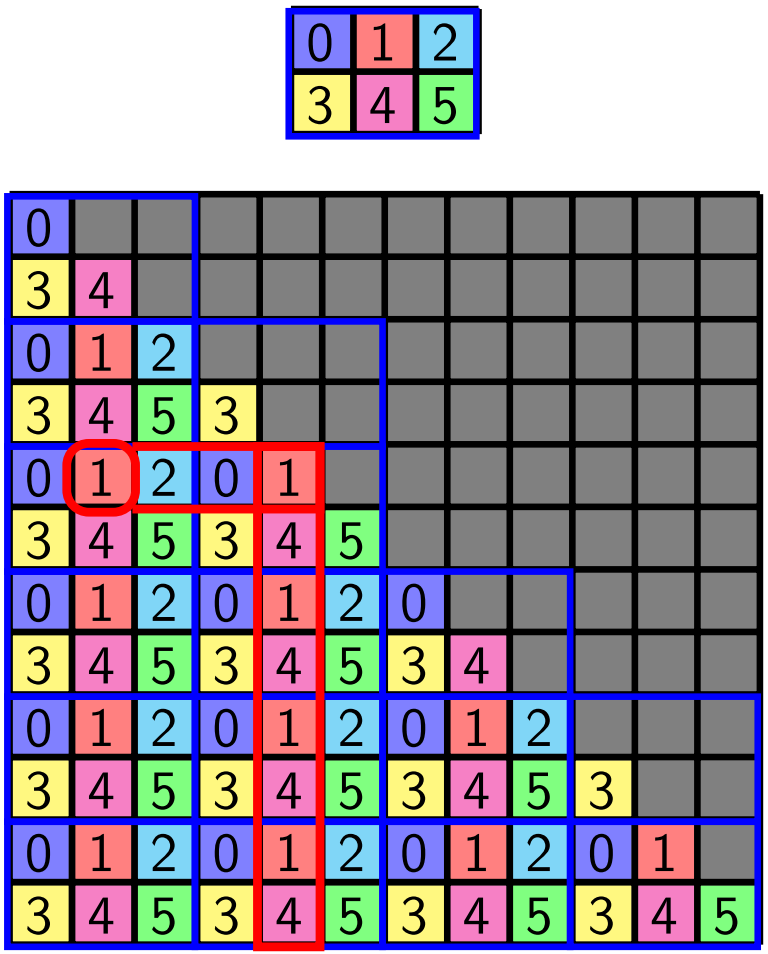


- For a **symmetric** matrix each line is present on both a line and a column.
- In a $p \times q$ distribution, each line is distributed among $p + q - 1$ nodes.

$(p \approx q \approx \sqrt{P})$

Method	Comm. Volume
GEMM C-Stat	$qm^2 + pmk$
GEMM A-Stat	$\approx 2\sqrt{P}mk$

2D Block-Cyclic Distribution

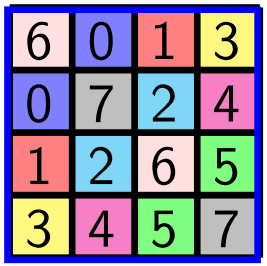


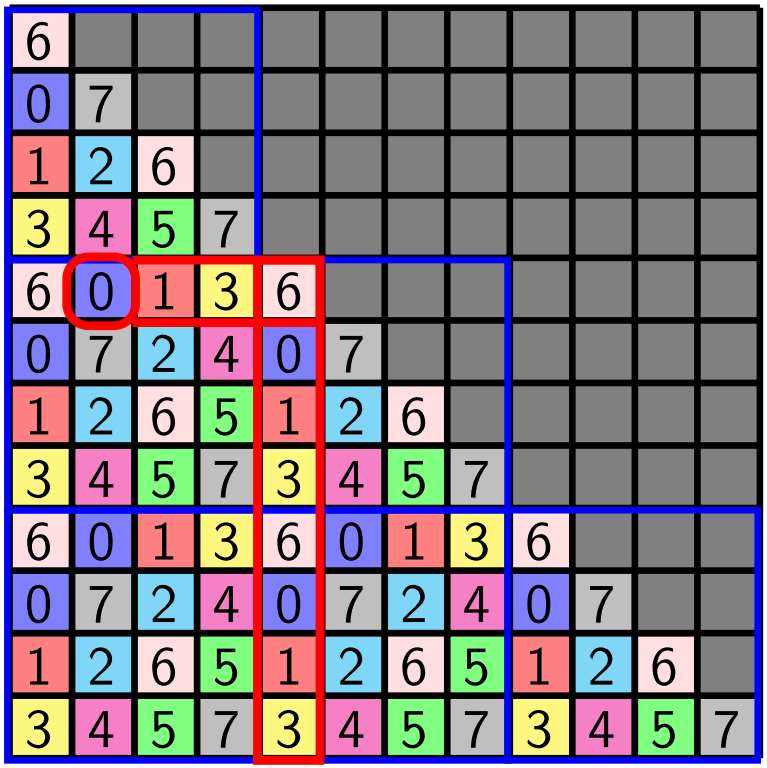
- For a **symmetric** matrix each line is present on both a line and a column.
- In a $p \times q$ distribution, each line is distributed among $p + q - 1$ nodes.

$(p \approx q \approx \sqrt{P})$

Method	Comm. Volume
GEMM C-Stat	$qm^2 + pmk$
GEMM A-Stat	$\approx 2\sqrt{P}mk$
SYMM 2DBC	$\approx 4\sqrt{P}mk$

Symmetric Block-Cyclic Distribution

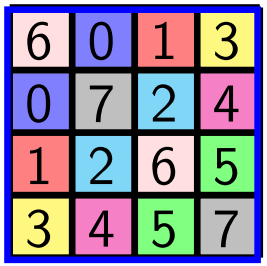


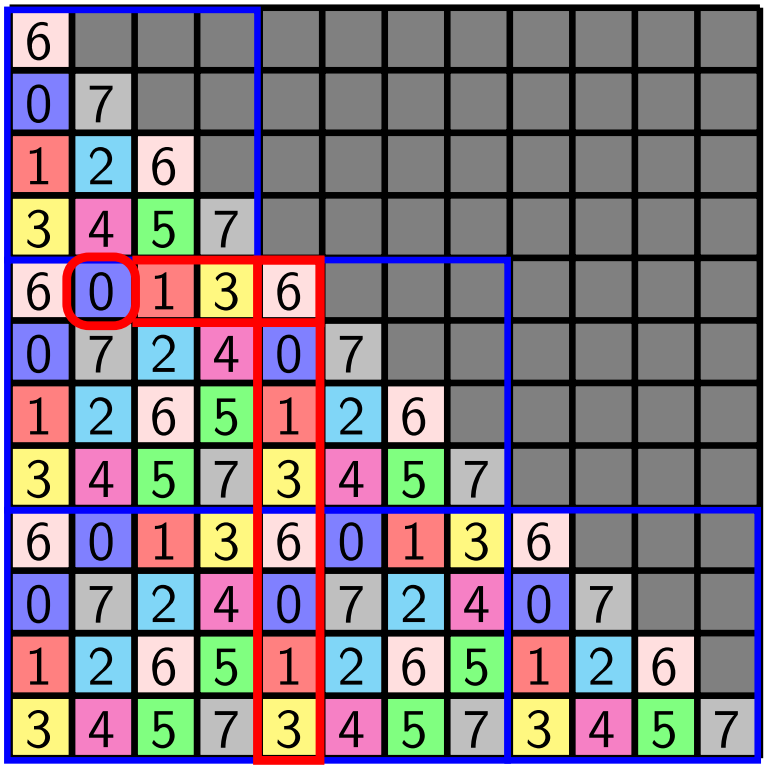


- Blocks of size $r \times r$ with $P = \frac{r^2}{2}$ Beaumont, Duchon, et al., 2022
- Optimises the number of nodes on which lines are distributed

Method	Comm. Volume
GEMM C-Stat.	$qm^2 + pmk$
GEMM A-Stat.	$\approx 2\sqrt{P}mk$
SYMM 2DBC	$\approx 4\sqrt{P}mk$

Symmetric Block-Cyclic Distribution

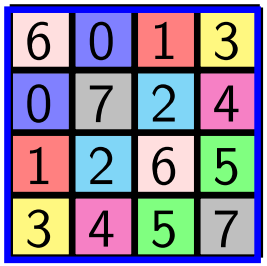


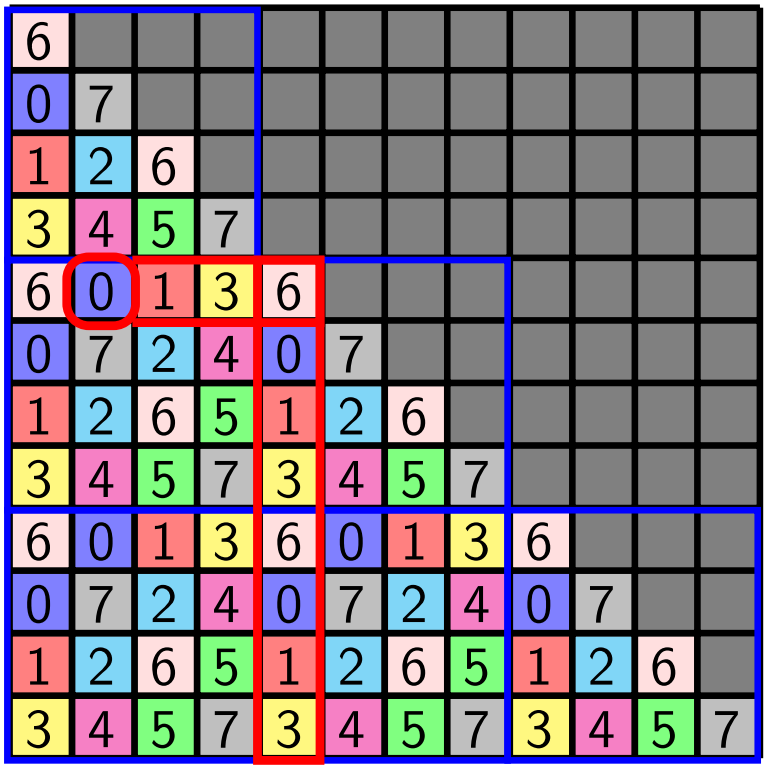


- Blocks of size $r \times r$ with $P = \frac{r^2}{2}$ Beaumont, Duchon, et al., 2022
- Optimises the number of nodes on which lines are distributed

Method	Comm. Volume
GEMM C-Stat.	$qm^2 + pmk$
GEMM A-Stat.	$\approx 2\sqrt{P}mk$
SYMM 2DBC	$\approx 4\sqrt{P}mk$
SYMM SBC	$\approx 2\sqrt{2P}mk$

Symmetric Block-Cyclic Distribution

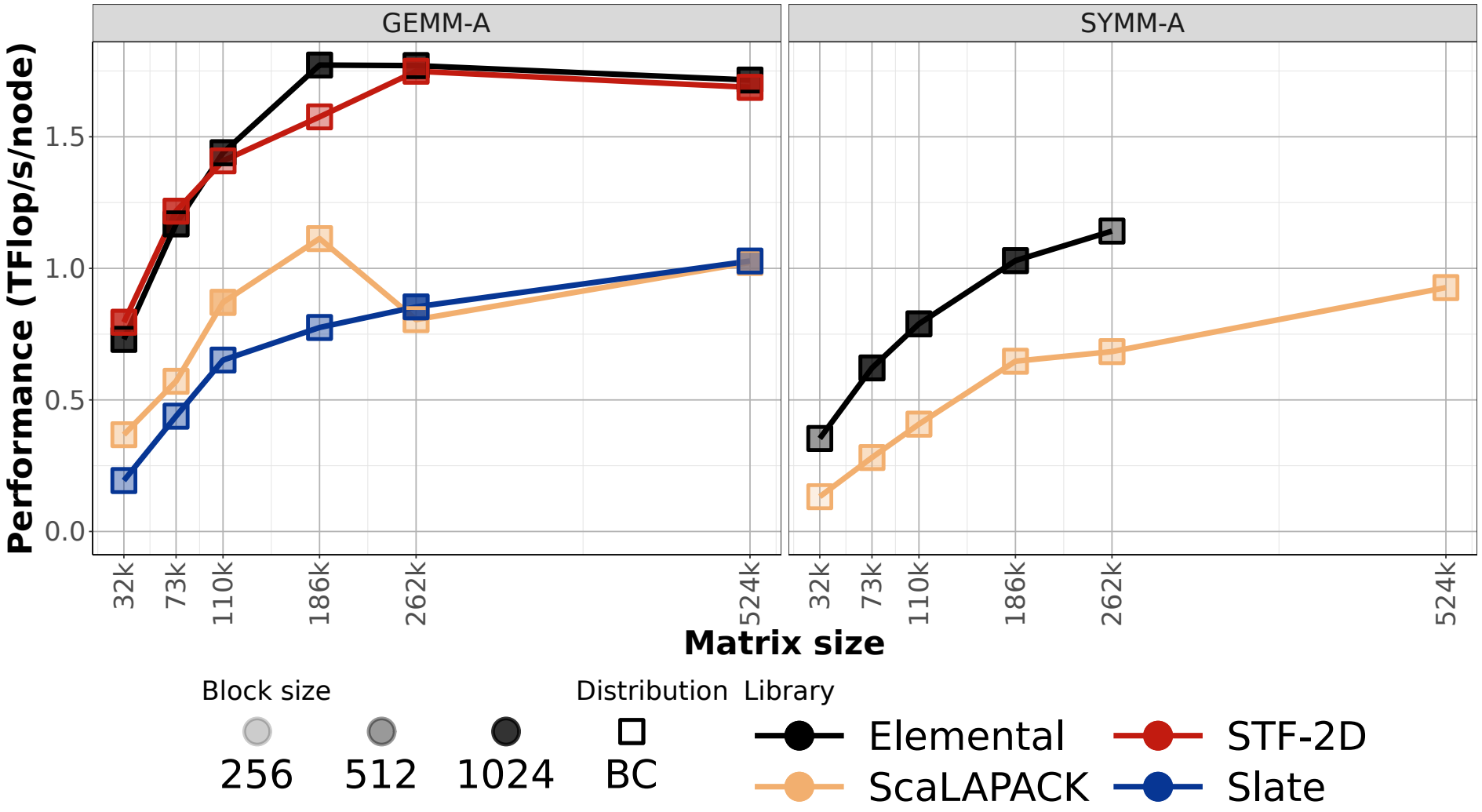




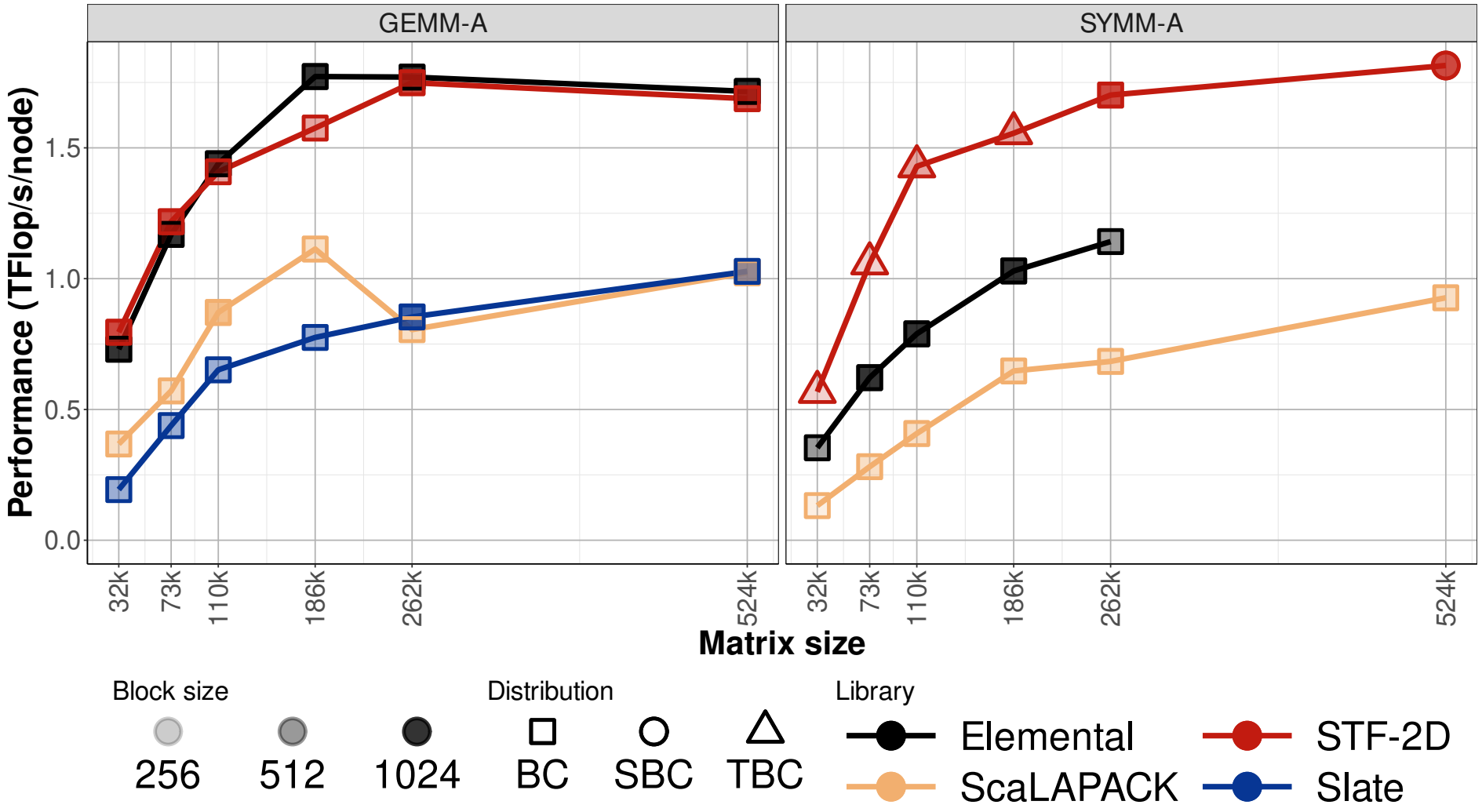
- Blocks of size $r \times r$ with $P = \frac{r^2}{2}$ Beaumont, Duchon, et al., 2022
- Optimises the number of nodes on which lines are distributed
- It is possible to reach an even lower communication volume using a TBC distribution Beaumont, Eyraud-Dubois, et al., 2022

Method	Comm. Volume
GEMM C-Stat.	$qm^2 + pmk$
GEMM A-Stat.	$\approx 2\sqrt{P}mk$
SYMM 2DBC	$\approx 4\sqrt{P}mk$
SYMM SBC	$\approx 2\sqrt{2P}mk$
SYMM TBC	$\approx 2\sqrt{P}mk$

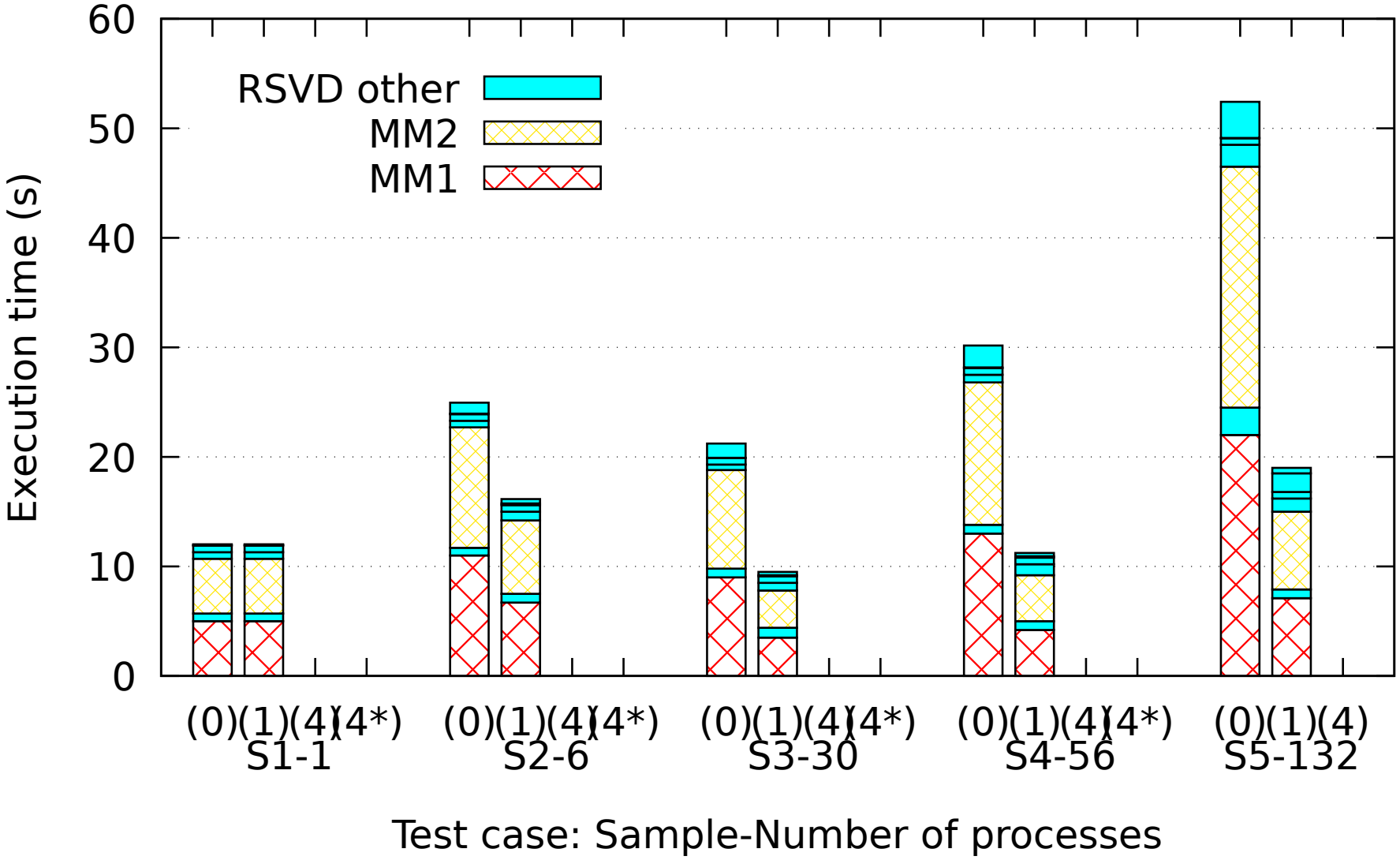
Performance of new A-Stationary SYMM



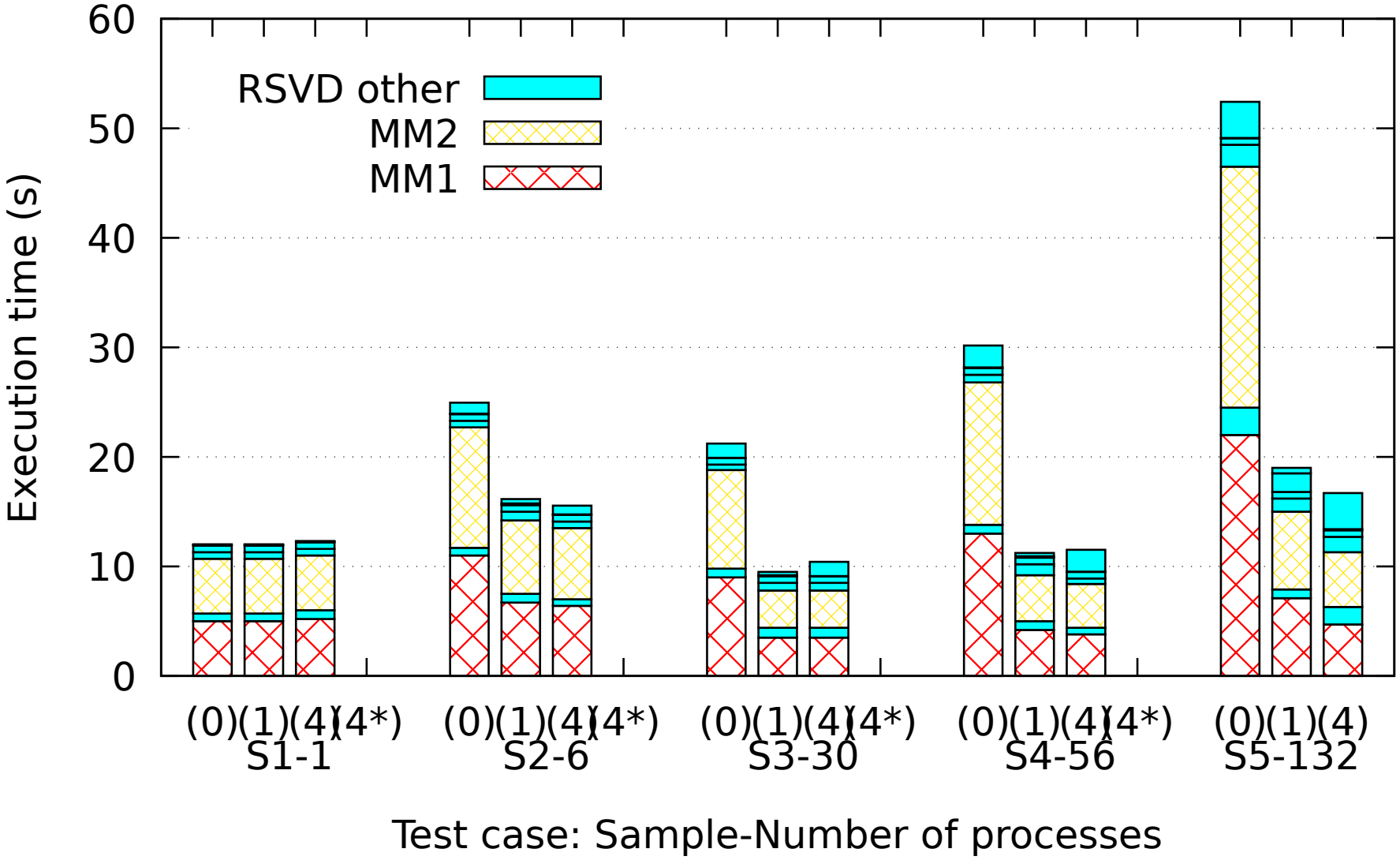
Performance of new A-Stationary SYMM



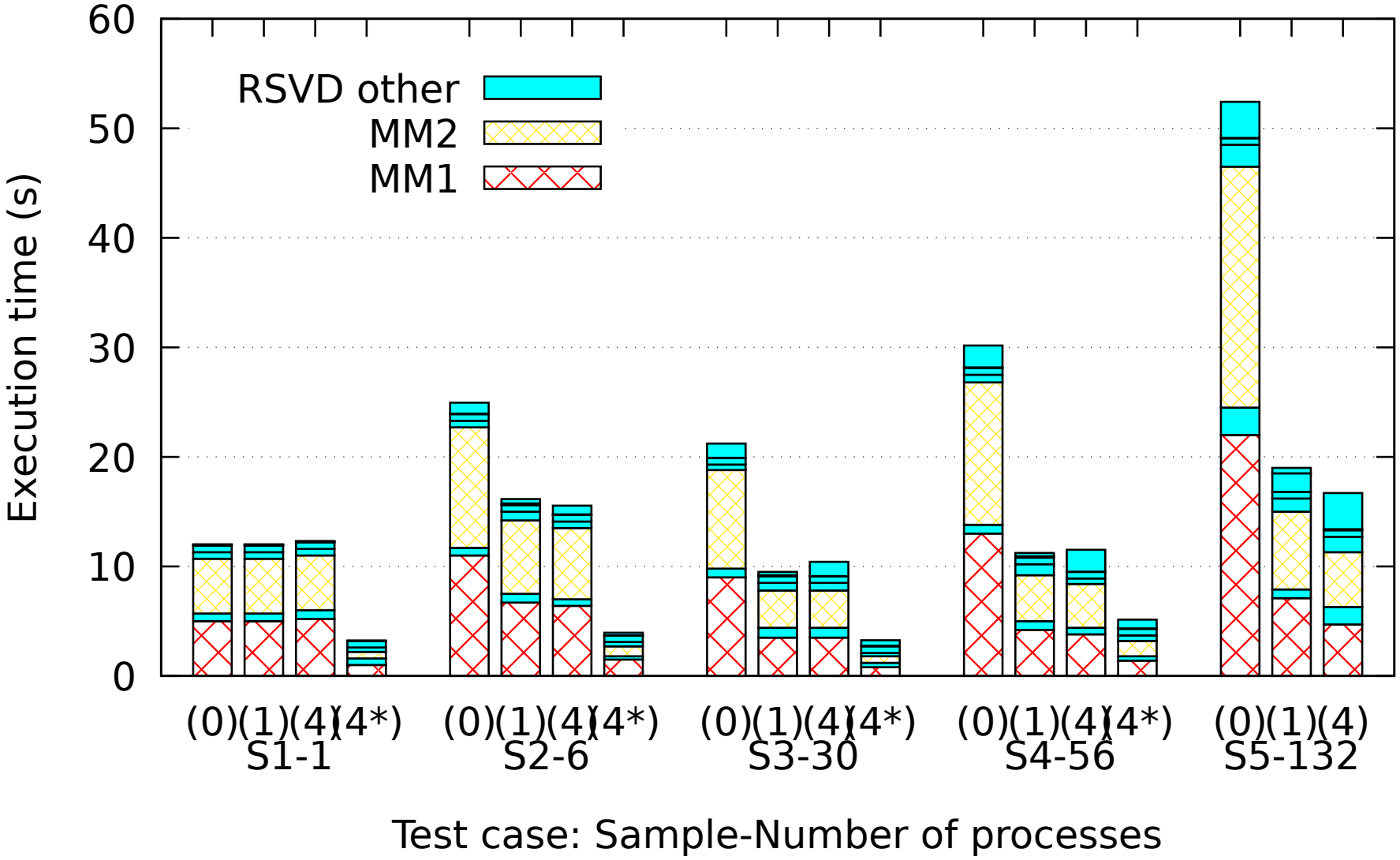
Performance of RSVD-MDS using our improved SYMM



Performance of RSVD-MDS using our improved SYMM



Performance of RSVD-MDS using our improved SYMM

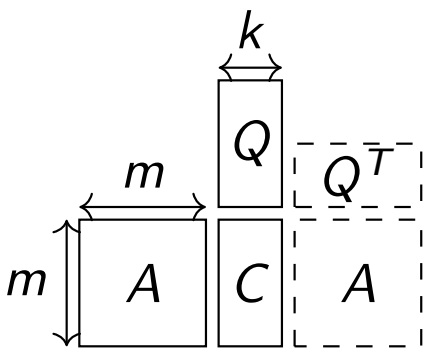


Summary of matrix multiplication improvements

- We implemented an A-Stationary GEMM implementation into Chameleon
- Using Abstract programming models we implemented a SYMM competitive with GEMM
- This work was accepted in IPDPS 2023

Reminder on random projection

- Approximate $A \approx AQQ^T$ (Halko et al., 2011; Martinsson, 2016)
- Define $C = AQ$
- Compute $SVD(C) = U_C \Sigma V_C^T$
- Multiply by Q^T on the right to get $RSVD(A)$



Evaluation of the symmetry of AQQ^T

A: symmetric
AQQ^T: ?

$$\begin{aligned} A &= A_{sym} + A_{skew} \\ A_{sym} &= \frac{A + A^T}{2} \\ A_{skew} &= \frac{A - A^T}{2} \end{aligned}$$

Evaluation of the symmetry of AQQ^T

A: symmetric
AQQ^T: ?

$$A = A_{sym} + A_{skew}$$

$$A_{sym} = \frac{A + A^T}{2}$$

$$A_{skew} = \frac{A - A^T}{2}$$

$$\chi = \frac{||A_{skew}||}{||A||}$$

Evaluation of the symmetry of AQQ^T

Skew-symmetry induced by the projection

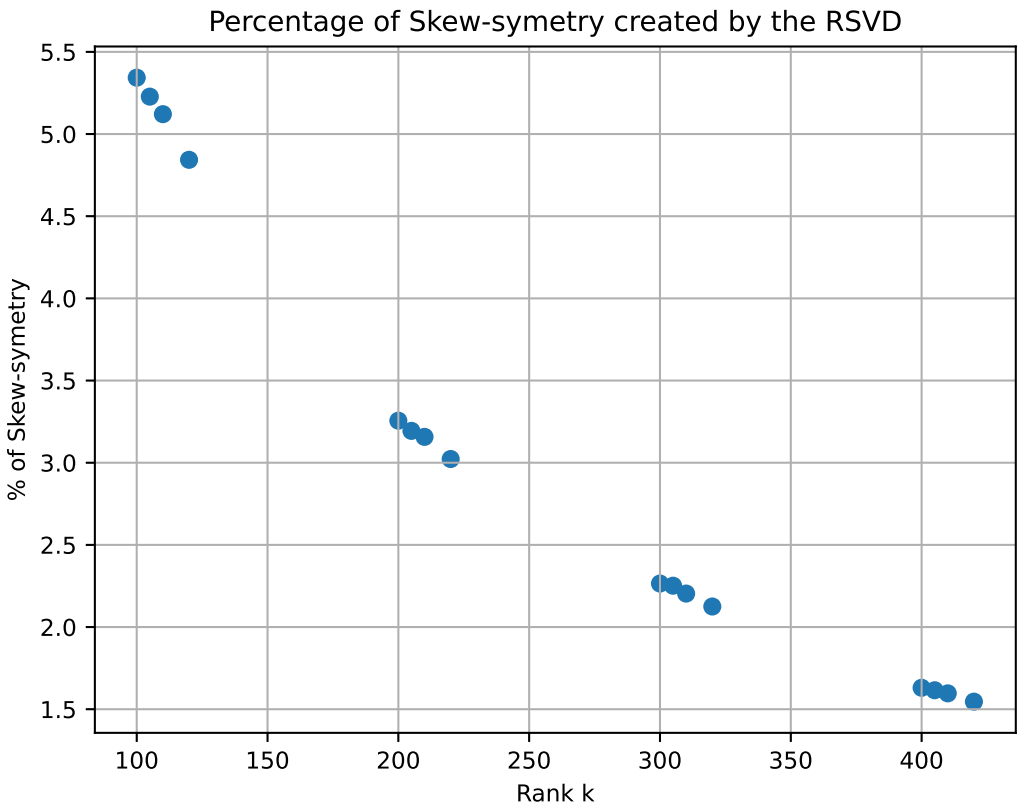
A: symmetric
AQQ^T: ?

$$A = A_{sym} + A_{skew}$$

$$A_{sym} = \frac{A + A^T}{2}$$

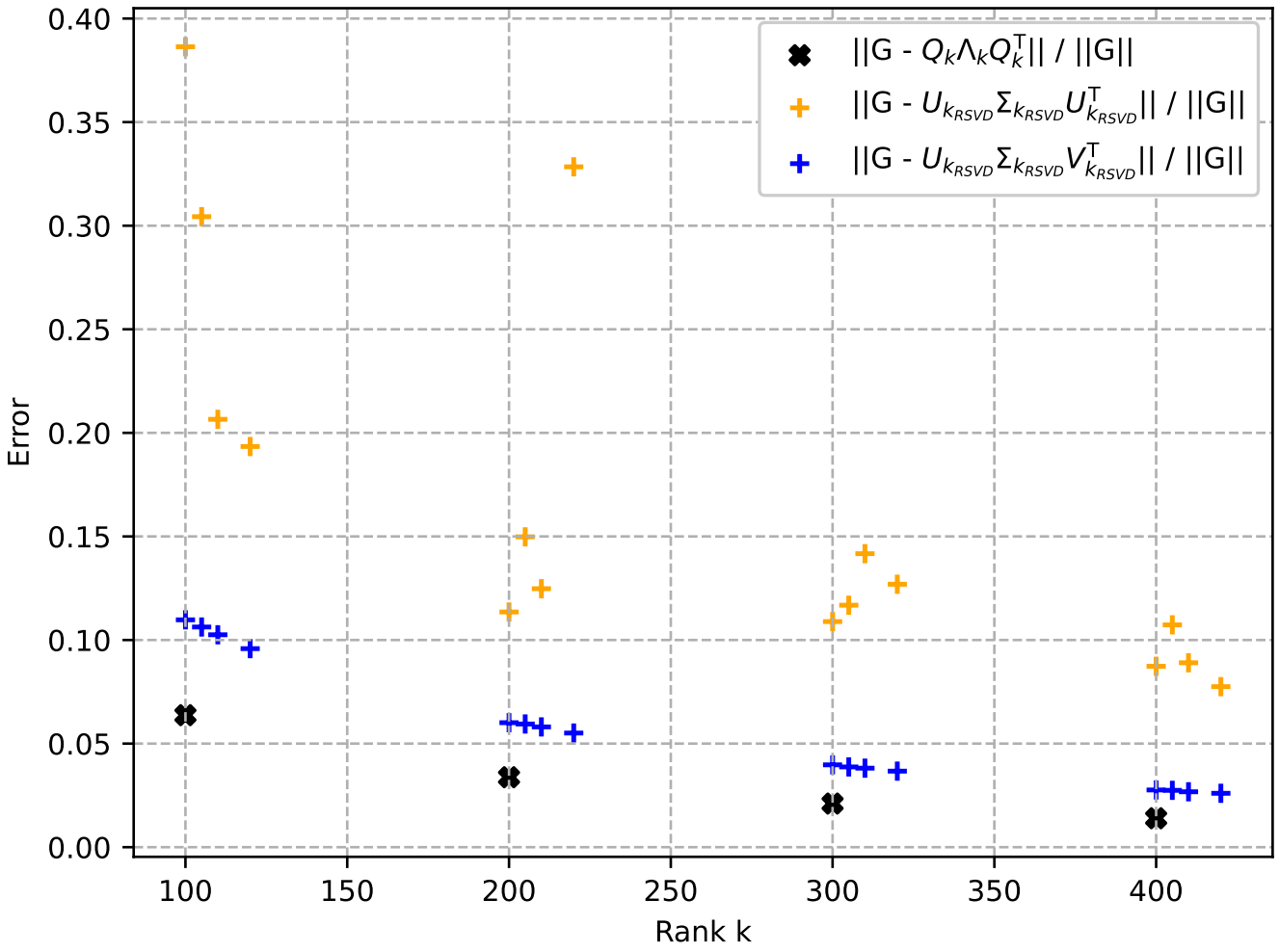
$$A_{skew} = \frac{A - A^T}{2}$$

$$\chi = \frac{||A_{skew}||}{||A||}$$



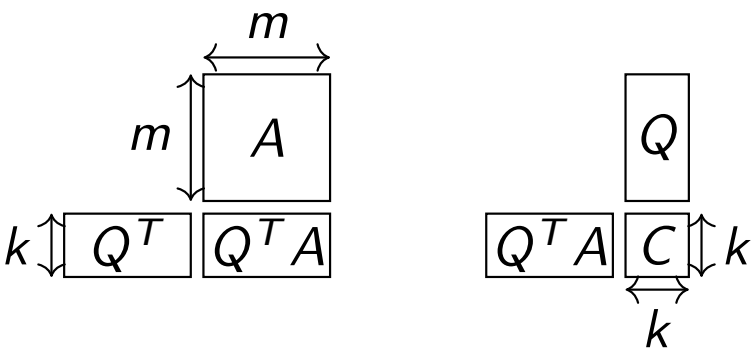
Impact on MDS

Impact of the loss of symmetry on MDS.



Proposed solution : double projection

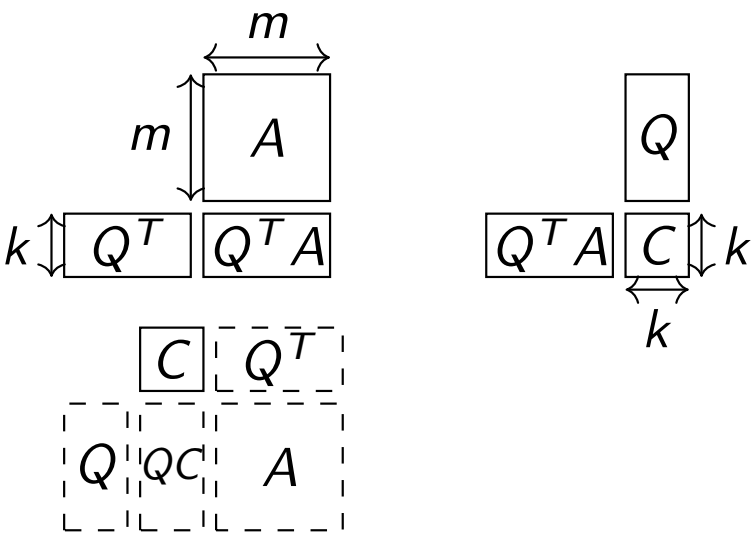
- Approximate $A \approx QQ^T AQQ^T$ (Halko et al., 2011; Martinsson, 2016)
- Define $C = Q^T AQ$
- Compute $sEVD(C) = Q_C \Lambda Q_C^T$
- Multiply by Q on both sides to get RsEVD(A)



Summary

Proposed solution : double projection

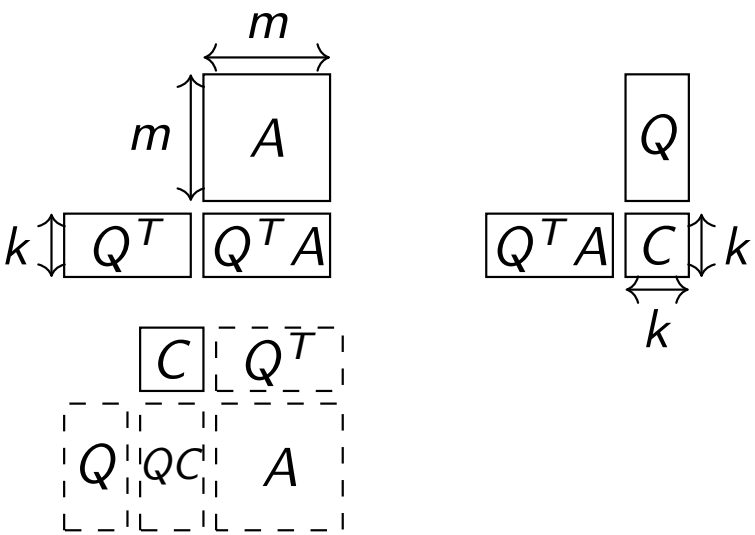
- Approximate $A \approx QQ^T AQQ^T$ (Halko et al., 2011; Martinsson, 2016)
- Define $C = Q^T AQ$
- Compute $sEVD(C) = Q_C \Lambda Q_C^T$
- Multiply by Q on both sides to get RsEVD(A)



Summary

Proposed solution : double projection

- Approximate $A \approx QQ^T AQQ^T$ (Halko et al., 2011; Martinsson, 2016)
- Define $C = Q^T AQ$
- Compute $sEVD(C) = Q_C \Lambda Q_C^T$
- Multiply by Q on both sides to get RsEVD(A)



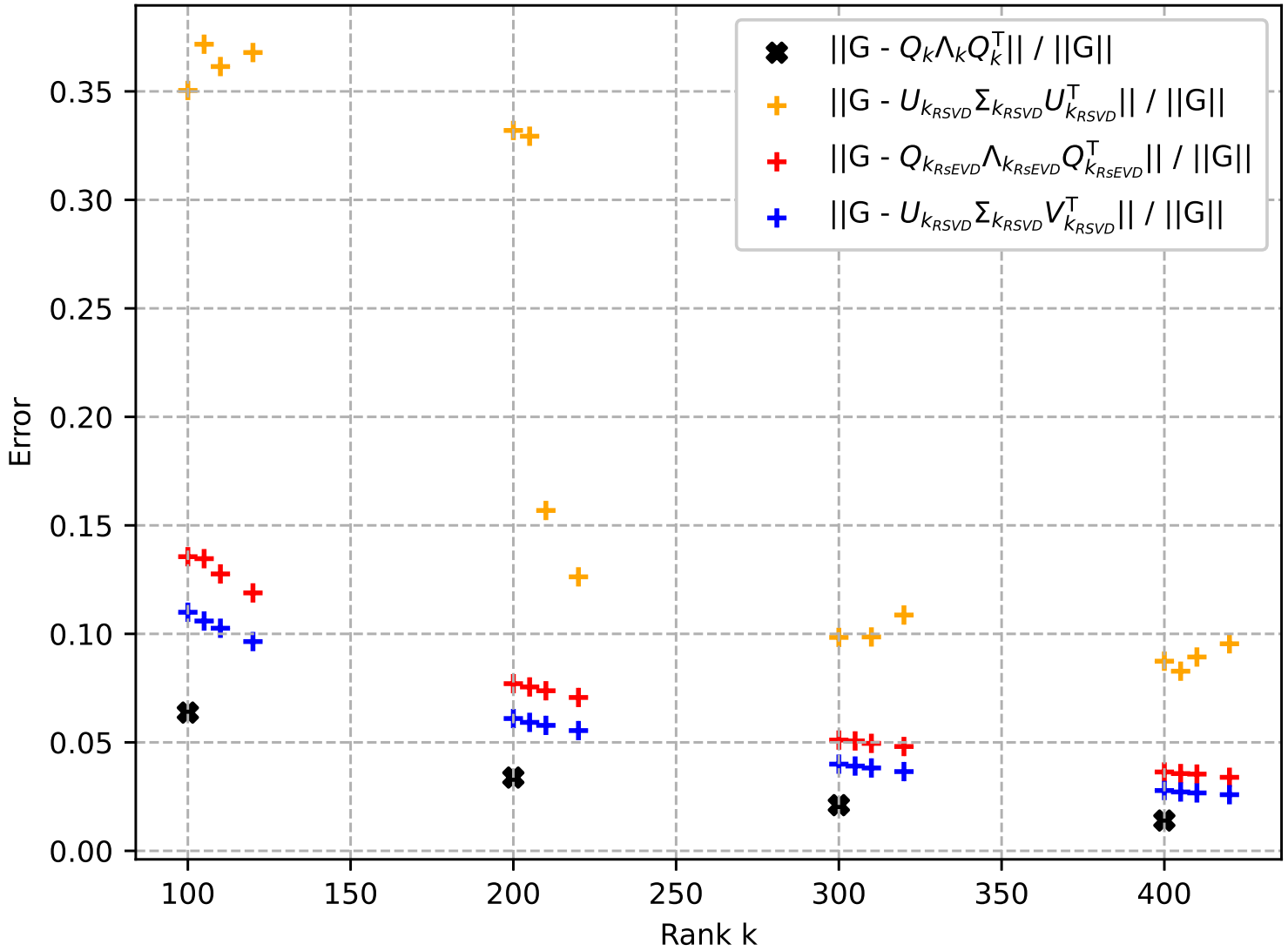
Summary

	Error	Symmetry preserving
RSVD	ϵ	?
REVD	2ϵ	✓

(Halko et al., 2011; Martinsson, 2016)

Evaluation of the error

Comparison of the different random projection error in the context of MDS



Experimental setup

About the test

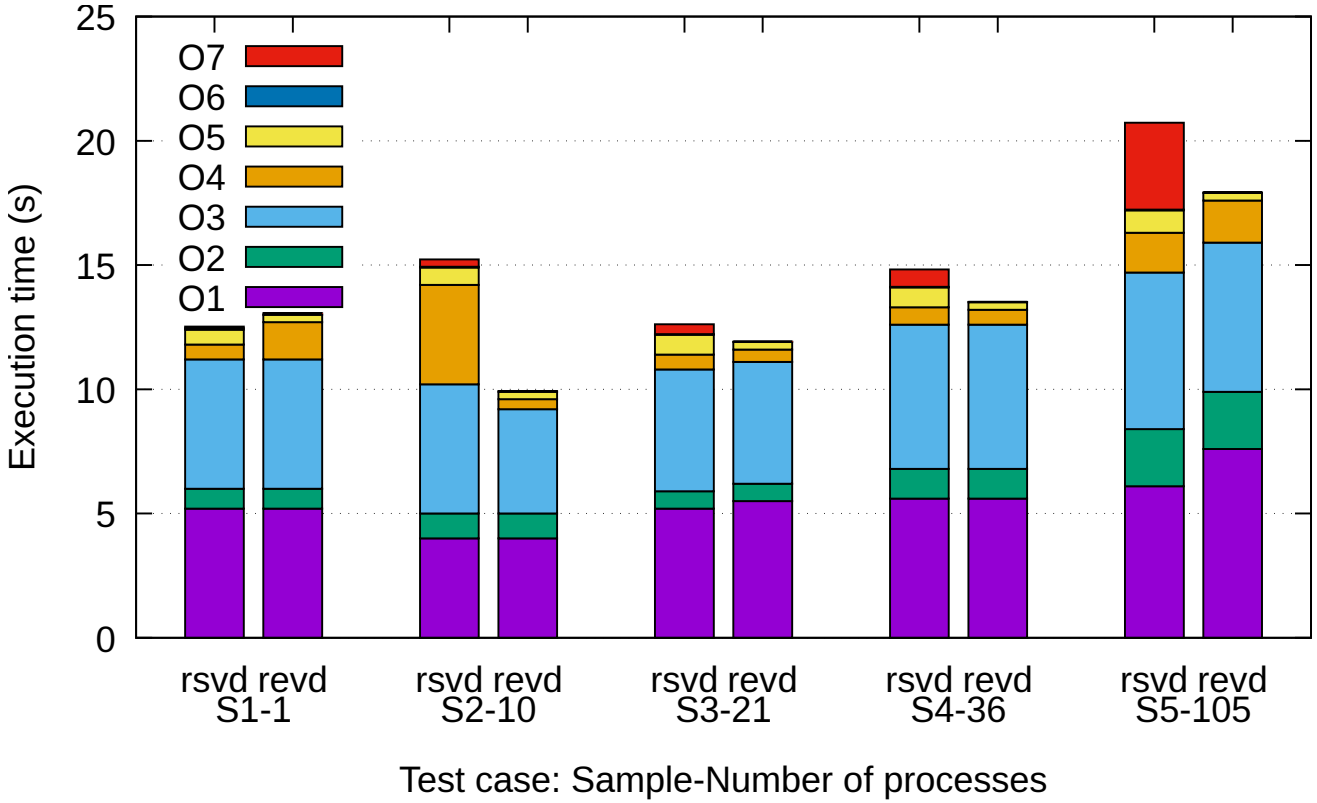
- Sample size ranges from 99, 954 (S1) to 1,043,192 (S5).
- MM1 and MM2 are presented as O1 and O3 respectively.

Machine used

- Results obtained on the Jean Zay supercomputer.
- Each node has 192 GB of memory
- Each node has two 20 cores Cascade Lake 6248 @ 2.5 GHz processors (40 cores per node)
- Used between 1 and 105 nodes

Work done in collaboration with Florent Pruvost

Performance results



Contributions

Improving the Matrix-Matrix multiplication step (Accepted into IPDPS 2023)

- Implementation of an A-Stationary GEMM into Chameleon
- Design and implementation of a A-Stationary SYMM with performance similar to GEMM

Task-based RsEVD for MDS

- Numerical study of RSVD and RsEVD
- HPC implementation of RsEVD and performance analysis

Conclusion

On matrix-Matrix Multiplication

- We showed that we can reduce the communication volume of SYMM to the same amount as the communication volume of GEMM.
- Using abstract programming models, we can seamlessly implement the different data distributions presented.
- We showed that our approach allowed us to reach performance for SYMM roughly matching those of GEMM, while only having to use half the amount of memory.
- We implemented our code in the context of an RSVD-MDS application and showed significant performance improvements.

On RsEVD for MDS

- We showed RSVD in the context of MDS may lead to error in the projection
- We presented an HPC RsEVD-MDS code and showed that it achieves similar performance than the RSVD-MDS approach.

Thank you for listening

Any questions ?

Acknowledgent: Projet Région Nouvelle-Aquitaine 2018-1R50119
"HPC scalable ecosystem"

Agullo, E., Buttari, A., Guermouche, A., Herrmann, J., & Jego, A. (2023). Task-based parallel programming for scalable matrix product algorithms. *ACM Transactions on Mathematical Software*. https://doi.org/10.1145/3583560

Agullo, E., Coulaud, O., Denis, A., Faverge, M., Franc, A. A., Frigerio, J.-M., Furmento, N., Thibault, S., Guilbaud, A., Jeannot, E., Peressoni, R., & Pruvost, F. (2022). *Task-based randomized singular value decomposition and multidimensional scaling* (Research Report No. 9482). Inria Bordeaux - Sud Ouest ; Inrae - BioGeCo.

The End II

Beaumont, O., Duchon, P., Eyraud-Dubois, L., Langou, J., & Vérité, M. (2022). Symmetric block-cyclic distribution: Fewer communications leads to faster dense cholesky factorization. *Supercomputing*.

Beaumont, O., Eyraud-Dubois, L., Vérité, M., & Langou, J. (2022). I/O-optimal algorithms for symmetric linear algebra kernels. *arXiv preprint arXiv:2202.10217*.

Blackford, L. S., Choi, J., Cleary, A. J., D'Azevedo, E. F., Demmel, J., Dhillon, I. S., Dongarra, J. J., Hammarling, S., Henry, G., Petitet, A., Stanley, K., Walker, D. W., & Whaley, R. C. (1997). Scalapack: A linear algebra library for message-passing computers. *Proceedings of the Eighth SIAM Conference on Parallel Processing for Scientific Computing, PPSC 1997, Hyatt Regency Minneapolis on Nicollel Mall Hotel, Minneapolis, Minnesota, USA, March 14-17, 1997*.

Blanchard, P., Chaumeil, P., Frigerio, J.-M., Rimet, F., Salin, F., Thérond, S., Coulaud, O., & Franc, A. (2018). *A geometric view of biodiversity: Scaling to metagenomics*.

The End III

Blanchard, P. (2017). *Fast hierarchical algorithms for the low-rank approximation of matrices, with applications to materials physics, geostatistics and data analysis.* (Doctoral dissertation). Université de Bordeaux. https://tel.archives-ouvertes.fr/tel-01534930

Franc, A. (2022). *Linear Dimensionality Reduction* (Research Report No. 9488). Inria Bordeaux Sud-Ouest. https://hal.inria.fr/hal-03784623

Franc, A., Blanchard, P., & Coulaud, O. (2020). Nonlinear mapping and distance geometry. *Optimization Letters*, *14*(2), 453–467.

Halko, N., Martinsson, P. G., & Tropp, J. A. (2011). Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, *53*(2), 217–288. https://doi.org/10.1137/090771806

Martinsson, P.-G. (2016). Randomized methods for matrix computations. https://doi.org/10.48550/ARXIV.1607.01649

Paradis, E. (2018). Multidimensional scaling with very large datasets. *Journal of Computational and Graphical Statistics*, *27*(4), 935–939.

The End IV

van de Geijn, R., & Watts, J. (1997). Summa: Scalable universal matrix multiplication algorithm. *CONCURRENCY: PRACTICE AND EXPERIENCE*, *9*(4), 255–274. http://www.netlib.org/lapack/lawnspdf/lawn96.pdf