Communication-Avoiding Parallel Sparse-Dense Matrix-Matrix Multiplication

Penporn Koanantakool[1,2],
Ariful Azad[2], Aydin Buluç[2],Dmitriy Morozov[2],
Sang-Yun Oh[2,3], Leonid Oliker[2], Katherine Yelick[1,2]

[1]Computer Science Division, University of California, Berkeley
[2]Lawrence Berkeley National Laboratory
[3]University of California, Santa Barbara

May 26, 2016

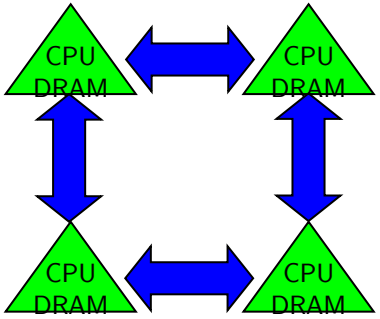
Outline

- Background and Motivation
- Algorithms
  - 2D SUMMA
  - 3D SUMMA
  - 1.5D variants
- Experimental Results : **100x speedup**
- Iterative Multiplication
- Conclusions

Parallel Algorithm Cost Model

- p distributed, homogenous processors connected through network
- **Per-processor** costs along **critical path**

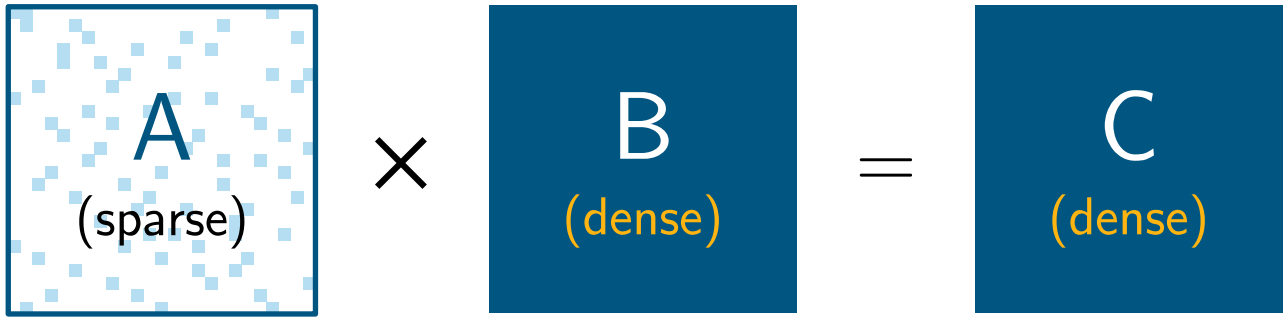
Image courtesy of: James Demmel

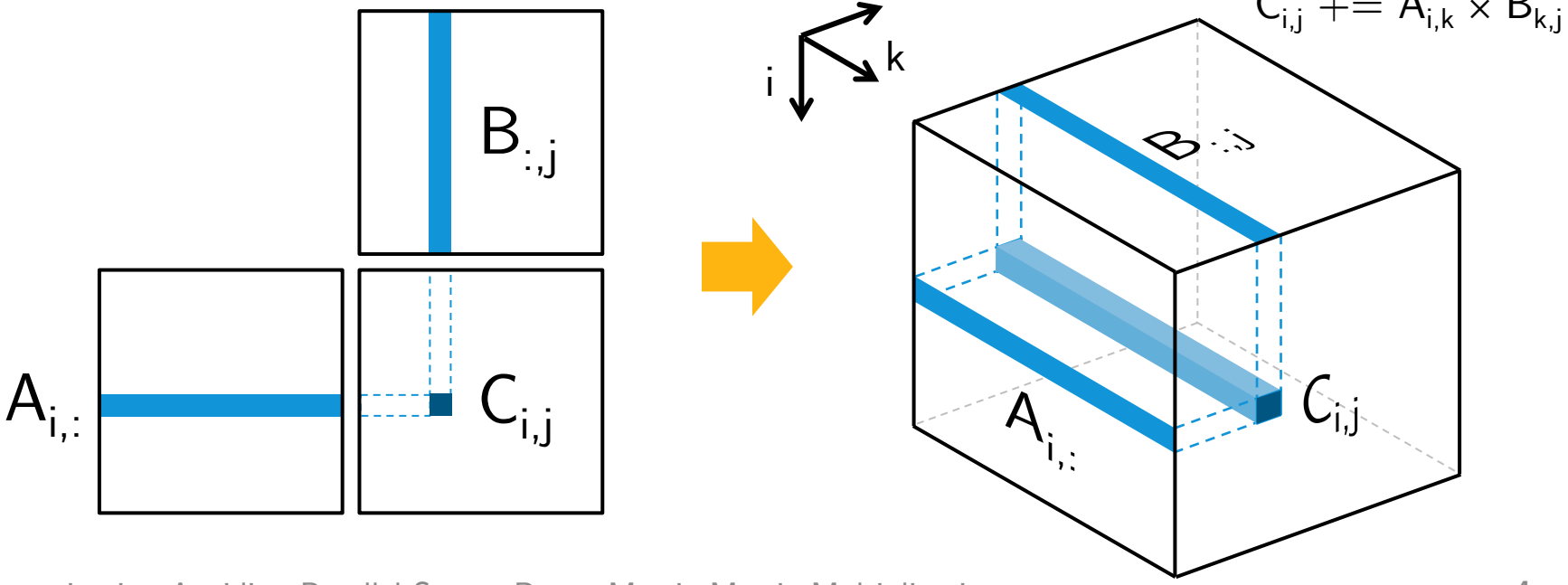


$$\text{time} = \begin{array}{l} \#\text{flops} \cdot \text{time\_per\_flop} + \\ \#\text{messages} \cdot \text{time\_per\_message} + \\ \#\text{words} \cdot \text{time\_per\_word} \end{array} \quad \left.\begin{array}{l} \\ \\ \end{array}\right\} \begin{array}{l} \text{Computation} \\ \text{Communication} \end{array}$$

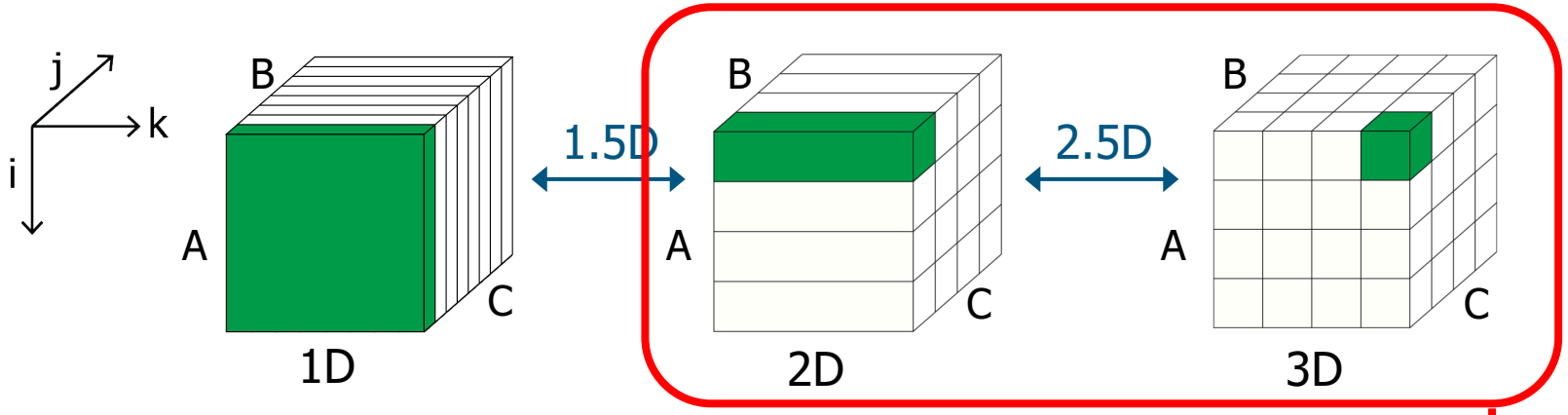
- time_per_flop, time_per_message, time_per_word are machine-specific constants.
- **Goal:** minimize **#messages (S)** and **#words (W)**

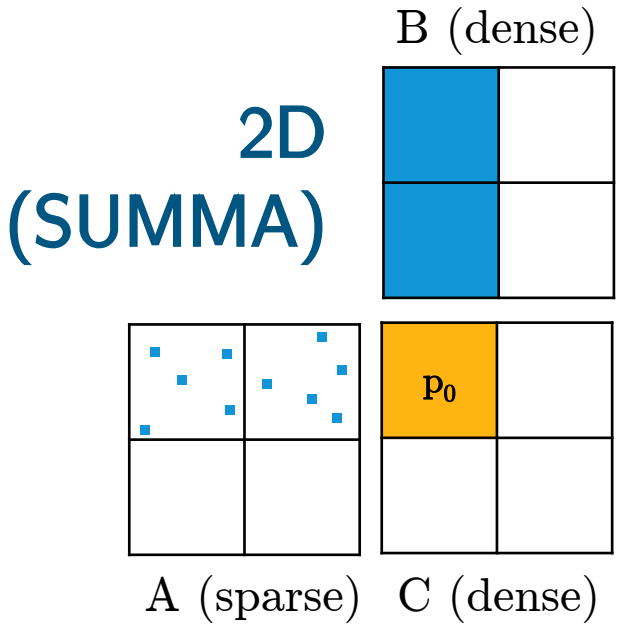
Sparse × Dense Matmul

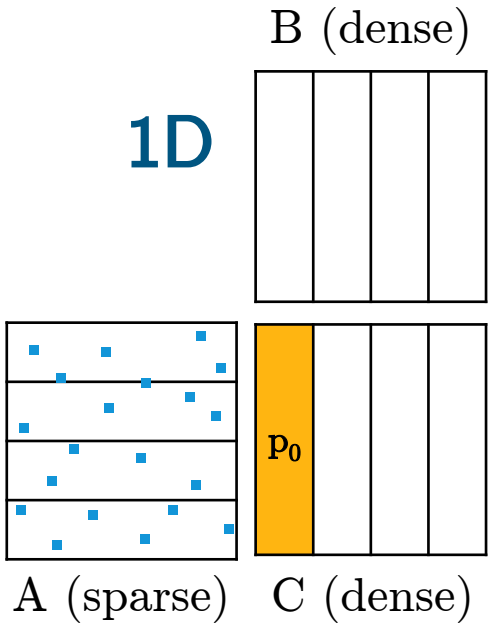




Matrix Multiplication Algorithms







Optimal for dense-dense/ sparse-sparse matmul

[Solomonik and Demmel 2011]

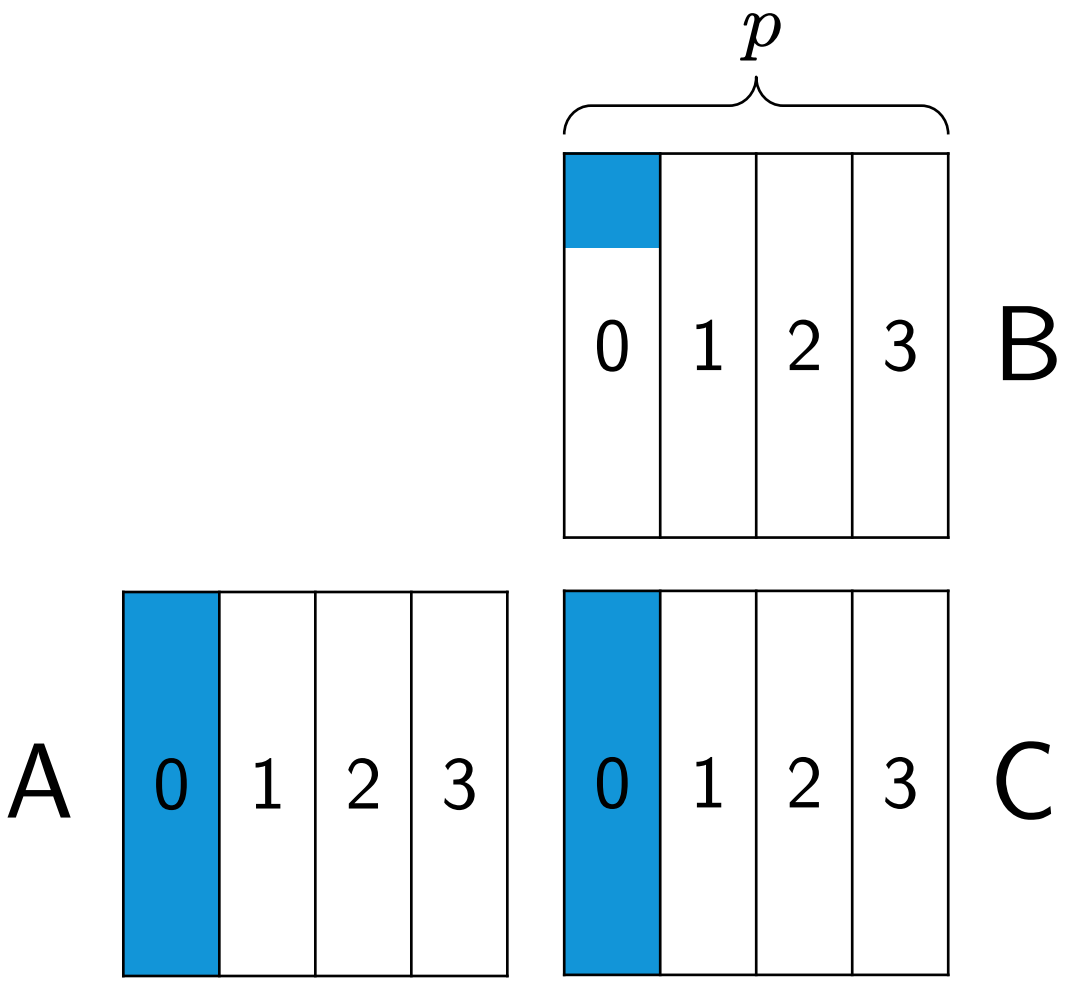
[Ballard et al. 2013]

*2D and 3D images courtesy of Grey Ballard

Existing Algorithms

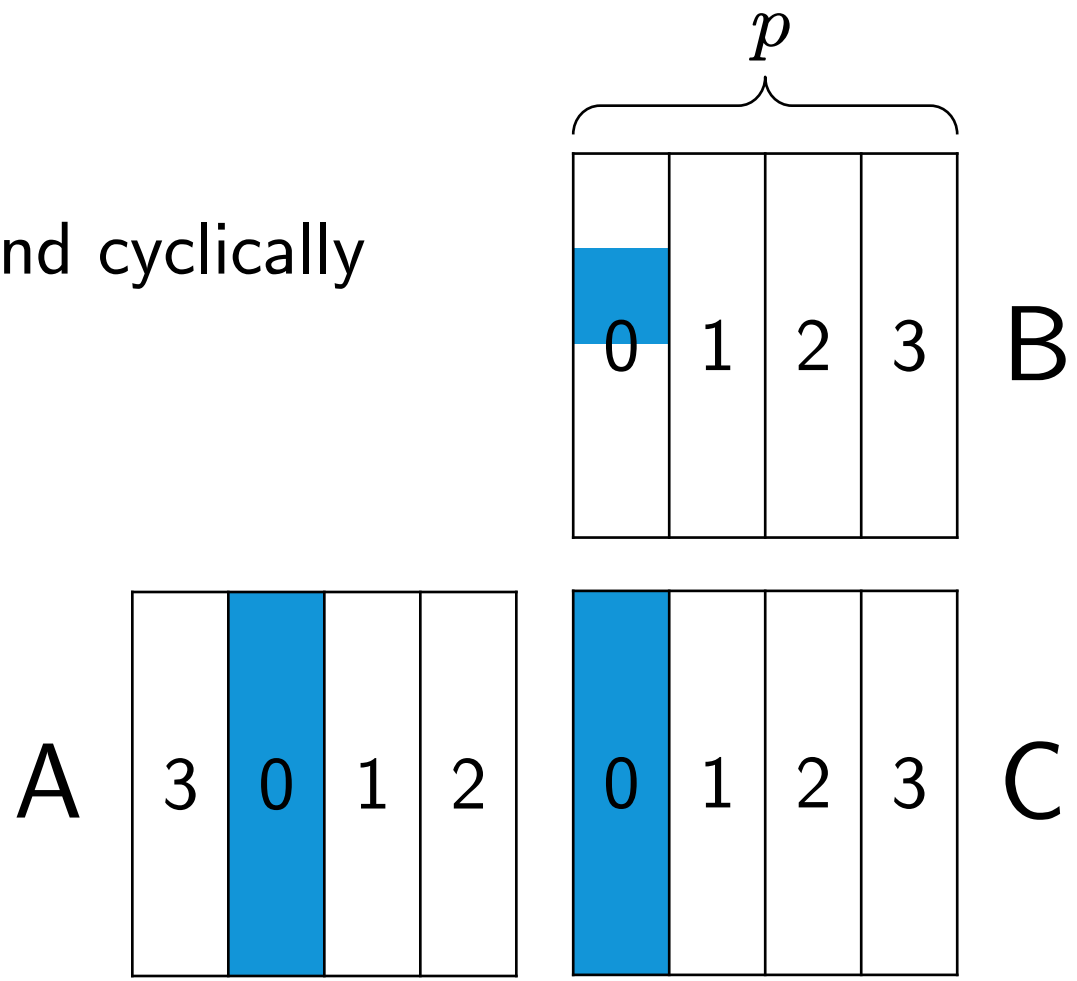
1D Ring Algorithm

- $p=4$
- 1D layout



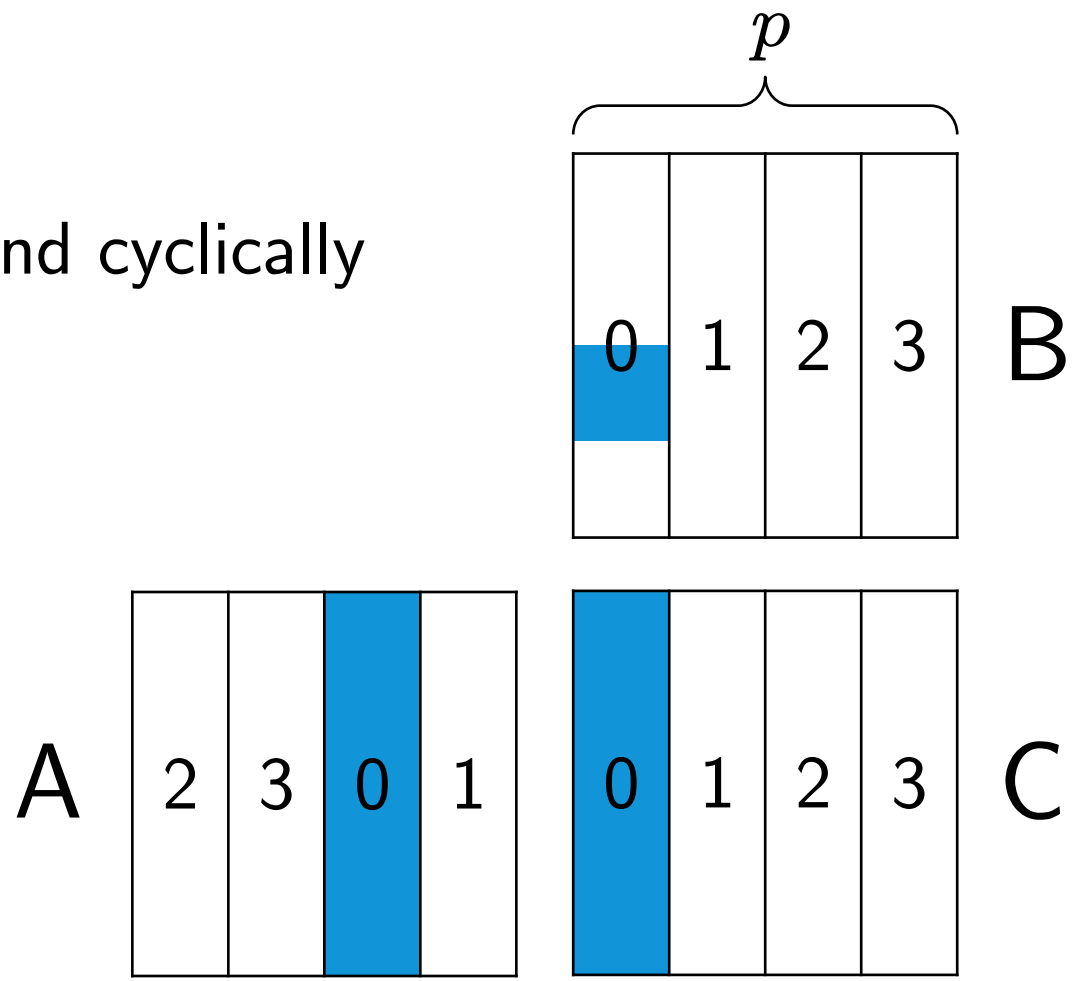
1D Ring Algorithm

- $p=4$
- 1D layout
- Shifts matrix around cyclically



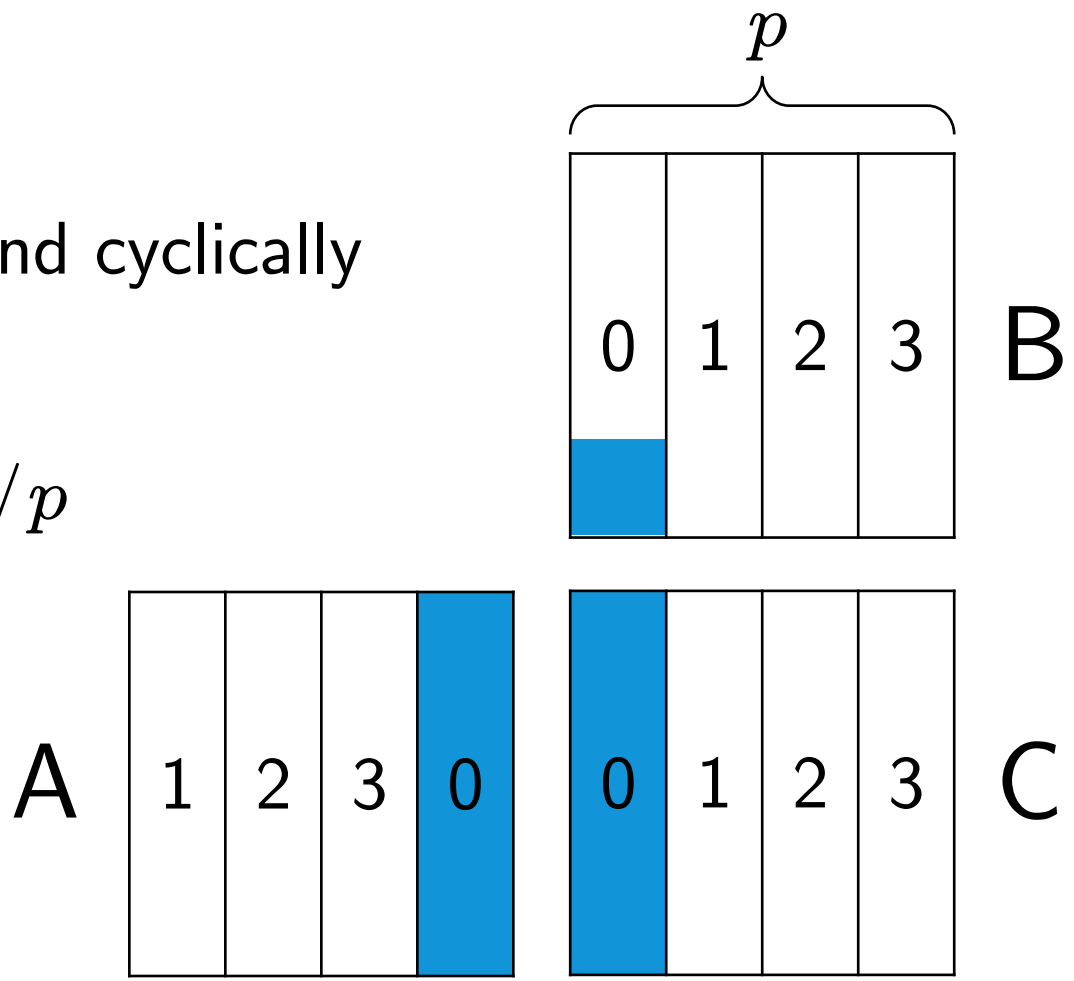
1D Ring Algorithm

- $p=4$
- 1D layout
- Shifts matrix around cyclically



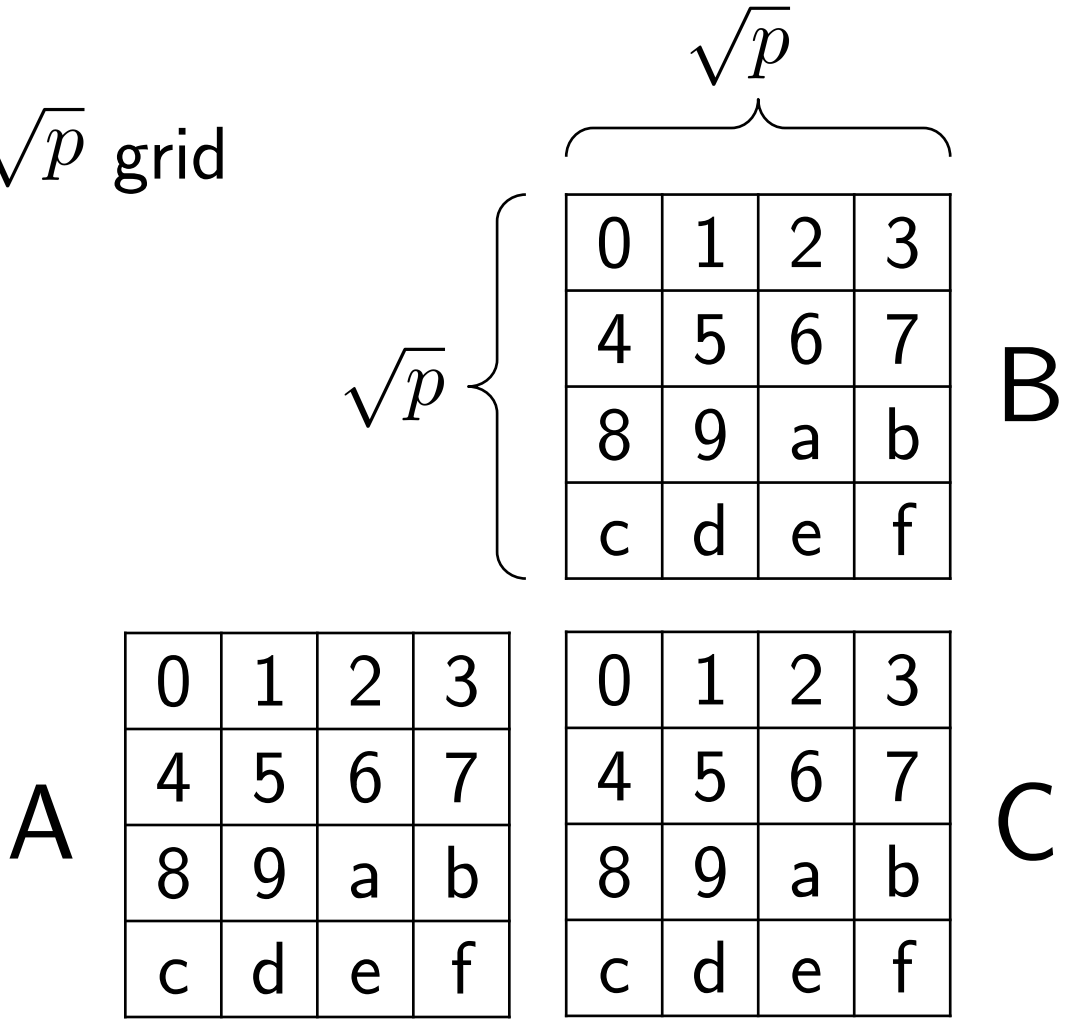
1D Ring Algorithm

- p=4
- 1D layout
- Shifts matrix around cyclically
- #msgs: p
- msg size: $\text{nnz}(A)/p$
- #words: $\text{nnz}(A)$



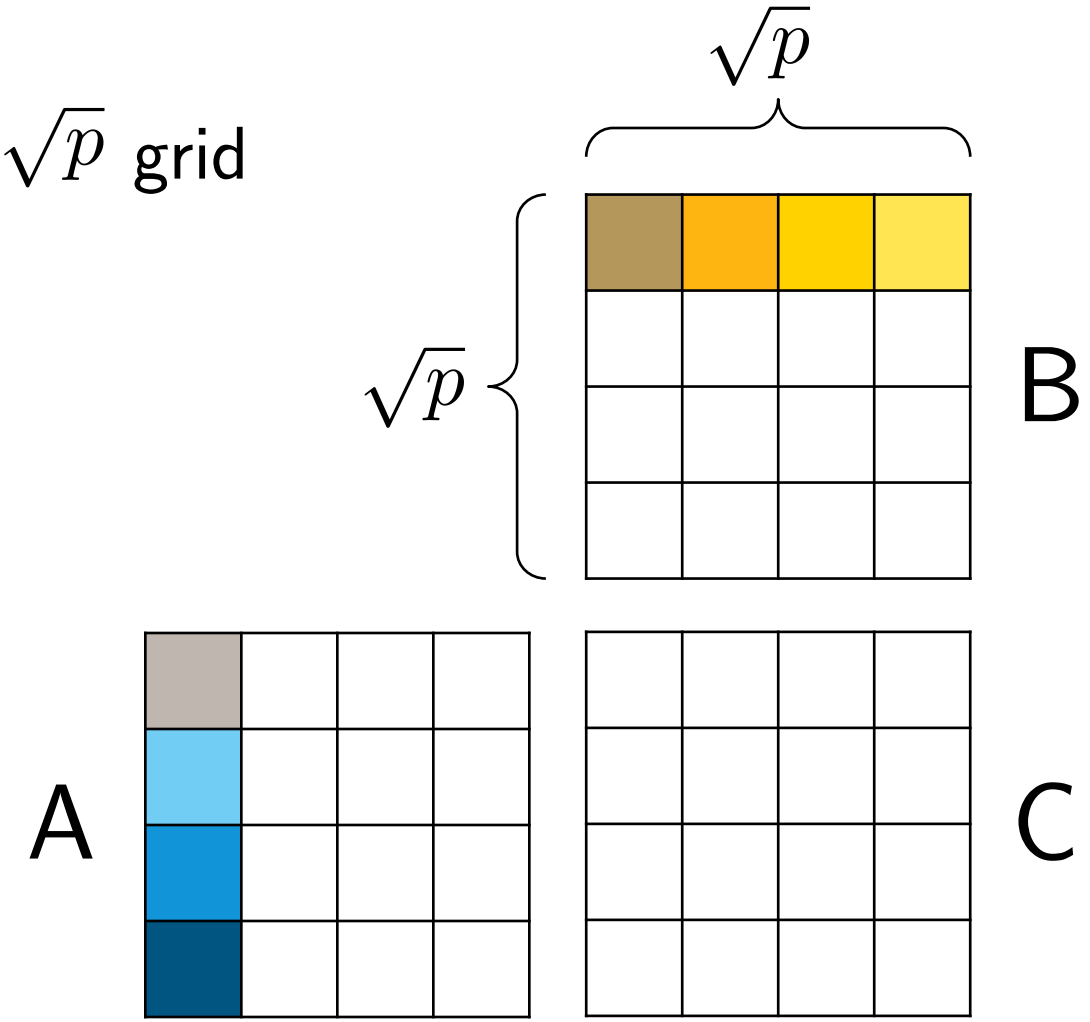
2D SUMMA Algorithm

- p=16
- 2D layout, $\sqrt{p} \times \sqrt{p}$ grid



2D SUMMA Algorithm

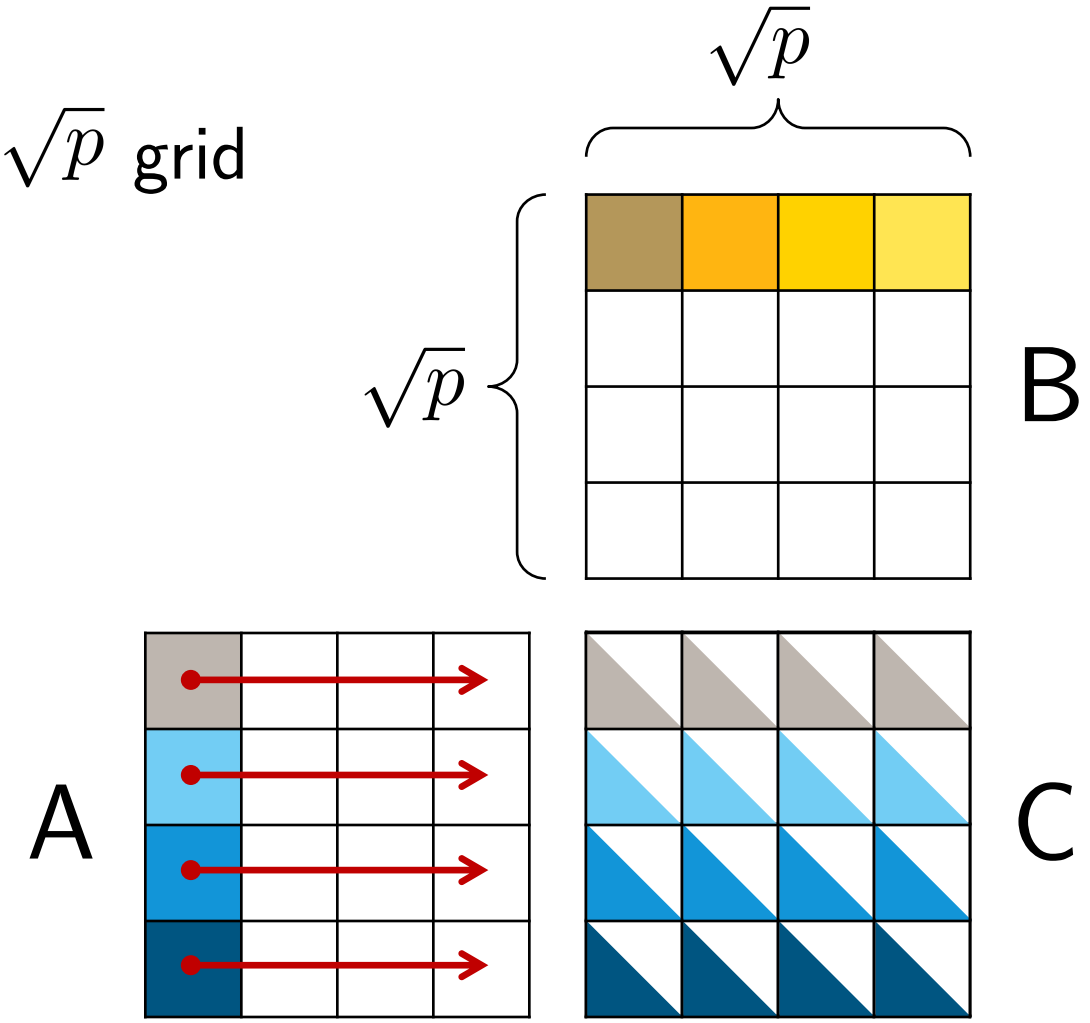
- p=16
- 2D layout, $\sqrt{p} \times \sqrt{p}$ grid
- $\sqrt{p}$ outer products



2D SUMMA Algorithm

- p=16
- 2D layout, $\sqrt{p} \times \sqrt{p}$ grid
- $\sqrt{p}$ outer products

A broadcasts row-wise

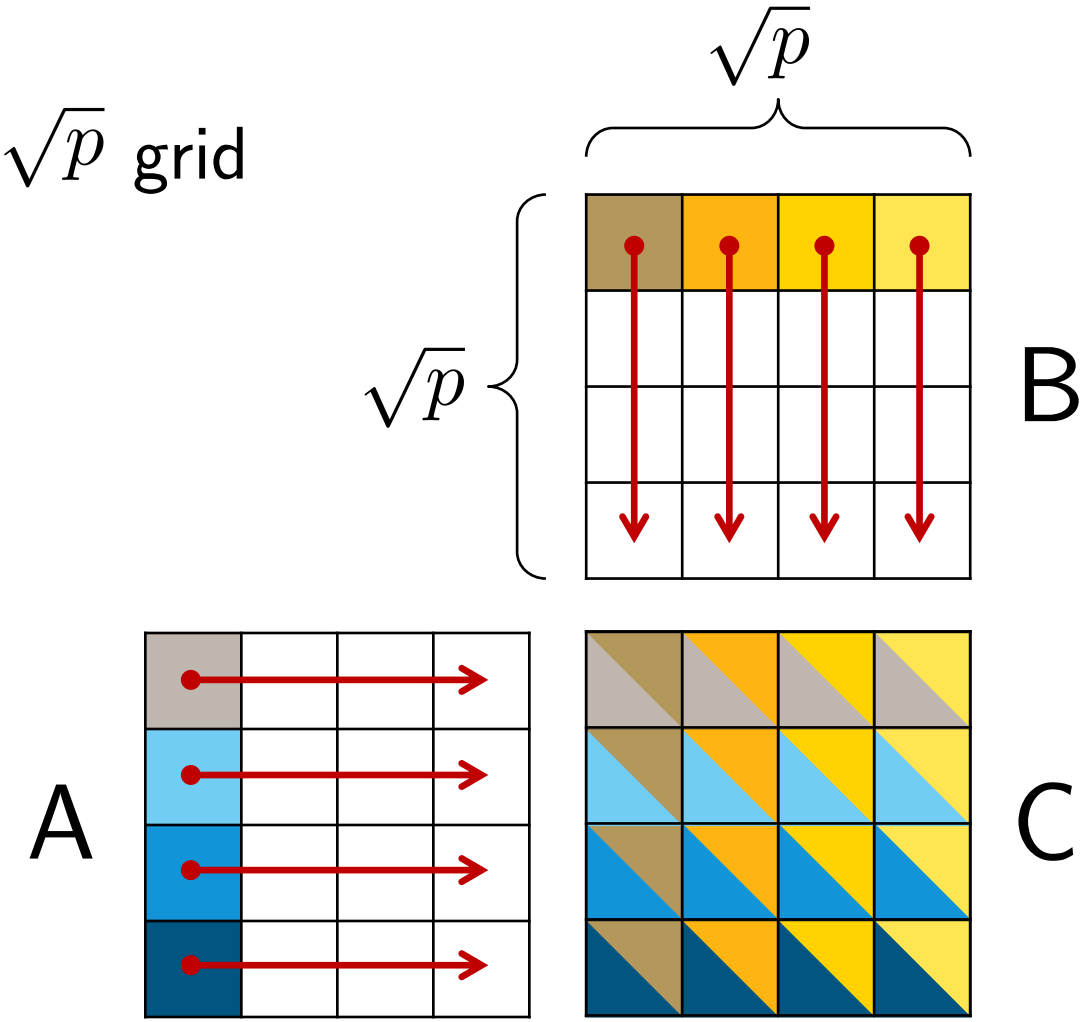


2D SUMMA Algorithm

- p=16
- 2D layout, $\sqrt{p} \times \sqrt{p}$ grid
- $\sqrt{p}$ outer products

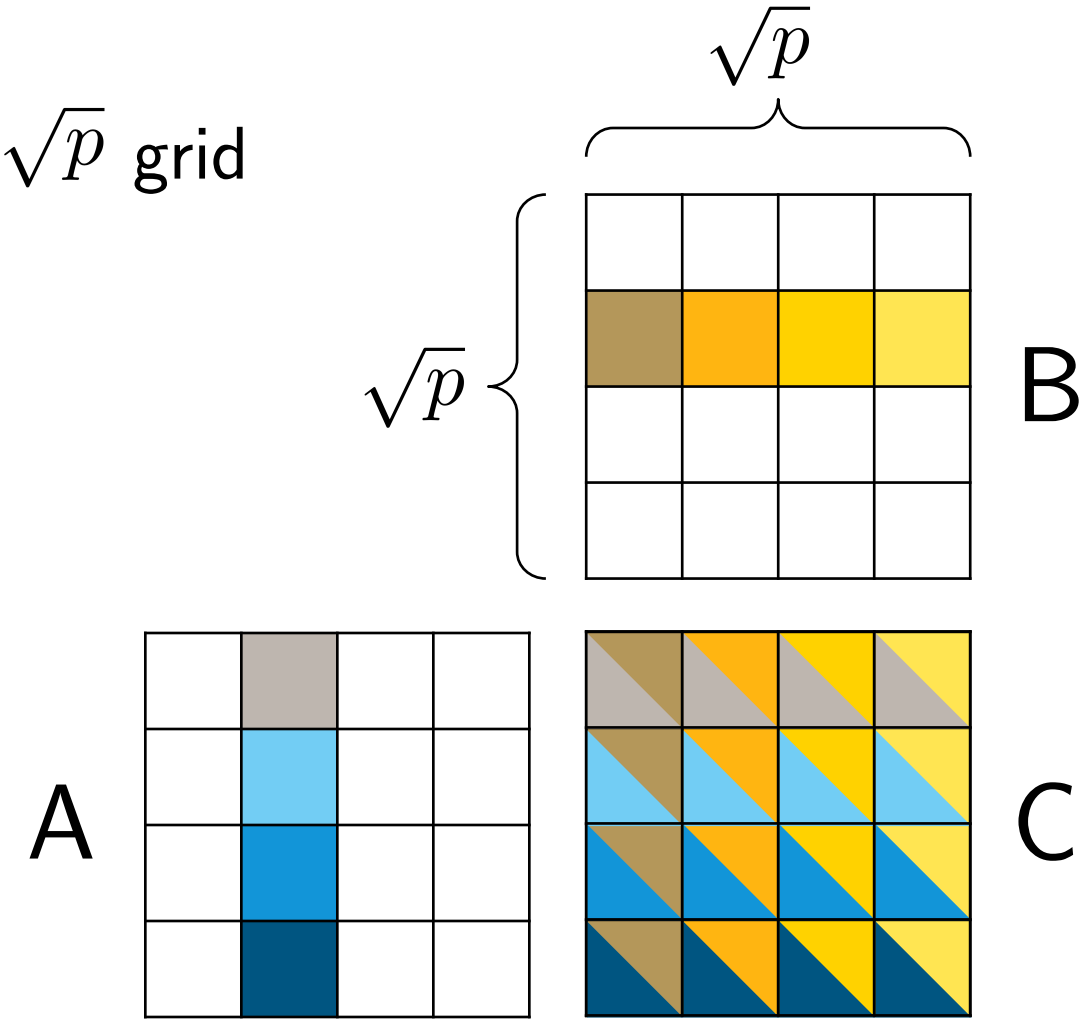
A broadcasts row-wise

B broadcasts column-wise



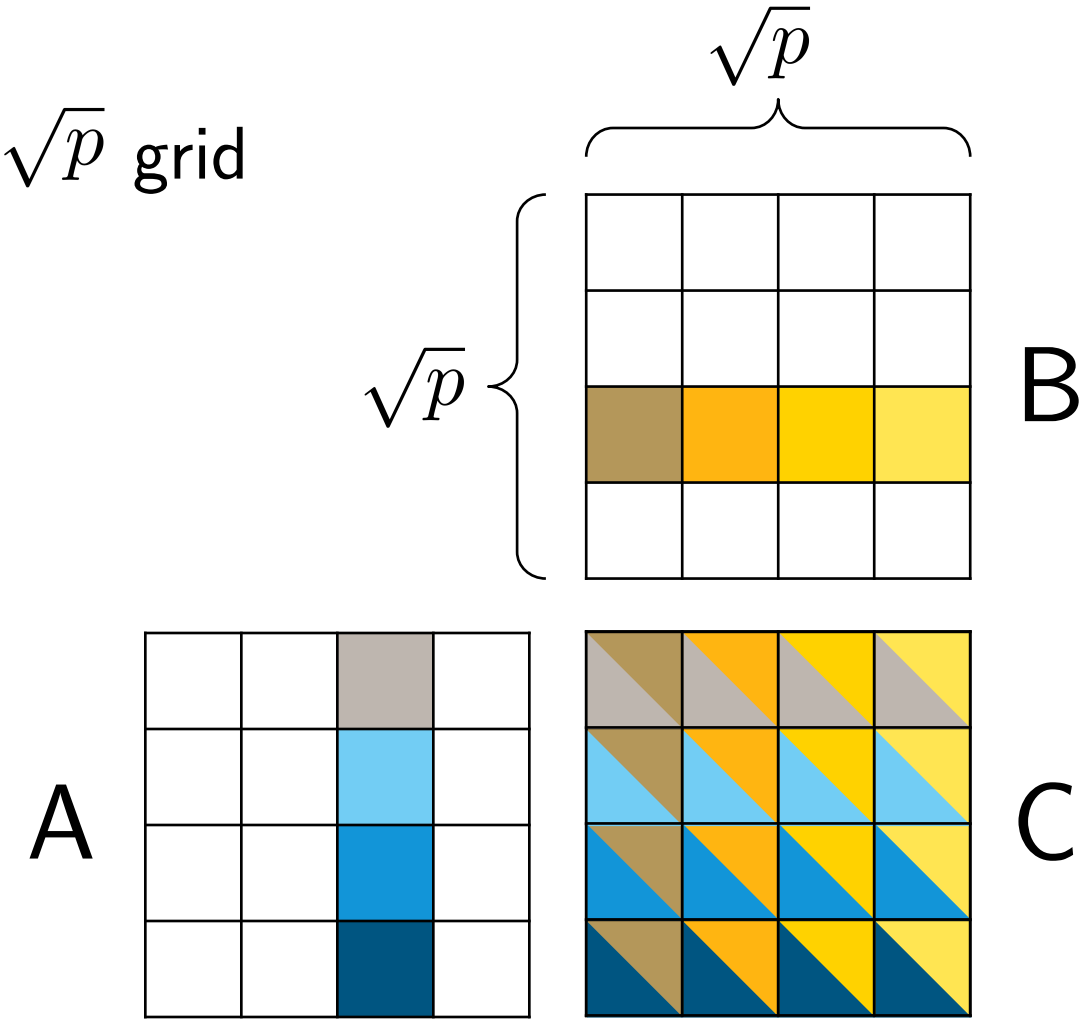
2D SUMMA Algorithm

- p=16
- 2D layout, $\sqrt{p} \times \sqrt{p}$ grid
- $\sqrt{p}$ outer products



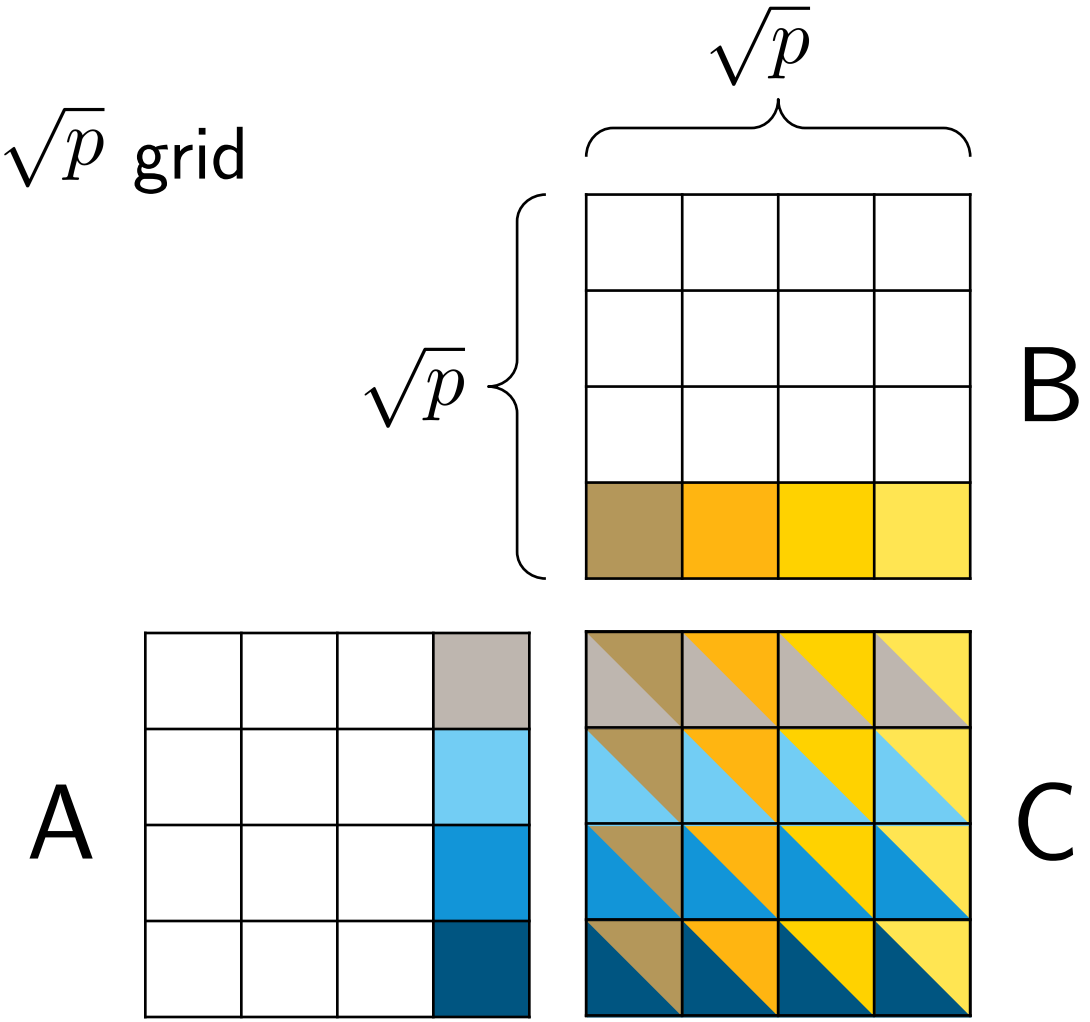
2D SUMMA Algorithm

- p=16
- 2D layout, $\sqrt{p} \times \sqrt{p}$ grid
- $\sqrt{p}$ outer products



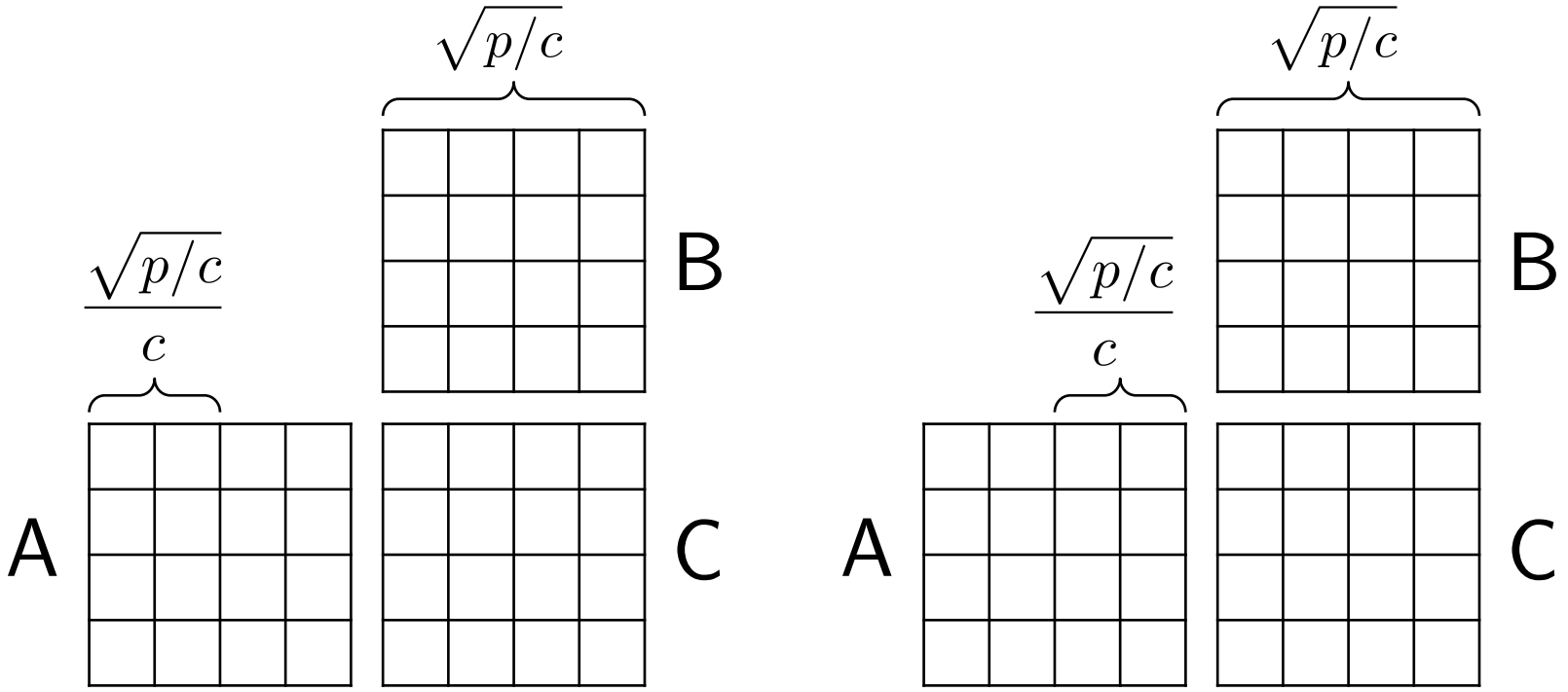
2D SUMMA Algorithm

- p=16
- 2D layout, $\sqrt{p} \times \sqrt{p}$ grid
- $\sqrt{p}$ outer products
- $\sqrt{p}$ broadcasts of size $\mathrm{nnz}(\mathrm{A})/p$
- $\sqrt{p}$ broadcasts of size $\mathrm{nnz}(\mathrm{B})/p$



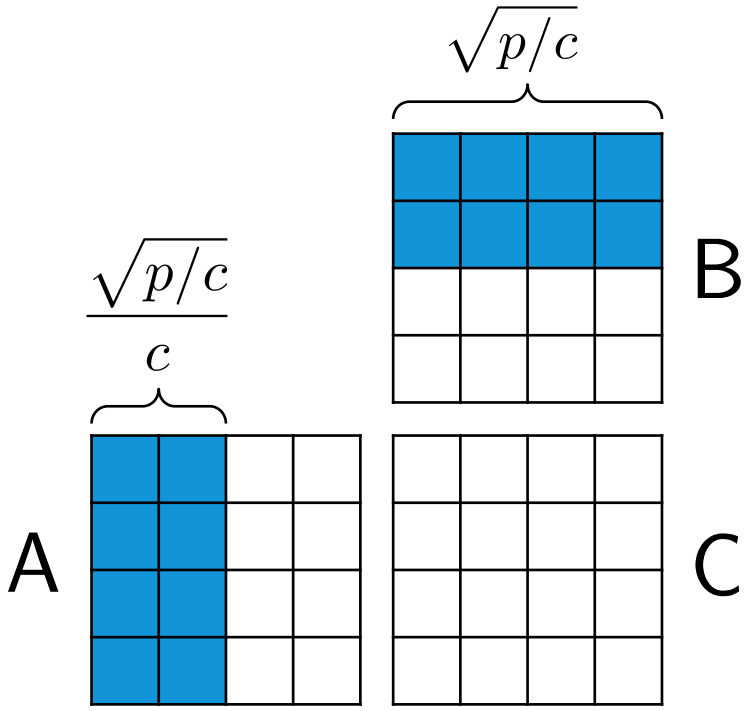
3D SUMMA Algorithm

- p=32, c=2 (c is #copies)
- $\sqrt{p/c} \times \sqrt{p/c} \times c$ grid (teams of size c)

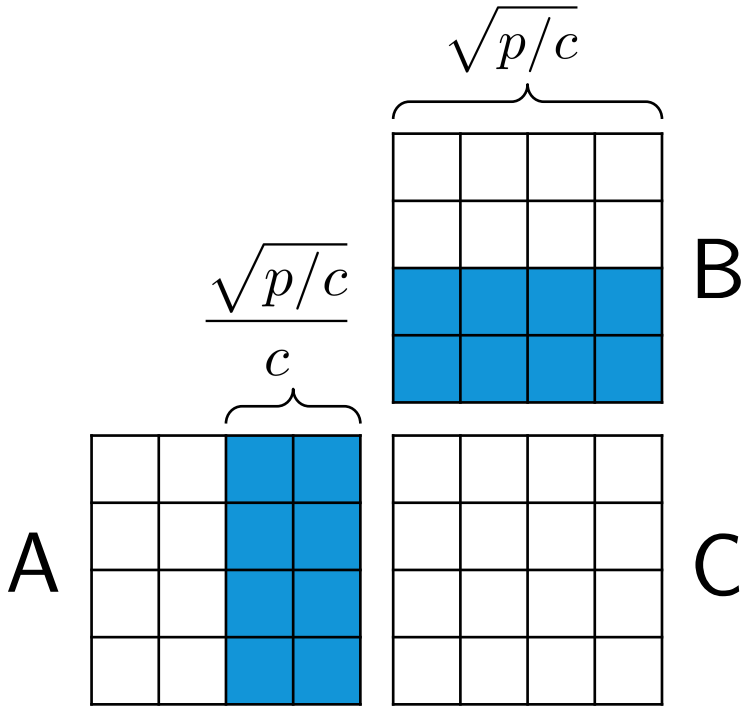


3D SUMMA Algorithm

- p=32, c=2 (c is #copies)
- $\sqrt{p/c} \times \sqrt{p/c} \times c$ grid (teams of size c)



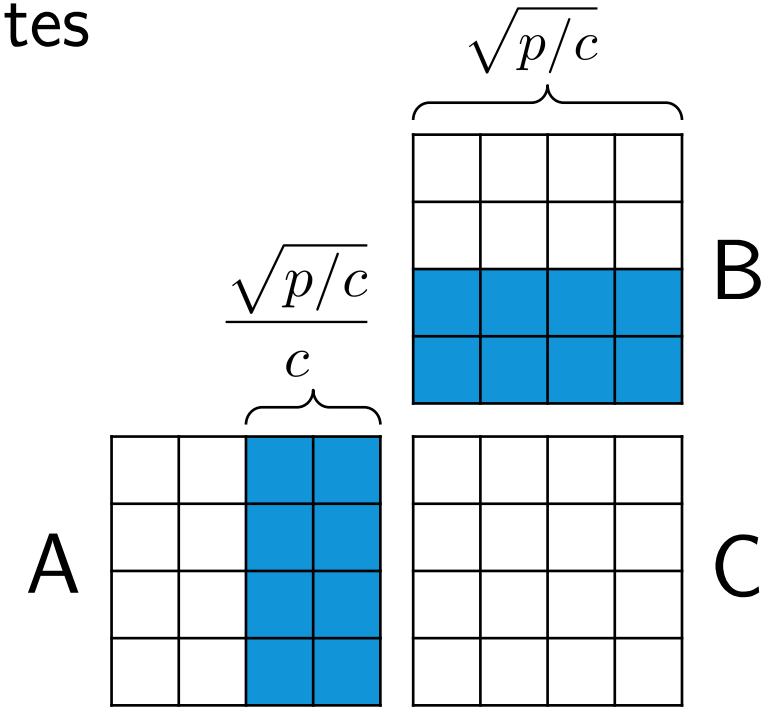
First team member calculates the first half outer products.



Second team member calculates the latter half outer products.

3D SUMMA Algorithm

- p=32, c=2 (c is #copies)
- $\sqrt{p/c} \times \sqrt{p/c} \times c$ grid (teams of size c)
- Each team member calculates $1/c$ of all outer products
- $\sqrt{p}/c^{3/2}$ broadcasts of size $\mathrm{nnz}(\mathrm{A})c/p$
- $\sqrt{p}/c^{3/2}$ broadcasts of size $\mathrm{nnz}(\mathrm{B})c/p$



Second team member calculates the latter half outer products.

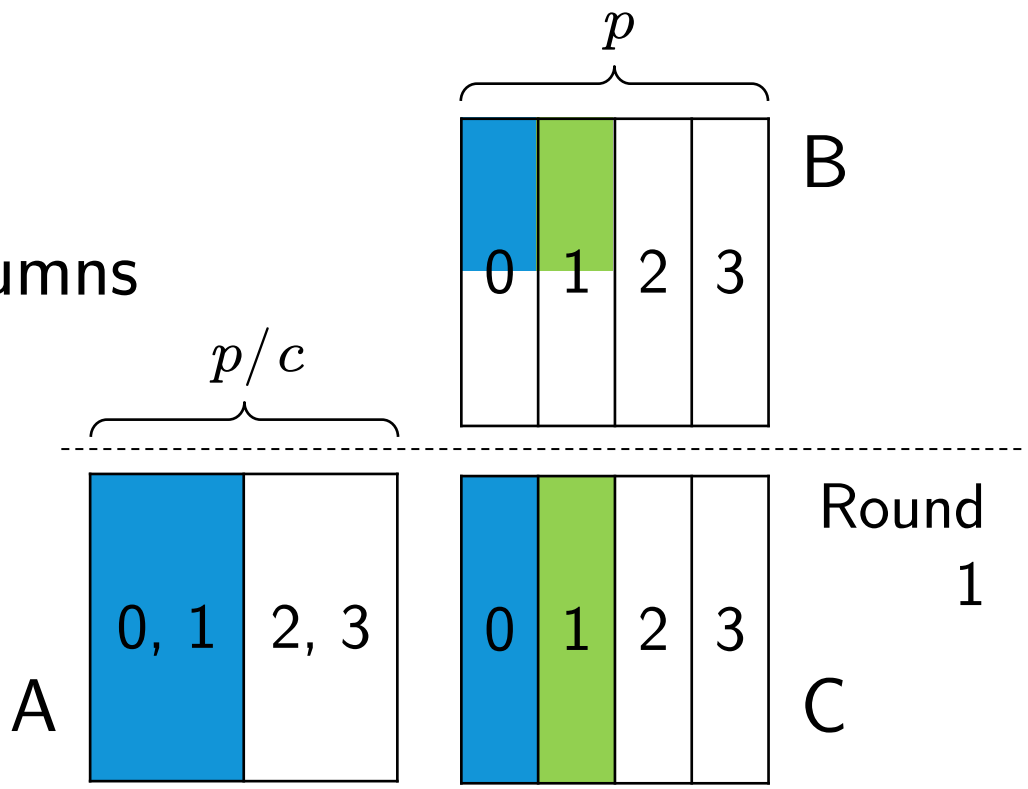
3D SUMMA: Costs

	#messages	#words
Replication		
– Replicate A:	$2 \log c$	$\times \frac{\text{nnz}(A)c}{p}$
– Replicate B:	$2 \log c$	$\times \frac{\text{nnz}(B)c}{p}$
Propagation		
– Broadcast A:	$\frac{\sqrt{p}}{c^{3/2}} \log \sqrt{\frac{p}{c}}$	$\times \frac{\text{nnz}(A)c}{p}$
– Broadcast B:	$\frac{\sqrt{p}}{c^{3/2}} \log \sqrt{\frac{p}{c}}$	$\times \frac{\text{nnz}(B)c}{p}$
Collection		
– Reduce C:	$\log c$	$\times \frac{\text{nnz}(C)c}{p}$

1.5D Algorithms

1.5D Col A

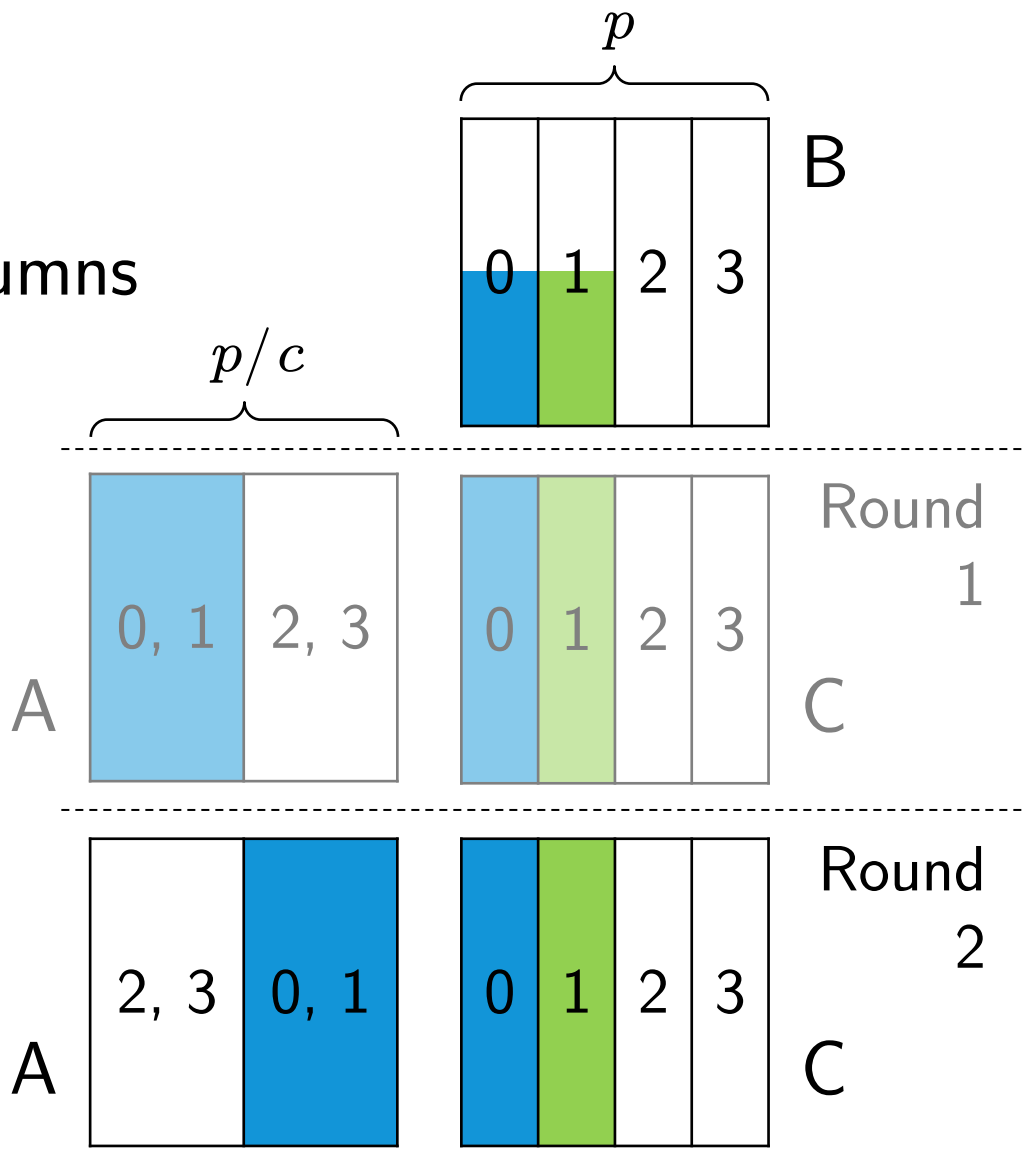
- $p{=}4,\ c{=}2$
- A has p/c block columns



1.5D Col A

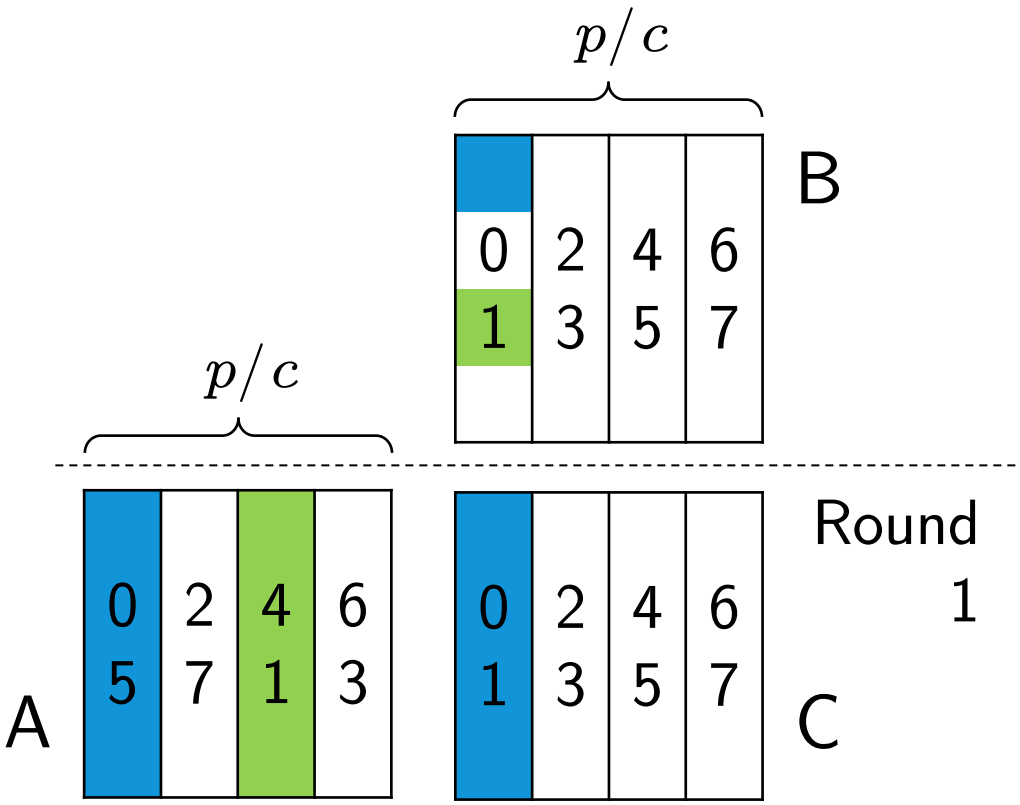
- $p{=}4,\ c{=}2$
- A has p/c block columns

- #msgs: p/c
- msg size: $\frac{\mathrm{nnz}(A)c}{p}$
- #words: $\mathrm{nnz}(A)$



1.5D Col ABC

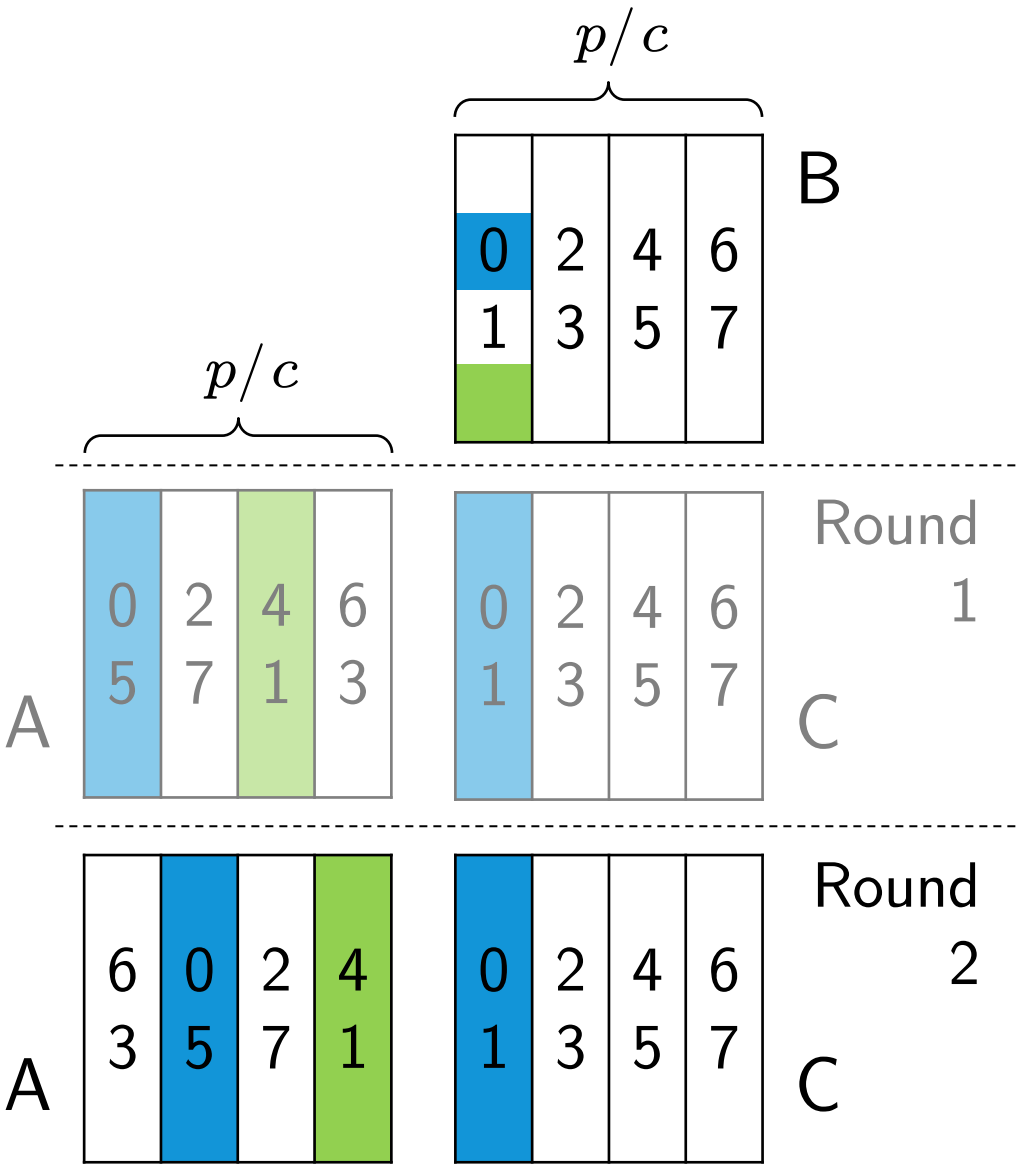
- $p{=}8,\ c{=}2$
- A and B both have p/c block columns
- Each team member does $1/c$ of work



1.5D Col ABC

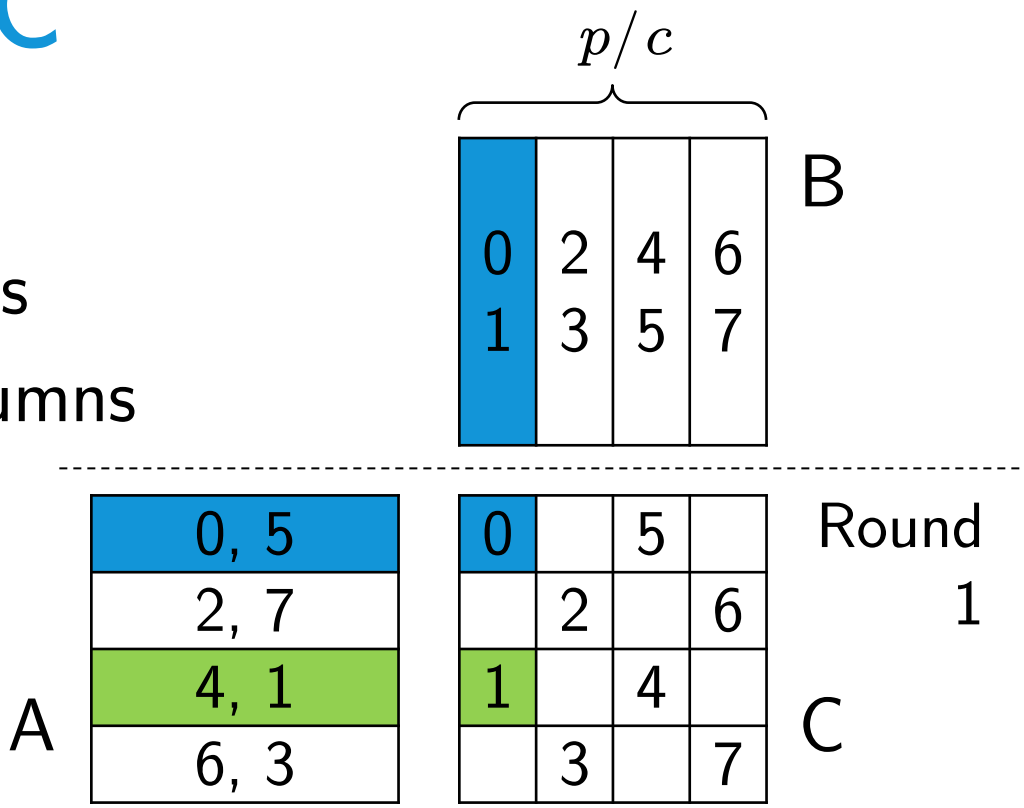
- p=8, c=2
- A and B both have p/c block columns
- Each team member does $1/c$ of work

- #msgs: p/c^2
- msg size: $\frac{\text{nnz}(A)c}{p}$
- #words: $\frac{\text{nnz}(A)}{c}$



1.5D Inner ABC

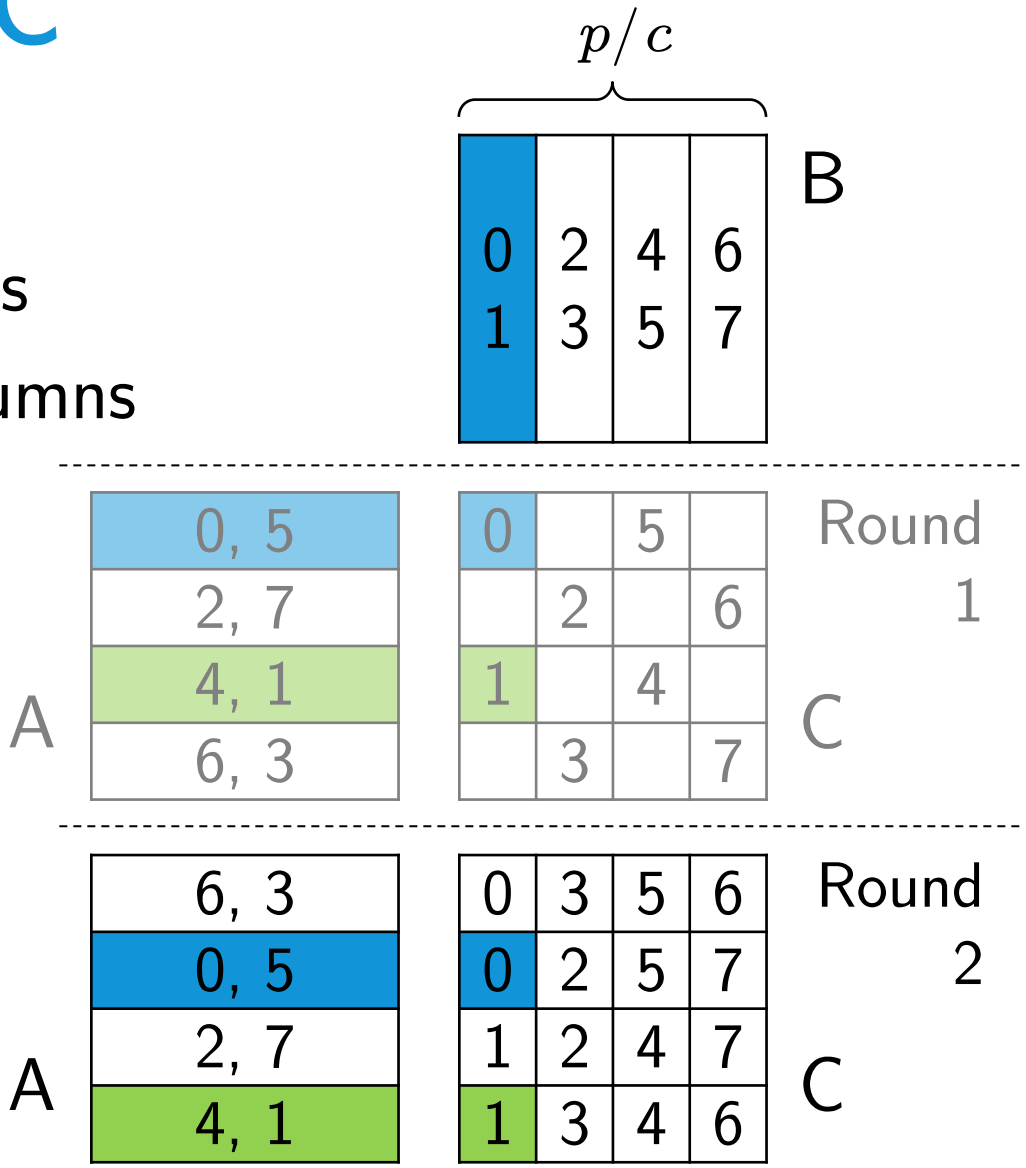
- p=8, c=2
- A has p/c block rows
- B has p/c block columns
- Each team member does $1/c$ of work



1.5D Inner ABC

- p=8, c=2
- A has p/c block rows
- B has p/c block columns
- Each team member does $1/c$ of work

- #msgs: p/c^2
- msg size: $\frac{\text{nnz}(A)c}{p}$
- #words: $\frac{\text{nnz}(A)}{c}$



Cost Comparison

- Latency cost

Algorithms	A		B		C
	Replication	Propagation	Replication	Propagation	Collection
3D SUMMA ABC	$2\log c$	$\frac{\sqrt{p}}{c^{3/2}}\log\sqrt{\frac{p}{c}}$	$2\log c$	$\frac{\sqrt{p}}{c^{3/2}}\log\sqrt{\frac{p}{c}}$	$\log c$
1.5D Col A	$2\log c$	$\frac{p}{c}$	-	-	-
1.5D Col ABC	$2\log c$	$\frac{p}{c^2}$	$2\log c$	-	$\log c$
1.5D Inner ABC	$2\log c$	$\frac{p}{c^2}$	$2\log c$	-	$\log c$

- Append $\times\frac{\mathrm{nnz}(A)c}{p}$, $\times\frac{\mathrm{nnz}(B)c}{p}$, $\times\frac{\mathrm{nnz}(C)c}{p}$ for bandwidth cost
- Even replication and collection can be the bottleneck if $\mathrm{nnz(A)} \ll \mathrm{nnz(B)}, \mathrm{nnz(C)}$

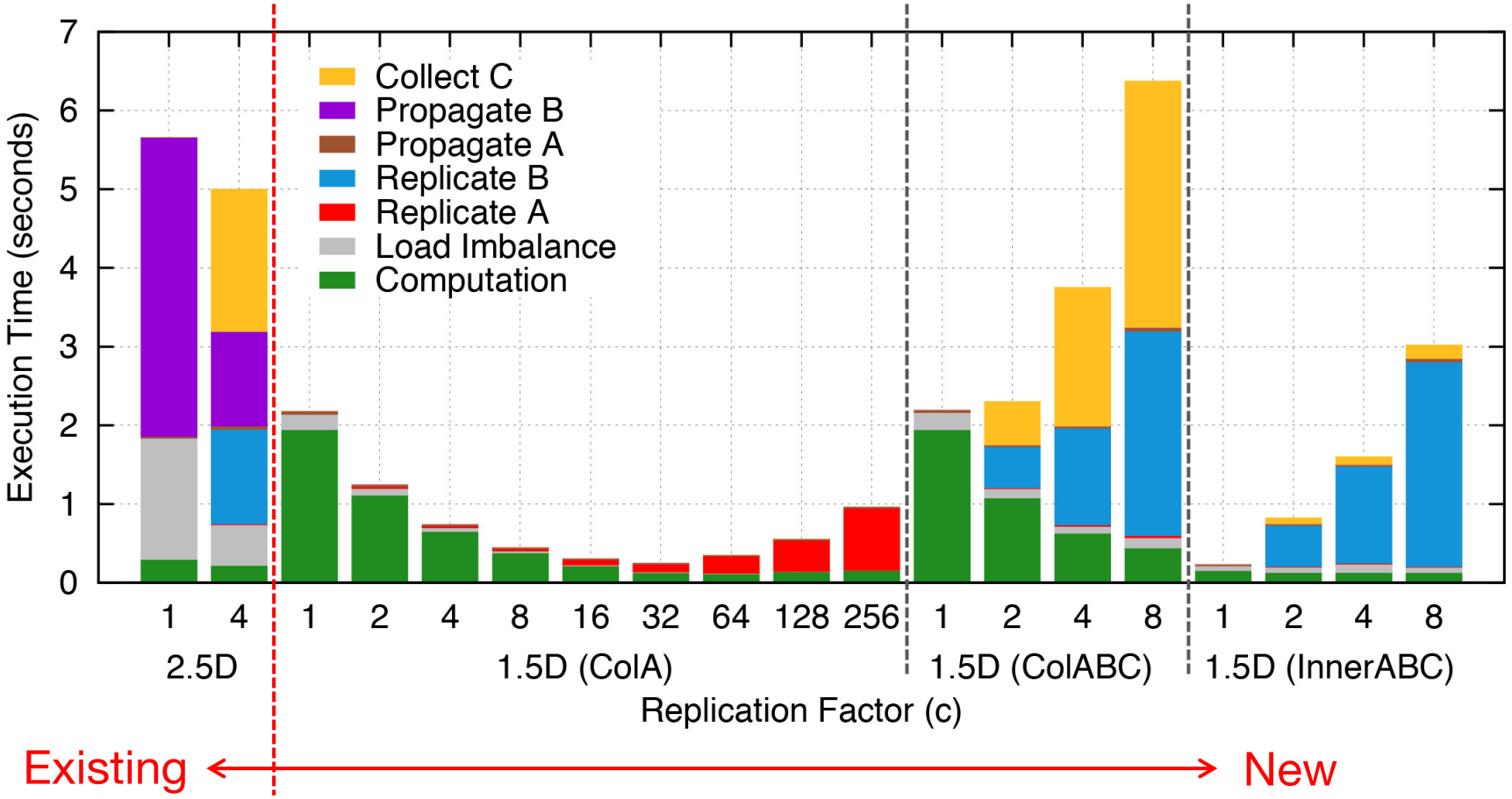
Experimental Results

Test environments

- Comparison
  - **1.5D variants:** Col A, Col ABC, and Inner ABC
  - **Baseline:** Best of 2D/2.5D/3D SUMMA ABC
- C++ MPI, hybrid (12 threads per MPI process)
- MKL's multithreaded routine for local computation (mkl_dcsrmm) (double-precision)
- CSR + row major
- **Edison @NERSC**
  - 2.57PFLOPS Cray XC30
  - 12 cores, 2 sockets each
  - Cray Aries, Dragonfly topology

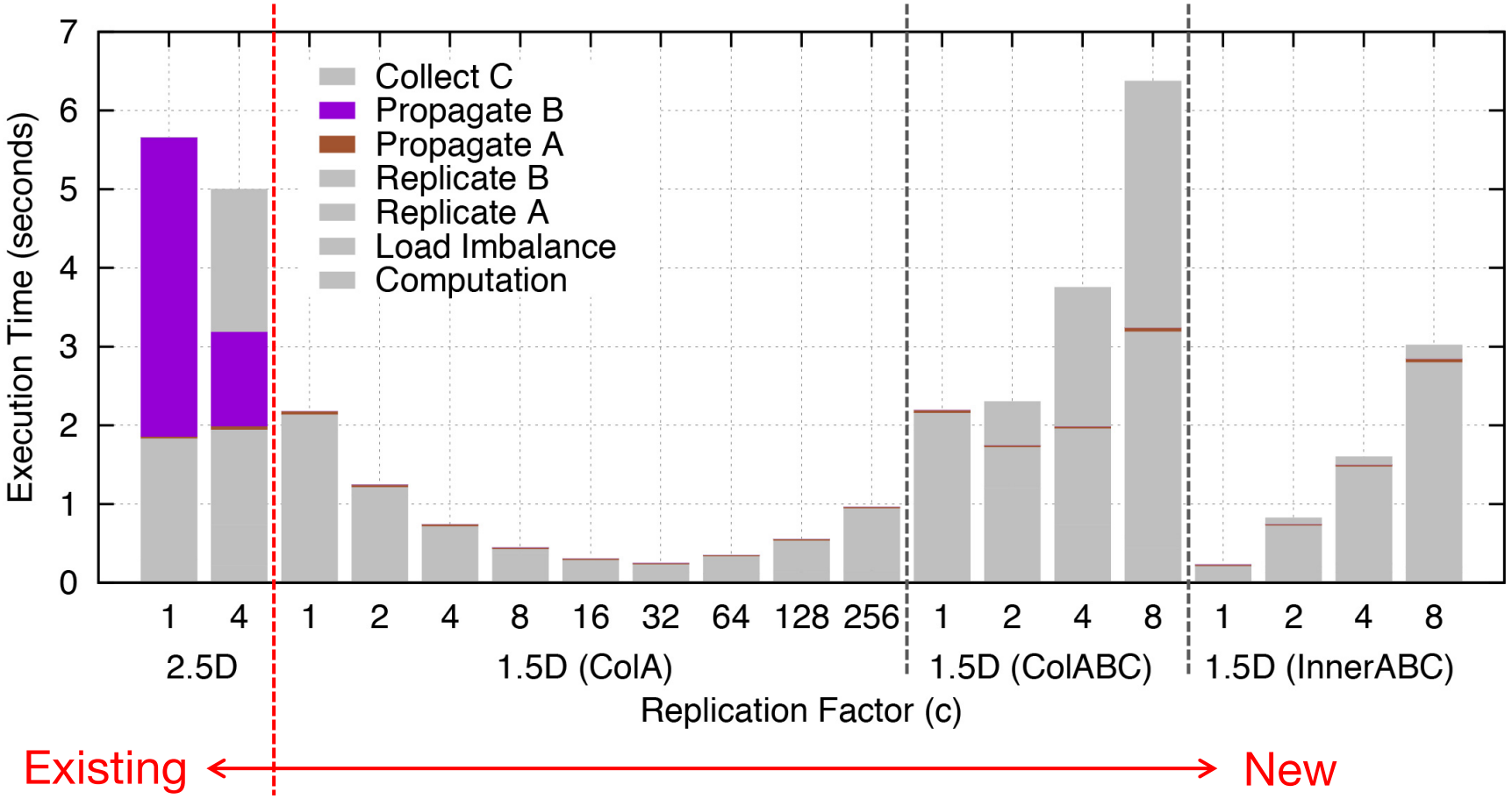
Cost Breakdown

- p=3,072, n=64k x 64k, 41 nnz per row, Cray XC30



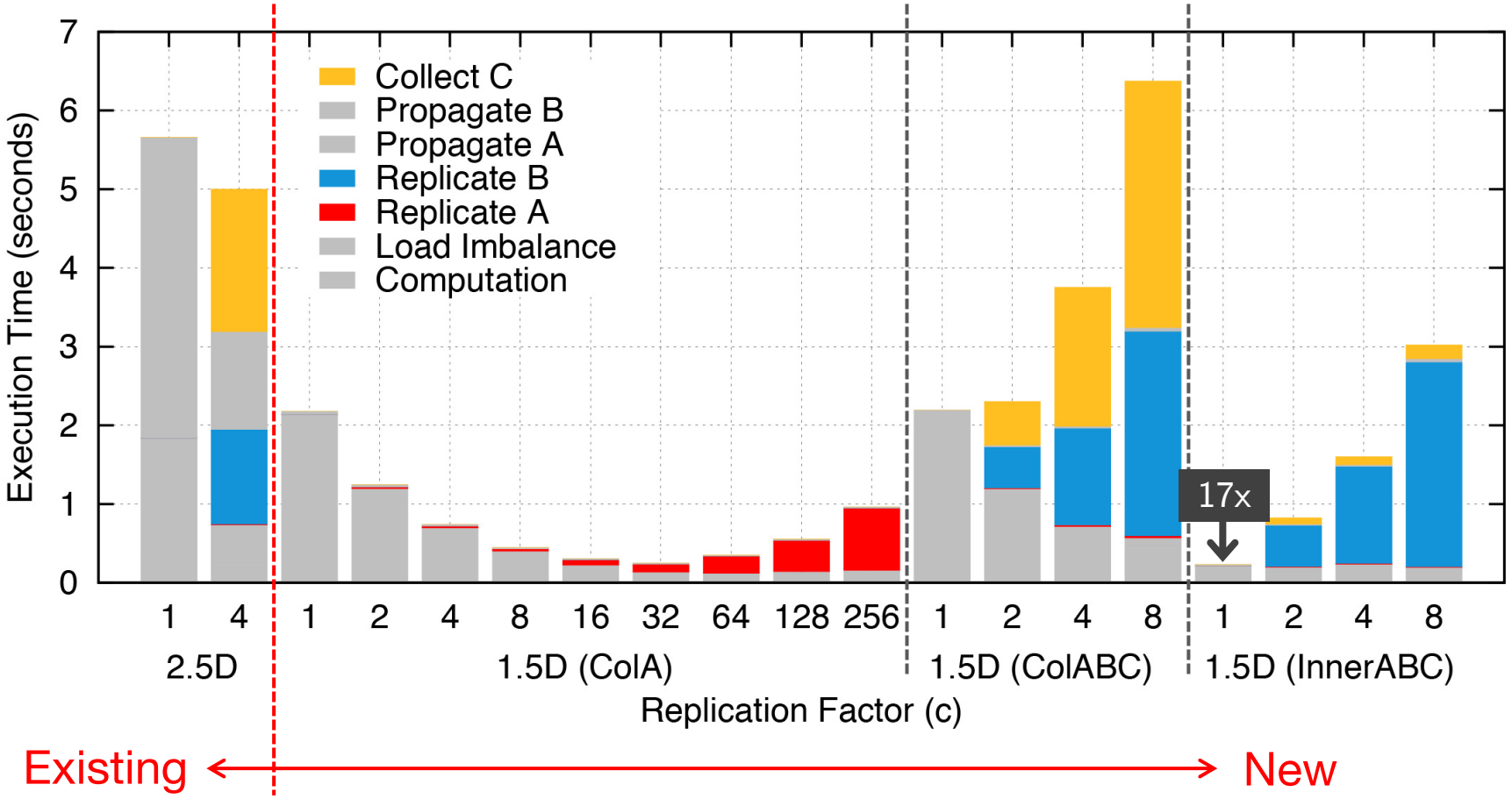
Moving Dense Matrix is Costly

- p=3,072, n=64k x 64k, 41 nnz per row, Cray XC30



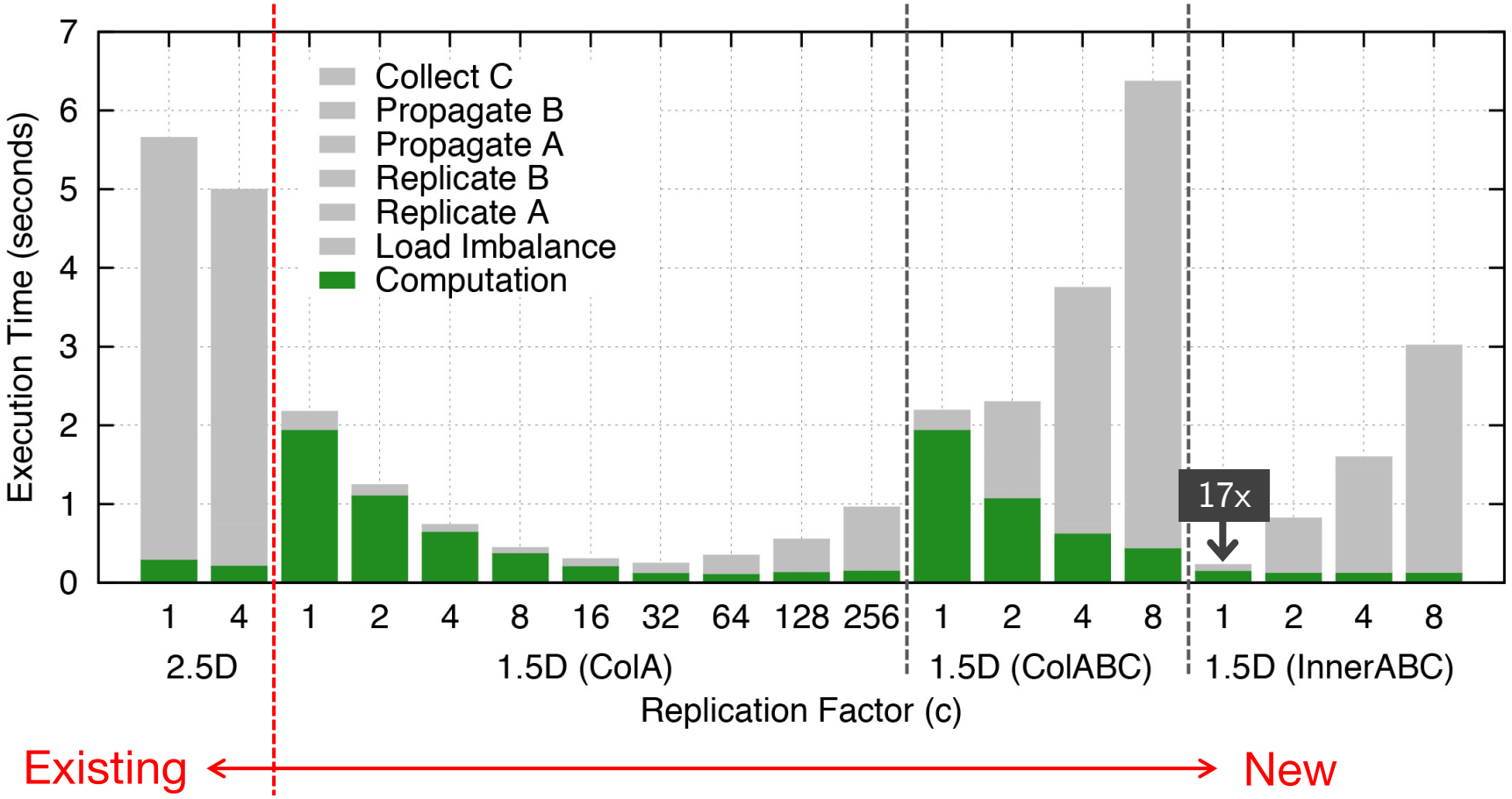
Replication might not pay off

- p=3,072, n=64k x 64k, 41 nnz per row, Cray XC30



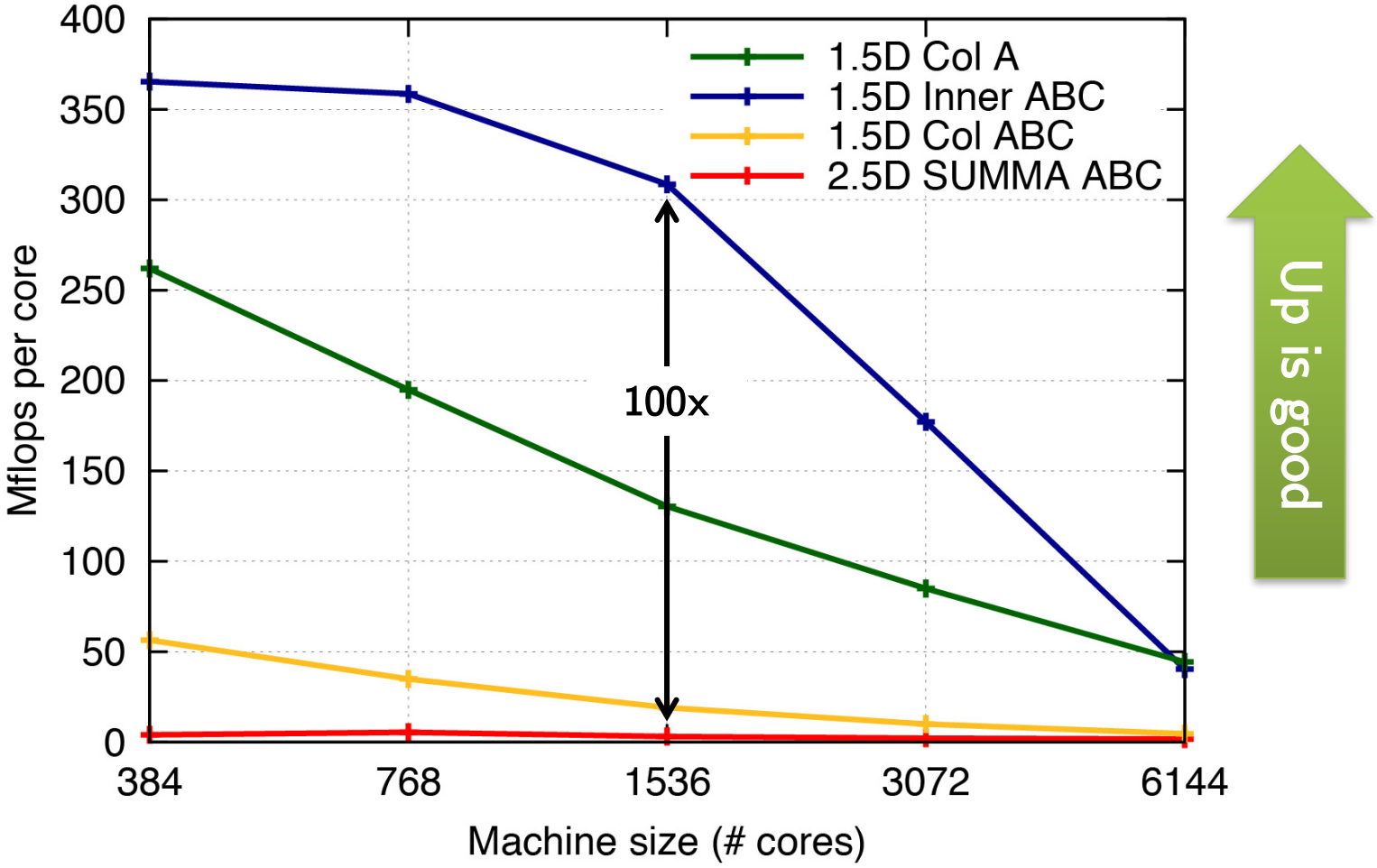
Different Computation Efficiency

- p=3,072, n=64k x 64k, 41 nnz per row, Cray XC30



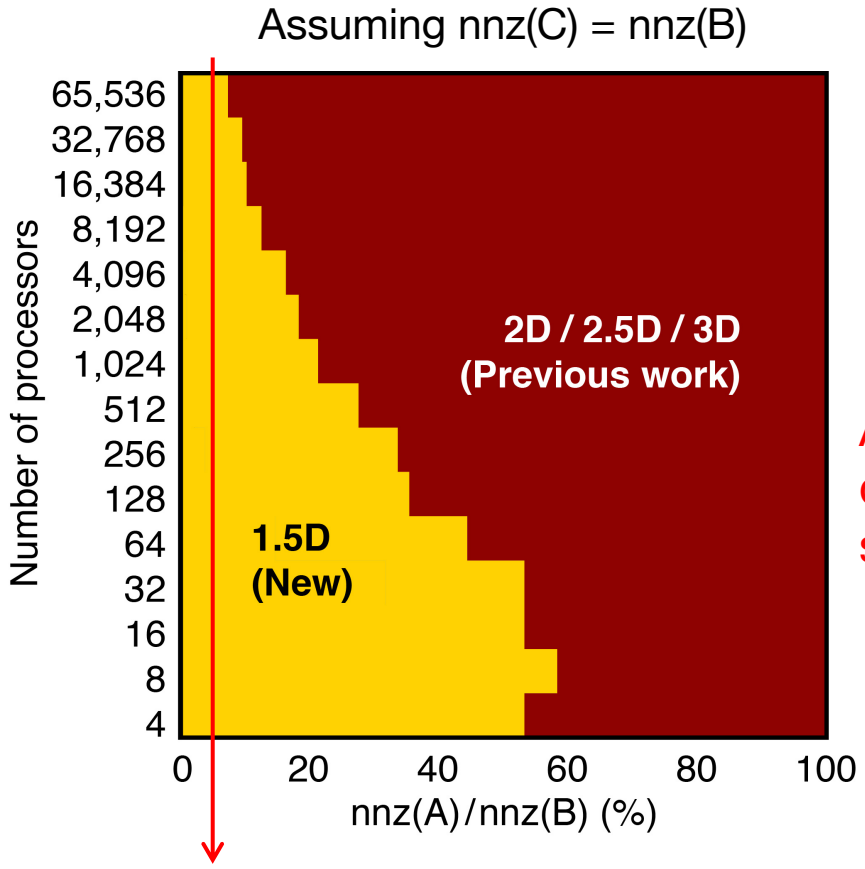
100x Improvement

- $A^{66k \times 172k}$, $B^{172k \times 66k}$, 0.0038% nnz, Cray XC30



When is 1.5D better?

- Best theoretical bandwidth cost areas



Also applicable to dense-dense and sparse-sparse

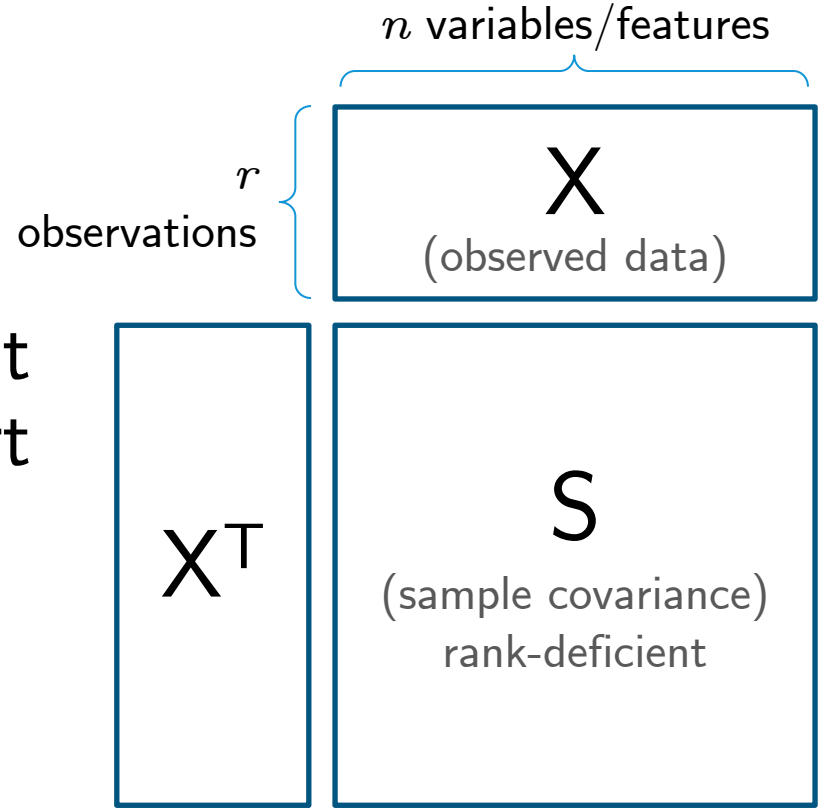
Most sparse matrices have < 5% nonzeroes

Iterative Multiplication

Sparse ICM Estimation

*[S. Oh et al 2014]

- ICM = inverse covariance matrix → Ω
  - Direct pairwise dependency
- Data could be
  - Brain fMRI scan
  - Gene expression
- S is usually rank-deficient or too expensive to invert directly
- Estimate Ω by setting an objective function and optimize



CONCORD-ISTA

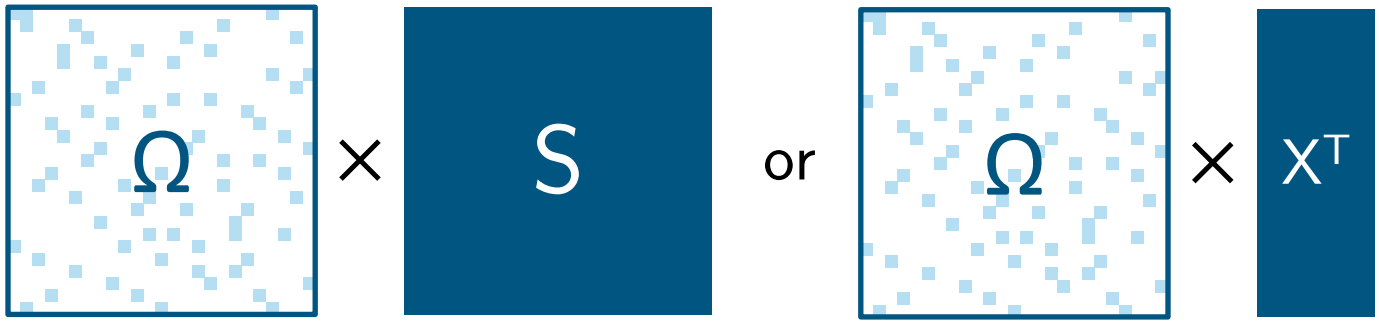
- **CONCORD** objective function

Ω_D : diagonal elements
Ω_X : off-diagonal elements

$$Q(\Omega) = \frac{n}{2}[-\log(\det \Omega_D)^2 + \mathrm{tr}(S\Omega^2) + \lambda\|\Omega_X\|_1]$$

Ω: estimated sparse inverse of S

- Iteratively compute ΩS [or $(\Omega X^T)X$]



- Replication (and sometimes collection) costs can be amortized.

Conclusions

- 1.5D beneficial when #nnz of matrices are **imbalanced**.
  - Also applicable to **dense-dense/sparse-sparse.**
- Observed up to **100×** speedup.
- 1.5D can handle moderately imbalanced load with **greedy partitioning**.
- **Iterative multiplication** gains even more benefit.
- Future work
  - **Parallel CONCORD-ISTA**
  - New communication **lower bounds**
  - Try **recursive** approach
  - Consider **different computation efficiency**

References I

- P. Koanantakool, A. Azad, A. Buluç, D. Morozov, S. Oh, L. Oliker, K. Yelick.
  Communication-avoiding parallel sparse-dense matrix-matrix multiplication.
  In IPDPS 2016.
- E. Solomonik and J. Demmel,
  Communication-optimal parallel 2.5D matrix multiplication and LU factorization algorithms.
  In Euro-Par, 2011, vol. 6853, pp. 90–109.
- G. Ballard, A. Buluc, J. Demmel, L. Grigori, B. Lipshitz, O. Schwartz, and S. Toledo.
  Communication optimal parallel multiplication of sparse random matrices.
  In Proceedings of the twenty-fifth annual ACM symposium on Parallelism in algorithms and architectures. ACM, 2013, pp. 222–231.

References II

- M. Driscoll, E. Georganas, P. Koanantakool, E. Solomonik, K. Yelick.
  A communication-optimal n-body algorithm for direct interactions.
  In IPDPS 2013.
- S. Oh, O. Dalal, K. Khare, and B. Rajaratnam,
  Optimization methods for sparse pseudo-likelihood graphical model selection.
  In NIPS 27, 2014, pp. 667–675.

Thank you!

Questions?

Suggestions/Applications?

Communication Costs

- 3 phases

Phase	Description	As c increases..
Replication	to have c copies	▲
Propagation	to multiply	▼
Collection	of matrix C	▲

- Non-replicating version only has propagation cost.
- Replication and collection happen between team members, i.e., only c processors ($c \ll p$), so they are usually cheaper than propagation.

CONCORD-ISTA

- **CONCORD** objective function

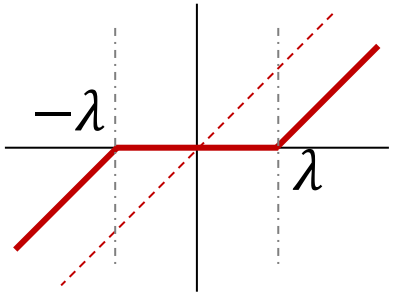
non-smooth

$$Q(\Omega) = \frac{n}{2}[-\log(\det \Omega_D)^2 + \mathrm{tr}(S\Omega^2) + \lambda\|\Omega_X\|_1]$$

Ω: estimated sparse inverse of S

Ω_D : diagonal elements
Ω_X : off-diagonal elements

- Proximal gradient method
  - **Smooth:** Gradient descent
  - **Non-smooth:** Soft-thresholding
    (**ISTA**: Iterative Soft-Thresholding Algorithm)



- Does not assume Gaussian distribution

Pseudocode

Ω: estimated sparse inverse of S

```
S = X^T X / r
Ω = I
do
    Ω_0 = Ω
    Compute gradient G
    try τ = {1, 1/2, 1/4, ...}  // find appropriate step size
        Ω = soft_thresholding(Ω_0 - τG, τλ)
    until descent condition satisfied
while max(|Ω_0 - Ω|) ≥ ε  // until no more progress
```

- Distributed operations
  - $S = X^T X / r$: dense-dense matrix multiplication
  - $S\Omega + \Omega S$: sparse-dense matrix multiplication