

Nyström Low-Rank Approximation: Communication Lower Bounds and Algorithms

Hussam AL DAAS¹, Grey BALLARD², Laura GRIGORI³, Md Taufique HUSSAIN², Suraj KUMAR⁴,
Md Marufur RAHMAN² and Kathryn ROUSE⁵

¹Rutherford Appleton Laboratory, UK

²Wake Forest University, USA

³Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland

⁴Inria & ENS Lyon, France

⁵Inmar Intelligence, USA

19th Scheduling for large-scale systems workshop

Communication lower bounds for a computation on P processors

minimum amount of communication required to perform the computation in parallel

Should avoid such settings

- If all operations are performed on a single processor, then no communication is required
- If data is duplicated on all processors, then communication may not be necessary

Our assumptions

- Load balanced computation: each processor performs equal amount of operations
- There is one copy of data in the system

- There exists one processor that has at most $1/P$ th amount of overall data – we examine the required data transfers for this processor to establish the lower bound

Communication costs

- Communication cost: latency cost (number of messages) and bandwidth cost (volume of data transfers)
- Focus on bandwidth costs – it dominates when messages are large

Communication cost of an algorithm

- Cost along the critical path – the path along which the communication (bandwidth) cost is maximum

Focus on $A\Omega$ and $\Omega^T A\Omega$ computations

$B = A\Omega$ computation

- $A \in \mathbb{R}^{n_1 \times n_2}$ and Ω is a $n_2 \times r$ random matrix
- Fundamental to all randomized linear algebra techniques including randomized SVD

$B = A\Omega$ and $C = \Omega^T A\Omega$ computations together

- A is a $n \times n$ matrix and Ω is a $n \times r$ random matrix
- Useful for Nyström approximation, $\tilde{A} = (A\Omega)(\Omega^T A\Omega)^\dagger (A\Omega)^T$

r is (much) less than the contracted dimension ($r \leq n_2$ and $r \leq n$).

Table of Contents

- 1 Existing results for classical matrix multiplication
- 2 $A\Omega$ computation
- 3 Computations of Nyström approximation
- 4 Experimental results
- 5 Conclusion

Table of Contents

- 1 Existing results for classical matrix multiplication
- 2 $A\Omega$ computation
- 3 Computations of Nyström approximation
- 4 Experimental results
- 5 Conclusion

Classical matrix multiplication bound

- $C = AB$ with $A \in \mathbb{R}^{n_1 \times n_2}$, $B \in \mathbb{R}^{n_2 \times n_3}$, and $C \in \mathbb{R}^{n_1 \times n_3}$

Let $k = \min(n_1, n_2, n_3) \leq \ell = \text{median}(n_1, n_2, n_3) \leq m = \max(n_1, n_2, n_3)$.

- ϕ_A, ϕ_B, ϕ_C : number of elements accessed by a processor in arrays A, B, C
- Minimum number of elements accessed in the critical path = $\phi_A + \phi_B + \phi_C$

$$\begin{array}{l} \text{1} \quad \phi_A + \phi_B + \phi_C = kl + \frac{km + lm}{P} \\ \frac{m}{\ell} \phi_A + \phi_B + \phi_C = 2\left(\frac{k^2 \ell m}{P}\right)^{1/2} + \frac{\ell m}{P} \\ \frac{\ell m}{k^2} \quad \phi_A + \phi_B + \phi_C = 3\left(\frac{k \ell m}{P}\right)^{2/3} \end{array}$$

- Communication lower bound = $\phi_A + \phi_B + \phi_C - \frac{n_1 n_2 + n_2 n_3 + n_1 n_3}{P}$

Can we improve this bound when one matrix is randomly generated?

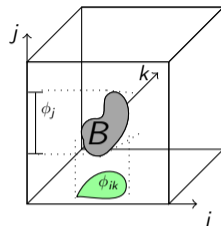
Table of Contents

- 1 Existing results for classical matrix multiplication
- 2 $A\Omega$ computation
 - Communication lower bounds
 - Optimal algorithms
- 3 Computations of Nyström approximation
 - $A\Omega$ and $\Omega^T A\Omega$ computations together
 - Optimal algorithms
- 4 Experimental results
- 5 Conclusion

Approach to obtain communication lower bounds

A special case of Hölder-Brascamp-Lieb (HBL) inequality:

- For the 3d object B , $\phi_j \phi_{ik} \geq \text{Volume}(B)$
- ϕ_j, ϕ_{ik} : projection sizes of B on j -axis and i - k plane
- Also implies that: $\phi_{ij} \phi_{ik} \geq \text{Volume}(B)$ and $\phi_{jk} \phi_{ik} \geq \text{Volume}(B)$



Constraints for load balanced computation

- $B = A\Omega$ with $A \in \mathbb{R}^{n_1 \times n_2}$, $B \in \mathbb{R}^{n_1 \times r}$, and Ω is a random matrix

for $i = 1:n_1$, for $k = 1:n_2$, for $j = 1:r$

$$B[i][j]_+ = A[i][k] * \Omega[k][j]$$

- ϕ_A, ϕ_B : projection sizes of computation on arrays A, B
- From the HBL inequality: $\phi_A \phi_B \geq \text{number of multiplications per processor} = \frac{n_1 n_2 r}{P}$
- Extra constraints: $\frac{n_1 n_2}{P} \leq \phi_A \leq n_1 n_2$, $\frac{n_1 r}{P} \leq \phi_B \leq n_1 r$

Optimization problem and communication lower bounds

Minimize $\phi_A + \phi_B$ s.t.

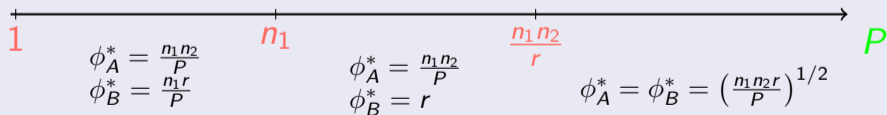
$$\phi_A \phi_B \geq \frac{n_1 n_2 r}{P}$$

$$\frac{n_1 n_2}{P} \leq \phi_A \leq n_1 n_2$$

$$\frac{n_1 r}{P} \leq \phi_B \leq n_1 r$$

Amount of non-random array accesses = $\phi_A + \phi_B$

- Estimate the solution and prove optimality using all Karush–Kuhn–Tucker conditions are satisfied
- For $n_2 \geq r$,



- Communication lower bound = $\phi_A + \phi_B$ – data owned by the processor = $\phi_A + \phi_B - \frac{n_1 n_2 + n_1 r}{P}$

Table of Contents

- 1 Existing results for classical matrix multiplication
- 2 $A\Omega$ computation
 - Communication lower bounds
 - Optimal algorithms
- 3 Computations of Nyström approximation
 - $A\Omega$ and $\Omega^T A\Omega$ computations together
 - Optimal algorithms
- 4 Experimental results
- 5 Conclusion

A naive way to minimize communication cost for $B = A\Omega$

$$\begin{pmatrix} B_1 \\ B_2 \\ B_3 \end{pmatrix} = \begin{pmatrix} A_1 \\ A_2 \\ A_3 \end{pmatrix} \cdot \Omega$$

Structure of the algorithm for each processor i :

- owns A_i row block ($1/P$ th portion of A), generates random matrix Ω , and performs $B_i = A_i \cdot \Omega$

When P is small, communication cost of the algorithm is 0 (**Optimal**).

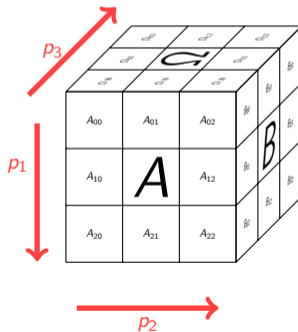
What about when P is large?

Design of communication optimal algorithms for $B = A\Omega$

Arrangements of 8 processors



- P is organized into $p_1 \times p_2 \times p_3$ logical grid
- Select p_1, p_2 and p_3 based on the communication lower bounds
- Allgather A on the set of processors along each slice of p_3
- Generate required portion of Ω on each processor
- Perform local computation
- Perform Reduce-Scatter along p_2 to obtain B



Matrix multiplication with a random matrix

Require: Π is a $p_1 \times p_2 \times p_3$ grid of processors, $|\Pi| = P$.

Require: A is evenly divided into a $p_1 \times p_2$ grid of rectangular blocks of dimension $n_1/p_1 \times n_2/p_2$, and each block A_{ij} is evenly divided across a set of p_3 processors. $A_{ij}^{(k)}$ is owned by processor rank (i, j, k) .

Ensure: $B = A\Omega$, B is evenly divided across a $p_1 \times p_3$ grid of blocks, and each block B_{ik} is evenly divided across a set of p_2 processors. $B_{ik}^{(j)}$ is owned by processor rank (i, j, k) .

- 1: **function** $B_{ik}^{(j)} = \text{RANDMATMUL}(A_{ij}^{(k)}, \Pi)$
- 2: $(i, j, k) = \text{MYRANK}(\Pi)$
- 3: $A_{ij} = \text{ALL-GATHER}(A_{ij}^{(k)}, \Pi_{ij*})$ ▷ Gather the required data of input matrix A
- 4: $\Omega_{jk} = \text{GENRANDOM}(n_2/p_2, r/p_3)$ ▷ Generate the required random submatrix
- 5: $B_{ik} = A_{ij} \cdot \Omega_{jk}$ ▷ Perform local matrix multiplication
- 6: $B_{ik}^{(j)} = \text{REDUCE-SCATTER}(B_{ik}, \Pi_{i*k})$ ▷ Sum results to compute $B_{ik}^{(j)}$
- 7: **end function**

Communication cost is optimal when p_1 , p_2 & p_3 are chosen based on lower bounds.

Table of Contents

- 1 Existing results for classical matrix multiplication
- 2 $A\Omega$ computation
 - Communication lower bounds
 - Optimal algorithms
- 3 Computations of Nyström approximation
 - $A\Omega$ and $\Omega^T A\Omega$ computations together
 - Optimal algorithms
- 4 Experimental results
- 5 Conclusion

Nyström approximation

Let $A \in \mathbb{R}^{n \times n}$ be a symmetric positive definite matrix and $\Omega \in \mathbb{R}^{n \times r}$ be a random matrix.

- Nyström approximation of A , $\tilde{A} = (A\Omega)(\Omega^T A\Omega)^\dagger (A\Omega)^T$
- Requires to compute both $A\Omega$ and $\Omega^T A\Omega$

Constraints for load balanced computation

$$B = A\Omega, \quad C = \Omega^T B$$

- ϕ_A, ϕ_B, ϕ_C : projection sizes of computations on arrays A, B, C
- Each processor performs $1/P$ th portion of each computation
- From the HBL inequality: $\phi_A \phi_B \geq \frac{n^2 r}{P}$, $\phi_B \phi_C \geq \frac{nr^2}{P}$
- Extra constraints: $\frac{n^2}{P} \leq \phi_A \leq n^2$, $\frac{nr}{P} \leq \phi_B \leq nr$, $\frac{r^2}{P} \leq \phi_C \leq r^2$

Optimization problem and communication lower bounds

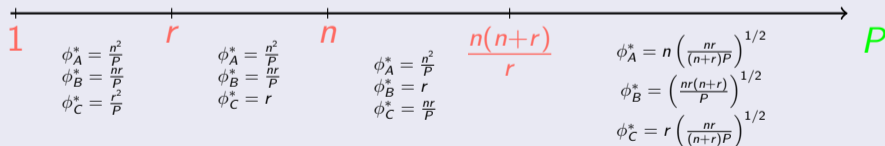
Minimize $\phi_A + \phi_B + \phi_C$ s.t.

$$\phi_A \phi_B \geq \frac{n^2 r}{P}, \quad \phi_B \phi_C \geq \frac{nr^2}{P}$$

$$\frac{n^2}{P} \leq \phi_A \leq n^2, \quad \frac{nr}{P} \leq \phi_B \leq nr, \quad \frac{r^2}{P} \leq \phi_C \leq r^2$$

Amount of non-random array accesses = $\phi_A + \phi_B + \phi_C$

- Estimate the solution and prove optimality using all Karush–Kuhn–Tucker conditions are satisfied
- For $n \geq r$,



- Communication lower bound = $\phi_A + \phi_B + \phi_C$ – data owned by the processor
 $= \phi_A + \phi_B + \phi_C - \frac{n^2 + nr + r^2}{P}$

Table of Contents

- 1 Existing results for classical matrix multiplication
- 2 $A\Omega$ computation
 - Communication lower bounds
 - Optimal algorithms
- 3 Computations of Nyström approximation
 - $A\Omega$ and $\Omega^T A\Omega$ computations together
 - Optimal algorithms
- 4 Experimental results
- 5 Conclusion

Parallel algorithms for $B = A\Omega$ and $C = \Omega^T B$

Require: Π is a $p_1 \times p_2 \times p_3$ processor grid, Ψ is a $q_1 \times q_2 \times q_3$ processor grid, and $|\Pi| = |\Psi| = P$.

Require: A is evenly divided into a $p_1 \times p_2$ grid of rectangular blocks, and each block A_{ij} is evenly divided across a set of p_3 processors with $A_{ij}^{(k)}$ is owned by processor rank (i, j, k) in Π .

Ensure: $B = A\Omega$, B is evenly divided across a $q_1 \times q_3$ grid of blocks, and each block $B_{i'k'}$ is evenly divided across a set of q_2 processors with $B_{i'k'}^{(j')}$ is owned by processor rank (i', j', k') in Ψ .

Ensure: $C = \Omega^T B$, C is evenly divided across a $q_2 \times q_3$ grid of blocks, and each block $C_{j'k'}$ is evenly divided across a set of q_1 processors with $C_{j'k'}^{(i')}$ is owned by processor rank (i', j', k') in Ψ .

- 1: **function** $(B_{i'k'}^{(j')}, C_{j'k'}^{(i')}) = \text{RANDCOMP NYS}(A_{ij}^{(k)}, \Pi, \Psi)$
- 2: $(i, j, k) = \text{MYRANK}(\Pi)$
- 3: Perform $B = A\Omega$ in $p_1 \times p_2 \times p_3$ grid of processors
- 4: //Change distribution of B so that it is suitable for $q_1 \times q_2 \times q_3$ grid
- 5: $(i', j', k') = \text{MYRANK}(\Psi)$
- 6: **if** for any $i, p_i \neq q_i$ **then** $B_{i'k'}^{(j')} = \text{REDISTRIBUTE}(B)$ **end if**
- 7: Perform $C = \Omega^T B$ in $q_1 \times q_2 \times q_3$ grid of processors
- 8: **end function**

Processor grid dimensions

- Not clear how to obtain processor grid dimensions that exactly match lower bounds ($p_i = q_i$ for $i = 0, 1, 2$)

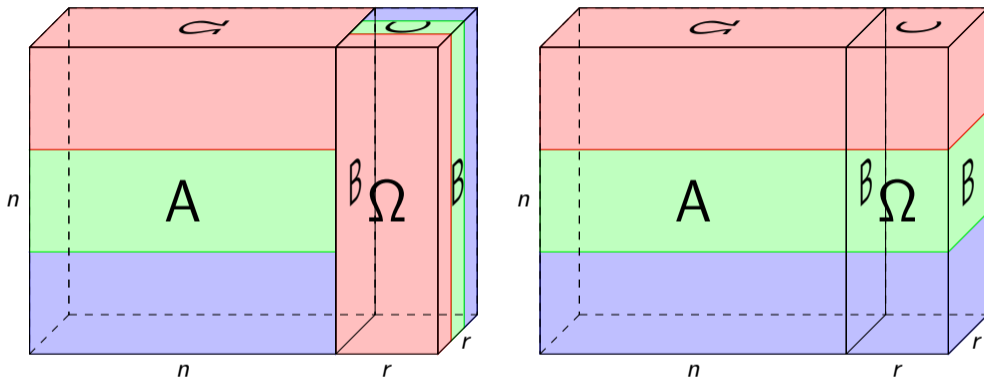
Approach 1 (*Redistribution*):

- Perform $B = A\Omega$ in $p_1 \times p_2 \times p_3$ grid of processors and $C = \Omega^T B$ in $q_1 \times q_2 \times q_3$ grid of processors
 - Communication cost is far from the lower bound by the sum of the communication cost of B and the redistribution cost

Approach 2 (*No-Redistribution*):

- $B = A\Omega$ is the dominating computation
 - Use the optimal processor grid of this computation to also perform $C = \Omega^T B$

Redistribution vs No-Redistribution on 3 processors



$$\text{Total number of iteration points} = n^2r + nr^2 = n(n+r)r$$

Table of Contents

- 1 Existing results for classical matrix multiplication
- 2 $A\Omega$ computation
 - Communication lower bounds
 - Optimal algorithms
- 3 Computations of Nyström approximation
 - $A\Omega$ and $\Omega^T A\Omega$ computations together
 - Optimal algorithms
- 4 Experimental results
- 5 Conclusion

NERSC Perlmutter GPU Partition

Processor	1 × AMD EPYC 7763 per node
# NUMA domains	4
# Cores	64 per node, 16 per NUMA domain
# Hyperthreads	2 per core, 32 per NUMA domain
# GPUs	4 × NVIDIA A100 per node
Memory	256 GB DDR4 per node, 40GB per GPU

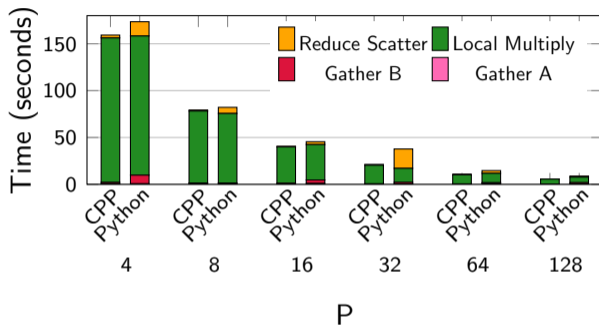
	CPU-only	GPU
Compiler	Intel C++ version 2024.1.0	NVIDIA CUDA C++ version 24.5-1
BLAS and PRNG	MKL 2024.1	CUDA toolkit 12.4
MPI	CRAY MPICH 8.1.30 (CUDA-aware for GPUs)	

Dataset and number of processors

- Small number of processors ($P < r$, practical regime)
 - For $B = A\Omega$ computation, communication cost = 0
 - For $B = A\Omega$ and $C = \Omega^T B$ computations, communication cost = nr/P (for *Redistribution*) and $r^2 - r^2/P$ (for *No-Redistribution*)

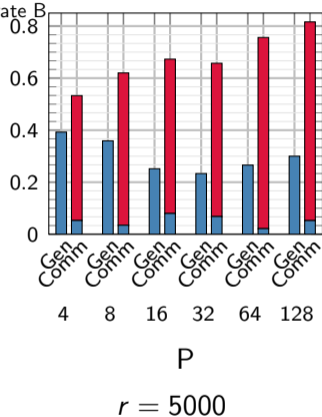
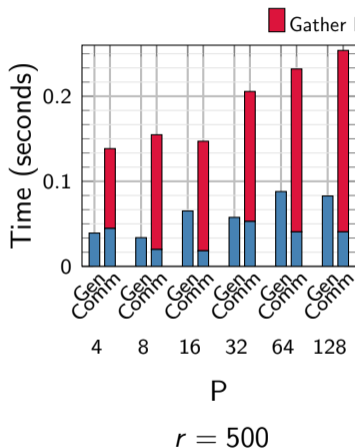
- n and r are selected based on CIFAR-10 dataset
 - $n = 50,000$ and $r = 500$ and 5000

C++ vs Python: 3D-distributed memory matrix multiplication



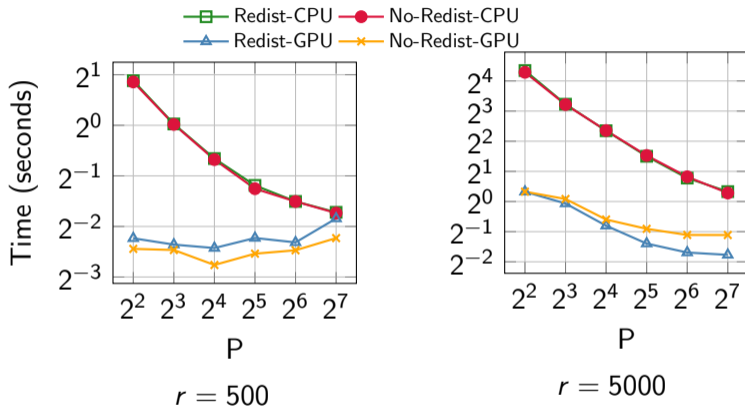
- Comparison to multiply two $50k \times 50k$ double precision matrices
- Use a processor grid as cubical as possible
- Python (mpi4py version 3.1.5) performance is hard to explain
- We focus on C++ implementation

Communicating vs redundantly generating Ω in $B = A\Omega$



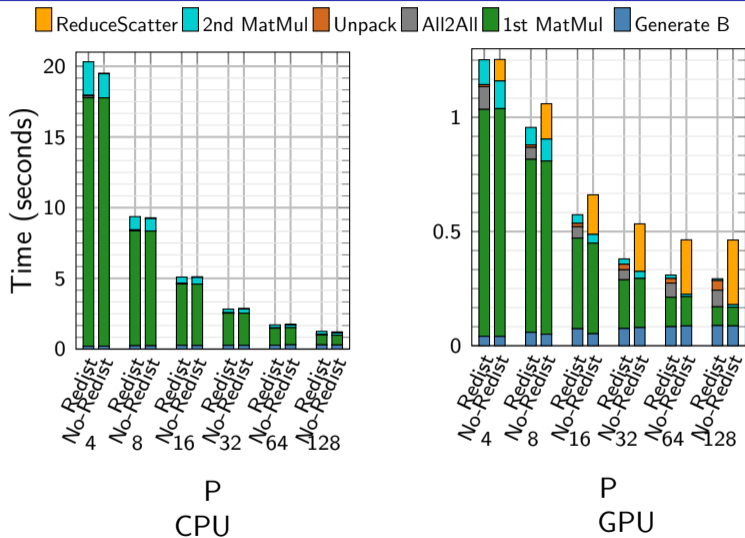
- Generating Ω is cheaper than communicating it

$B = A\Omega$ and $C = \Omega^T B$ computations



- CPU-only implementations scale better
- *Redist* approach scales better on GPU

Runtime breakdown for both approaches with $r = 5000$



- CPU-only implementation is dominated by the local matrix multiplication

Table of Contents

- 1 Existing results for classical matrix multiplication
- 2 $A\Omega$ computation
 - Communication lower bounds
 - Optimal algorithms
- 3 Computations of Nyström approximation
 - $A\Omega$ and $\Omega^T A\Omega$ computations together
 - Optimal algorithms
- 4 Experimental results
- 5 Conclusion

Conclusion

- Communication lower bounds and optimal algorithms for $B = A\Omega$ and $C = \Omega^T B$
- Parallel scalability of our algorithms on both CPUs and GPUs
- *Redist* approach scales better on GPUs

Future Work

- Exploit symmetry to further reduce computation and communication costs
- Study how to perform all computations of generalized Nyström approximations for nonsymmetric matrices efficiently
- Extend our approaches to other sketching methods that use structured matrices, such as CountSketch and Subsampled Randomized Hadamard Transform methods

Thank You!