Communication costs of parallel matrix multiplications

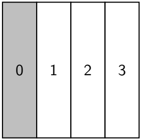
Suraj Kumar

Inria & ENS Lyon
Email:*suraj.kumar@ens-lyon.fr*

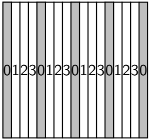
CR12: September 2024

https://surakuma.github.io/courses/daamtc.html

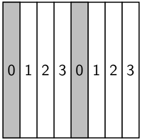
Popular parallel distributions of matrices



1D column block layout

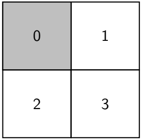


1D column cyclic layout



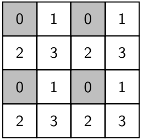
1D column block cyclic layout

Row versions of the previous layouts



0	1
2	3

2D row and column block layout



0	1	0	1
2	3	2	3
0	1	0	1
2	3	2	3

2D row and column block cyclic layout

Note: Process 0 owns the shaded submatrices.

Table of Contents

Extension of sequential lower bounds

- Sequential lower bound on bandwidth $= \Omega\left(\frac{2mn\ell}{\sqrt{M}}\right) = \Omega\left(\frac{\#\text{operations}}{\sqrt{M}}\right)$
- Sequential lower bound on latency $= \Omega\left(\frac{\#\text{operations}}{M^{3/2}}\right)$

Extension to paralllel machines

Lemma

Consider a traditional $n \times n$ matrix multiplication performed on P processors with distributed memory. A processor with memory M that performs W elementary products must send or receive $\Omega\left(\frac{W}{\sqrt{M}}\right)$ elements.

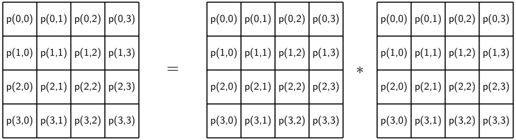
Theorem

Consider a traditional $n \times n$ matrix multiplication on P processors, each with a memory M. Some processor has $\Omega\left(\frac{n^3/P}{\sqrt{M}}\right)$ volume of I/O.

- Lower bound on latency $= \Omega\left(\frac{n^3/P}{M^{3/2}}\right)$
- Bound is useful only when M is not very large

Matrix multiplication with 2D layout

- Consider processors are arranged in a 2-dimensional grid
- Processors exchange data along rows and columns



p(0,0)	p(0,1)	p(0,2)	p(0,3)
p(1,0)	p(1,1)	p(1,2)	p(1,3)
p(2,0)	p(2,1)	p(2,2)	p(2,3)
p(3,0)	p(3,1)	p(3,2)	p(3,3)

=

p(0,0)	p(0,1)	p(0,2)	p(0,3)
p(1,0)	p(1,1)	p(1,2)	p(1,3)
p(2,0)	p(2,1)	p(2,2)	p(2,3)
p(3,0)	p(3,1)	p(3,2)	p(3,3)

*

p(0,0)	p(0,1)	p(0,2)	p(0,3)
p(1,0)	p(1,1)	p(1,2)	p(1,3)
p(2,0)	p(2,1)	p(2,2)	p(2,3)
p(3,0)	p(3,1)	p(3,2)	p(3,3)

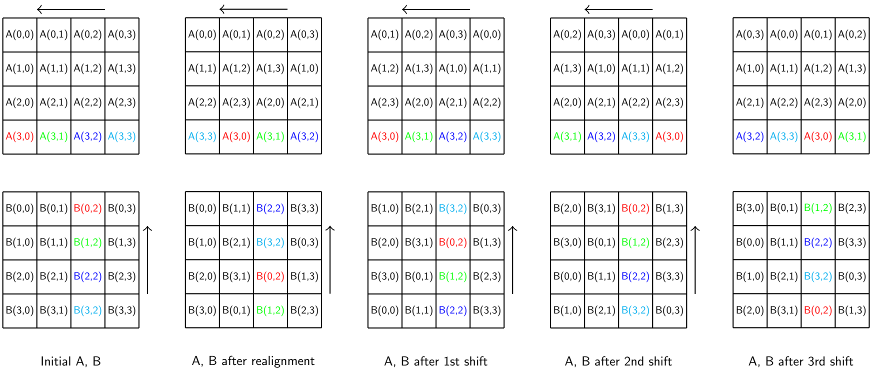
- P processors are arranged in $\sqrt{P} \times \sqrt{P}$ grid

Cannon's 2D matrix multiplication algorithm

- Processors organized on a square 2D grid of size $\sqrt{P} \times \sqrt{P}$
- A, B, C matrices distributed by blocks of size $N/\sqrt{P} \times N/\sqrt{P}$
- Processor $P(i,j)$ initially holds blocks $A(i,j), B(i,j)$ and computes $C(i,j)$
- First realign matrices:
  - Shift $A(i,j)$ block to the left by i
  - Shift $B(i,j)$ block to the top by j

  After realignment: $P(i,j)$ holds blocks $A(i,i+j)$ and $B(i+j,j)$
- At each step :
  - Compute one block product
  - Shift A blocks left
  - Shift B blocks up

Cannon’s matrix multiplication algorithm



A(0,0)	A(0,1)	A(0,2)	A(0,3)
A(1,0)	A(1,1)	A(1,2)	A(1,3)
A(2,0)	A(2,1)	A(2,2)	A(2,3)
A(3,0)	A(3,1)	A(3,2)	A(3,3)

B(0,0)	B(0,1)	B(0,2)	B(0,3)
B(1,0)	B(1,1)	B(1,2)	B(1,3)
B(2,0)	B(2,1)	B(2,2)	B(2,3)
B(3,0)	B(3,1)	B(3,2)	B(3,3)

Initial A, B

A(0,0)	A(0,1)	A(0,2)	A(0,3)
A(1,1)	A(1,2)	A(1,3)	A(1,0)
A(2,2)	A(2,3)	A(2,0)	A(2,1)
A(3,3)	A(3,0)	A(3,1)	A(3,2)

B(0,0)	B(1,1)	B(2,2)	B(3,3)
B(1,0)	B(2,1)	B(3,2)	B(0,3)
B(2,0)	B(3,1)	B(0,2)	B(1,3)
B(3,0)	B(0,1)	B(1,2)	B(2,3)

A, B after realignment

A(0,1)	A(0,2)	A(0,3)	A(0,0)
A(1,2)	A(1,3)	A(1,0)	A(1,1)
A(2,3)	A(2,0)	A(2,1)	A(2,2)
A(3,0)	A(3,1)	A(3,2)	A(3,3)

B(1,0)	B(2,1)	B(3,2)	B(0,3)
B(2,0)	B(3,1)	B(0,2)	B(1,3)
B(3,0)	B(0,1)	B(1,2)	B(2,3)
B(0,0)	B(1,1)	B(2,2)	B(3,3)

A, B after 1st shift

A(0,2)	A(0,3)	A(0,0)	A(0,1)
A(1,3)	A(1,0)	A(1,1)	A(1,2)
A(2,0)	A(2,1)	A(2,2)	A(2,3)
A(3,1)	A(3,2)	A(3,3)	A(3,0)

B(2,0)	B(3,1)	B(0,2)	B(1,3)
B(3,0)	B(0,1)	B(1,2)	B(2,3)
B(0,0)	B(1,1)	B(2,2)	B(3,3)
B(1,0)	B(2,1)	B(3,2)	B(0,3)

A, B after 2nd shift

A(0,3)	A(0,0)	A(0,1)	A(0,2)
A(1,0)	A(1,1)	A(1,2)	A(1,3)
A(2,1)	A(2,2)	A(2,3)	A(2,0)
A(3,2)	A(3,3)	A(3,0)	A(3,1)

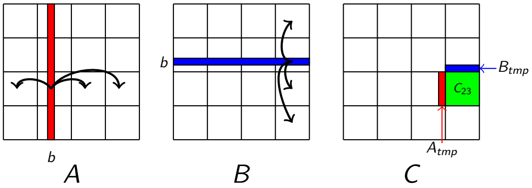
B(3,0)	B(0,1)	B(1,2)	B(2,3)
B(0,0)	B(1,1)	B(2,2)	B(3,3)
B(1,0)	B(2,1)	B(3,2)	B(0,3)
B(2,0)	B(3,1)	B(0,2)	B(1,3)

A, B after 3rd shift

$$C(3,2) = A(3,1) * B(1,2) + A(3,2) * B(2,2) + A(3,3) * B(3,2) + A(3,0) * B(0,2)$$

- Total data transfer costs $= \mathcal{O}(n^2/\sqrt{P})$
- Not clear how to extend it for rectangular matrices

Scalable Universal Matrix Multiplication Algorithm (SUMMA)



- P is arranged in $\sqrt{P} \times \sqrt{P}$ grid
- Each processor owns $n/\sqrt{P} \times n/\sqrt{P}$ submatrices of A, B and C
- b=block size ($\leq n/\sqrt{P}$)

Algorithm structure

- Each owner of A block broadcasts data to whole processor row
- Each owner of B block broadcasts data to whole processor column
- Receive block of A in A_{tmp}, receive block of B in B_{tmp}
- Compute $C_{local} + = C_{local} + A_{tmp} * B_{tmp}$

Communication costs of SUMMA algorithm

- Total number of steps $= \sqrt{P} \cdot \frac{n/\sqrt{P}}{b} = \frac{n}{b}$
- Total data transfer costs $= \mathcal{O}(n^2/\sqrt{P})$
- Easily extendable with rectangular matrices

Theorem

Consider a traditional matrix multiplication on P processors each with $O(n^2/P)$ storage, some processor has $\Omega(n^2/\sqrt{P})$ I/O volume.

Proof: Previous result: $\Omega(n^3/P\sqrt{M})$ with $M = n^2/P$.

- $\mathcal{O}(n^2/\sqrt{P})$ I/O volume of both Cannon's algorithm and SUMMA
- Both algorithms are bandwidth optimal
- Can we do better?

Table of Contents

Notations & Settings

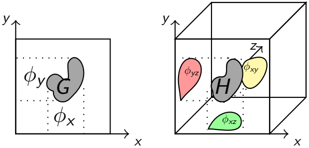
- $C = AB$, where $A \in \mathbb{R}^{n_1 \times n_2}$, $B \in \mathbb{R}^{n_2 \times n_3}$, and $C \in \mathbb{R}^{n_1 \times n_3}$
- Let $d_1 = \min(n_1, n_2, n_3) \leq d_2 = median(n_1, n_2, n_3) \leq d_3 = \max(n_1, n_2, n_3)$

Settings

- P number of processors
- The algorithm load balances the computation
- One copy of data is in the system
    - There exists a processor whose input data at the start plus output data at the end must be at most $\frac{d_1d_2+d_1d_3+d_2d_3}{P}$ words – will analyze data transfers for this processor
- Each processor has large local memory – enough to store all the required data
- Focus on bandwidth cost (volume of data transfers)

Constraints for matrix multiplications

- Loomis-Whitney inequalitiy: for $d-1$ dimensional projections
    - For the 2d object G, $Area(G) \leq \phi_x\phi_y$
    - For the 3d object H, $Volume(H) \leq \sqrt{\phi_{xy}\phi_{yz}\phi_{xz}}$



$$\text{for } i = 0{:}n_1-1, \text{ for } k = 0{:}n_2-1, \text{ for } j = 0{:}n_3-1$$
$$C[i][j] += A[i][k] * B[k][j]$$

- Total number of multiplications $= n_1n_2n_3$
- Each processor performs $\frac{n_1n_2n_3}{P}$ amount of multiplications
- Optimization problem:

$$Minimize\ \phi_A + \phi_B + \phi_C \quad \text{s.t.}$$
$$\phi_A^{\frac{1}{2}}\phi_B^{\frac{1}{2}}\phi_C^{\frac{1}{2}} \geq \frac{n_1n_2n_3}{P}$$

Extra constraints

$$\text{for } i = 0{:}n_1 - 1, \text{ for } k = 0{:}n_2 - 1, \text{ for } j = 0{:}n_3 - 1$$

$$C[i][j] += A[i][k] * B[k][j]$$

- Each element of A (resp. B) is involved in n_3 (resp. n_1) multiplications
    - To perform at least $\frac{n_1 n_2 n_3}{P}$ multiplications: $\phi_A \geq \frac{n_1 n_2}{P}, \phi_B \geq \frac{n_2 n_3}{P}$
- Each element of C is the sum of n_2 multiplications, therefore $\phi_C \geq \frac{n_1 n_3}{P}$
- Projections can be at max the size of the arrays: $\phi_A \leq n_1 n_2$, $\phi_B \leq n_2 n_3$, $\phi_C \leq n_1 n_3$

Optimization problem for communication lower bounds

- Projections (ϕ_A, ϕ_B, ϕ_C) indicate the amount of array accesses
- Communication lower bound $= \phi_A + \phi_B + \phi_C -$ data owned by the processor

$$\begin{aligned}
&\textit{Minimize } \phi_A + \phi_B + \phi_C \ \text{ s.t.} \\
&\phi_A^{\frac{1}{2}}\phi_B^{\frac{1}{2}}\phi_C^{\frac{1}{2}} \geq \frac{n_1 n_2 n_3}{P} \\
&\frac{n_1 n_2}{P} \leq \phi_A \leq n_1 n_2 \\
&\frac{n_2 n_3}{P} \leq \phi_B \leq n_2 n_3 \\
&\frac{n_1 n_3}{P} \leq \phi_C \leq n_1 n_3
\end{aligned}$$

Generalized version (in terms of d_1, d_2, d_3)

$$\begin{aligned}
&\textit{Minimize } \phi_1 + \phi_2 + \phi_3 \ \text{ s.t.} \\
&\phi_1^{\frac{1}{2}}\phi_2^{\frac{1}{2}}\phi_3^{\frac{1}{2}} \geq \frac{d_1 d_2 d_3}{P} \\
&\frac{d_1 d_2}{P} \leq \phi_1 \leq d_1 d_2 \\
&\frac{d_1 d_3}{P} \leq \phi_2 \leq d_1 d_3 \\
&\frac{d_2 d_3}{P} \leq \phi_3 \leq d_2 d_3 \\
&d_1 \leq d_2 \leq d_3
\end{aligned}$$

Amount of accesses and communication lower bounds

- Estimate the solution based on Lagrange multipliers
- Prove optimality using all Karush–Kuhn–Tucker (KKT) conditions are satisfied

Amount of accesses $=\phi_1 + \phi_2 + \phi_3$

$1 \longrightarrow \frac{d_3}{d_2} \longrightarrow \frac{d_2 d_3}{d_1^2} \longrightarrow P$

$\phi_1 = d_1 d_2$
$\phi_2 = \frac{d_1 d_3}{P}$
$\phi_3 = \frac{d_2 d_3}{P}$

$\phi_1 = \phi_2 = (\frac{d_1^2 d_2 d_3}{P})^{1/2}$
$\phi_3 = \frac{d_2 d_3}{P}$

$\phi_1 = \phi_2 = \phi_3 = (\frac{d_1 d_2 d_3}{P})^{2/3}$

Communication lower bounds (amount of data transfers)

$1 \longrightarrow \frac{d_3}{d_2} \longrightarrow \frac{d_2 d_3}{d_1^2} \longrightarrow P$

$\text{LB}=d_1 d_2 - \frac{d_1 d_2}{P}$

$\text{LB}=2(\frac{d_1^2 d_2 d_3}{P})^{1/2} - \frac{d_1 d_2 + d_1 d_3}{P}$

$\text{LB}=3(\frac{d_1 d_2 d_3}{P})^{2/3} - \frac{d_1 d_2 + d_1 d_3 + d_2 d_3}{P}$

Convex and quasiconvex functions

Definition (Eq. 3.2, Boyd and Vandenberghe, 2004.)

A differentiable function $f : \mathbb{R}^d \to \mathbb{R}$ is *convex* if its domain is a convex set and for all $\boldsymbol{x}, \boldsymbol{y} \in \textbf{dom } f$,

$$f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \langle \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle.$$

Definition (Eq. 3.20, Boyd and Vandenberghe, 2004.)

A differentiable function $g : \mathbb{R}^d \to \mathbb{R}$ is *quasiconvex* if its domain is a convex set and for all $\boldsymbol{x}, \boldsymbol{y} \in \textbf{dom } g$,

$$g(\boldsymbol{y}) \leq g(\boldsymbol{x}) \text{ implies that } \langle \nabla g(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle \leq 0.$$

Lemma (Lemma 2, Ballard et al., SPAA 2022.)

The function $g_0(\boldsymbol{x}) = L - x_1 x_2 x_3$, for some constant L, is quasiconvex in the positive octant.

KKT conditions

Definition (Eq. 5.49, Boyd and Vandenberghe, 2004.)

Consider an optimization problem of the form

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) \quad \text{subject to} \quad \boldsymbol{g}(\boldsymbol{x}) \leq 0 \tag{1}$$

where $f : \mathbb{R}^d \to \mathbb{R}$ and $\boldsymbol{g} : \mathbb{R}^d \to \mathbb{R}^c$ are both differentiable. Define the dual variables $\boldsymbol{\mu} \in \mathbb{R}^c$, and let $\boldsymbol{J_g}$ be the Jacobian of $\boldsymbol{g}$. The *Karush-Kuhn-Tucker (KKT)* conditions of $(\boldsymbol{x}, \boldsymbol{\mu})$ are as follows:

- *Primal feasibility*: $\boldsymbol{g}(\boldsymbol{x}) \leq 0$;
- *Dual feasibility*: $\boldsymbol{\mu} \geq 0$;
- *Stationarity*: $\nabla f(\boldsymbol{x}) + \boldsymbol{\mu} \cdot \boldsymbol{J_g}(\boldsymbol{x}) = 0$;
- *Complementary slackness*: $\mu_i g_i(\boldsymbol{x}) = 0$ for all $i \in \{1, \ldots, c\}$.

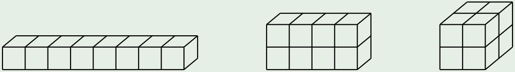
Lemma (Lemma 3, Ballard et al., SPAA 2022.)

Consider an optimization problem of the form given in Equation 1. If f is a convex function and each g_i is a quasiconvex function, then the KKT conditions are sufficient for optimality.

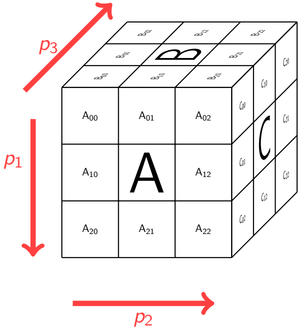
Table of Contents

Design of communication optimal algorithms for $C = AB$

Arrangements of 8 processors



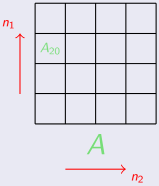
- P is organized into $p_1 \times p_2 \times p_3$ logical grid
- Select p_1, p_2 and p_3 based on the communication lower bounds
- Allgather A on the set of processors along each slice of p_3
- Allgather B on the set of processors along each slice of p_1
- Perform local computation
- Perform Reduce-Scatter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



Communication optimal algorithms

Data Distribution (P is organized into a $p_1 \times p_2 \times p_3$ grid)

- Each processor has $\frac{1}{P}$th amount of input and output variables
- $A_{20} = A(2\frac{n_1}{p_1} : 3\frac{n_1}{p_1} - 1, 0 : \frac{n_2}{p_2} - 1)$ is evenly distributed among $(2, 0, *)$ processors
- $B_{01} = B(0 : \frac{n_2}{p_2} - 1, \frac{n_3}{p_3} : 2\frac{n_3}{p_3}) - 1$ is evenly distributed among $(*, 0, 1)$ processors



Assignment 2 – deadline Sept. 26

Questions:

- Write pseudo code for 3-dimensional parallel matrix multiplication algorithm.
- Determine expressions for processor grid dimensions based on the lower bounds when P is large and compute the data transfer costs of the algorithm with these dimensions.

Cost analysis and Open questions

Cost analysis along the critical path

- Total amount of multiplications per processor $= \frac{n_1 n_2 n_3}{p_1 p_2 p_3} = \frac{n_1 n_2 n_3}{P}$
- Total data transfers $= \frac{n_1 n_2}{p_1 p_2} + \frac{n_2 n_3}{p_2 p_3} + \frac{n_1 n_3}{p_1 p_3} - \frac{n_1 n_2 + n_2 n_3 + n_1 n_3}{P}$
- Total memory required on each processor $= \mathcal{O}\left(\left(\frac{n_1 n_2 n_3}{P} \right)^{\frac{2}{3}} \right)$

Open Questions

- Are communication lower bounds achievable for all matrix dimensions?
- How to adapt when we have less memory on each processor?

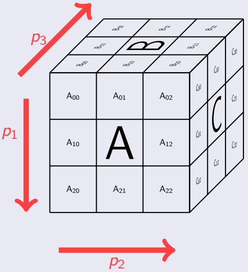
Table of Contents

Limited memory scenarios

- $C = AB$, where $A \in \mathbb{R}^{n_1 \times n_2}$, $B \in \mathbb{R}^{n_2 \times n_3}$, and $C \in \mathbb{R}^{n_1 \times n_3}$
- Amount of memory on each processor $= \mathcal{O}\left(c\frac{n_1 n_3}{P}\right)$
- $\frac{n_1 n_3}{P} << c\frac{n_1 n_3}{P} << \left(\frac{n_1 n_2 n_3}{P}\right)^{\frac{2}{3}}$
- Data transfer lower bound $= \Omega\left(\frac{n_1 n_2 n_3}{P\sqrt{M}}\right) = \Omega\left(n_2\sqrt{\frac{n_1 n_3}{Pc}}\right)$

Algorithm structure

- The same 3-dimensional algorithm
- P ia arranged in $p_1 \times p_2 \times p_3$ logical grid
- Set $p_2 = c$
- $p_1 p_3 = P/c$ processors perform multiplication of $n_1 \times \frac{n_2}{c}$ submatrix of A with $\frac{n_2}{c} \times n_3$ submatrix of B
- Perform Reduce-Scatter operation along p_2 to obtain C



Processor grid dimensions and data transfer costs

- Total amount of multiplications on each processor $= \frac{n_1}{p_1} \cdot \frac{n_2}{c} \cdot \frac{n_3}{p_3} = \frac{n_1 n_2 n_3}{P}$
- To minimize data transfer costs
  - $\#$ access of A on each processor $= \#$ access of B on each processor $=> \frac{n_1}{p_1} \cdot \frac{n_2}{c} = \frac{n_2}{c} \cdot \frac{n_1}{p_1}$
  - $p_1 p_3 = P/c$
  - $p_1 = \left(\frac{n_1}{n_3} \cdot \frac{P}{c}\right)^{\frac{1}{2}}$
  - $p_3 = \left(\frac{n_3}{n_1} \cdot \frac{P}{c}\right)^{\frac{1}{2}}$
- $\#$ accessed elements on each processor $= \frac{n_1 n_2}{p_1 c} + \frac{n_2 n_3}{p_3 c} + c\frac{n_1 n_3}{P}$
  $= 2n_2\sqrt{\frac{n_1 n_3}{Pc}} + c\frac{n_1 n_3}{P}$
- Data transfer costs on each processor $= \#$ *accessed elements* - *owned data*
- *owned data* $= \frac{n_1 n_2 + n_2 n_3 + n_1 n_3}{P}$

$$c\frac{n_1 n_3}{P} << \left(\frac{n_1 n_2 n_3}{P}\right)^{\frac{2}{3}} => c\frac{n_1 n_3}{P} << n_2\sqrt{\frac{n_1 n_3}{Pc}}$$

- Data transfer costs of the algorithm match the leading terms in the lower bounds