

Multiple Tensor Times Matrix computation

Suraj Kumar

Inria & ENS Lyon

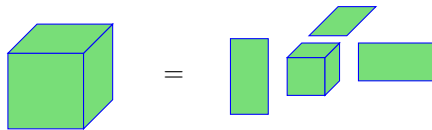
Email: suraj.kumar@inria.fr

CR12: October 2023

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

Tucker decomposition of $\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times \dots \times n_d}$

It represents a tensor with d matrices (usually orthonormal) and a small core tensor.



Tucker decomposition of a 3-dimensional tensor.

$$\mathcal{X} = \mathcal{Y} \times_1 \mathbf{A}^{(1)} \dots \times_d \mathbf{A}^{(d)}$$

$$\mathcal{X}(i_1, \dots, i_d) = \sum_{\alpha_1=1}^{r_1} \dots \sum_{\alpha_d=1}^{r_d} \mathcal{Y}(\alpha_1, \dots, \alpha_d) \mathbf{A}^{(1)}(i_1, \alpha_1) \dots \mathbf{A}^{(d)}(i_d, \alpha_d)$$

It can be concisely expressed as $\mathcal{X} = \llbracket \mathcal{Y}; \mathbf{A}^{(1)}, \dots, \mathbf{A}^{(d)} \rrbracket$.

Here r_j for $1 \leq j \leq d$ denote a set of ranks. Matrices $\mathbf{A}^{(j)} \in \mathbb{R}^{n_j \times r_j}$ for $1 \leq j \leq d$ are usually orthonormal and known as factor matrices. The tensor $\mathcal{Y} \in \mathbb{R}^{r_1 \times r_2 \times \dots \times r_d}$ is called the core tensor.

Algorithm 1 HOSVD method to compute a Tucker decomposition

Require: input tensor $\mathcal{X} \in \mathbb{R}^{n_1 \times \dots \times n_d}$, desired rank (r_1, \dots, r_d)

Ensure: $\mathcal{X} = \mathcal{Y} \times_1 A^{(1)} \times_2 A^{(2)} \dots \times_d A^{(d)}$

- 1: **for** $k = 1$ to d **do**
 - 2: $A^{(k)} \leftarrow r_k$ leading left singular vectors of $X_{(k)}$
 - 3: **end for**
 - 4: $\mathcal{Y} = \mathcal{X} \times_1 A^{(1)\top} \times_2 A^{(2)\top} \dots \times_d A^{(d)\top}$
-

- When $r_i < \text{rank}(A_{(i)})$ for one or more i , the decomposition is called the truncated-HOSVD (T-HOSVD)
- The collective operation $\mathcal{X} \times_1 A^{(1)\top} \times_2 A^{(2)\top} \dots \times_d A^{(d)\top}$ is known as Multiple Tensor-Times-Matrix (Multi-TTM) computation

Sequentially T-HOSVD (ST-HOSVD) for Tucker decomposition

- This method is more work efficient than T-HOSVD
- In each step, it reduces the size of one dimension of the tensor

Algorithm 2 ST-HOSVD method to compute a Tucker decomposition

Require: input tensor $\mathcal{X} \in \mathbb{R}^{n_1 \times \dots \times n_d}$, desired rank (r_1, \dots, r_d)

Ensure: $\llbracket \mathcal{Y}; \mathbf{A}^{(1)}, \dots, \mathbf{A}^{(d)} \rrbracket$: a (r_1, \dots, r_d) -rank Tucker decomposition of \mathcal{X}

- 1: $\mathcal{W} \leftarrow \mathcal{X}$
 - 2: **for** $k = 1$ to d **do**
 - 3: $\mathbf{A}^{(k)} \leftarrow r_k$ leading singular vectors of $W_{(k)}$
 - 4: $\mathcal{W} \leftarrow \mathcal{W} \times_k \mathbf{A}^{(k)\top}$
 - 5: **end for**
 - 6: $\mathcal{Y} = \mathcal{W}$
-

We can note that ST-HOSVD also performs Multi-TTM computation by doing a sequence of TTM operations, i.e, $\mathcal{Y} = ((\mathcal{X} \times_1 \mathbf{A}^{(1)\top}) \times_2 \mathbf{A}^{(2)\top}) \dots \times_d \mathbf{A}^{(d)\top}$.

- Multi-TTM becomes the overwhelming bottleneck computation when
 - Matrix SVD costs are reduced using randomization via sketching or
 - U_k are computed with eigen value decompositions of $B_{(k)}B_{(k)}^T$

Multi-TTM computation

Let $\mathcal{Y} \in \mathbb{R}^{r_1 \times \dots \times r_d}$ be the output tensor, $\mathcal{X} \in \mathbb{R}^{n_1 \times \dots \times n_d}$ be the input tensor, and $A^{(k)} \in \mathbb{R}^{n_k \times r_k}$ be the matrix of the k th mode. Then the Multi-TTM computation can be represented as

$$\mathcal{Y} = \mathcal{X} \times_1 A^{(1)\top} \dots \times_d A^{(d)\top}$$
$$\text{or } \mathcal{X} = \mathcal{Y} \times_1 A^{(1)} \dots \times_d A^{(d)}.$$

We will focus only on the first representation in this course. Our results and analysis extend straightforwardly to the latter case.

Two approaches to perform this computation:

- TTM-in-Sequence approach – performed by a sequence of TTM operations

$$\mathcal{Y} = ((\mathcal{X} \times_1 A^{(1)\top}) \times_2 A^{(2)\top}) \dots \times_d A^{(d)\top}$$

- All-at-once approach

$$\mathcal{Y}(r'_1, \dots, r'_d) = \sum_{\{n'_k \in [n_k]\}_{k \in [d]}} \mathcal{X}(n'_1, \dots, n'_d) \prod_{j \in [d]} A^{(j)}(n'_j, r'_j)$$

$[d]$ denotes $\{1, 2, \dots, d\}$. We represent $n_1 n_2 \dots n_d$ and $r_1 r_2 \dots r_d$ by n and r , respectively. We mainly focus on all-at-once approach.

All-at-once Multi-TTM pseudo code

for $n'_1 = 1:n_1, \dots, \text{for } n'_d = 1:n_d,$

for $r'_1 = 1:r_1, \dots, \text{for } r'_d = 1:r_d,$

$$\mathcal{Y}(r'_1, \dots, r'_d) += \mathcal{X}(n'_1, \dots, n'_d) \cdot A^{(1)}(n'_1, r'_1) \cdot \dots \cdot A^{(N)}(n'_d, r'_d)$$

Δ matrix for Multi-TTM

$$\Delta = \begin{matrix} & A^{(1)} & \dots & A^{(i)} & \dots & A^{(d)} & \mathcal{X} & \mathcal{Y} \\ \begin{matrix} n'_1 \\ \vdots \\ n'_i \\ \vdots \\ n'_d \\ r'_1 \\ \vdots \\ r'_i \\ \vdots \\ r'_d \end{matrix} & \begin{pmatrix} 1 & & & & & & 1 \\ & \ddots & & & & & \vdots \\ & & 1 & & & & 1 \\ & & & \ddots & & & \vdots \\ & & & & 1 & & 1 \\ 1 & & & & & & 1 \\ & \ddots & & & & & \vdots \\ & & 1 & & & & 1 \\ & & & \ddots & & & \vdots \\ & & & & 1 & & 1 \end{pmatrix} & \end{matrix} = \begin{pmatrix} I_{d \times d} & 1_d & 0_d \\ I_{d \times d} & 0_d & 1_d \end{pmatrix}$$

Question: Let $\mathcal{Y} \in \mathbb{R}^{r \times r \times r}$, $\mathcal{X} \in \mathbb{R}^{n \times n \times n}$ and $A \in \mathbb{R}^{n \times r}$. What are the different approaches to perform the following Multi-TTM computation?

$$\mathcal{Y} = \mathcal{X} \times_1 A^T \times_2 A^T \times_3 A^T$$

Compute the exact number of arithmetic operation for each approach.

Table of Contents

1 Parallel Multi-TTM computation

Settings to compute parallel communication lower bound

- Without loss of generality, we assume that $n_1 r_1 \leq n_2 r_2 \leq \dots \leq n_d r_d$
- The input tensor is larger than the output tensor, i.e., $n \geq r$
- The algorithm load balances the computation – each processor performs $1/P$ th number of loop iterations
- 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 $\frac{n+r+\sum_{i=1}^d n_i r_i}{P}$ words – will analyze amount of data transfers for this processor
- Assume that the innermost computation is atomic – all the multiplications are performed on only one processor

Table of Contents

1 Parallel Multi-TTM computation

- 3-dimensional Multi-TTM
- d -dimensional Multi-TTM

Optimization problems (Ballard et. al., 2023)

Lemma

Consider the following optimization problem:

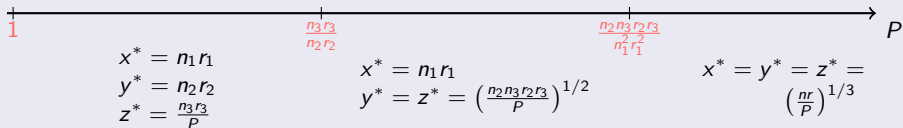
$$\min_{x,y,z} x + y + z \text{ such that}$$

$$\frac{nr}{P} \leq xyz, \quad 0 \leq x \leq n_1 r_1, \quad 0 \leq y \leq n_2 r_2, \quad 0 \leq z \leq n_3 r_3,$$

where $n_1 r_1 \leq n_2 r_2 \leq n_3 r_3$, and $n_1, n_2, n_3, r_1, r_2, r_3, P \geq 1$. The optimal solution (x^*, y^*, z^*) depends on the relative values of the constraints, yielding three cases:

- 1 if $P < \frac{n_3 r_3}{n_2 r_2}$, then $x^* = n_1 r_1, y^* = n_2 r_2, z^* = \frac{n_3 r_3}{P}$;
- 2 if $\frac{n_3 r_3}{n_2 r_2} \leq P < \frac{n_2 n_3 r_2 r_3}{n_1^2 r_1^2}$, then $x^* = n_1 r_1, y^* = z^* = \left(\frac{n_2 n_3 r_2 r_3}{P}\right)^{\frac{1}{2}}$;
- 3 if $\frac{n_2 n_3 r_2 r_3}{n_1^2 r_1^2} \leq P$, then $x^* = y^* = z^* = \left(\frac{nr}{P}\right)^{\frac{1}{3}}$;

which can be visualized as follows.



Optimization problems (Ballard et. al., 2023)

Lemma

Consider the following optimization problem:

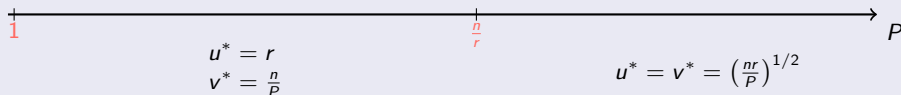
$$\min_{u,v} u + v \text{ such that}$$

$$\frac{nr}{P} \leq uv, \quad 0 \leq u \leq r, \quad 0 \leq v \leq n,$$

where $n \geq r$, and $n, r, P \geq 1$. The optimal solution (u^*, v^*) depends on the relative values of the constraints, yielding two cases:

- ① if $P < \frac{n}{r}$, then $u^* = r, v^* = \frac{n}{P}$;
- ② if $\frac{n}{r} \leq P$, then $u^* = v^* = \left(\frac{nr}{P}\right)^{\frac{1}{2}}$;

which can be visualized as follows.



Both lemma can be proved using the KKT conditions.

Theorem

Any computationally load balanced atomic Multi-TTM algorithm that starts and ends with one copy of the data distributed across processors involving 3-dimensional tensors with dimensions n_1, n_2, n_3 and r_1, r_2, r_3 performs at least $A + B - \left(\frac{n}{P} + \frac{r}{P} + \sum_{j=1}^3 \frac{n_j r_j}{P} \right)$ sends or receives where

$$A = \begin{cases} n_1 r_1 + n_2 r_2 + \frac{n_3 r_3}{P} & \text{if } P < \frac{n_3 r_3}{n_2 r_2} \\ n_1 r_1 + 2 \left(\frac{n_2 n_3 r_2 r_3}{P} \right)^{\frac{1}{2}} & \text{if } \frac{n_3 r_3}{n_2 r_2} \leq P < \frac{n_2 n_3 r_2 r_3}{n_1^2 r_1^2} \\ 3 \left(\frac{n r}{P} \right)^{\frac{1}{3}} & \text{if } \frac{n_2 n_3 r_2 r_3}{n_1^2 r_1^2} \leq P \end{cases}$$
$$B = \begin{cases} r + \frac{n}{P} & \text{if } P < \frac{n}{r} \\ 2 \left(\frac{n r}{P} \right)^{\frac{1}{2}} & \text{if } \frac{n}{r} \leq P. \end{cases}$$

Communication lower bound proof

Let F be the set of loop indices performed by a processor and $|F| = nr/P$. Define $\phi_{\mathbf{x}}(F)$, $\phi_{\mathbf{y}}(F)$ and $\phi_j(F)$ to be the projections of F onto the indices of the arrays \mathbf{x} , \mathbf{y} , and $A^{(j)}$ for $1 \leq j \leq 3$. Δ matrix can be represented as,

$$\Delta = \begin{pmatrix} I_{3 \times 3} & I_3 & 0_3 \\ I_{3 \times 3} & 0_3 & I_3 \end{pmatrix}.$$

Let $\mathcal{C} = \{s \in [0, 1]^5 : \Delta \cdot s \geq 1\}$. Here Δ is not full rank, we consider all vectors $v = [a \ a \ a \ 1-a \ 1-a]^T \in \mathcal{C}$ where $0 \leq a \leq 1$ such that $\Delta \cdot v = 1$. From HBL inequality, we obtain

$$\frac{nr}{P} \leq \left(\prod_{j \in [3]} |\phi_j(F)| \right)^a (|\phi_{\mathbf{x}}(F)| |\phi_{\mathbf{y}}(F)|)^{1-a}.$$

This is equivalent to $\frac{nr}{P} \leq \prod_{j \in [3]} |\phi_j(F)|$ and $\frac{nr}{P} \leq |\phi_{\mathbf{x}}(F)| |\phi_{\mathbf{y}}(F)|$. We also have $|\phi_{\mathbf{x}}(F)| \leq n$, $|\phi_{\mathbf{y}}(F)| \leq r$, and $|\phi_j(F)| \leq n_j r_j$ for $1 \leq j \leq 3$. We want to minimize $|\phi_{\mathbf{x}}(F)| + |\phi_{\mathbf{y}}(F)| + \sum_{j \in [3]} |\phi_j(F)|$. Employing the previous two lemmas and subtracting the owned data of the processor yields the mentioned bound.

Multi-TTM with cubical tensors

Corollary

Any computationally load balanced atomic Multi-TTM algorithm that starts and ends with one copy of the data distributed across processors involving 3-dimensional cubical tensors with dimensions $n^{\frac{1}{3}} \times n^{\frac{1}{3}} \times n^{\frac{1}{3}}$ and $r^{\frac{1}{3}} \times r^{\frac{1}{3}} \times r^{\frac{1}{3}}$ (with $n \geq r$) performs at least

$$3 \left(\frac{nr}{P} \right)^{\frac{1}{3}} + r - \frac{3(nr)^{\frac{1}{3}} + r}{P}$$

sends or receives when $P < \frac{n}{r}$ and at least

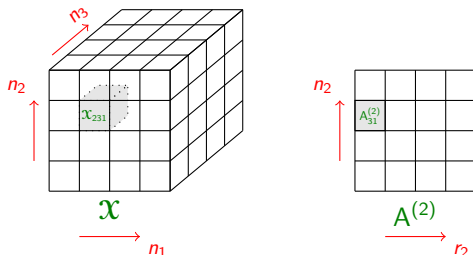
$$3 \left(\frac{nr}{P} \right)^{\frac{1}{3}} + 2 \left(\frac{nr}{P} \right)^{\frac{1}{2}} - \frac{n + 3(nr)^{\frac{1}{3}} + r}{P}$$

sends or receives when $P \geq \frac{n}{r}$.

We will mainly focus on $P < \frac{n}{r}$ case throughout the slides.

Data distribution model

P processors are organized in a 6-dimensional $p_1 \times p_2 \times p_3 \times q_1 \times q_2 \times q_3$ logical processor grid.



Subtensor \mathcal{X}_{231} is distributed evenly among processors $(2, 3, 1, *, *, *)$. Similarly, submatrix $A_{31}^{(2)}$ is distributed evenly among processors $(*, 3, *, *, 1, *)$.

Parallel Multi-TTM algorithm

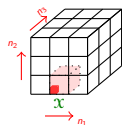
Algorithm 3 Parallel Atomic 3-dimensional Multi-TTM

Require: \mathcal{X} , $A^{(1)}$, $A^{(2)}$, $A^{(3)}$, $p_1 \times p_2 \times p_3 \times q_1 \times q_2 \times q_3$ logical processor grid

Ensure: \mathcal{Y} such that $\mathcal{Y} = \mathcal{X} \times_1 A^{(1)\top} \times_2 A^{(2)\top} \times_3 A^{(3)\top}$

- 1: $(p'_1, p'_2, p'_3, q'_1, q'_2, q'_3)$ is my processor id
 - 2: //All-gather input tensor \mathcal{X}
 - 3: $\mathcal{X}_{p'_1 p'_2 p'_3} = \text{All-Gather}(\mathcal{X}, (p'_1, p'_2, p'_3, *, *, *))$
 - 4: //All-gather input matrices
 - 5: $A_{p'_1 q'_1}^{(1)} = \text{All-Gather}(A^{(1)}, (p'_1, *, *, q'_1, *, *))$
 - 6: $A_{p'_2 q'_2}^{(2)} = \text{All-Gather}(A^{(2)}, (*, p'_2, *, *, q'_2, *))$
 - 7: $A_{p'_3 q'_3}^{(3)} = \text{All-Gather}(A^{(3)}, (*, *, p'_3, *, *, q'_3))$
 - 8: //Local computations in a temporary tensor \mathcal{T}
 - 9: $\mathcal{T} = \text{Local-Multi-TTM}(\mathcal{X}_{p'_1 p'_2 p'_3}, A_{p'_1 q'_1}^{(1)}, A_{p'_2 q'_2}^{(2)}, A_{p'_3 q'_3}^{(3)})$
 - 10: //Reduce-scatter the output tensor in $\mathcal{Y}_{q'_1 q'_2 q'_3}$
 - 11: $\text{Reduce-Scatter}(\mathcal{Y}_{q'_1 q'_2 q'_3}, \mathcal{T}, (*, *, *, q'_1, q'_2, q'_3))$
-

Steps of the algorithm



(a) Perform All-Gather on processors $(2, 1, 1, *, *, *)$ to obtain \mathcal{X}_{211} .



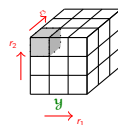
(b) Perform All-Gather on processors $(2, *, *, 1, *, *)$ to obtain $A_{21}^{(1)}$.



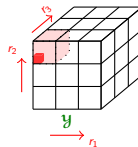
(c) Perform All-Gather on processors $(*, 1, *, *, 3, *)$ to obtain $A_{13}^{(2)}$.



(d) Perform All-Gather on processors $(*, *, 1, *, *, 1)$ to obtain $A_{11}^{(3)}$.



(e) Perform local Multi-TTM to compute partial \mathcal{Y}_{131} .



(f) Perform Reduce-Scatter on processors $(*, *, *, 1, 3, 1)$ to compute/distribute \mathcal{Y}_{131} .

Steps of the algorithm for processor $(2, 1, 1, 1, 3, 1)$, where $p_1 = p_2 = p_3 = q_1 = q_2 = q_3 = 3$. Highlighted areas correspond to the data blocks on which the processor is operating. The dark red highlighting represents the input/output data initially/finally owned by the processor, and the light red highlighting corresponds to received/sent data from/to other processors in All-Gather/Reduce-Scatter collectives to compute \mathcal{Y}_{131} .

Cost analysis

The bandwidth cost of the algorithm is

$$\frac{n}{p} + \frac{n_1 r_1}{p_1 q_1} + \frac{n_2 r_2}{p_2 q_2} + \frac{n_3 r_3}{p_3 q_3} + \frac{r}{q} - \left(\frac{n + n_1 r_1 + n_2 r_2 + n_3 r_3 + r}{P} \right).$$

Here $p = p_1 p_2 p_3$ and $q = q_1 q_2 q_3$. The algorithm is communication optimal when we select p_i and q_i based on lower bounds.

Arithmetic operations

The dimensions of $\mathcal{X}_{p'_1 p'_2 p'_3}$ and \mathcal{T} are $\frac{n_1}{p_1} \times \frac{n_2}{p_2} \times \frac{n_3}{p_3}$ and $\frac{r_1}{q_1} \times \frac{r_2}{q_2} \times \frac{r_3}{q_3}$, respectively.

The dimension of $A_{p'_k q'_k}^{(k)}$ is $\frac{n_i}{p_i} \times \frac{r_i}{q_i}$ for $i = 1, 2, 3$.

- Local Multi-TTM can be performed as a sequence of TTM operations
- Assuming TTM operations are performed in their order, first with $A^{(1)}$, then with $A^{(2)}$, and in the end with $A^{(3)}$,

$$\text{Total arithmetic operations} = 2 \left(\frac{n_1 n_2 n_3 r_1}{p_1 p_2 p_3 q_1} + \frac{n_2 n_3 r_1 r_2}{p_2 p_3 q_1 q_2} + \frac{n_3 r_1 r_2 r_3}{p_3 q_1 q_2 q_3} \right).$$

Multi-TTM cost in TuckerMPI library

- State-of-the-art library for parallel Tucker decomposition
- Implement ST-HOSVD algorithm – employs TTM-in-sequence approach to perform Multi-TTM
- Assume TTMs are performed in increasing mode order

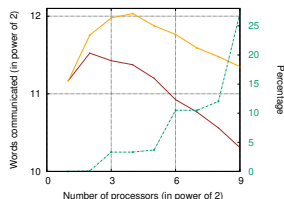
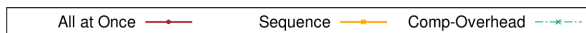
It uses a $\tilde{p}_1 \times \tilde{p}_2 \times \tilde{p}_3$ logical processor grid. The bandwidth cost is

$$\frac{r_1 n_2 n_3}{\tilde{p}_2 \tilde{p}_3} + \frac{n_1 r_1}{\tilde{p}_1} + \frac{r_1 r_2 n_3}{\tilde{p}_1 \tilde{p}_3} + \frac{n_2 r_2}{\tilde{p}_2} + \frac{r_1 r_2 r_3}{\tilde{p}_1 \tilde{p}_2} + \frac{n_3 r_3}{\tilde{p}_3} - \frac{r_1 n_2 n_3 + r_1 r_2 n_3 + r_1 r_2 r_3 + n_1 r_1 + n_2 r_2 + n_3 r_3}{P}.$$

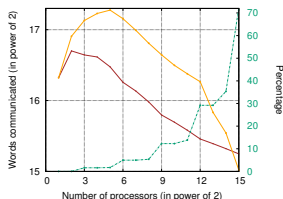
The parallel computational cost is

$$2 \left(\frac{r_1 n_1 n_2 n_3 + r_1 r_2 n_2 n_3 + r_1 r_2 r_3 n_3}{P} \right).$$

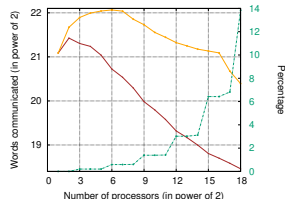
Comparison of All-at-once and TTM-in-Sequence



(a) $n_i = 2^8$, $r_i = 2^3$.



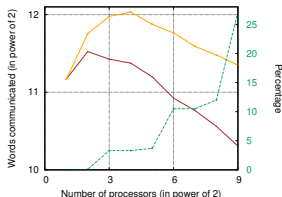
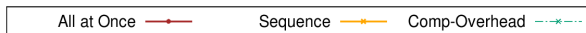
(b) $n_i = 2^{11}$, $r_i = 2^5$.



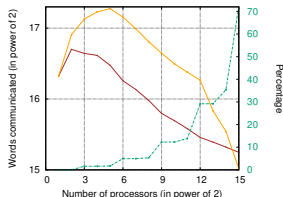
(c) $n_i = 2^{15}$, $r_i = 2^6$.

Communication cost comparison of all-at-once approach (the presented algorithm) and TTM-in-Sequence approach (of TuckerMPI). *Comp-Overhead* shows the percentage of computational overhead of the all-at-once approach with respect to the TTM-in-Sequence approach. Cost of an approach represents the minimum cost among all possible processor configurations.

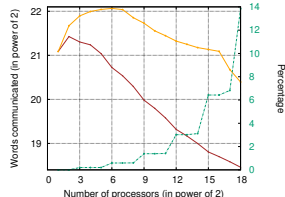
Comparison of All-at-once and TTM-in-Sequence



(a) $n_i = 2^8$, $r_i = 2^3$.



(b) $n_i = 2^{11}$, $r_i = 2^5$.



(c) $n_i = 2^{15}$, $r_i = 2^6$.

- Not any clear winner for all settings
- All-at-once approach performs significantly less communication for small P
- Computational overhead of all-at-once approach is negligible for small P
- TTM-in-Sequence approach is better for large P

Table of Contents

1 Parallel Multi-TTM computation

- 3-dimensional Multi-TTM
- d -dimensional Multi-TTM

Communication lower bound

Theorem

Any computationally load balanced atomic Multi-TTM algorithm that starts and ends with one copy of the data distributed across processors and involves d -dimensional tensors with dimensions n_1, n_2, \dots, n_d and r_1, r_2, \dots, r_d performs at least $A + B - \left(\frac{n}{P} + \frac{r}{P} + \sum_{j=1}^d \frac{n_j r_j}{P} \right)$ sends or receives where

$$A = \begin{cases} \sum_{j=1}^{d-1} n_j r_j + \frac{N_1 R_1}{P} & \text{if } P < \frac{N_1 R_1}{n_{d-1} r_{d-1}}, \\ \sum_{j=1}^{(d-i)} n_j r_j + i \left(\frac{N_i R_i}{P} \right)^{\frac{1}{i}} & \text{if } \frac{N_{i-1} R_{i-1}}{(n_{d+1-i} r_{d+1-i})^{i-1}} \leq P < \frac{N_i R_i}{(n_{d-i} r_{d-i})^i}, \\ & \text{for some } 2 \leq i \leq d-1, \\ d \left(\frac{N_d R_d}{P} \right)^{\frac{1}{d}} & \text{if } \frac{N_{d-1} R_{d-1}}{(n_1 r_1)^{d-1}} \leq P. \end{cases}$$
$$B = \begin{cases} r + \frac{n}{P} & \text{if } P < \frac{n}{r}, \\ 2 \left(\frac{nr}{P} \right)^{\frac{1}{2}} & \text{if } \frac{n}{r} \leq P. \end{cases}$$

Parallel Multi-TTM algorithm

Algorithm 4 Parallel Atomic d-dimensional Multi-TTM

Require: $\mathcal{X}, A^{(1)}, \dots, A^{(d)}, p_1 \times \dots \times p_d \times q_1 \times \dots \times q_d$ logical processor grid

Ensure: \mathcal{Y} such that $\mathcal{Y} = \mathcal{X} \times_1 A^{(1)\top} \dots \times_d A^{(d)\top}$

- 1: $(p'_1, \dots, p'_d, q'_1, \dots, q'_d)$ is my processor id
 - 2: //All-gather input tensor \mathcal{X}
 - 3: $\mathcal{X}_{p'_1 \dots p'_d} = \text{All-Gather}(\mathcal{X}, (p'_1, \dots, p'_d, *, \dots, *))$
 - 4: //All-gather all input matrices
 - 5: **for** $i = 1, \dots, d$ **do**
 - 6: $A_{p'_i q'_i}^{(i)} = \text{All-Gather}(A^{(i)}, (*, \dots, *, p'_i, * \dots, *, q'_i, *))$
 - 7: **end for**
 - 8: //Perform local computations in a temporary tensor \mathcal{T}
 - 9: $\mathcal{T} = \text{Local-Multi-TTM}(\mathcal{X}_{p'_1 \dots p'_d}, A_{p'_1 q'_1}^{(1)}, \dots, A_{p'_d q'_d}^{(d)})$
 - 10: //Reduce-scatter the output tensor in $\mathcal{Y}_{q'_1 \dots q'_d}$
 - 11: $\text{Reduce-Scatter}(\mathcal{Y}_{q'_1 \dots q'_d}, \mathcal{T}, (*, \dots, *, q'_1, \dots, q'_d))$
-

The algorithm is communication optimal when p_i and q_i are selected based on the lower bound.

- Cost analysis of several ways to perform Multi-TTM
 - Unifying all-at-once and sequence approaches
 - Study of communication-computation trade-off

- Optimal costs for algorithms to compute Tucker decompositions