Communication-Optimal Algorithms for CP Decompositions of Dense Tensors

Grey Ballard, Koby Hayashi, Ramakrishnan Kannan,
Nicholas Knight, Kathryn Rouse

WAKE FOREST UNIVERSITY

May 8, 2018

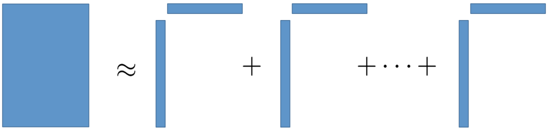
SIAM Conference on Applied Linear Algebra
MS04: Constrained Low-Rank Matrix and Tensor Approximations

Summary

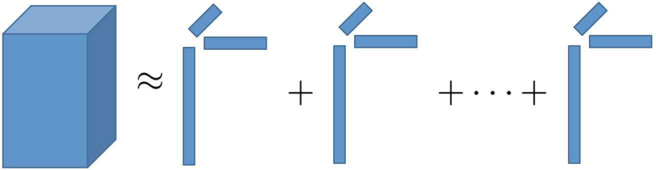
- We establish communication lower bounds for matricized-tensor times Khatri-Rao product (MTTKRP)
  - key kernel for computing CP decomposition
- We present optimal parallel dense MTTKRP algorithm
  - attains the lower bound to within constant factors
- We implement and benchmark optimal CP-ALS algorithm
  - remains computation bound and scales well
  - dimension tree optimization avoids redundant computation

CP Decomposition: sum of outer products

Matrix: $$\mathbf{M} \approx \sum_{r=1}^{R} \mathbf{u}_r(\sigma_r \mathbf{v}_r^T)$$

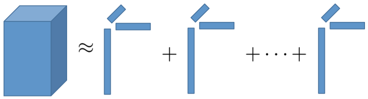


Tensor: $$\mathcal{X} \approx \sum_{r=1}^{R} \mathbf{u}_r \circ \mathbf{v}_r \circ \mathbf{w}_r$$



This is known as the CANDECOMP or PARAFAC or canonical polyadic or CP decomposition

CP Optimization Problem



For fixed rank R, we want to solve

$$\min_{\mathbf{U},\mathbf{V},\mathbf{W}} \left\| \mathcal{X} - \sum_{r=1}^{R} \mathbf{u}_r \circ \mathbf{v}_r \circ \mathbf{w}_r \right\|$$

which is a nonlinear, nonconvex optimization problem

- in the matrix case, the SVD gives us the optimal solution
- in the tensor case, need iterative optimization scheme

Alternating Least Squares (ALS)

Fixing all but one factor matrix, we have a linear LS problem:

$$\min_{\mathbf{V}} \left\| \mathcal{X} - \sum_{r=1}^{R} \hat{\mathbf{u}}_r \circ \mathbf{v}_r \circ \hat{\mathbf{w}}_r \right\|$$

or equivalently

$$\min_{\mathbf{V}} \left\| \mathbf{X}_{(2)} - \mathbf{V}(\hat{\mathbf{W}} \odot \hat{\mathbf{U}})^{\mathsf{T}} \right\|_F$$

$\odot$ is the Khatri-Rao product, a column-wise Kronecker product or row-wise Hadamard (element-wise) product

ALS works by alternating over factor matrices, updating one at a time by solving the corresponding linear LS problem

CP-ALS

Repeat

1. Solve $\mathbf{U}(\mathbf{V}^\mathsf{T}\mathbf{V} * \mathbf{W}^\mathsf{T}\mathbf{W}) = \mathbf{X}_{(1)}(\mathbf{W} \odot \mathbf{V})$ for $\mathbf{U}$
2. Solve $\mathbf{V}(\mathbf{U}^\mathsf{T}\mathbf{U} * \mathbf{W}^\mathsf{T}\mathbf{W}) = \mathbf{X}_{(2)}(\mathbf{W} \odot \mathbf{U})$ for $\mathbf{V}$
3. Solve $\mathbf{W}(\mathbf{U}^\mathsf{T}\mathbf{U} * \mathbf{V}^\mathsf{T}\mathbf{V}) = \mathbf{X}_{(3)}(\mathbf{V} \odot \mathbf{U})$ for $\mathbf{W}$

Linear least squares problems solved via normal equations using identity $(\mathbf{A} \odot \mathbf{B})^\mathsf{T}(\mathbf{A} \odot \mathbf{B}) = \mathbf{A}^\mathsf{T}\mathbf{A} * \mathbf{B}^\mathsf{T}\mathbf{B}$, where $*$ is Hadamard product

All optimization schemes that compute the gradient must also compute MTTKRP in all modes: e.g.,

$$\frac{\partial f}{\partial \mathbf{V}} = \mathbf{V}(\mathbf{U}^\mathsf{T}\mathbf{U} * \mathbf{W}^\mathsf{T}\mathbf{W}) - \mathbf{X}_{(2)}(\mathbf{W} \odot \mathbf{U})$$

MTTKRP bottleneck

- How do we compute MTTKRP efficiently?

- How do we parallelize MTTKRP efficiently?
  - how do we load balance computation?
  - how do we minimize communication?

MTTKRP via Matrix Multiplication

MTTKRP: $\mathbf{M} = \mathbf{X}_{(2)}(\mathbf{W} \odot \mathbf{U})$

Standard approach to MTTKRP for dense tensors

1. "form" matricized tensor (a matrix)
2. compute Khatri-Rao product (a matrix)
3. call matrix-matrix multiplication subroutine

Can we communicate less by exploiting tensor structure?
(avoiding forming explicit Khatri-Rao product)

MTTKRP for 3-way Tensors

Matrix equation:

$$\mathbf{M} = \mathbf{X}_{(2)}(\mathbf{W} \odot \mathbf{U})$$

Element equation:

$$m_{jr} = \sum_{i=1}^{I} \sum_{k=1}^{K} x_{ijk} u_{ir} w_{kr}$$

Example pseudocode:

for $i = 1$ to I **do**
  for $j = 1$ to J **do**
    for $k = 1$ to K **do**
      for $r = 1$ to R **do**
        $\mathbf{M}(j, r) \mathrel{+}= \mathcal{X}(i, j, k) \cdot \mathbf{U}(i, r) \cdot \mathbf{W}(k, r)$

MTTKRP for N-way Tensors

Matrix equation:

$$\mathbf{M}^{(n)} = \mathbf{X}_{(n)}(\mathbf{U}^{(N)} \odot \cdots \odot \mathbf{U}^{(n+1)} \odot \mathbf{U}^{(n-1)} \odot \cdots \odot \mathbf{U}^{(1)})$$

Element equation:

$$m^{(n)}_{i_n r} = \sum x_{i_1 \dots i_N} \prod_{m \neq n} u^{(m)}_{i_m r}$$

Example pseudocode:

for $i_1 = 1$ to I_1 **do**

$\ddots$

for $i_N = 1$ to I_N **do**

for $r = 1$ to R **do**

$\mathbf{M}^{(n)}(i_n, r) \mathrel{+}= \mathcal{X}(i_1, \dots, i_N) \cdot \mathbf{U}^{(1)}(i_1, r) \cdots \mathbf{U}^{(N)}(i_N, r)$

Communication Lower Bounds for MTTKRP

MTTKRP is a set of nested loops that accesses arrays

- Nick's PhD thesis was "Communication-Optimal Loop Nests"
- References: thesis [Kni15] and paper [CDK$^+$13]

From Nick's thesis...

- tabulate how the arrays are accessed
- use Hölder-Brascamp-Lieb-type inequality in LB proof
- solve linear program to get tightest lower bound

MTTKRP Lower Bound Proofs

- Lower bound argument follows [CDK$^+$13] almost directly

MTTKRP Lower Bound Proofs

- Lower bound argument follows [CDK^{+}13] almost directly
- Gotcha: the number of nested loops is not constant
  - Fixed using a technique similar to one used for tightening the constant in matrix multiplication lower bound [SvdG17]

MTTKRP Lower Bound Proofs

- Lower bound argument follows [CDK+13] almost directly
- Gotcha: the number of nested loops is not constant
  - Fixed using a technique similar to one used for tightening the constant in matrix multiplication lower bound [SvdG17]
- Gotcha: memory-independent bounds most relevant
  - inspiration from matrix multiplication [BDH+12, DEF+13]

MTTKRP Lower Bound Proofs

- Lower bound argument follows [CDK$^+$13] almost directly
- Gotcha: the number of nested loops is not constant
  - Fixed using a technique similar to one used for tightening the constant in matrix multiplication lower bound [SvdG17]
- Gotcha: memory-independent bounds most relevant
  - inspiration from matrix multiplication [BDH$^+$12, DEF$^+$13]
- Key assumption: algorithm is not allowed to pre-compute and re-use temporary values
  - e.g., forming explicit Khatri-Rao product
  - e.g., computing and re-using “partial” MTTKRP

Parallel Communication Lower Bound

Theorem

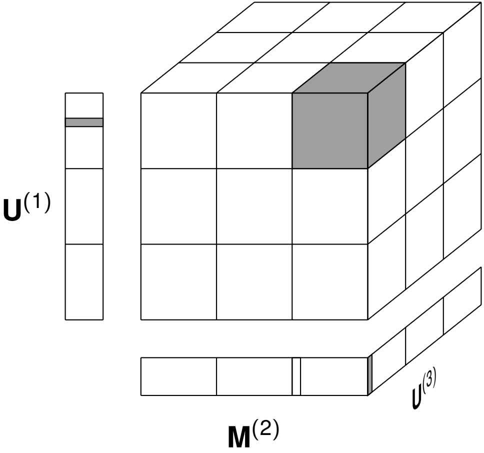
Any parallel MTTKRP algorithm involving a tensor with $I_k = I^{1/N}$ for all k and that evenly distributes one copy of the input and output performs at least

$$\Omega\left(\left(\frac{NIR}{P}\right)^{\frac{N}{2N-1}} + NR\left(\frac{I}{P}\right)^{1/N}\right)$$

sends and receives. (Either term can dominate.)

- N is the number of modes
- I is the number of tensor entries
- I_k is the dimension of the kth mode
- R is the rank of the CP model
- P is the number of processors

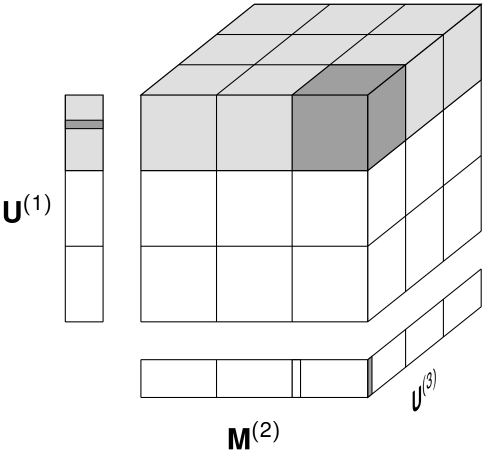
Communication-Optimal Parallel Algorithm (3D)



Each processor

1. Starts with one subtensor and subset of rows of each input factor matrix

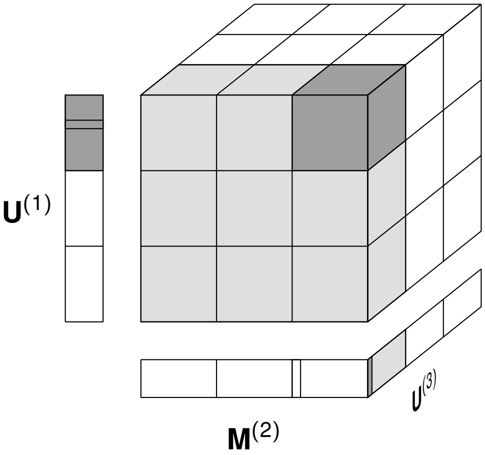
Communication-Optimal Parallel Algorithm (3D)



Each processor

1. Starts with one subtensor and subset of rows of each input factor matrix
2. All-Gathers all the rows needed from $\mathbf{U}^{(1)}$

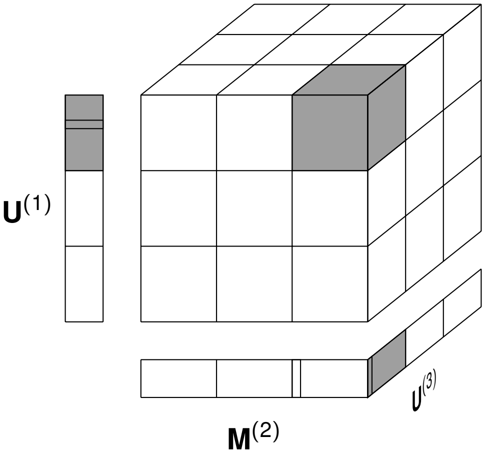
Communication-Optimal Parallel Algorithm (3D)



Each processor

1. Starts with one subtensor and subset of rows of each input factor matrix
2. All-Gathers all the rows needed from $\mathbf{U}^{(1)}$
3. All-Gathers all the rows needed from $\mathbf{U}^{(3)}$

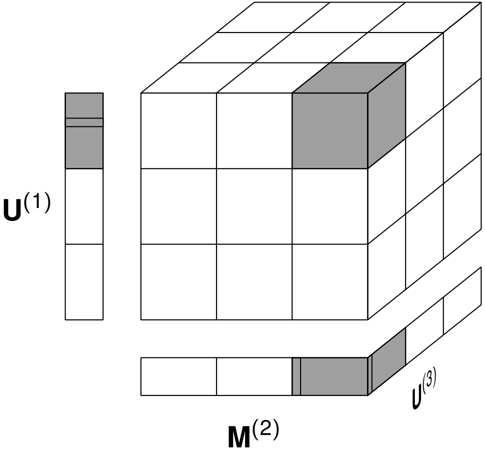
Communication-Optimal Parallel Algorithm (3D)



Each processor

1. Starts with one subtensor and subset of rows of each input factor matrix
2. All-Gathers all the rows needed from $\mathbf{U}^{(1)}$
3. All-Gathers all the rows needed from $\mathbf{U}^{(3)}$
4. Computes its contribution to rows of $\mathbf{M}^{(2)}$ (local MTTKRP)

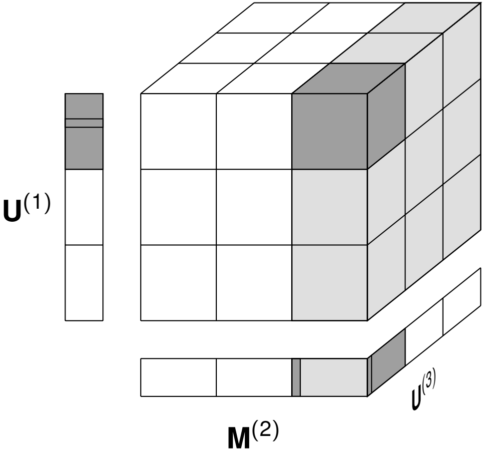
Communication-Optimal Parallel Algorithm (3D)



Each processor

1. Starts with one subtensor and subset of rows of each input factor matrix
2. All-Gathers all the rows needed from $\mathbf{U}^{(1)}$
3. All-Gathers all the rows needed from $\mathbf{U}^{(3)}$
4. Computes its contribution to rows of $\mathbf{M}^{(2)}$ (local MTTKRP)

Communication-Optimal Parallel Algorithm (3D)



Each processor

1. Starts with one subtensor and subset of rows of each input factor matrix
2. All-Gathers all the rows needed from $\mathbf{U}^{(1)}$
3. All-Gathers all the rows needed from $\mathbf{U}^{(3)}$
4. Computes its contribution to rows of $\mathbf{M}^{(2)}$ (local MTTKRP)
5. Reduce-Scatters to compute and distribute $\mathbf{M}^{(2)}$ evenly

Theoretical Comparisons

	Lower Bound	New Algorithm	Standard (MM*)
Words ("small" P)	$\Omega\left(NR\left(\frac{I}{P}\right)^{1/N}\right)$	$O\left(NR\left(\frac{I}{P}\right)^{1/N}\right)$	$O\left(I^{1/N}R\right)$

- For relatively small P (or small R) and even dimensions, parallel "stationary" algorithm attains lower bound
  - same algorithm for sparse [SK16] and dense 3D [LKL$^+$17]

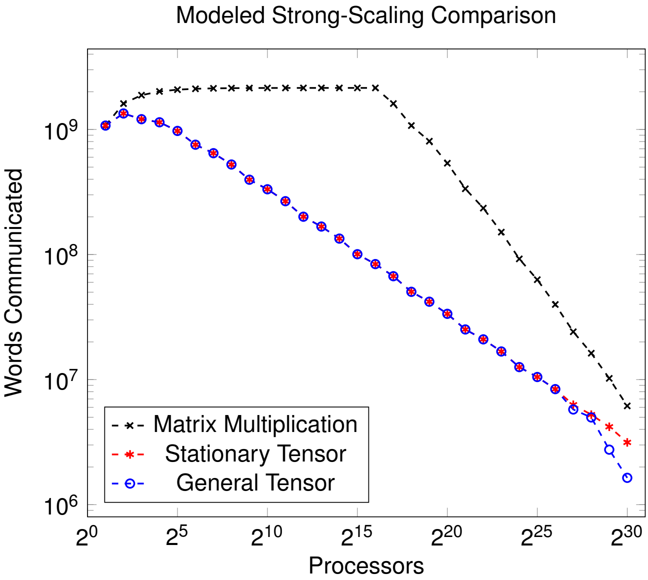
Theoretical Comparisons

	Lower Bound	New Algorithm	Standard (MM*)
Words ("small" P)	$\Omega\left(NR\left(\frac{I}{P}\right)^{1/N}\right)$	$O\left(NR\left(\frac{I}{P}\right)^{1/N}\right)$	$O\left(I^{1/N}R\right)$
Words ("large" P)	$\Omega\left(\left(\frac{NIR}{P}\right)^{\frac{N}{2N-1}}\right)$	$O\left(\left(\frac{NIR}{P}\right)^{\frac{N}{2N-1}}\right)$	$O\left(\left(\frac{IR}{P}\right)^{2/3}\right)$

- For relatively small P (or small R) and even dimensions, parallel "stationary" algorithm attains lower bound
  - same algorithm for sparse [SK16] and dense 3D [LKL$^+$17]
- For larger P (or R), then we need more general algorithm to attain lower bound
  - involves communicating the tensor

*communication-optimal matrix multiplication from [DEF$^+$13]

Modeled Communication Costs



What about for a full CP-ALS iteration?

A full iteration of CP-ALS includes computing all N MTTKRPs

Lower Bound
Lower bound for single MTTKRP applies to computing all N

Algorithm
We can compute all N with same communication as just 1

- lots of data overlap across MTTKRPs
- more computation required, but not that much more

Avoiding re-communication across MTTKRPs

while not converged **do**
  for $n = 1$ to N **do**
    % *Compute new factor matrix in nth mode*
    $\mathbf{M} = \text{Local-MTTKRP}(\mathcal{X}_{p_1 \cdots p_N}, \{\mathbf{U}_{p_i}^{(i)}\}, n)$
    $\mathbf{M}_{\mathbf{p}}^{(n)} = \text{Reduce-Scatter}(\mathbf{M}, \textsc{Proc-Slice}(n, p_n))$
    $\mathbf{S}^{(n)} = \mathbf{G}^{(1)} * \cdots * \mathbf{G}^{(n-1)} * \mathbf{G}^{(n+1)} * \cdots * \mathbf{G}^{(N)}$
    $\mathbf{U}_{\mathbf{p}}^{(n)} = \text{Local-Update}(\mathbf{S}^{(n)}, \mathbf{M}_{\mathbf{p}}^{(n)})$
    % *Organize data for later modes*
    $\mathbf{H} = \mathbf{U}_{\mathbf{p}}^{(n)\mathsf{T}} \mathbf{U}_{\mathbf{p}}^{(n)}$
    $\mathbf{G}^{(n)} = \text{All-Reduce}(\mathbf{H}, \textsc{All-Procs})$
    $\mathbf{U}_{p_n}^{(n)} = \text{All-Gather}(\mathbf{U}_{\mathbf{p}}^{(n)}, \textsc{Proc-Slice}(n, p_n))$

Compute factor matrix, communicate it **once** for use in all other $N-1$ modes

Avoiding recomputation across MTTKRPs

We re-use communication and computation across MTTKRPs

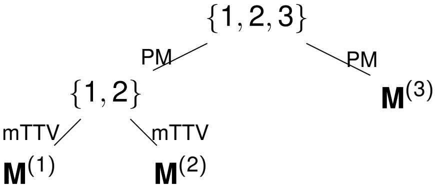
$$\mathbf{M}^{(1)} = \underline{\mathbf{X}_{(1)} \left(\mathbf{U}^{(3)}\right.} \odot \left.\mathbf{U}^{(2)}\right) \qquad \text{and} \qquad \mathbf{M}^{(2)} = \underline{\mathbf{X}_{(2)} \left(\mathbf{U}^{(3)}\right.} \odot \left.\mathbf{U}^{(1)}\right)$$

Avoiding recomputation across MTTKRPs

We re-use communication <u>and</u> computation across MTTKRPs

$$\mathbf{M}^{(1)} = \underline{\mathbf{X}_{(1)} \big(\mathbf{U}^{(3)}} \odot \mathbf{U}^{(2)}\big) \quad \text{and} \quad \mathbf{M}^{(2)} = \underline{\mathbf{X}_{(2)} \big(\mathbf{U}^{(3)}} \odot \mathbf{U}^{(1)}\big)$$

We organize intermediate values in "dimension tree" [PTC13, LCP$^+$17, KU18]

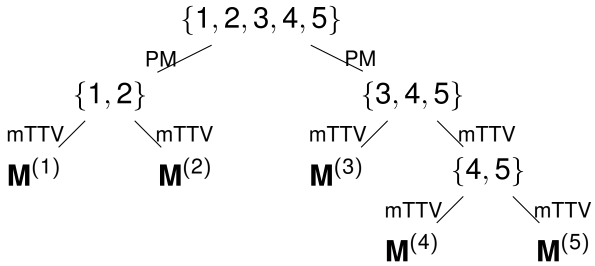


PM = Partial MTTKRP   mTTV = multi-Tensor-Times-Vector

We re-use communication <u>and</u> computation across MTTKRPs

$$\mathbf{M}^{(1)} = \underline{\mathbf{X}_{(1)} \Big(\mathbf{U}^{(3)}} \odot \mathbf{U}^{(2)}\Big) \qquad \text{and} \qquad \mathbf{M}^{(2)} = \underline{\mathbf{X}_{(2)} \Big(\mathbf{U}^{(3)}} \odot \mathbf{U}^{(1)}\Big)$$

We organize intermediate values in “dimension tree” [PTC13, LCP$^+$17, KU18]



PM = Partial MTTKRP   mTTV = multi-Tensor-Times-Vector

Implementation

- Uses CP-ALS for non-negative CP problems
  - minimize least squares loss function
  - use block principal pivoting [KP11] to solve subproblems
- Avoids redundant communication across MTTKRPs
- Avoids redundant computation across MTTKRPs using dimension trees

Strong Scaling Results (3D)

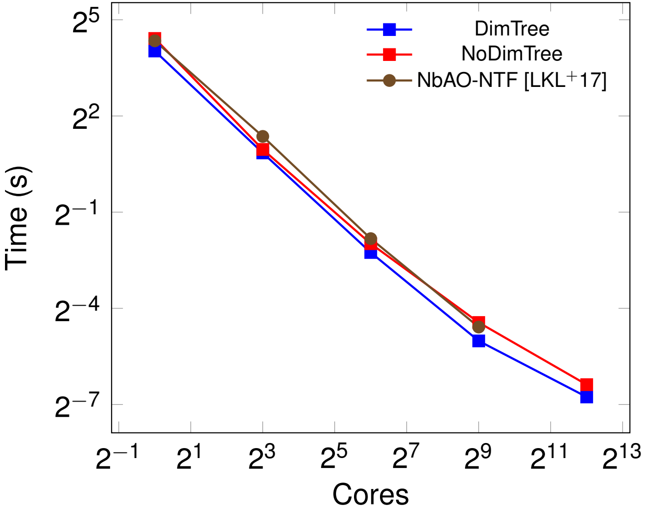


Figure: $1024 \times 1024 \times 1024$ tensor on $2^k \times 2^k \times 2^k$ proc grids ($R = 32$)

Strong Scaling Results (5D)

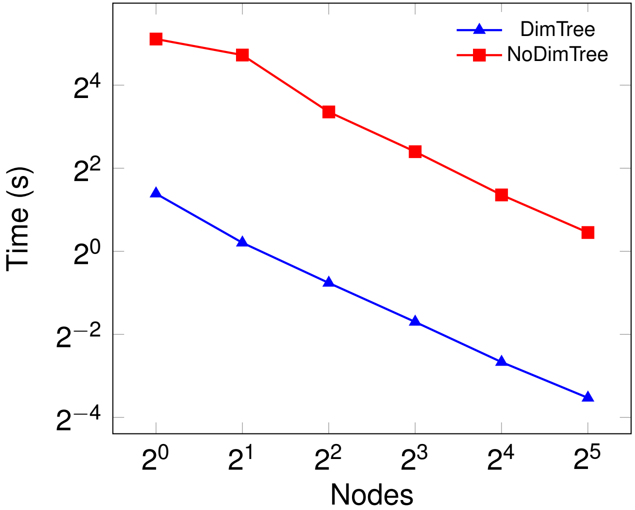


Figure: $64 \times 64 \times 64 \times 64 \times 64$ tensor ($R = 32$)

Varying Rank Results (3D)

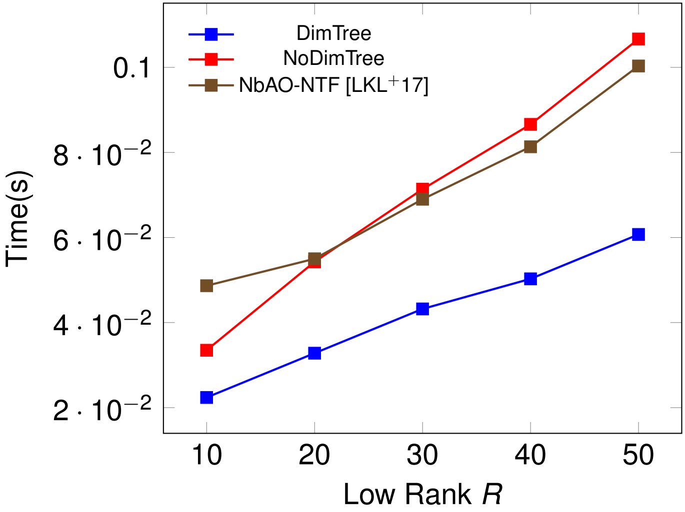


Figure: 30,012 × 1200 × 500 tensor on 120 × 6 × 2 proc grid

Varying Rank Results (4D)

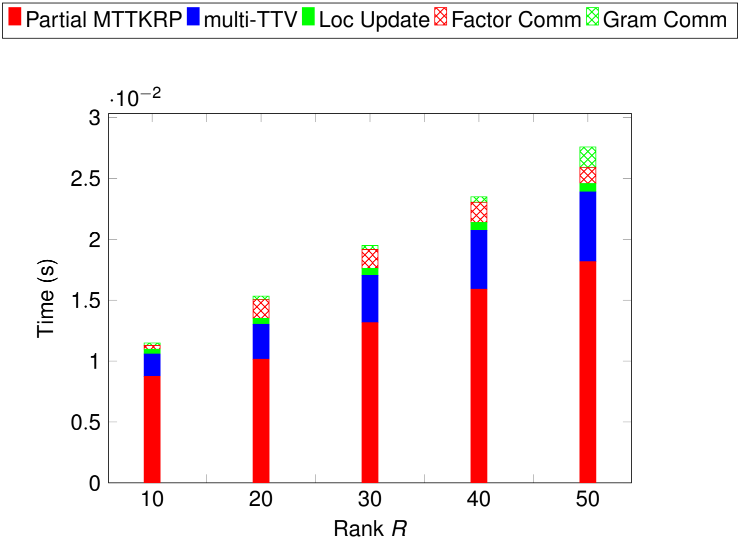


Figure: $1344 \times 1024 \times 33 \times 9$ tensor on $8 \times 8 \times 1 \times 1$ proc grid

Summary

- We establish communication lower bounds for matricized-tensor times Khatri-Rao product (MTTKRP)
  - key kernel for computing CP decomposition
- We present optimal parallel dense MTTKRP algorithm
  - attains the lower bound to within constant factors
- We implement and benchmark optimal CP-ALS algorithm
  - remains computation bound and scales well
  - dimension tree optimization avoids redundant computation

References I

G. Ballard, J. Demmel, O. Holtz, B. Lipshitz, and O. Schwartz.
Brief announcement: strong scaling of matrix multiplication algorithms and memory-independent communication lower bounds.
In Proceedings of the 24th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA '12, pages 77–79, New York, NY, USA, June 2012. ACM.

M. Christ, J. Demmel, N. Knight, T. Scanlon, and K. Yelick.
Communication lower bounds and optimal algorithms for programs that reference arrays - part 1.
Technical Report UCB/EECS-2013-61, EECS Department, University of California, Berkeley, May 2013.

J. Demmel, D. Eliahu, A. Fox, S. Kamil, B. Lipshitz, O. Schwartz, and O. Spillinger.
Communication-optimal parallel recursive rectangular matrix multiplication.
In Proceedings of the 27th IEEE International Symposium on Parallel and Distributed Processing, IPDPS '13, pages 261–272, 2013.

Nicholas Knight.
Communication-Optimal Loop Nests.
PhD thesis, EECS Department, University of California, Berkeley, Aug 2015.

Jingu Kim and Haesun Park.
Fast nonnegative matrix factorization: An active-set-like method and comparisons.
SIAM Journal on Scientific Computing, 33(6):3261–3281, 2011.

Oguz Kaya and Bora Uçar.
Parallel candecomp/parafac decomposition of sparse tensors using dimension trees.
SIAM Journal on Scientific Computing, 40(1):C99–C130, 2018.

Jiajia Li, Jee Choi, Ioakeim Perros, Jimeng Sun, and Richard Vuduc.
Model-driven sparse CP decomposition for higher-order tensors.
In IEEE International Parallel and Distributed Processing Symposium, IPDPS, pages 1048–1057, May 2017.

References II

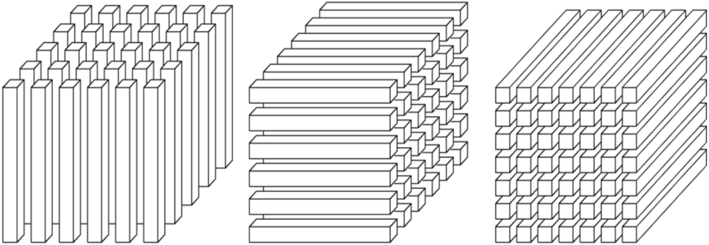
A. P. Liavas, G. Kostoulas, G. Lourakis, K. Huang, and N. D. Sidiropoulos.
Nesterov-based alternating optimization for nonnegative tensor factorization: Algorithm and parallel implementation.
IEEE Transactions on Signal Processing, Nov 2017.

Anh-Huy Phan, Petr Tichavsky, and Andrzej Cichocki.
Fast alternating LS algorithms for high order CANDECOMP/PARAFAC tensor factorizations.
IEEE Transactions on Signal Processing, 61(19):4834–4846, Oct 2013.

Shaden Smith and George Karypis.
A medium-grained algorithm for distributed sparse tensor factorization.
In IEEE 30th International Parallel and Distributed Processing Symposium, pages 902–911, May 2016.

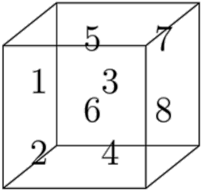
Tyler Michael Smith and Robert A. van de Geijn.
Pushing the bounds for matrix-matrix multiplication.
Technical Report 1702.02017, arXiv, 2017.

Fibers



Mode-1 Fibers Mode-2 Fibers Mode-3 Fibers

A tensor can be decomposed into the fibers of each mode
(fibers are vectors – fix all indices but one)

$\mathcal{X} =$ 

$$\mathbf{X}_{(1)} = \begin{bmatrix} 1 & 3 & 5 & 7 \\ 2 & 4 & 6 & 8 \end{bmatrix}$$

$$\mathbf{X}_{(2)} = \begin{bmatrix} 1 & 2 & 5 & 6 \\ 3 & 4 & 7 & 8 \end{bmatrix}$$

$$\mathbf{X}_{(3)} = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{bmatrix}$$

A tensor can be reshaped into a matrix,
called a matricized tensor or unfolding, for a given mode,
where each column is a fiber

Sequential Communication Lower Bound

Theorem

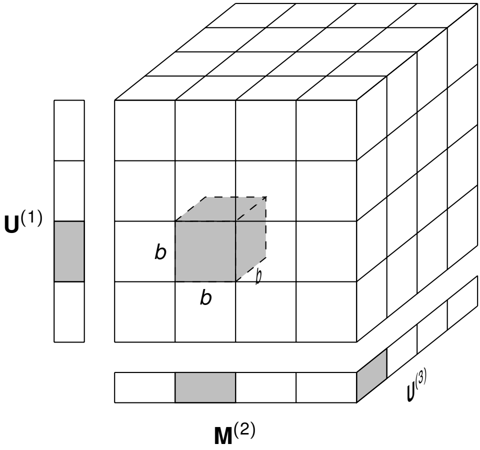
For sufficiently large I, any sequential MTTKRP algorithm performs at least

$$\Omega\left(\frac{NIR}{M^{1-1/N}}\right)$$

loads and stores to/from slow memory.

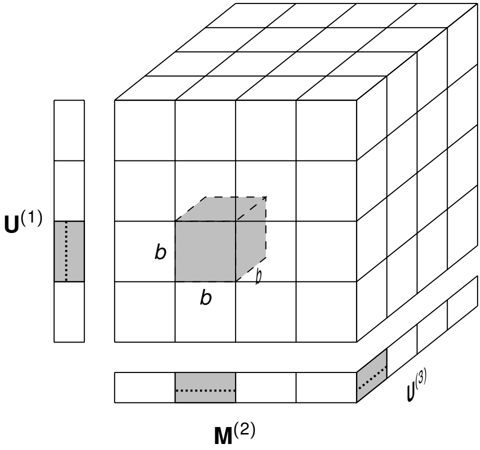
- N is the number of modes
- I is the number of tensor entries
- R is the rank of the CP model
- M is the size of the fast memory

Communication-Optimal Sequential Algorithm (3D)



1. Loop over $b \times \cdots \times b$ blocks of the tensor

Communication-Optimal Sequential Algorithm (3D)



1. Loop over $b \times \cdots \times b$ blocks of the tensor
2. With block in memory, loop over subcolumns of input factor matrices, updating corresponding subcolumn of output matrix

- choose $b \approx M^{1/N}$

Theoretical Comparisons

	Lower Bound	New Algorithm	Standard (MM)
Flops	-	NIR	$2IR$
Words	$\Omega\left(\frac{NIR}{M^{1-1/N}}\right)$	$O\left(I + \frac{NIR}{M^{1-1/N}}\right)$	$O\left(I + \frac{IR}{M^{1/2}}\right)$
Temp Mem	-	-	$\frac{IR}{I_n}$

Theoretical Comparisons

	Lower Bound	New Algorithm	Standard (MM)
Flops	-	NIR	$2IR$
Words	$\Omega\left(\frac{NIR}{M^{1-1/N}}\right)$	$O\left(I + \frac{NIR}{M^{1-1/N}}\right)$	$O\left(I + \frac{IR}{M^{1/2}}\right)$
Temp Mem	-	-	$\frac{IR}{I_n}$

- New algorithm performs $N/2$ more flops than standard
- For relatively small R, I term dominates communication
  - we expect this to be the typical case in practice
- For relatively large R, new algorithm communicates less
  - better exponent on M

MTTKRP Loop Nest

for $i_1 = 1$ to I_1 **do**

$\ddots$

for $i_N = 1$ to I_N **do**

for $r = 1$ to R **do**

$$\mathbf{M}^{(n)}(i_n, r) \mathrel{+}= \mathcal{X}(i_1, \ldots, i_N) * \mathbf{U}^{(1)}(i_1, r) * \cdots * \mathbf{U}^{(N)}(i_N, r)$$

$$\Delta = \begin{array}{c|cccccc}
 & i_1 & \cdots & i_n & \cdots & i_N & r \\
\hline
\mathbf{U}^{(1)} & 1 & & & & & 1 \\
\vdots & & \ddots & & & & \vdots \\
\mathbf{M}^{(n)} & & & 1 & & & 1 \\
\vdots & & & & \ddots & & \vdots \\
\mathbf{U}^{(N)} & & & & & 1 & 1 \\
\mathcal{X} & 1 & \cdots & 1 & \cdots & 1 &
\end{array}$$