Algorithmic and HPC Challenges in Parallel Tensor Computations

Oguz Kaya

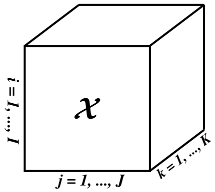
INRIA Bordeaux, France

May 3, 2017

Tensor

What is a tensor?

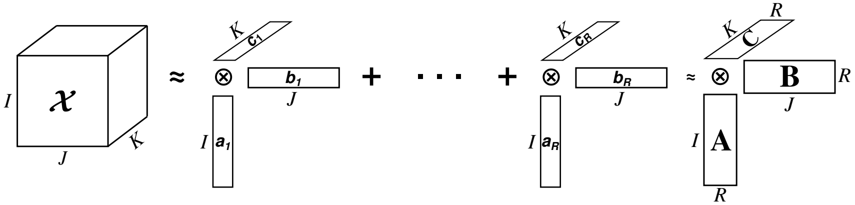
- A vector is a 1-dimensional tensor.
- A matrix is a 2-dimensional tensor.
- A tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ has N dimensions.



$\mathcal{X} \in \mathbb{R}^{I \times J \times K}$

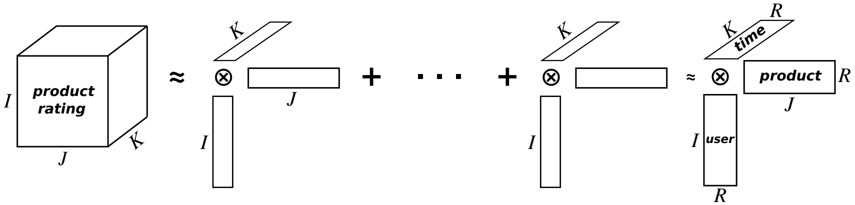
We are mostly interested in the case when $\mathcal{X}$ is **sparse** and of **low rank**.

Tensor Decompositions



- Generalization of matrix decompositions to higher dimensions
- Provide **low-rank representation** of high dimensional data
- CP Decomposition
  - Provides a **rank-R representation** of a tensor with **R rank-1 terms** summed.
  - Minimum R yielding an equality is called the **rank of $\mathcal{X}$**.
- **Goal:** Compute CP decomposition efficiently for a sparse $\mathcal{X}$.

Applications



- Recommender systems
- Analyzing web links
- Link prediction in temporal graphs
- Data compression
- Signal processing, quantum chemistry, etc.

Outline

CP Decomposition

Algorithm CP-ALS for 3rd order tensors

Input: $\mathcal{X}$: A sparse tensor
R: The rank of approximation

Output: CP decomposition $[\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$

repeat

$\hat{\mathbf{A}} \leftarrow \mathbf{X}_{(1)}(\mathbf{C} \odot \mathbf{B})$ ▶ $\hat{\mathbf{A}} \in \mathbb{R}^{I \times R}$

$\mathbf{A} \leftarrow \hat{\mathbf{A}}\ \mathbf{M}_A$ ▶ $\mathbf{M}_A \in \mathbb{R}^{R \times R}$

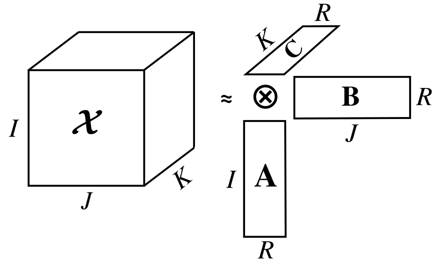
$\hat{\mathbf{B}} \leftarrow \mathbf{X}_{(2)}(\mathbf{C} \odot \mathbf{A})$ ▶ $\hat{\mathbf{B}} \in \mathbb{R}^{J \times R}$

$\mathbf{B} \leftarrow \hat{\mathbf{B}}\ \mathbf{M}_B$ ▶ $\mathbf{M}_B \in \mathbb{R}^{R \times R}$

$\hat{\mathbf{C}} \leftarrow \mathbf{X}_{(3)}(\mathbf{B} \odot \mathbf{A})$ ▶ $\hat{\mathbf{C}} \in \mathbb{R}^{K \times R}$

$\mathbf{C} \leftarrow \hat{\mathbf{C}}\ \mathbf{M}_C$ ▶ $\mathbf{M}_C \in \mathbb{R}^{R \times R}$

until no improvement or max iterations achieved



- $\mathbf{A}, \mathbf{B}, \mathbf{C}$ are initialized (randomly or using HOSVD).
- Algorithm iteratively updates $\mathbf{A}, \mathbf{B}, \mathbf{C}$ until convergence.
- $\hat{\mathbf{A}} \leftarrow \mathbf{X}_{(1)}(\mathbf{C} \odot \mathbf{B})$ is called **matricized tensor-times Khatri-Rao product (MTTKRP)**.

CP Decomposition

Algorithm CP-ALS for 3rd order tensors

Input: $\mathcal{X}$: A sparse tensor

R: The rank of approximation

Output: CP decomposition $[\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$

repeat

$\hat{\mathbf{A}} \leftarrow \mathbf{X}_{(1)}(\mathbf{C} \odot \mathbf{B})$ ▶ $\hat{\mathbf{A}} \in \mathbb{R}^{I \times R}$

$\mathbf{A} \leftarrow \hat{\mathbf{A}}\ \mathbf{M}_A$ ▶ $\mathbf{M}_A \in \mathbb{R}^{R \times R}$

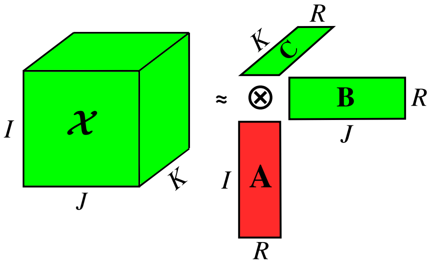
$\hat{\mathbf{B}} \leftarrow \mathbf{X}_{(2)}(\mathbf{C} \odot \mathbf{A})$ ▶ $\hat{\mathbf{B}} \in \mathbb{R}^{J \times R}$

$\mathbf{B} \leftarrow \hat{\mathbf{B}}\ \mathbf{M}_B$ ▶ $\mathbf{M}_B \in \mathbb{R}^{R \times R}$

$\hat{\mathbf{C}} \leftarrow \mathbf{X}_{(3)}(\mathbf{B} \odot \mathbf{A})$ ▶ $\hat{\mathbf{C}} \in \mathbb{R}^{K \times R}$

$\mathbf{C} \leftarrow \hat{\mathbf{C}}\ \mathbf{M}_C$ ▶ $\mathbf{M}_C \in \mathbb{R}^{R \times R}$

until no improvement or max iterations achieved



- $\mathbf{A}, \mathbf{B}, \mathbf{C}$ are initialized (randomly or using HOSVD).
- Algorithm iteratively updates $\mathbf{A}, \mathbf{B}, \mathbf{C}$ until convergence.
- $\hat{\mathbf{A}} \leftarrow \mathbf{X}_{(1)}(\mathbf{C} \odot \mathbf{B})$ is called **matricized tensor-times Khatri-Rao product (MTTKRP)**.

CP Decomposition

Algorithm CP-ALS for 3rd order tensors

Input: $\mathcal{X}$: A sparse tensor
R: The rank of approximation

Output: CP decomposition $[\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$

repeat

$\hat{\mathbf{A}} \leftarrow \mathbf{X}_{(1)}(\mathbf{C} \odot \mathbf{B})$ ▶ $\hat{\mathbf{A}} \in \mathbb{R}^{I \times R}$

$\mathbf{A} \leftarrow \hat{\mathbf{A}}\ \mathbf{M}_A$ ▶ $\mathbf{M}_A \in \mathbb{R}^{R \times R}$

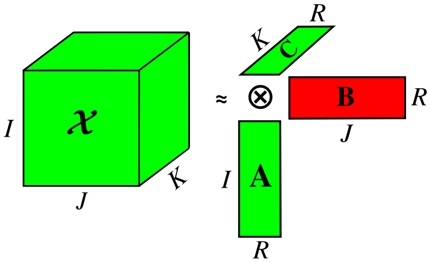
$\hat{\mathbf{B}} \leftarrow \mathbf{X}_{(2)}(\mathbf{C} \odot \mathbf{A})$ ▶ $\hat{\mathbf{B}} \in \mathbb{R}^{J \times R}$

$\mathbf{B} \leftarrow \hat{\mathbf{B}}\ \mathbf{M}_B$ ▶ $\mathbf{M}_B \in \mathbb{R}^{R \times R}$

$\hat{\mathbf{C}} \leftarrow \mathbf{X}_{(3)}(\mathbf{B} \odot \mathbf{A})$ ▶ $\hat{\mathbf{C}} \in \mathbb{R}^{K \times R}$

$\mathbf{C} \leftarrow \hat{\mathbf{C}}\ \mathbf{M}_C$ ▶ $\mathbf{M}_C \in \mathbb{R}^{R \times R}$

until no improvement or max iterations achieved



- $\mathbf{A}, \mathbf{B}, \mathbf{C}$ are initialized (randomly or using HOSVD).
- Algorithm iteratively updates $\mathbf{A}, \mathbf{B}, \mathbf{C}$ until convergence.
- $\hat{\mathbf{A}} \leftarrow \mathbf{X}_{(1)}(\mathbf{C} \odot \mathbf{B})$ is called **matricized tensor-times Khatri-Rao product (MTTKRP)**.

CP Decomposition

Algorithm CP-ALS for 3rd order tensors

Input: $\mathcal{X}$: A sparse tensor
R: The rank of approximation

Output: CP decomposition $[\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$

repeat

$\hat{\mathbf{A}} \leftarrow \mathbf{X}_{(1)}(\mathbf{C} \odot \mathbf{B})$ ▶ $\hat{\mathbf{A}} \in \mathbb{R}^{I \times R}$

$\mathbf{A} \leftarrow \hat{\mathbf{A}}\ \mathbf{M}_A$ ▶ $\mathbf{M}_A \in \mathbb{R}^{R \times R}$

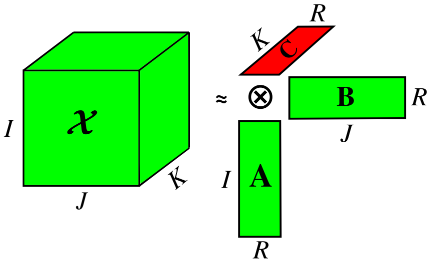
$\hat{\mathbf{B}} \leftarrow \mathbf{X}_{(2)}(\mathbf{C} \odot \mathbf{A})$ ▶ $\hat{\mathbf{B}} \in \mathbb{R}^{J \times R}$

$\mathbf{B} \leftarrow \hat{\mathbf{B}}\ \mathbf{M}_B$ ▶ $\mathbf{M}_B \in \mathbb{R}^{R \times R}$

$\hat{\mathbf{C}} \leftarrow \mathbf{X}_{(1)}(\mathbf{B} \odot \mathbf{A})$ ▶ $\hat{\mathbf{C}} \in \mathbb{R}^{K \times R}$

$\mathbf{C} \leftarrow \hat{\mathbf{C}}\ \mathbf{M}_C$ ▶ $\mathbf{M}_C \in \mathbb{R}^{R \times R}$

until no improvement or max iterations achieved



- $\mathbf{A}, \mathbf{B}, \mathbf{C}$ are initialized (randomly or using HOSVD).
- Algorithm iteratively updates $\mathbf{A}, \mathbf{B}, \mathbf{C}$ until convergence.
- $\hat{\mathbf{A}} \leftarrow \mathbf{X}_{(1)}(\mathbf{C} \odot \mathbf{B})$ is called **matricized tensor-times Khatri-Rao product (MTTKRP)**.

CP Decomposition

Algorithm CP-ALS for 3rd order tensors

Input: $\mathcal{X}$: A sparse tensor

R: The rank of approximation

Output: CP decomposition $[\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$

repeat

$\hat{\mathbf{A}} \leftarrow \mathbf{X}_{(1)}(\mathbf{C} \odot \mathbf{B})$ ▶ $\hat{\mathbf{A}} \in \mathbb{R}^{I \times R}$

$\mathbf{A} \leftarrow \hat{\mathbf{A}}\, \mathbf{M}_A$ ▶ $\mathbf{M}_A \in \mathbb{R}^{R \times R}$

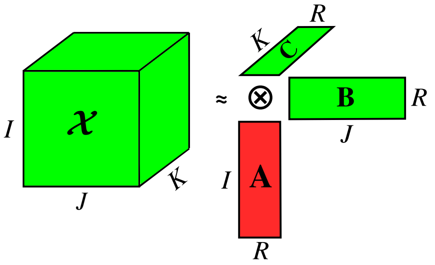
$\hat{\mathbf{B}} \leftarrow \mathbf{X}_{(2)}(\mathbf{C} \odot \mathbf{A})$ ▶ $\hat{\mathbf{B}} \in \mathbb{R}^{J \times R}$

$\mathbf{B} \leftarrow \hat{\mathbf{B}}\, \mathbf{M}_B$ ▶ $\mathbf{M}_B \in \mathbb{R}^{R \times R}$

$\hat{\mathbf{C}} \leftarrow \mathbf{X}_{(3)}(\mathbf{B} \odot \mathbf{A})$ ▶ $\hat{\mathbf{C}} \in \mathbb{R}^{K \times R}$

$\mathbf{C} \leftarrow \hat{\mathbf{C}}\, \mathbf{M}_C$ ▶ $\mathbf{M}_C \in \mathbb{R}^{R \times R}$

until no improvement or max iterations achieved

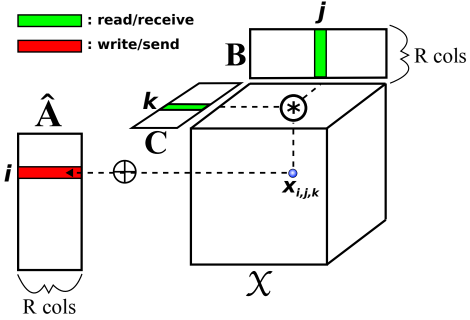


- $\mathbf{A}, \mathbf{B}, \mathbf{C}$ are initialized (randomly or using HOSVD).
- Algorithm iteratively updates $\mathbf{A}, \mathbf{B}, \mathbf{C}$ until convergence.
- $\hat{\mathbf{A}} \leftarrow \mathbf{X}_{(1)}(\mathbf{C} \odot \mathbf{B})$ is called **matricized tensor-times Khatri-Rao product (MTTKRP)**.

MTTKRP ($\hat{\mathbf{A}} \leftarrow \mathbf{X}_{(1)}(\mathbf{C} \odot \mathbf{B})$)

Matricized Tensor-Times Khatri-Rao Product (MTTKRP)

- $\hat{\mathbf{A}} = \mathbf{X}_{(1)}(\mathbf{C} \odot \mathbf{B})$, $\hat{\mathbf{A}} \in \mathbb{R}^{I \times R}$
- Each $x_{i,j,k} \in \boldsymbol{\mathcal{X}}$ multiplies vectors $\mathbf{B}(j,:)$ and $\mathbf{C}(k,:)$, then updates $\hat{\mathbf{A}}(i,:)$.
- How to parallelize?

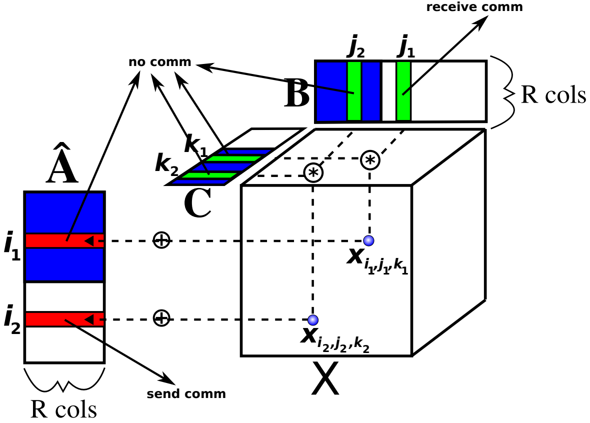


Outline

Parallel MTTKRP ($\hat{\mathbf{A}} \leftarrow \mathbf{X}_{(1)}(\mathbf{C} \odot \mathbf{B})$)

Consider a process p (blue).

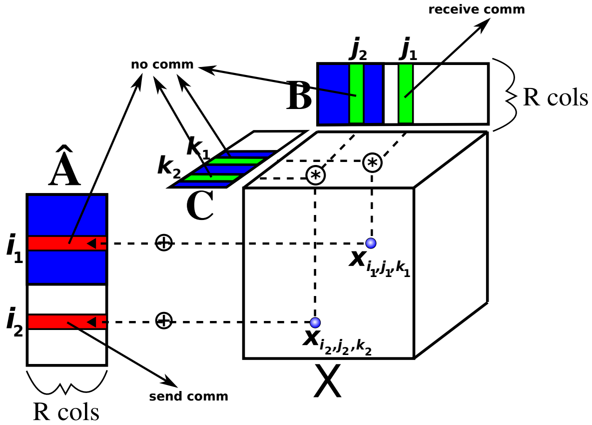
- $\mathcal{X}$ is partitioned; process has the subtensor $\mathcal{X}_p$.
- **A**, **B**, and **C** are partitioned; process p has I_p, J_p, and K_p rows of **A**, **B**, and **C**.



Parallel MTTKRP ($\hat{\mathbf{A}} \leftarrow \mathbf{X}_{(1)}(\mathbf{C} \odot \mathbf{B})$)

Consider a process p (blue).

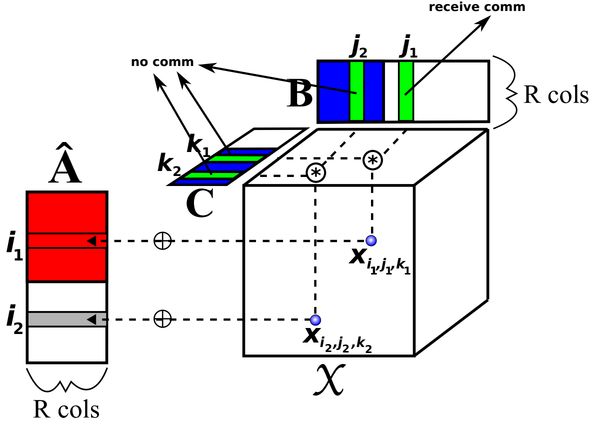
- $x_{i_1,j_1,k_1} \in \mathcal{X}_p$ generates a local result; no communication.
- $x_{i_2,j_2,k_2} \in \mathcal{X}_p$ generates a non-local result; communication needed (**fold**).
- $2R$ ops per nonzero. (for N-dims, $(N-1)R$).



Parallel MTTKRP ($\hat{\mathbf{A}} \leftarrow \mathbf{X}_{(1)}(\mathbf{C} \odot \mathbf{B})$)

Consider a process p (blue).

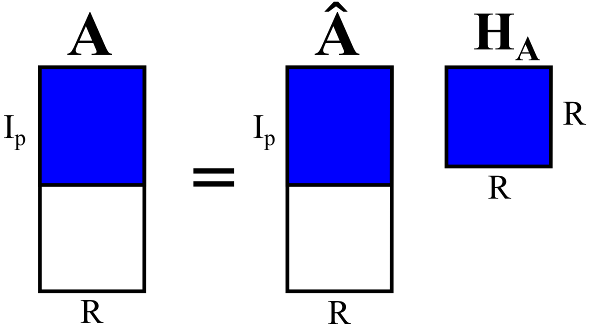
- $x_{i_1,j_1,k_1} \in \mathcal{X}_p$ generates a local result; no communication.
- $x_{i_2,j_2,k_2} \in \mathcal{X}_p$ generates a non-local result; communication needed (**fold**).
- $2R$ ops per nonzero. (for N-dims, $(N-1)R$).



Parallel GEMM ($\mathbf{A} \leftarrow \hat{\mathbf{A}}\mathbf{H}_A$)

Consider a process p (blue).

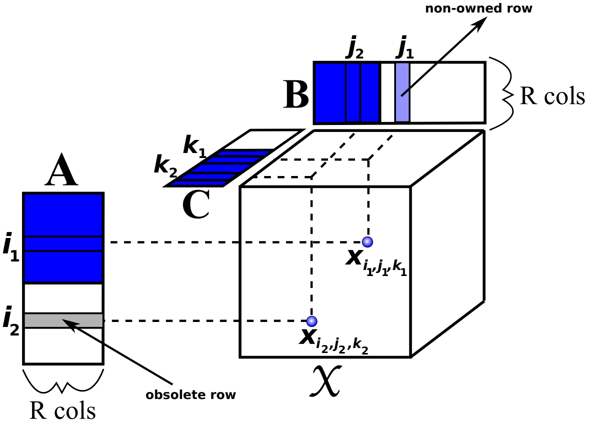
- $\mathbf{A}^T\mathbf{A}$, $\mathbf{B}^T\mathbf{B}$, $\mathbf{C}^T\mathbf{C} \in \mathbb{R}^{R \times R}$ available at each process.
- $\mathbf{H}_A \leftarrow (\mathbf{B}^T\mathbf{B} * \mathbf{C}^T\mathbf{C})^{\dagger}$ computed locally.
- Row-parallel $\mathbf{A} \leftarrow \hat{\mathbf{A}}\mathbf{H}_A$ with I_p rows, $O(I_p R^2)$ ops.



Post-communication (expand)

Consider a process p (blue).

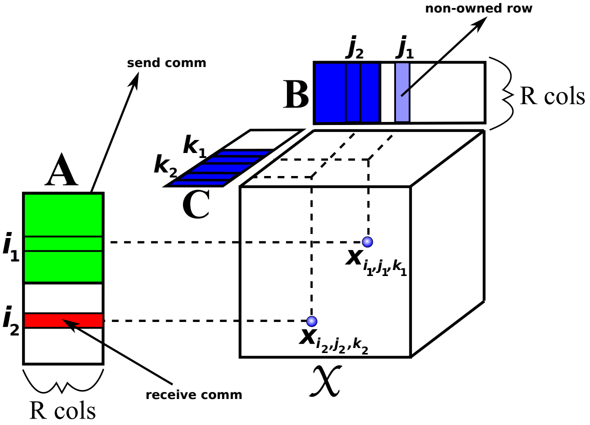
- $\mathbf{A}(i_2, :)$ needs to be received (to later compute $\hat{\mathbf{B}}$ and $\hat{\mathbf{C}}$).



Post-communication (expand)

Consider a process p (blue).

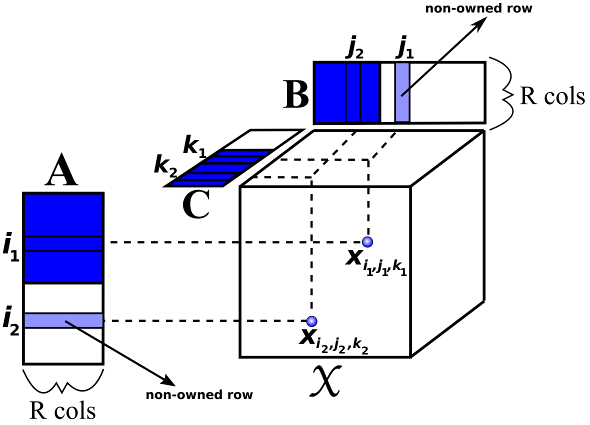
- $\mathbf{A}(i_2,:)$ needs to be received (to later compute $\hat{\mathbf{B}}$ and $\hat{\mathbf{C}}$).



Post-communication (expand)

Consider a process p (blue).

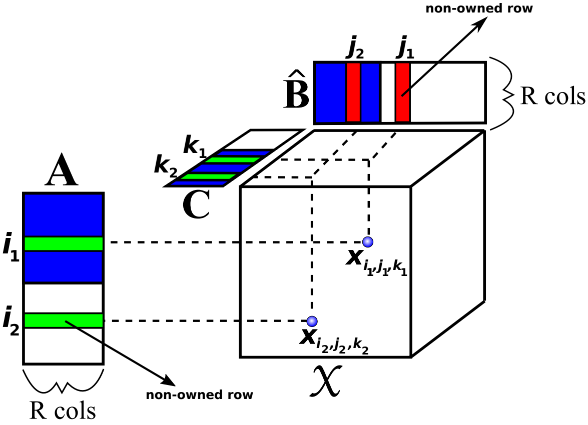
- $\mathbf{A}(i_2, :)$ needs to be received (to later compute $\hat{\mathbf{B}}$ and $\hat{\mathbf{C}}$).



Post-communication (expand)

Consider a process p (blue).

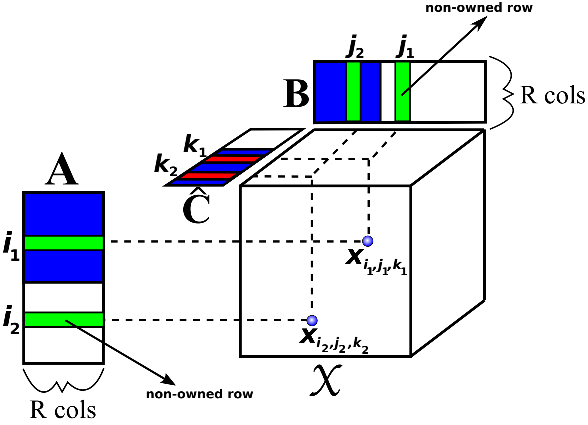
- $\mathbf{A}(i_2, :)$ needs to be received (to later compute $\hat{\mathbf{B}}$ and $\hat{\mathbf{C}}$).



Post-communication (expand)

Consider a process p (blue).

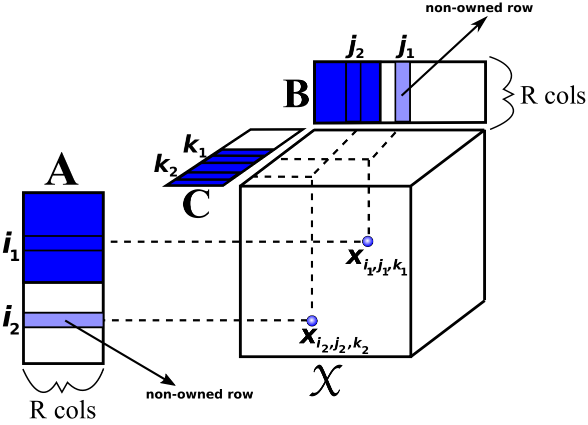
- $\mathbf{A}(i_2, :)$ needs to be received (to later compute $\hat{\mathbf{B}}$ and $\hat{\mathbf{C}}$).



Partitioning - Computation

Consider a process p (blue).

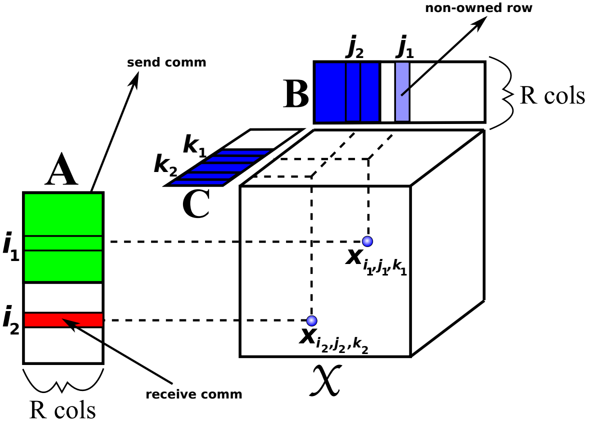
- Each nonzero incurs $(N-1)R$ ops in MTTKRP.
- Each matrix row incurs R^2 ops in GEMM and SYRK.
- **Goal:** Balance $|\mathcal{X}_p|$, I_p, J_p, K_p among all processes.



Partitioning - Communication

Consider a process p (blue).

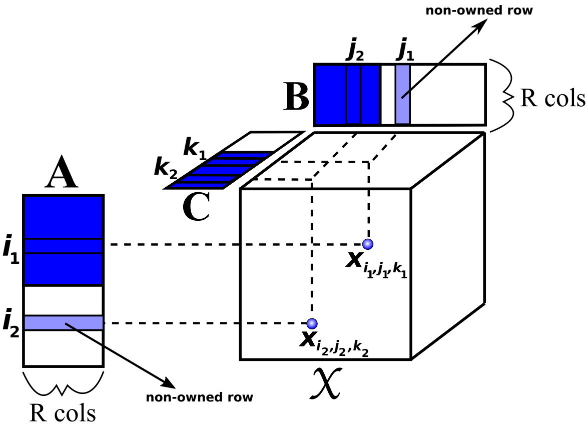
- Each non-owned row "touched" by an owned nonzero incurs a communication.
- Multiple "touches" do not increase the communication volume.



Partitioning - Memory

Consider a process p (blue).

- Stores $|\mathcal{X}_p|$ nonzeros.
- Stores I_p rows of **A**.
- Communicated rows are also stored!
- **Goal:** Balance $|\mathcal{X}_p|$, I_p, J_p, K_p, and **communication volume**!

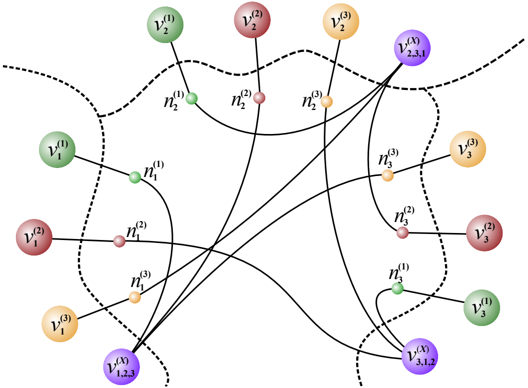


Hypergraph Partitioning - Fine-Grain Model

$$\mathcal{X} = \{(1,2,3),(2,3,1),(3,1,2)\} \in \mathbb{R}^{3\times3\times3}.$$

Fine-grain hypergraph involves:

- Unit vertex per nonzero
- Unit vertex per matrix row
- Hyperedge per matrix row, connected to matrix row's vertex and all nonzeros' that "touch" that row.
- **Goal:** Balance **each** vertex type, **minimize** the **cutsize**.

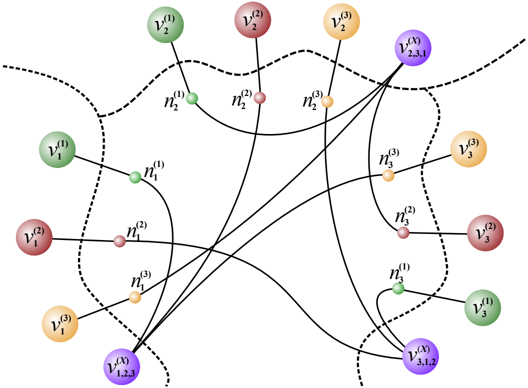


Hypergraph Partitioning - Fine-Grain Model

$$\boldsymbol{\mathcal{X}} = \{(1,2,3),(2,3,1),(3,1,2)\} \in \mathbb{R}^{3\times3\times3}.$$

Hypergraph partitioning is the "holy grail" of performance.
Balancing each vertex type balances

- MTTKRP load.
- sparse tensor storage.
- matrix storage.
- dense matrix operations.



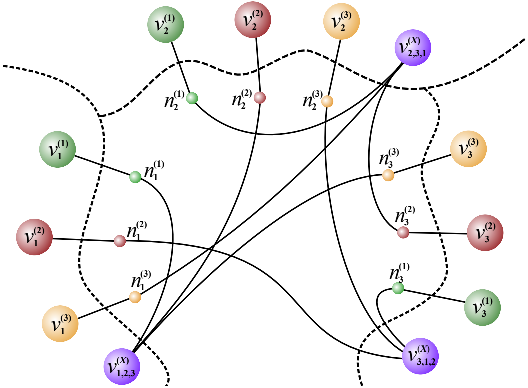
Hypergraph Partitioning - Fine-Grain Model

$$\mathcal{X} = \{(1,2,3),(2,3,1),(3,1,2)\} \in \mathbb{R}^{3\times3\times3}.$$

Hypergraph partitioning is the "holy grail" of performance.
Minimizing cutsize minimizes

- total fold and expand communication volume
- total non-local matrix row storage.

Balancing cutsize balances all these instead.

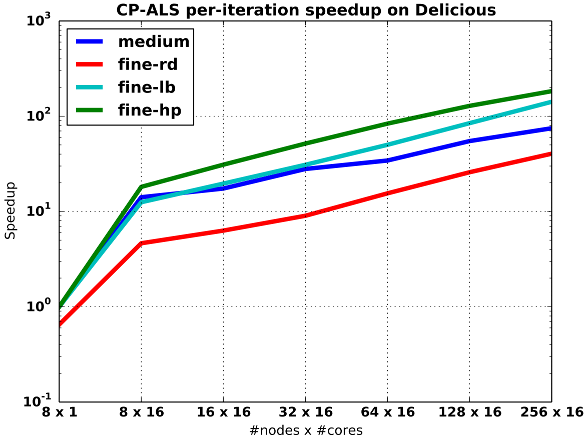


Tensors

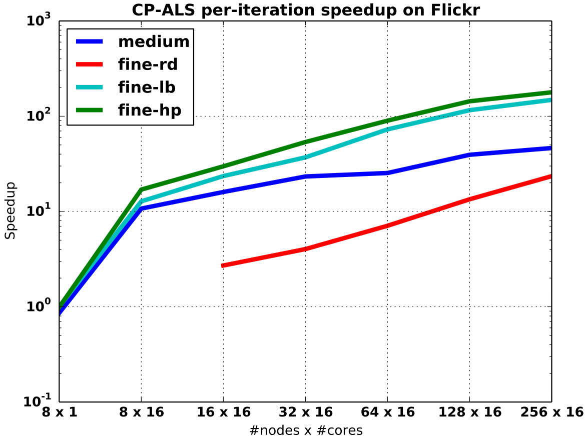
Real-world tensors used in the experiments.

Tensor	I_1	I_2	I_3	I_4	#nonzeros
Delicious	1.4K	532K	17M	2.4M	140M
Flickr	731	319K	28M	1.6M	112M
Netflix	480K	17K	2K	-	100M
NELL	3.2M	301	638K	-	78M
Amazon	6.6M	2.4M	23K	-	1.3B

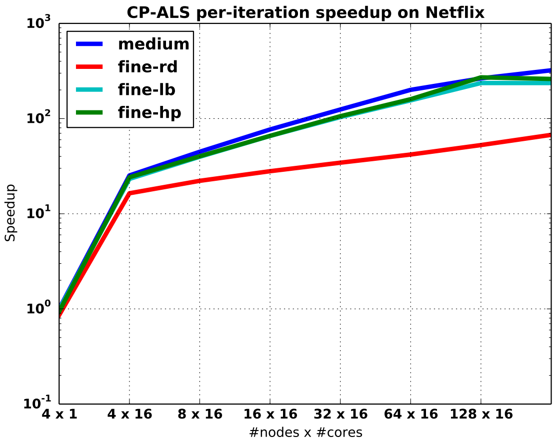
Results


CP-ALS per-iteration speedup on Delicious
medium
fine-rd
fine-lb
fine-hp
Speedup
#nodes x #cores
8 x 1
8 x 16
16 x 16
32 x 16
64 x 16
128 x 16
256 x 16
10^{-1}
10^{0}
10^{1}
10^{2}
10^{3}

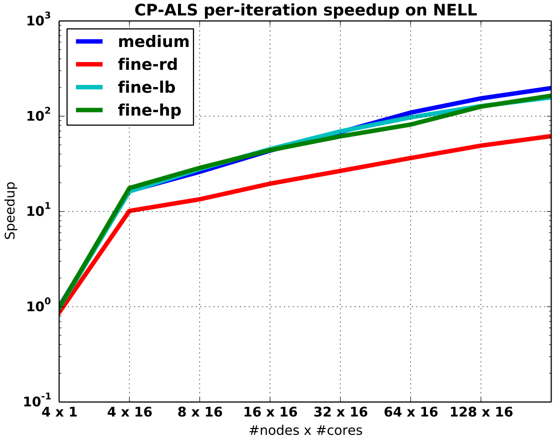
Results


CP-ALS per-iteration speedup on Flickr
medium
fine-rd
fine-lb
fine-hp
Speedup
10^{3}
10^{2}
10^{1}
10^{0}
10^{-1}
8 x 1
8 x 16
16 x 16
32 x 16
64 x 16
128 x 16
256 x 16
#nodes x #cores

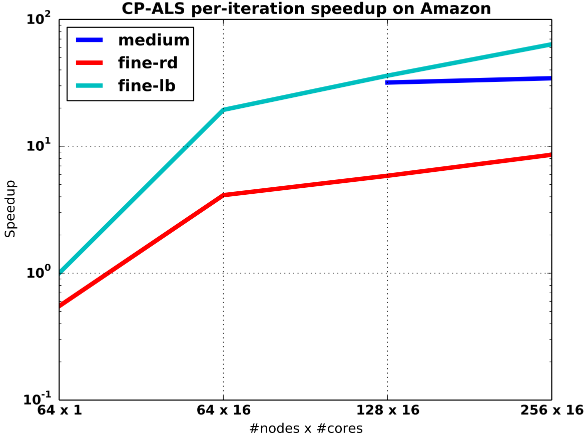
Results


CP-ALS per-iteration speedup on Netflix
medium
fine-rd
fine-lb
fine-hp
Speedup
10^3
10^2
10^1
10^0
10^-1
4 x 1
4 x 16
8 x 16
16 x 16
32 x 16
64 x 16
128 x 16
#nodes x #cores

Results


CP-ALS per-iteration speedup on NELL
medium
fine-rd
fine-lb
fine-hp
Speedup
10^{3}
10^{2}
10^{1}
10^{0}
10^{-1}
4 x 1
4 x 16
8 x 16
16 x 16
32 x 16
64 x 16
128 x 16
#nodes x #cores

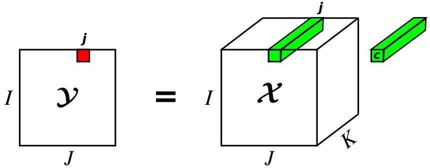
Results


CP-ALS per-iteration speedup on Amazon
medium
fine-rd
fine-lb
Speedup
10^{2}
10^{1}
10^{0}
10^{-1}
64 x 1
64 x 16
128 x 16
256 x 16
#nodes x #cores

Outline

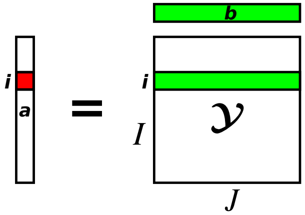
Tensor-Times-Vector Multiplication (TTV)

- Reduces dimensionality by one
- Performed in a particular dimension.
- $\mathcal{Y} = \mathcal{X} \times_3 \mathbf{c}$
- $\mathcal{Y}(i,j) = \mathbf{c}^T \mathcal{X}(i,j,:)$
  $= \sum_{k=1}^{K} \mathcal{X}(i,j,k)\mathbf{c}(k)$
- Sparsity of $\mathcal{Y}$ determined by sparsity of $\mathcal{X}$, i.e., $\text{nnz}(\mathcal{Y}) \leq \text{nnz}(\mathcal{X})$
- **Cost:** $\Theta(\text{nnz}(\mathcal{X}))$



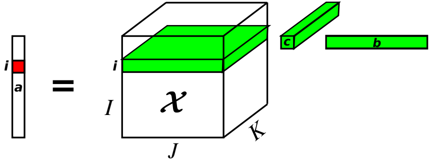
Tensor-Times-Vector Multiplication (TTV)

- $\mathbf{a} = \mathcal{Y} \times_2 \mathbf{b}$
- $\mathbf{a}(i) = \mathbf{b}^T \mathcal{Y}(i,:)$
  $= \sum_{j=1}^{J} \mathcal{Y}(i,j)\mathbf{b}(j)$
- TTV equivalent to matrix-vector multiplication
- **Cost:** $\Theta(\text{nnz}(\mathcal{Y})) = O(\text{nnz}(\mathcal{X}))$



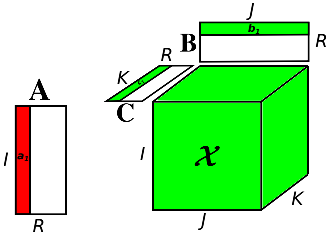
TTV in All-But-One Dimensions

- $\mathbf{a} = \boldsymbol{\mathcal{X}} \times_2 \mathbf{b} \times_3 \mathbf{c}$
- $\mathbf{a}(i) = \sum_{j=1}^{J} \sum_{k=1}^{K} \boldsymbol{\mathcal{X}}(i,j,k)\mathbf{b}(j)\mathbf{c}(k)$
- $N-1$ TTVs performed together.
- **Cost:** $\Theta(N\text{nnz}(\boldsymbol{\mathcal{X}}))$



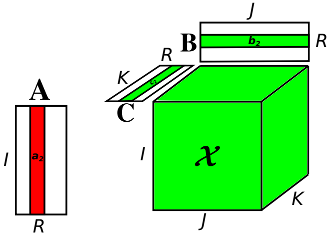
MTTKRP

- Column-wise TTV of $\mathcal{X}$ in all-but-one dimensions
- $\mathbf{a}_r \leftarrow \mathcal{X} \times_2 \mathbf{b}_r \times_3 \mathbf{c}_r$ for $r = 1, \dots, R$.
- Updating $\mathbf{a}_r$ takes $N-1$ TTVs.
- $RN(N-1)$ TTVs per iteration in total
- For simplicity, considering $R = 1$ henceforth (MTTKRP with vectors $\mathbf{a}, \mathbf{b}, \mathbf{c}$, etc.)



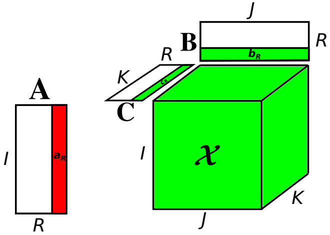
MTTKRP

- Column-wise TTV of $\mathcal{X}$ in all-but-one dimensions
- $\mathbf{a}_r \leftarrow \mathcal{X} \times_2 \mathbf{b}_r \times_3 \mathbf{c}_r$ for $r = 1, \ldots, R$.
- Updating $\mathbf{a}_r$ takes $N - 1$ TTVs.
- $RN(N - 1)$ TTVs per iteration in total
- For simplicity, considering $R = 1$ henceforth (MTTKRP with vectors $\mathbf{a}, \mathbf{b}, \mathbf{c}$, etc.)



MTTKRP

- Column-wise TTV of $\boldsymbol{\mathcal{X}}$ in all-but-one dimensions
- $\mathbf{a}_r \leftarrow \boldsymbol{\mathcal{X}} \times_2 \mathbf{b}_r \times_3 \mathbf{c}_r$ for $r = 1, \ldots, R$.
- Updating $\mathbf{a}_r$ takes $N - 1$ TTVs.
- $RN(N-1)$ TTVs per iteration in total
- For simplicity, considering $R = 1$ henceforth (MTTKRP with vectors $\mathbf{a}, \mathbf{b}, \mathbf{c}$, etc.)



Coordinate Storage (COOR)

$\mathbf{a} \leftarrow 0$
for $x_{i,j,k,l} \in \mathcal{X}$ **do**
$\quad \mathbf{a}(i) \mathrel{+}= x_{i,j,k,l}\mathbf{b}(j)\mathbf{c}(k)\mathbf{d}(l)$

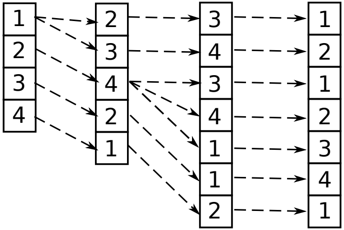
- **Storage cost:** $\Theta(N\text{nnz}(\mathcal{X}))$
- **MTTKRP cost:** $\Theta(N^2\text{nnz}(\mathcal{X}))$

1	2	3	1
1	3	4	2
2	4	3	1
2	4	4	2
2	4	1	3
3	2	1	4
4	1	2	1

$\mathcal{X}$

Compressed Sparse Fiber (CSF, Smith and Karypis, '15)

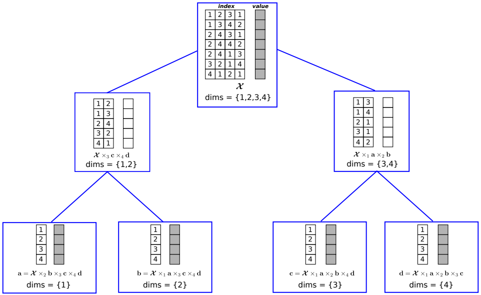
- Generalization of CSR/CSC
- Exploits index overlaps after TTVs
- Possible to use one representation across all dimensions
- Employed in SPLATT library
- **Storage cost:** $O(N\mathrm{nnz}(\boldsymbol{\mathcal{X}}))$
- **MTTKRP cost:** $O(N^2\mathrm{nnz}(\boldsymbol{\mathcal{X}}))$



$\mathcal{X}$

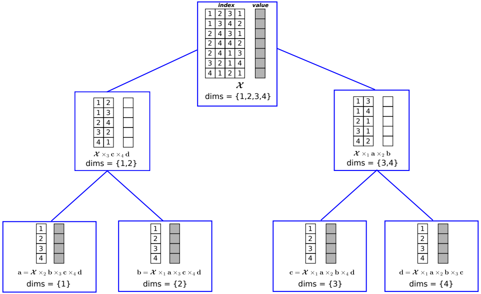
Dimension Tree (DT)

- Hierarchical storage, partitions dimensions at each level
- Single representation for all dimensions
- Each node corresponds to a set of TTVs
- Leaves correspond to factor matrices
- Index compression through leaves



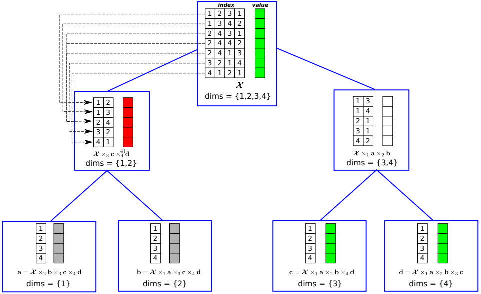
Dimension Tree (DT)

- Each node is computed using its parent.



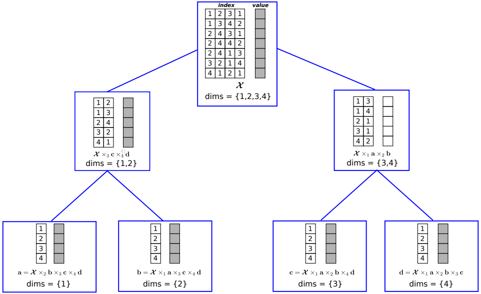
Dimension Tree (DT)

- Each node is computed using its parent.



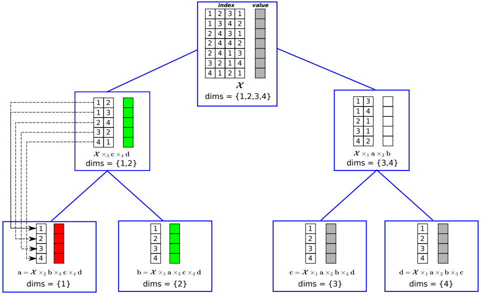
Dimension Tree (DT)

- Each node is computed using its parent.



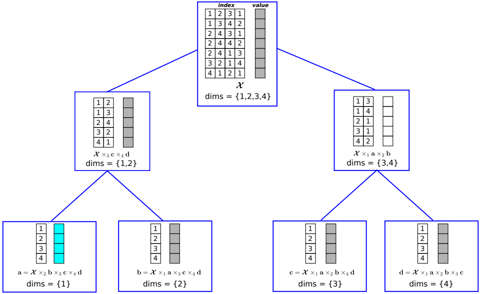
Dimension Tree (DT)

- Each node is computed using its parent.



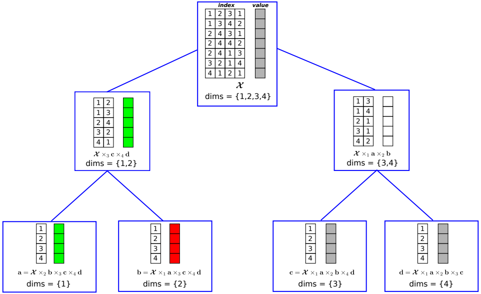
Dimension Tree (DT)

- Each node is computed using its parent.



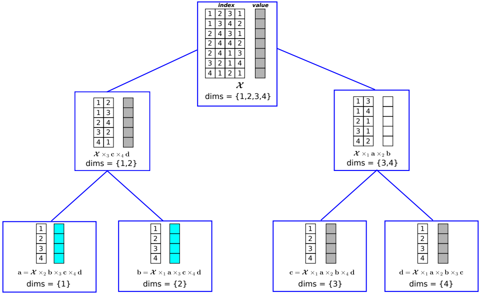
Dimension Tree (DT)

- Each node is computed using its parent.



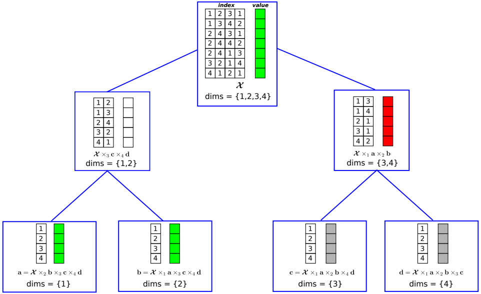
Dimension Tree (DT)

- Each node is computed using its parent.



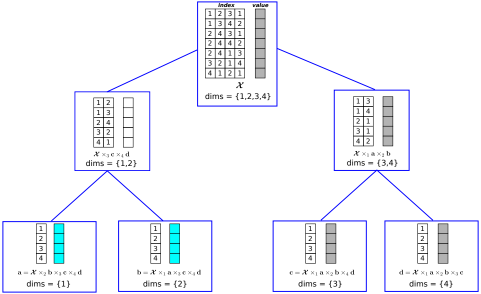
Dimension Tree (DT)

- Each node is computed using its parent.



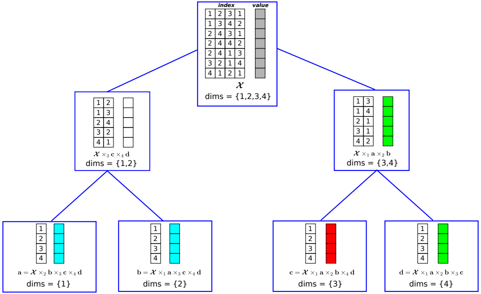
Dimension Tree (DT)

- Each node is computed using its parent.



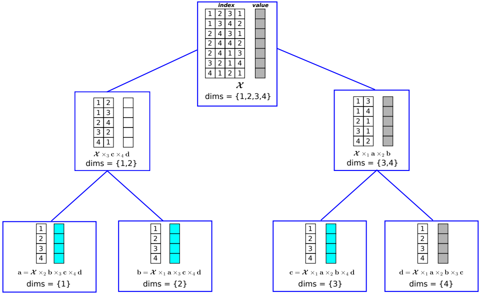
Dimension Tree (DT)

- Each node is computed using its parent.



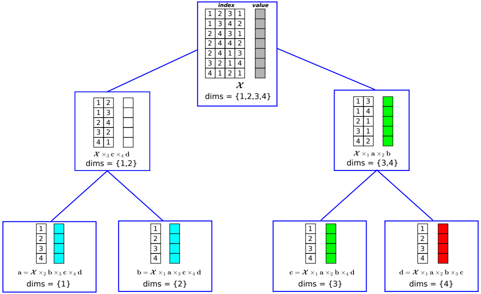
Dimension Tree (DT)

- Each node is computed using its parent.



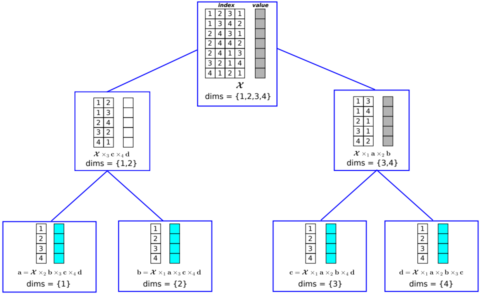
Dimension Tree (DT)

- Each node is computed using its parent.



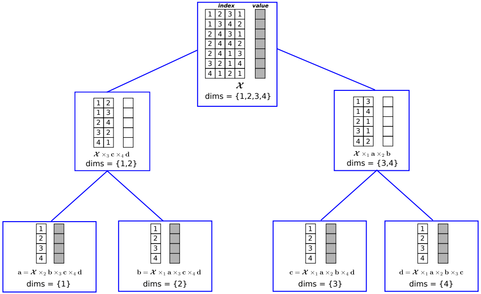
Dimension Tree (DT)

- Each node is computed using its parent.

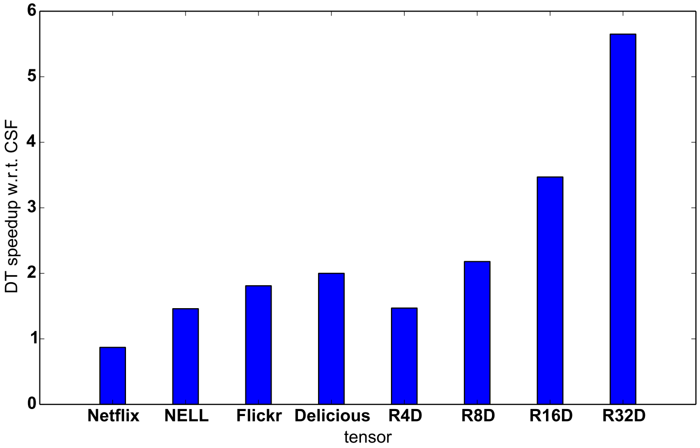


Dimension Tree (DT)

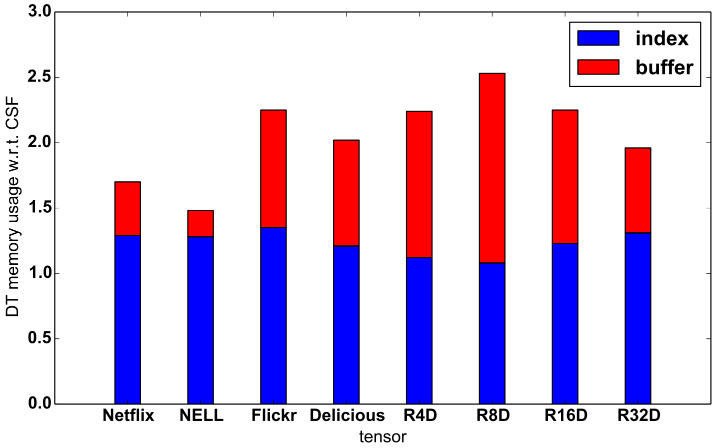
- Each node is computed using its parent.
- N index arrays per level
- **Storage cost (index):** $O(N \log N \mathrm{nnz}(\mathcal{X}))$ (vs. $O(N \mathrm{nnz}(\mathcal{X}))$ in CSF)
- With post-order traversal of leaves
  - N TTVs per level
  - $\log N$ value arrays allocated
- **Storage cost (value):** $O(\log N \mathrm{nnz}(\mathcal{X}))$
- **MTTKRP cost:** $O(N \log N \mathrm{nnz}(\mathcal{X}))$ (vs. $O(N^2 \mathrm{nnz}(\mathcal{X}))$ in CSF)
- $O(N/\log N)$ **faster** than CSF.



Experiments - Runtime ($R = 20$)



Experiments - Memory Usage ($R = 20$)



Outline

Conclusion

- Flexible fine-grained parallel algorithm to compute sparse tensor decompositions
- Hypergraph models of computation and communication
- A new tree data structure and computational scheme for sparse tensors
- $O(N/\log N)$ faster MTTKRP using $O(\log N)$-times more storage
- **5.65x speedup** on 32-D tensors using up to **2.5x more memory**
- Applicable to dense tensors, optimal algorithms in $O(3^D)$ time using $O(2^D)$ space
- All implemented in PACOS and HYPERTENSOR.

Contact

Oguz Kaya
Post-doctoral Researcher
HiePACS Team, INRIA Bordeaux, France
oguz.kaya@inria.fr
www.oguzkaya.com