Sparse Tensor Factorization: Algorithms, Data Structures, and Challenges

Shaden Smith & George Karypis

University of Minnesota
Department of Computer Science & Engineering
shaden@cs.umn.edu

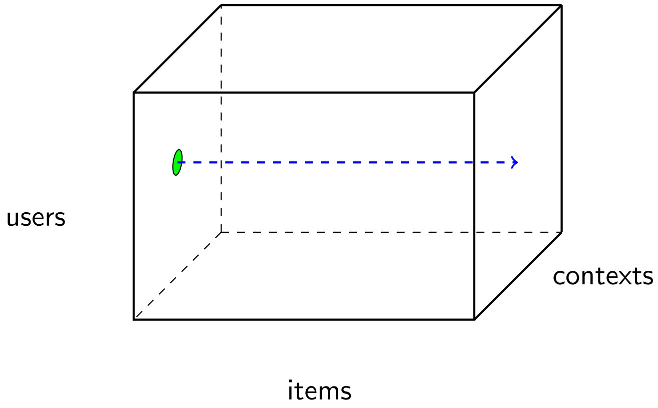
Talk Outline

1. Introduction
2. Compressed Sparse Fiber
3. Cache-Friendly Reordering & Tiling
4. Distributed-Memory MTTKRP
5. Conclusions

Table of Contents

Tensor Introduction

- Tensors are the generalization of matrices to $\geq 3D$
- Tensors have m dimensions (or *modes*) and are $I_1 \times \ldots \times I_m$
  - We'll usually stick to $I \times J \times K$ in this talk

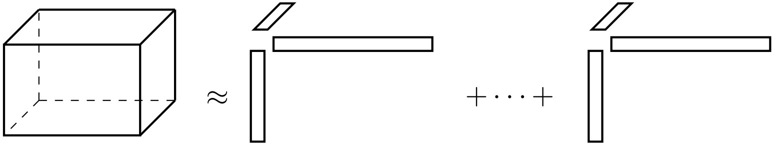


Applications

Dataset	I	J	K	nnz
NELL-2	12K	9K	28K	77M
Beer	33K	66K	960K	94M
Netflix	480K	18K	2K	100M
Delicious	532K	17M	3M	140M
NELL-1	3M	2M	25M	143M
Amazon	5M	18M	2M	1.7B

Canonical Polyadic Decomposition (CPD)

- We compute matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$, each with F columns
  - We will use $\mathbf{A}^{(1)}, \dots, \mathbf{A}^{(m)}$ when ≥ 3 modes



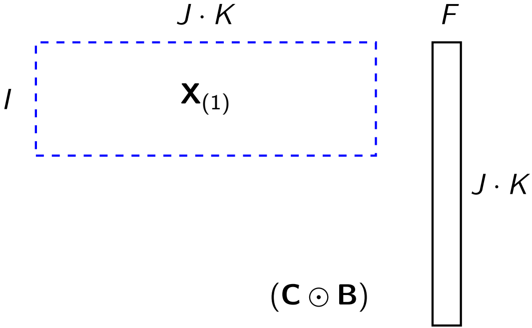
- Usually computed via *alternating least squares* (ALS)

Matricized Tensor Times Khatri-Rao Product

MTTKRP

- MTTKRP is the core computation of each iteration

$$\mathbf{A} = \mathbf{X}_{(1)} \left(\mathbf{C} \odot \mathbf{B} \right)$$



Alternating Least Squares

while not converged **do**
$$\mathbf{A}^\intercal = \left(\mathbf{C}^\intercal\mathbf{C} * \mathbf{B}^\intercal\mathbf{B}\right)^{-1}\left(\mathbf{X}_{(1)}(\mathbf{C} \odot \mathbf{B})\right)^\intercal$$
$$\mathbf{B}^\intercal = \left(\mathbf{C}^\intercal\mathbf{C} * \mathbf{A}^\intercal\mathbf{A}\right)^{-1}\left(\mathbf{X}_{(2)}(\mathbf{C} \odot \mathbf{A})\right)^\intercal$$
$$\mathbf{C}^\intercal = \left(\mathbf{B}^\intercal\mathbf{B} * \mathbf{A}^\intercal\mathbf{A}\right)^{-1}\left(\mathbf{X}_{(3)}(\mathbf{B} \odot \mathbf{A})\right)^\intercal$$
end while

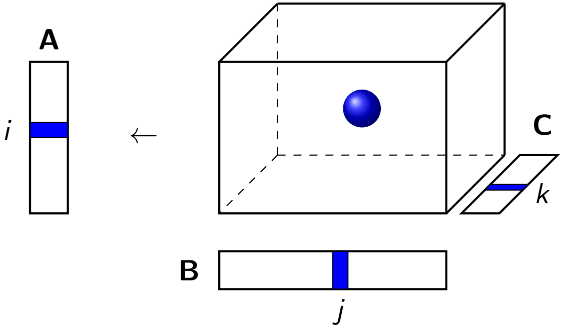
Tensor Storage – Coordinate Form

i	j	k	l	v
1	1	1	2	1.
1	1	1	3	1.
1	2	1	3	3.
1	2	2	1	8.
2	2	1	1	1.
2	2	1	3	3.
2	2	2	2	8.

Why don't we unfold?

- We need a representation of $\boldsymbol{\mathcal{X}}$ for each mode
- NELL has dimensions $3M \times 2M \times 25M$
  - Add a fourth mode and we exceed 2^{64}

MTTKRP



$$\mathbf{A}(i,:) \leftarrow \mathbf{A}(i,:) + \boldsymbol{\mathcal{X}}(i,j,k)\left[\mathbf{B}(j,:) * \mathbf{C}(k,:)\right]$$

Limitations

- Memory bandwidth
- Parallelism

Table of Contents

Can we do better?

- Consider three nonzeros in the fiber $\mathcal{X}(i, j, :)$ (a vector)

$$\mathbf{A}(i,:) \leftarrow \mathbf{A}(i,:) + \mathcal{X}(i,j,k_1) \; [\mathbf{B}(j,:) * \mathbf{C}(k_1,:)]$$
$$\mathbf{A}(i,:) \leftarrow \mathbf{A}(i,:) + \mathcal{X}(i,j,k_2) \; [\mathbf{B}(j,:) * \mathbf{C}(k_2,:)]$$
$$\mathbf{A}(i,:) \leftarrow \mathbf{A}(i,:) + \mathcal{X}(i,j,k_3) \; [\mathbf{B}(j,:) * \mathbf{C}(k_3,:)]$$

Can we do better?

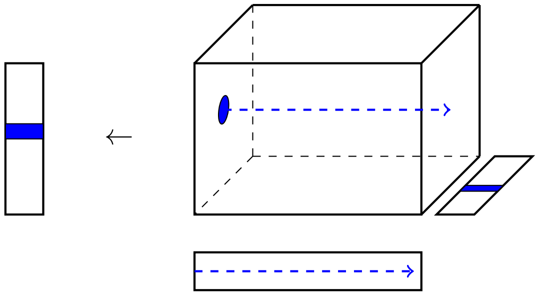
- Consider three nonzeros in the fiber $\mathcal{X}(i,j,:)$ (a vector)

$$\mathbf{A}(i,:) \leftarrow \mathbf{A}(i,:) + \mathcal{X}(i,j,k_1)\ [\mathbf{B}(j,:) * \mathbf{C}(k_1,:)]$$
$$\mathbf{A}(i,:) \leftarrow \mathbf{A}(i,:) + \mathcal{X}(i,j,k_2)\ [\mathbf{B}(j,:) * \mathbf{C}(k_2,:)]$$
$$\mathbf{A}(i,:) \leftarrow \mathbf{A}(i,:) + \mathcal{X}(i,j,k_3)\ [\mathbf{B}(j,:) * \mathbf{C}(k_3,:)]$$

- A little factoring...

$$\mathbf{A}(i,:) \leftarrow \mathbf{A}(i,:) + \ \mathbf{B}(j,:) * \left[\sum_{x=1}^{3} \mathcal{X}(i,j,k_x)\mathbf{C}(k_x,:) \right]$$

SPLATT: The **S**urprisingly **P**aralle**L** sp**A**rse **T**ensor **T**oolkit



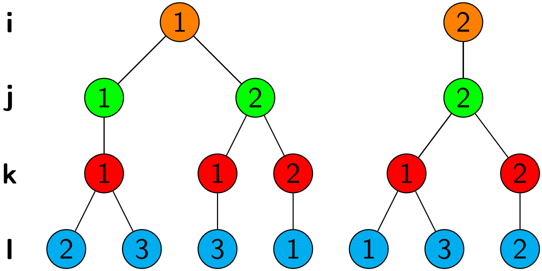
[Smith, Ravindran, Sidiropoulos, and Karypis 2015]

- Fibers are sparse vectors
- Slice $\boldsymbol{\mathcal{X}}(i,:,:)$ is almost a CSR matrix...
- But, we need m representations of $\boldsymbol{\mathcal{X}}$

Compressed Sparse Fiber (CSF)

i	j	k	l
1	1	1	2
1	1	1	3
1	2	1	3
1	2	2	1
2	2	1	1
2	2	1	3
2	2	2	2

→



[Smith and Karypis 2015]

- Modes are recursively compressed
- Values are stored in the leaves (not shown)

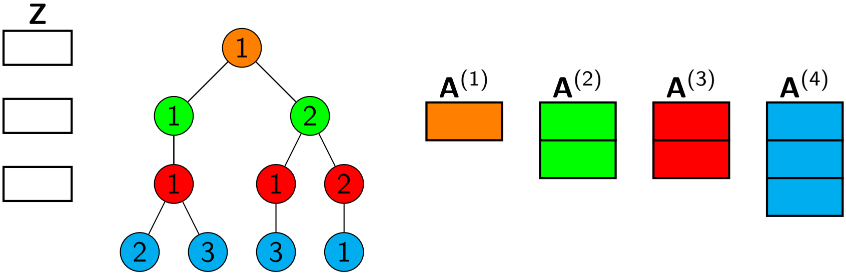
MTTKRP with a CSF Tensor

Objective

- We want to perform MTTKRP on each tensor mode with only one CSF representation
- There are three types of nodes in a tree: *root*, *internal*, and *leaf*
  - Each will have a tailored algorithm
  - *root* and *leaf* are special cases of *internal*

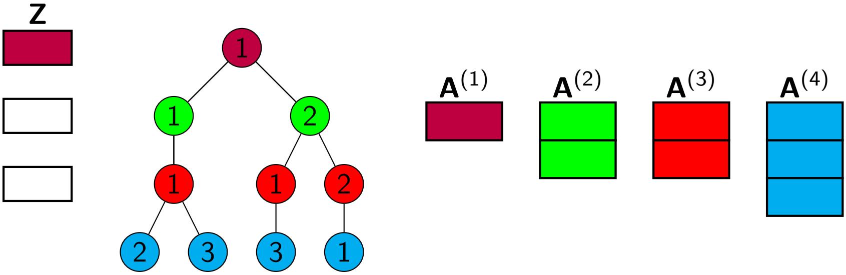
CSF-LEAF

- The leaf nodes determine the output location



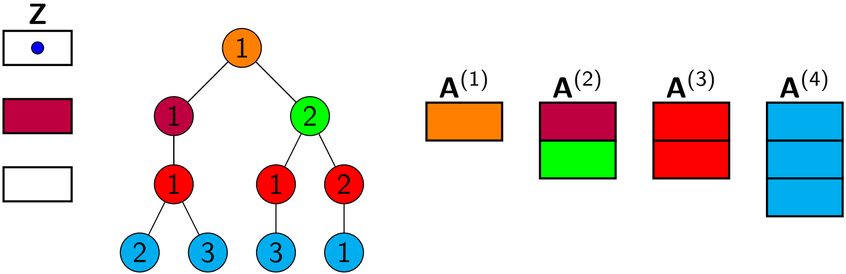
CSF-LEAF

- Hadamard products are pushed down the tree



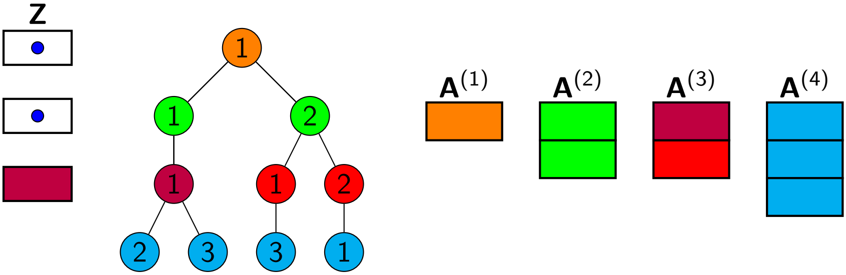
CSF-LEAF

- Hadamard products are pushed down the tree



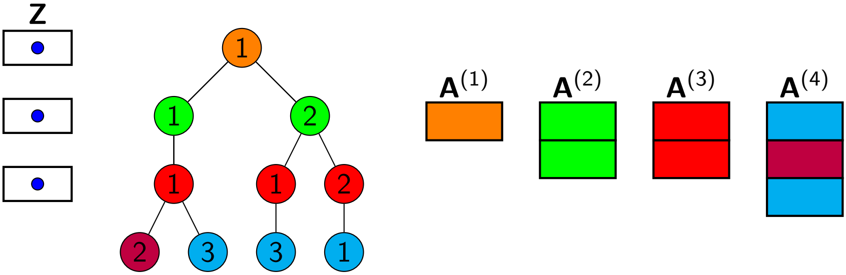
CSF-LEAF

- Hadamard products are pushed down the tree



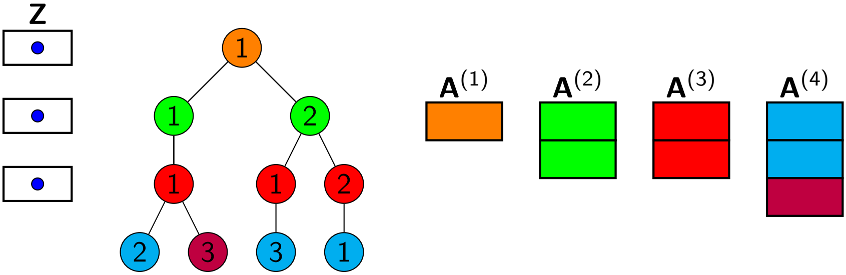
CSF-LEAF

- Leaves designate write locations



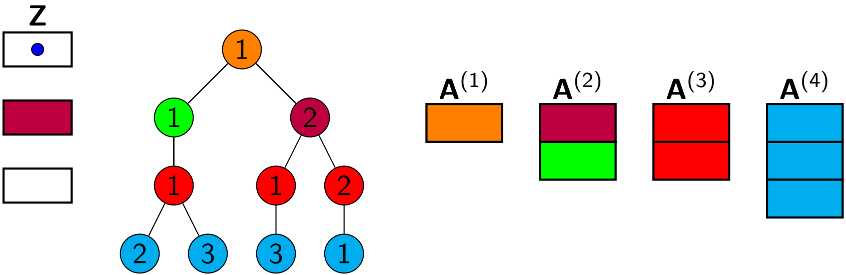
CSF-LEAF

- Leaves designate write locations



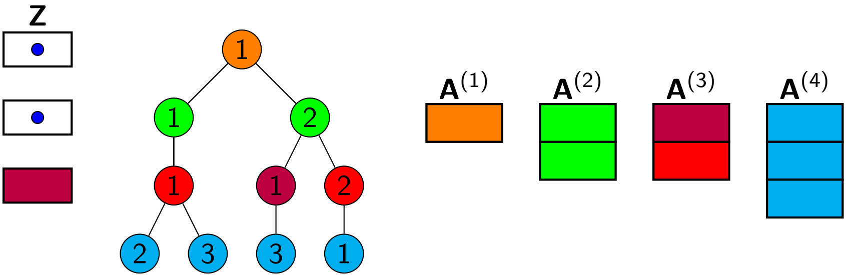
CSF-LEAF

- The traversal continues...



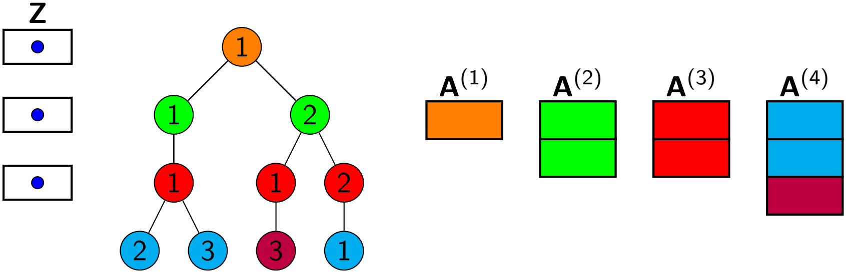
CSF-LEAF

- The traversal continues...



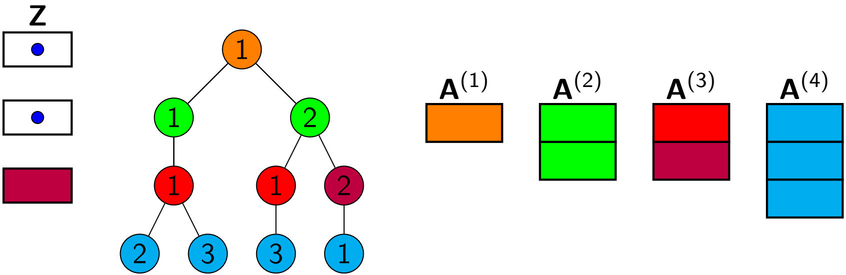
CSF-LEAF

- The traversal continues...



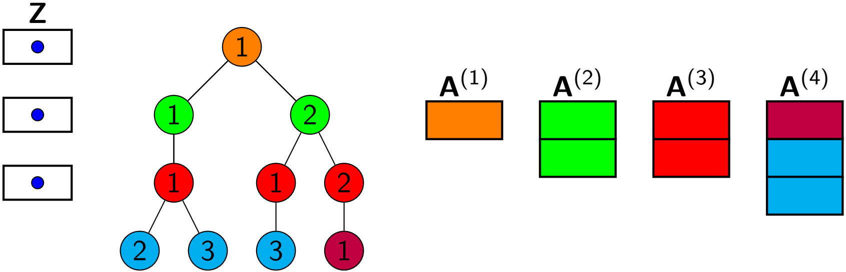
CSF-LEAF

- The traversal continues...



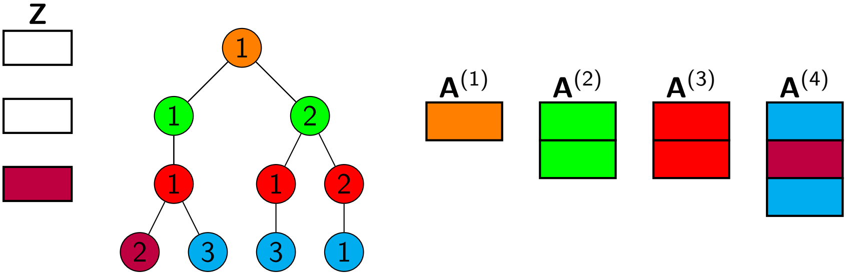
CSF-LEAF

- The traversal continues...



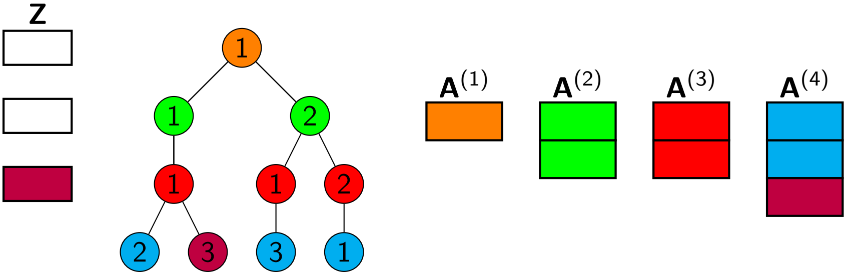
CSF-ROOT

- Inner products are accumulated in a buffer



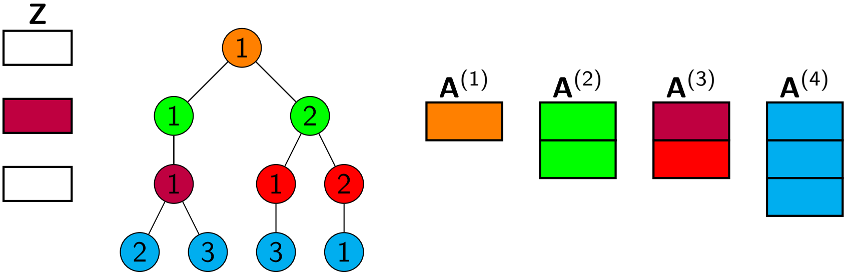
CSF-ROOT

- Inner products are accumulated in a buffer



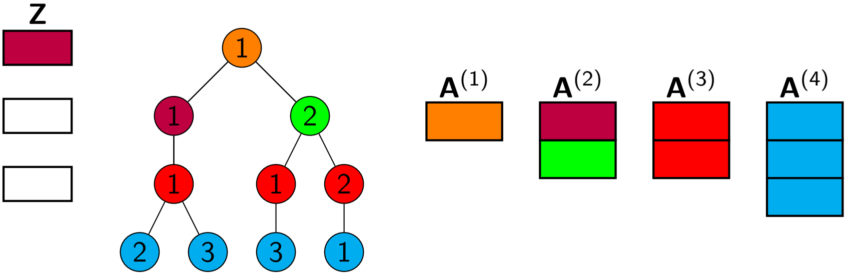
CSF-ROOT

- Hadamard products are then propagated up the CSF tree



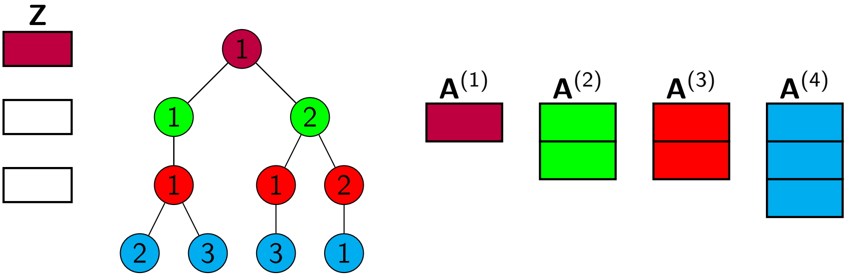
CSF-ROOT

- Hadamard products are then propagated up the CSF tree



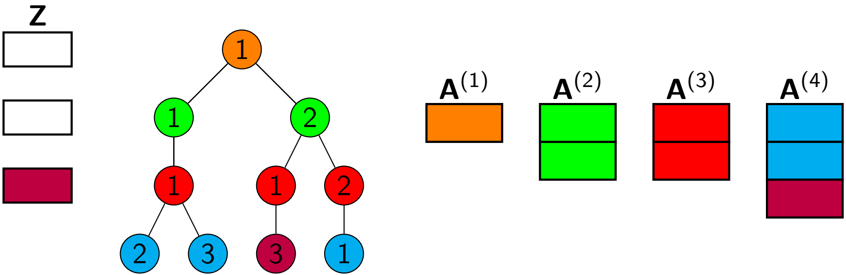
CSF-ROOT

- Results are written to $\mathbf{A}^{(1)}$



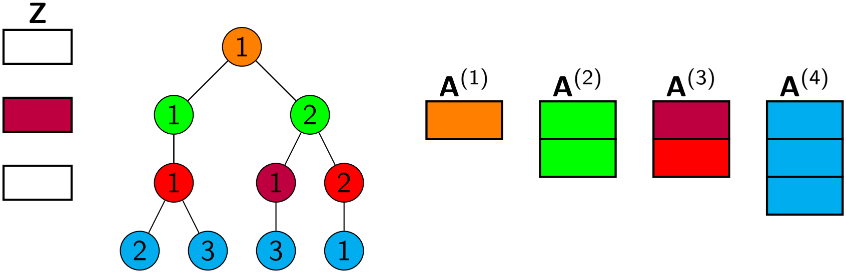
CSF-ROOT

- The traversal continues...



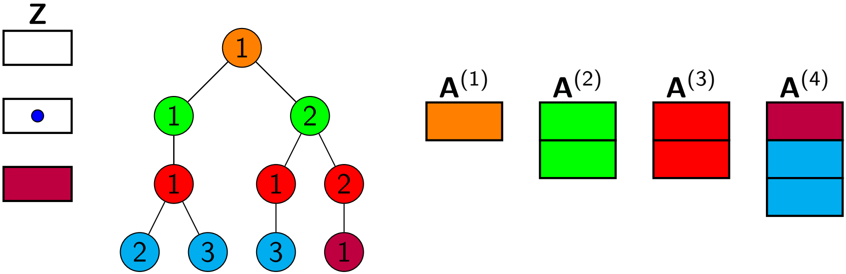
CSF-ROOT

- The traversal continues...



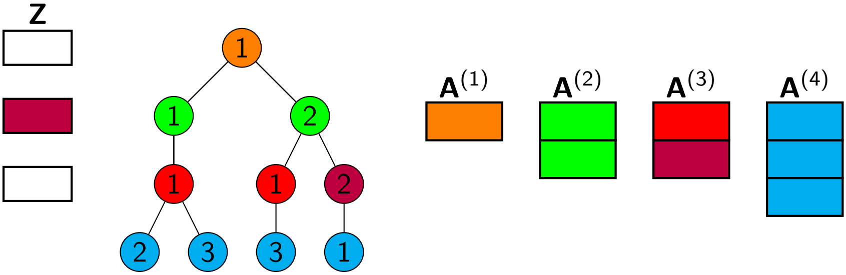
CSF-ROOT

- Partial results are kept in buffer



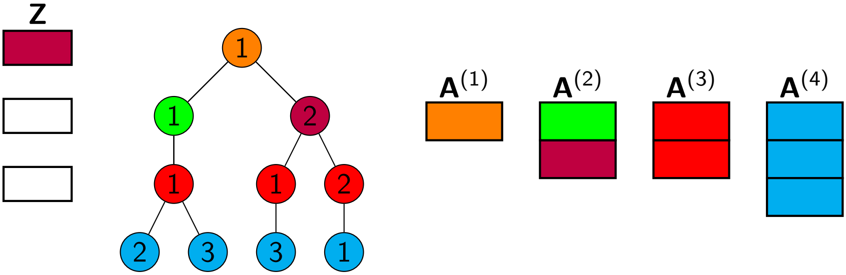
CSF-ROOT

- Inner products are accumulated in a buffer



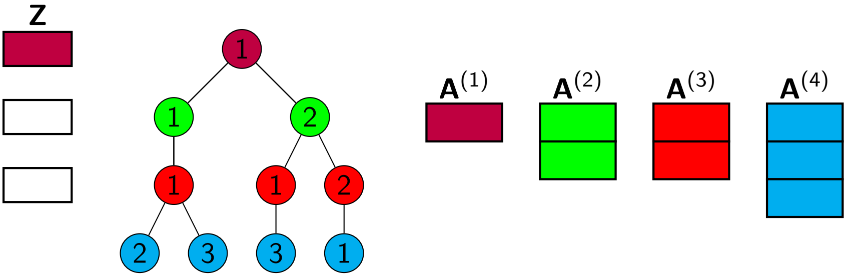
CSF-ROOT

- Inner products are accumulated in a buffer



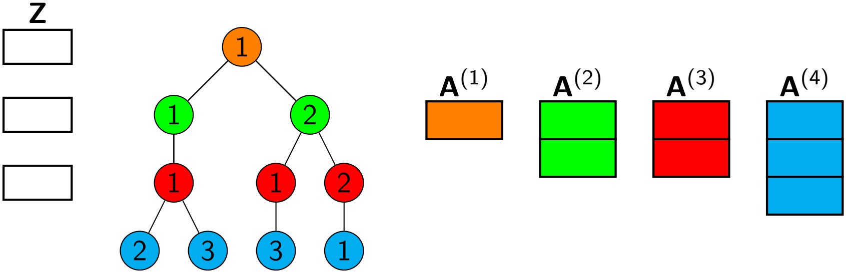
CSF-ROOT

- Results are written to $\mathbf{A}^{(1)}$



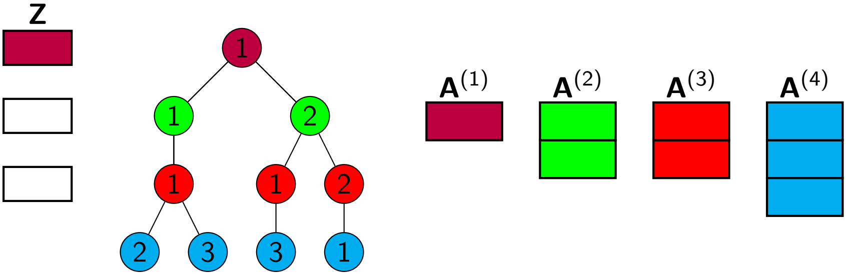
CSF-INTERNAL

- Internal nodes use a combination of CSF-ROOT and CSF-LEAF



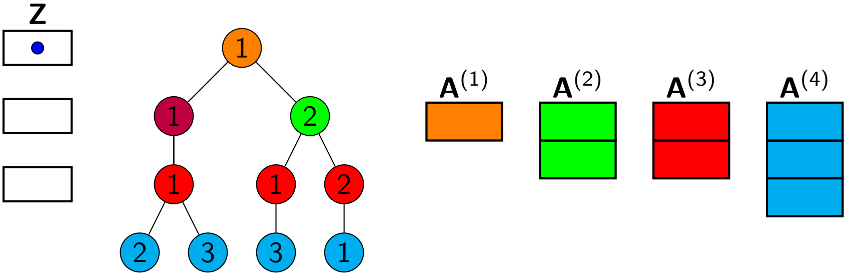
CSF-INTERNAL

- Hadamard products are pushed down to the output level



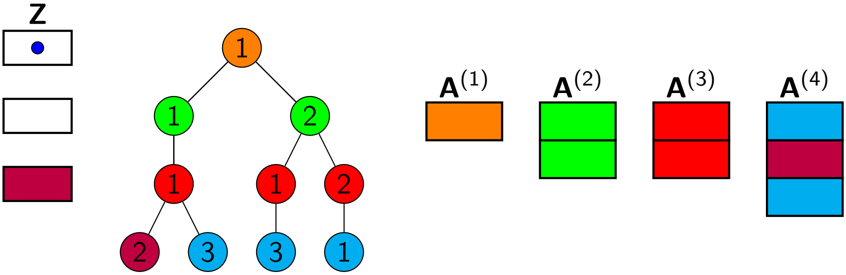
CSF-INTERNAL

- CSF-ROOT next pulls up to the output level



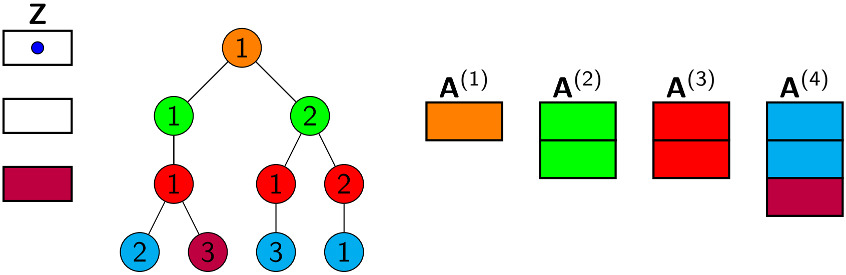
CSF-INTERNAL

- CSF-ROOT next pulls up to the output level



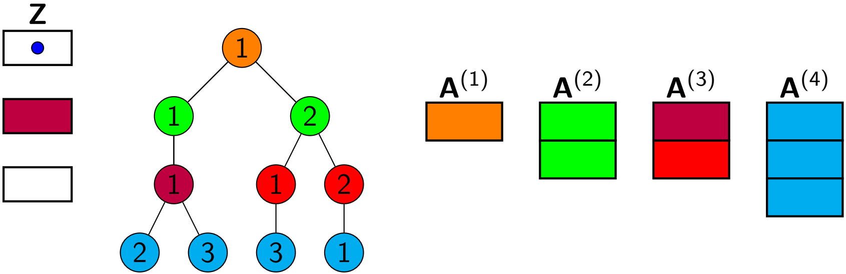
CSF-INTERNAL

- CSF-ROOT next pulls up to the output level



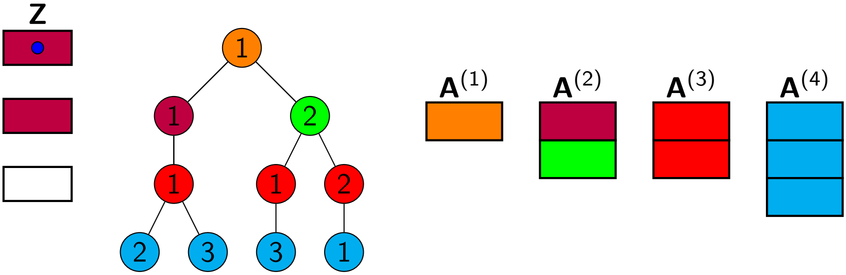
CSF-INTERNAL

- CSF-ROOT next pulls up to the output level



CSF-INTERNAL

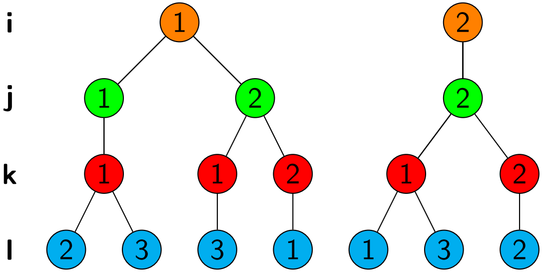
- CSF-ROOT next pulls up to the output level



Parallelism – Challenges?

i	j	k	l
1	1	1	2
1	1	1	3
1	2	1	3
1	2	2	1
2	2	1	1
2	2	1	3
2	2	2	2

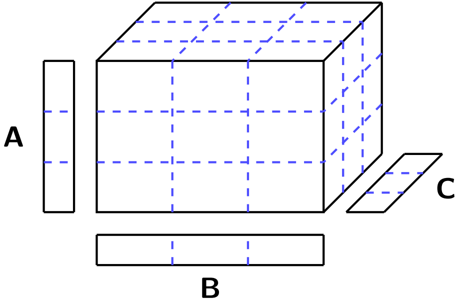
$\rightarrow$



- CSF-ROOT can be parallelized over the trees
- CSF-INTERNAL and CSF-LEAF require more thought...

Parallelism – Tiling

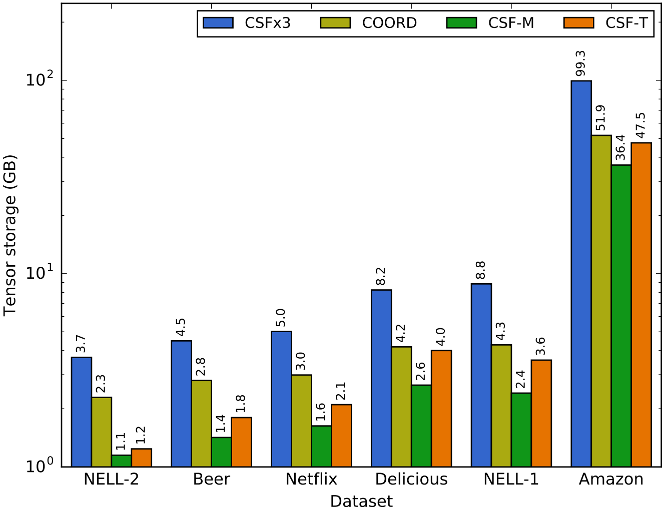
- For p threads, we do a p-way tiling of each tensor mode
- Distributing the tiles allows us to eleminate the need for mutexes



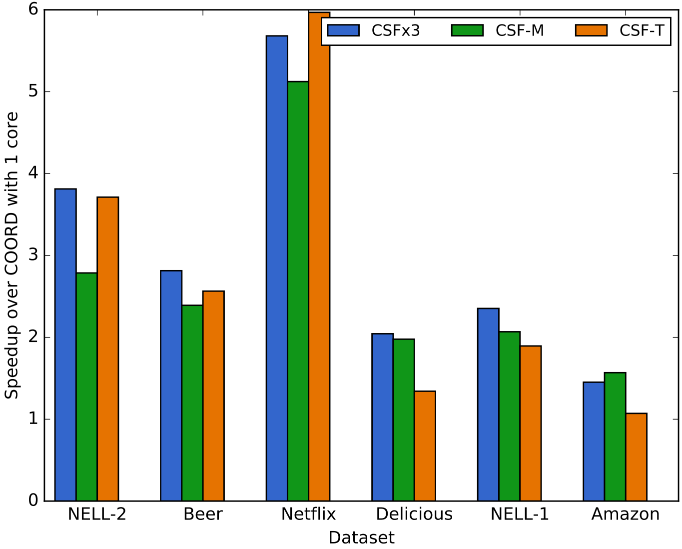
Datasets

Dataset	I	J	K	nnz
NELL-2	12K	9K	28K	77M
Beer	33K	66K	960K	94M
Netflix	480K	18K	2K	100M
Delicious	532K	17M	3M	140M
NELL-1	3M	2M	25M	143M
Amazon	5M	18M	2M	1.7B

Storage Comparison



Serial MTTKRP



Parallel MTTKRP

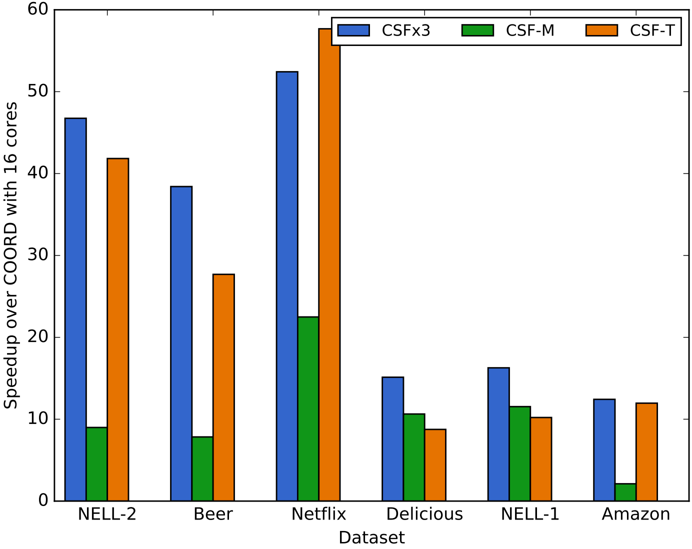


Table of Contents

Tensor Reordering

$$\left[\begin{array}{cccc|cccc|cccc}
 & 3 & & 3 & & & & & 2 & & & 2 \\
 & & & & 1 & & 1 & & & & & \\
 & & & & 1 & & 1 & & 2 & & & 2 \\
 & 3 & & 3 & & & & & & & &
\end{array}\right]$$

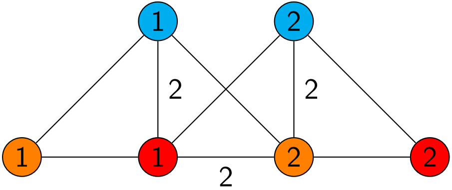
$$\left[\begin{array}{cccc|cccc|cccc}
3 & 3 & & & & & & & & & & \\
3 & 3 & & & & 2 & 2 & & & & & \\
 & & & & & 2 & 2 & & & & 1 & 1 \\
 & & & & & & & & & & 1 & 1
\end{array}\right]$$

We *reorder* the tensor to improve the access patterns on the factors

Tensor Reordering

i	j	k
1	1	1
1	2	1
2	2	1
2	2	2

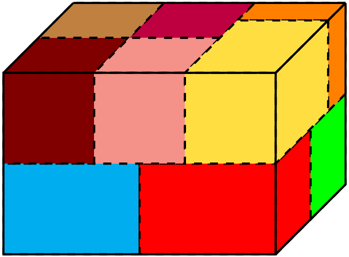
→



Graph Partitioning

- We model the sparsity structure of $\boldsymbol{\mathcal{X}}$ with a tripartite graph
  - Slices are vertices, nonzeros connect slices with a triangle
- Partitioning the graph finds regions with shared indices
- We reorder the tensor to group indices in the same partition

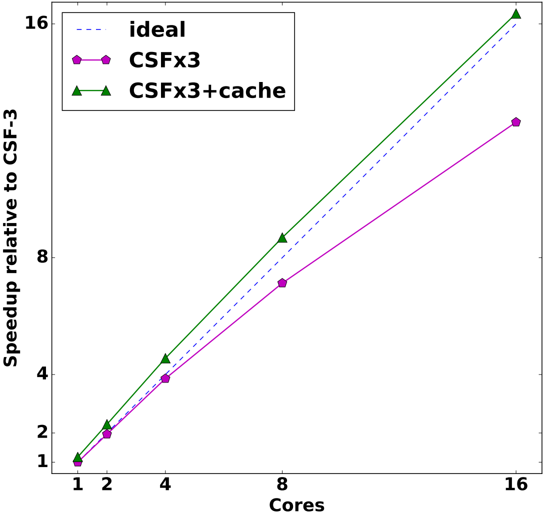
Cache Blocking over Tensors



Sparsity is Hard

- Tiling lets us schedule nonzeros to reuse indices already in cache
- Cost: more fibers
- Tensor sparsity forces us to *grow* tiles

Scaling: NELL-2, Speedup vs Untiled



Scaling: Netflix, Speedup vs Untiled

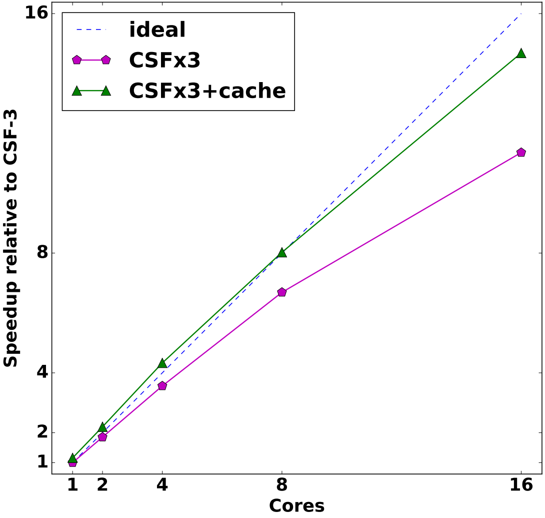
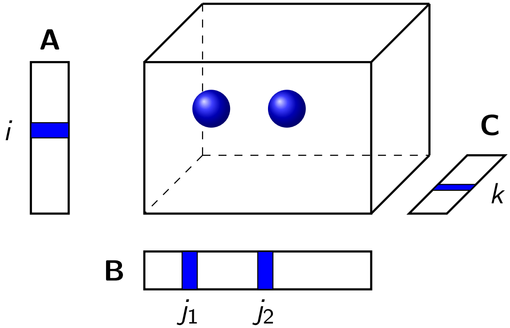


Table of Contents

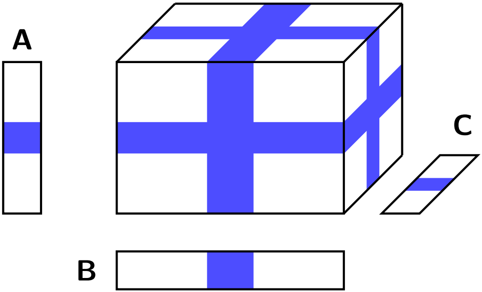
MTTKRP Communication



$$\mathbf{A}(i,:) \leftarrow \mathbf{A}(i,:) + \boldsymbol{\mathcal{X}}(i,j_1,k)\,[\mathbf{B}(j_1,:) * \mathbf{C}(k,:)]$$

$$\mathbf{A}(i,:) \leftarrow \mathbf{A}(i,:) + \boldsymbol{\mathcal{X}}(i,j_2,k)\,[\mathbf{B}(j_2,:) * \mathbf{C}(k,:)]$$

Coarse-Grained Decomposition



[Choi & Vishwanathan 2014, Shin & Kang 2014]

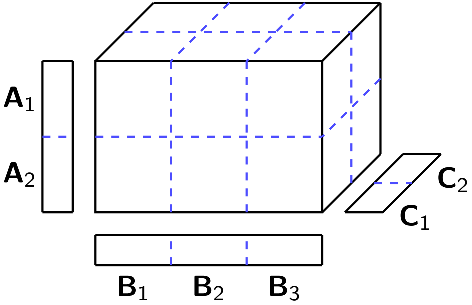
- Processes own complete slices of $\mathcal{X}$ and aligned factor rows
- I/p rows communicated to $p-1$ processes after each update

Fine-Grained Decomposition

[Kaya & Uçar 2015]

- Most flexible: non-zeros individually assigned to processes
- Two communication steps
  1. Aggregate partial computations after MTTKRP
  2. Exchange new factor values
- Hypergraph partitioning is used to minimize communication
  - Non-zeros mapped to vertices
  - $I+J+K$ hyperedges

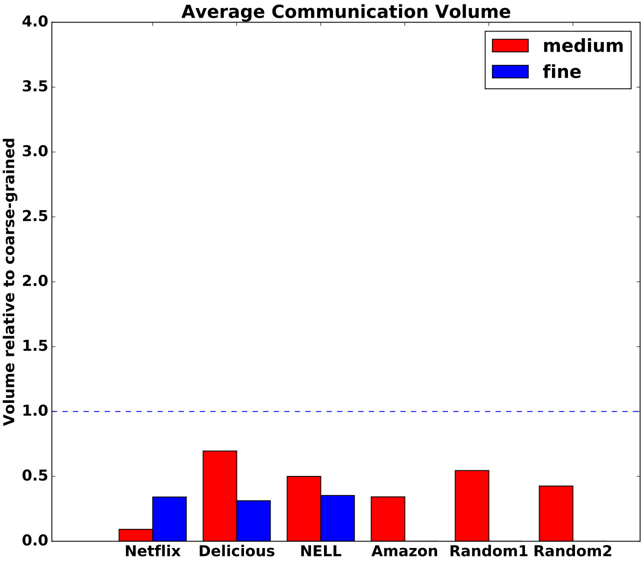
Medium-Grained Decomposition



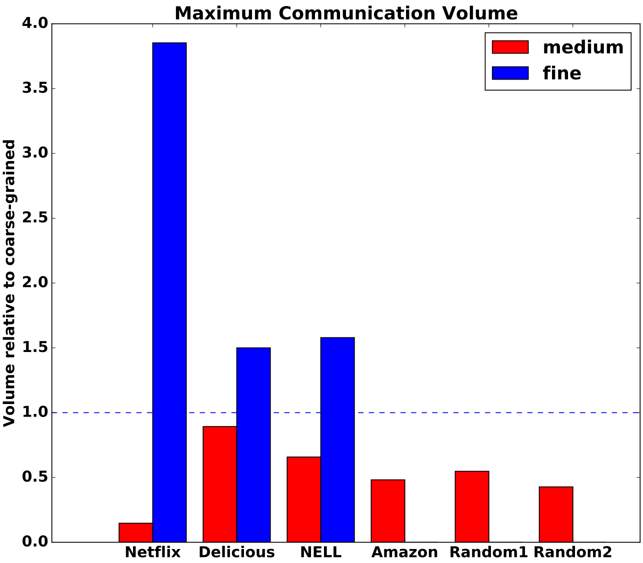
[Smith & Karypis 2016]

- Distribute over a grid of $p = q \times r \times s$ partitions
- $r \times s$ processes divide each $\mathbf{A}_1, \ldots, \mathbf{A}_q$
- Two communication steps like fine-grained
  - $\mathcal{O}(I/p)$ rows communicated to $r \times s$ processes

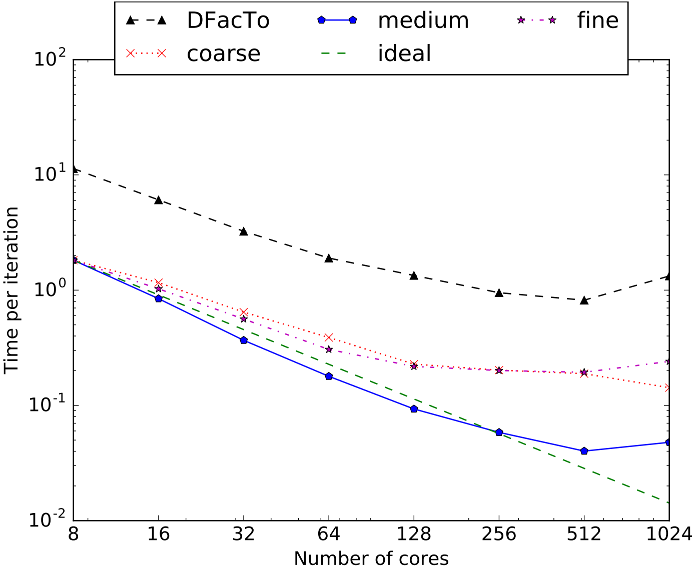
Average Communication Volume



Maximum Communication Volume



Strong Scaling: Netflix



Strong Scaling: Amazon

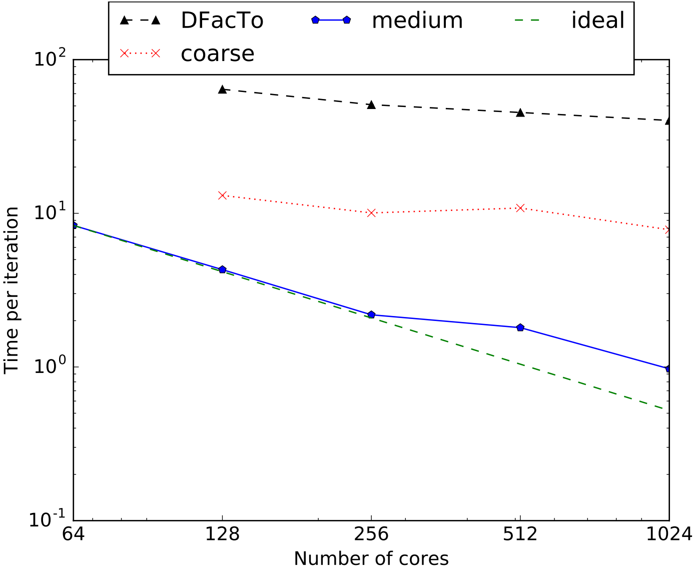


Table of Contents

Wrapping Up

- SPLATT is $40\times$ to $80\times$ faster than competing distributed-memory codes
- $50\times$ to $300\times$ faster single-node performance than Matlab
  - $> 1000\times$ faster with a supercomputer!
- New applications possible
  1. Healthcare
  2. Recommender systems
  3. Yours!

Future Work

Still many open problems!

- Manycore architectures
- Coupled factorization
- What's beyond ALS?

Where does Intel fit in?

- Intel is in a unique position to make significant contributions
- Kernels have unstructured access patterns and are memory-bound
  - Mostly :-)
- High-bandwidth memory and hardware-transactional memory are exciting technologies for tensor folk

`http://cs.umn.edu/~splatt/`

Questions?

Backup Slides

Convergence Check

```
1: while not converged do
2:     ...
3: end while
```

- Checking for convergence is not trivial!

$$||\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{Z}}||_F^2 = \underbrace{\langle \boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{X}} \rangle}_{\text{constant}} + \underbrace{\langle \boldsymbol{\mathcal{Z}}, \boldsymbol{\mathcal{Z}} \rangle}_{||\boldsymbol{\mathcal{Z}}||_F^2} - 2 \underbrace{\langle \boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Z}} \rangle}_{?}$$

Convergence Check – Tensor Norm

$$||\mathcal{Z}||_F^2 = \lambda^T \left(\mathbf{A}^T\mathbf{A} * \mathbf{B}^T\mathbf{B} * \mathbf{C}^T\mathbf{C} \right) \lambda$$

- The cost is negligible **if** we have cached $\mathbf{A}^T\mathbf{A}$, etc.
  - $O(F^2)$ vs $O(IF^2)$ flops

Convergence Check – Inner Product

$$\langle \boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Z}} \rangle = \sum_{f=1}^{F} \lambda(f) \left(\sum_{\text{nnz}(\boldsymbol{\mathcal{X}})} \boldsymbol{\mathcal{X}}(i,j,k)\mathbf{A}(i,f)\mathbf{B}(j,f)\mathbf{C}(k,f) \right)$$

- Cost: $O(F\,\text{nnz}(\boldsymbol{\mathcal{X}}))$, with a higher constant than MTTKRP

Convergence Check – Inner Product

$$\langle \boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Z}} \rangle = \sum_{f=1}^{F} \lambda(f) \underbrace{\left(\sum_{\text{nnz}(\boldsymbol{\mathcal{X}})} \boldsymbol{\mathcal{X}}(i,j,k)\mathbf{A}(i,f)\mathbf{B}(j,f)\mathbf{C}(k,f) \right)}_{\text{does this look familiar?}}$$

Convergence Check – Inner Product

Smith and Karypis 2016

- Keep the MTTKRP result from the last mode, $\hat{\mathbf{C}}$
  - $\hat{\mathbf{C}}$ has the latest $\mathbf{A}$ and $\mathbf{B}$ values

$$\hat{\mathbf{C}}(k,:) = \sum_{\text{nnz}(\boldsymbol{\mathcal{X}}(:,:,k))} \boldsymbol{\mathcal{X}}(i,j,k)\left[\mathbf{A}(i,:) * \mathbf{B}(j,:)\right]$$

Convergence Check – Inner Product

Smith and Karypis 2016

- Keep the MTTKRP result from the last mode, $\hat{\mathbf{C}}$
  - $\hat{\mathbf{C}}$ has the latest **A** and **B** values

$$\hat{\mathbf{C}}(k,:) = \sum_{\text{nnz}(\boldsymbol{\mathcal{X}}(:,:,k))} \boldsymbol{\mathcal{X}}(i,j,k)\left[\mathbf{A}(i,:) * \mathbf{B}(j,:)\right]$$

- Now we just need to account for λ and the new **C** values

$$\langle \boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Z}} \rangle = \mathbf{1}^T \left(\mathbf{C} * \hat{\mathbf{C}}\right) \lambda$$

Convergence Check – Inner Product

Smith and Karypis 2016

- Keep the MTTKRP result from the last mode, $\hat{\mathbf{C}}$
  - $\hat{\mathbf{C}}$ has the latest $\mathbf{A}$ and $\mathbf{B}$ values

$$\hat{\mathbf{C}}(k,:) = \sum_{\text{nnz}(\boldsymbol{\mathcal{X}}(:,:,k))} \boldsymbol{\mathcal{X}}(i,j,k) \left[\mathbf{A}(i,:) * \mathbf{B}(j,:)\right]$$

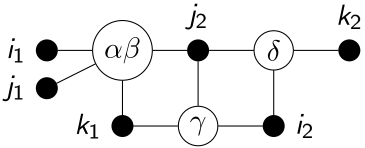
- Now we just need to account for λ and the new $\mathbf{C}$ values

$$\langle \boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Z}} \rangle = \mathbf{1}^T \left(\mathbf{C} * \hat{\mathbf{C}}\right) \lambda$$

- Cost: $O(IF)$, **much** cheaper than $O(\text{nnz}(\boldsymbol{\mathcal{X}})F)$

Tensor Reordering – Mode Dependent

$$\left[\begin{array}{cc|cc} \alpha & \beta & 0 & 0 \\ 0 & \gamma & 0 & \delta \end{array}\right]$$



Hypergraph Partitioning

- Instead, create a new reordering for each mode of computation
- Fibers are now vertices and slices are hyperedges
- Overheads?

Choosing the Shape of the Decomposition

Objective

- We need to find q, r, s such that $q \times r \times s = p$
- Tensors modes are often very skewed (480k Netflix users vs 2k days)
  - We want to assign processes proportionally
  - 1D decompositions actually work well for many tensors

Algorithm

1. Start with a $1 \times 1 \times 1$ shape
2. Compute the prime factorization of p
3. For each prime factor f, starting from the largest, multiply the most imbalanced mode by f