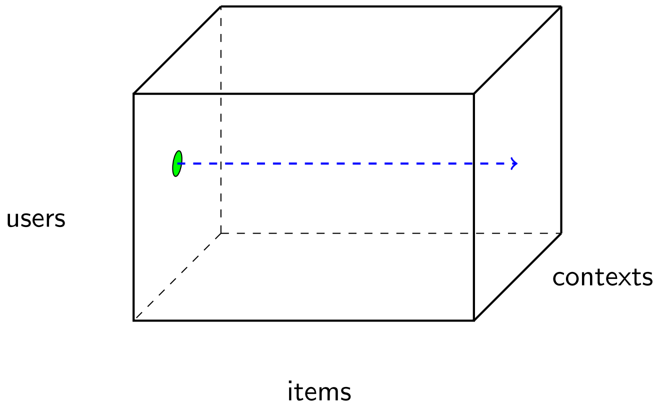
Efficient Factorization with Compressed Sparse Tensors

Shaden Smith George Karypis

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

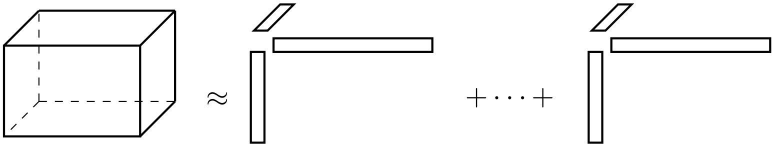
Tensor Introduction

- Tensors are the generalization of matrices to $\geq 3D$
- Tensors have m dimensions (or *modes*) and are $I_1 \times \ldots \times I_m$.



Canonical Polyadic Decomposition (CPD)

- We compute matrices $\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(m)}$, each with F columns

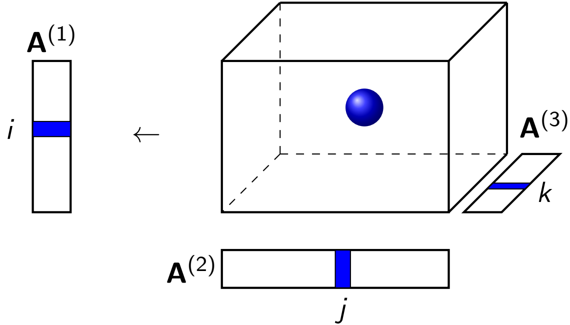


- Usually computed via *alternating least squares* (ALS)
- Matricized Tensor Times Khatri-Rao Product (MTTKRP) is the core computation of each iteration

Uncompressed Tensors

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.

Uncompressed Tensors – MTTKRP



$$\mathbf{A}^{(1)}(i,:) \leftarrow \mathbf{A}^{(1)}(i,:) + \boldsymbol{\mathcal{X}}(i,j,k)\left[\mathbf{A}^{(2)}(j,:) * \mathbf{A}^{(3)}(k,:)\right]$$

Can we do better?

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

$$\mathbf{A}^{(1)}(i,:) \leftarrow \mathbf{A}^{(1)}(i,:) + \boldsymbol{\mathcal{X}}(i,j,k_1)\left[\mathbf{A}^{(2)}(j,:) * \mathbf{A}^{(3)}(k_1,:)\right]$$

$$\mathbf{A}^{(1)}(i,:) \leftarrow \mathbf{A}^{(1)}(i,:) + \boldsymbol{\mathcal{X}}(i,j,k_2)\left[\mathbf{A}^{(2)}(j,:) * \mathbf{A}^{(3)}(k_2,:)\right]$$

$$\mathbf{A}^{(1)}(i,:) \leftarrow \mathbf{A}^{(1)}(i,:) + \boldsymbol{\mathcal{X}}(i,j,k_3)\left[\mathbf{A}^{(2)}(j,:) * \mathbf{A}^{(3)}(k_3,:)\right]$$

Can we do better?

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

$$\mathbf{A}^{(1)}(i,:) \leftarrow \mathbf{A}^{(1)}(i,:) + \boldsymbol{\mathcal{X}}(i,j,k_1) \left[\mathbf{A}^{(2)}(j,:) * \mathbf{A}^{(3)}(k_1,:)\right]$$

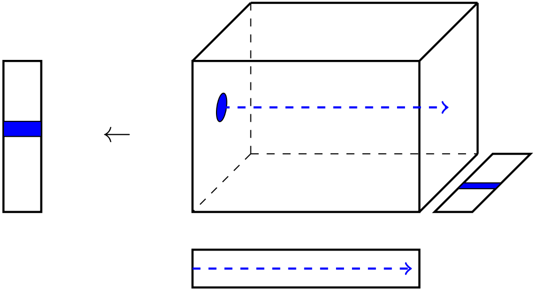
$$\mathbf{A}^{(1)}(i,:) \leftarrow \mathbf{A}^{(1)}(i,:) + \boldsymbol{\mathcal{X}}(i,j,k_2) \left[\mathbf{A}^{(2)}(j,:) * \mathbf{A}^{(3)}(k_2,:)\right]$$

$$\mathbf{A}^{(1)}(i,:) \leftarrow \mathbf{A}^{(1)}(i,:) + \boldsymbol{\mathcal{X}}(i,j,k_3) \left[\mathbf{A}^{(2)}(j,:) * \mathbf{A}^{(3)}(k_3,:)\right]$$

- A little factoring...

$$\mathbf{A}^{(1)}(i,:) \leftarrow \mathbf{A}^{(1)}(i,:) + \mathbf{A}^{(2)}(j,:) * \left[\sum_{x=1}^{3} \boldsymbol{\mathcal{X}}(i,j,k_x) \mathbf{A}^{(3)}(k_x,:)\right]$$

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



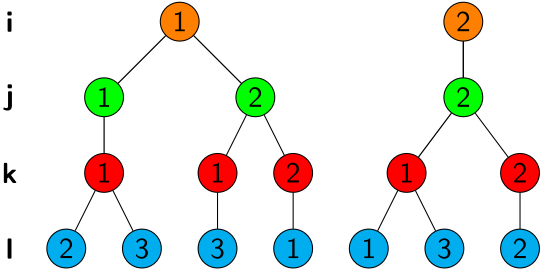
Data Structure

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

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

$\rightarrow$



MTTKRP with a CSF Tensor

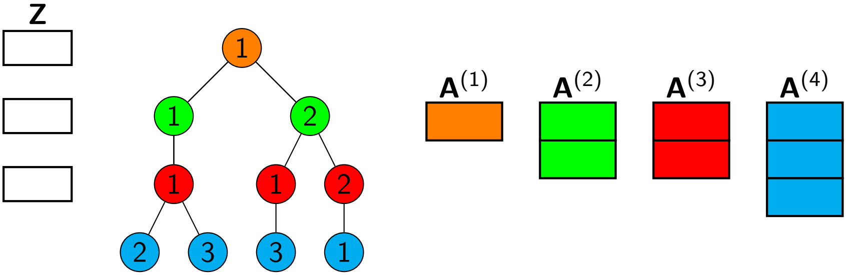
Objective

- 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

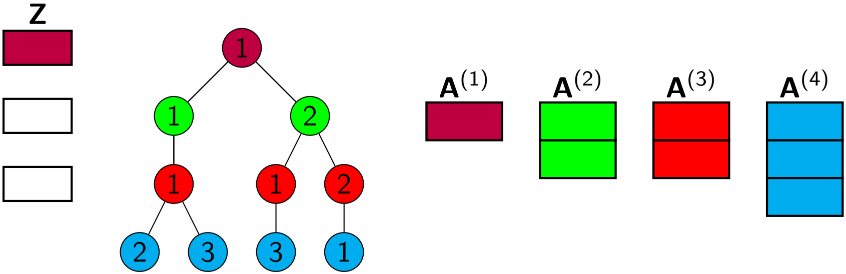
$$\mathbf{A}^{(4)}(l,:) \leftarrow \mathbf{A}^{(4)}(l,:) + \boldsymbol{\mathcal{X}}(i,j,k,l)\left[\ \mathbf{A}^{(1)}(i,:) * \mathbf{A}^{(2)}(j,:) * \mathbf{A}^{(3)}(k,:)\right]$$

- The leaf nodes determine the output location



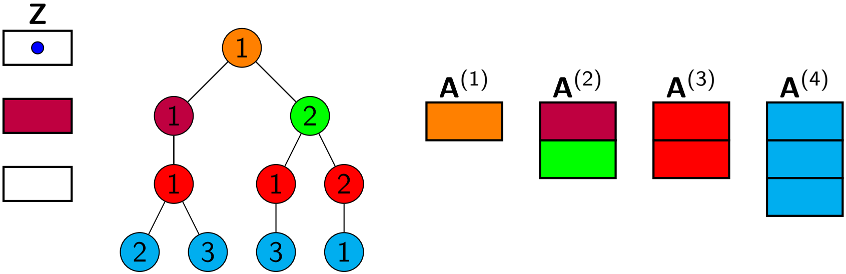
CSF-LEAF

- Hadamard products are pushed down the tree



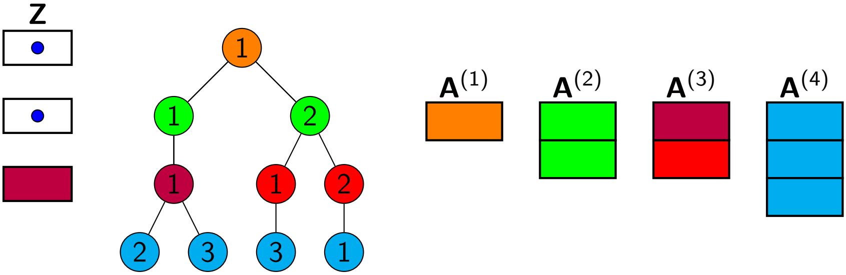
CSF-LEAF

- Hadamard products are pushed down the tree



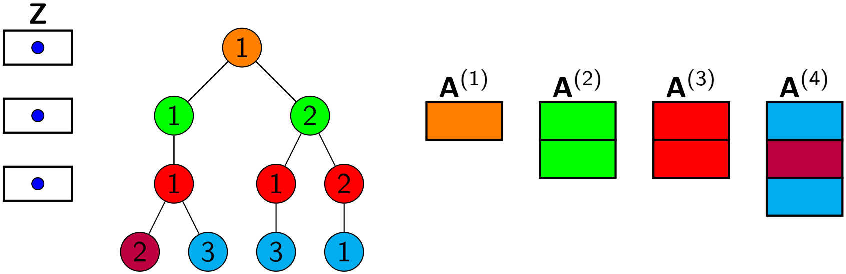
CSF-LEAF

- Hadamard products are pushed down the tree



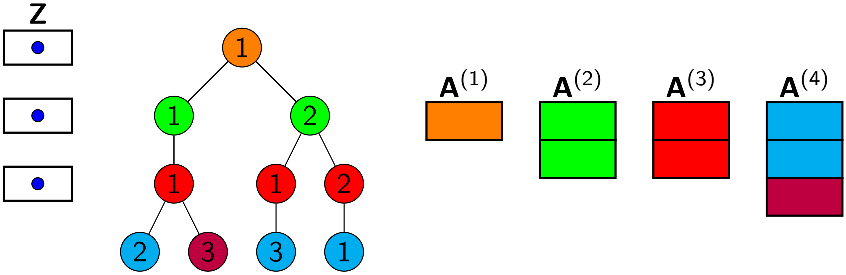
CSF-LEAF

- Leaves designate write locations



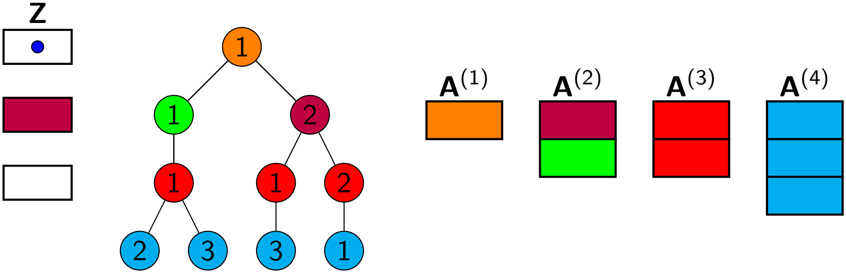
CSF-LEAF

- Leaves designate write locations



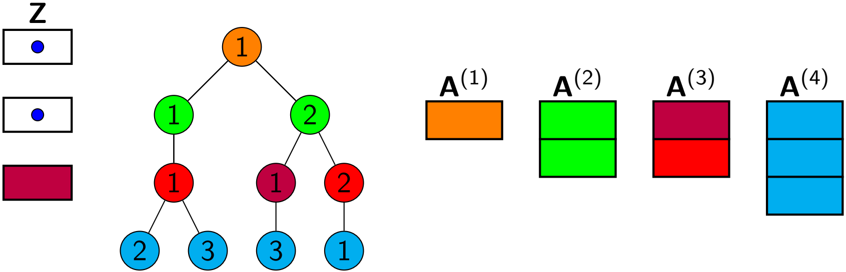
CSF-LEAF

- The traversal continues...



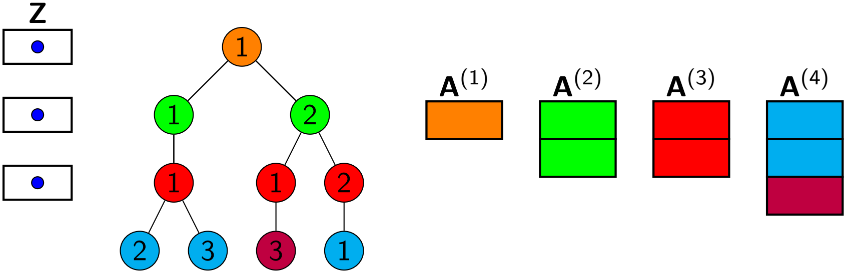
CSF-LEAF

- The traversal continues...



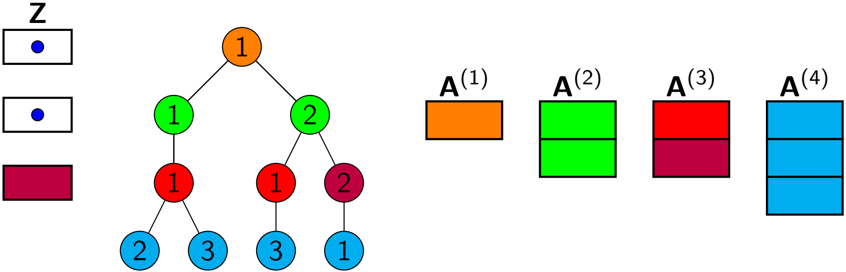
CSF-LEAF

- The traversal continues...



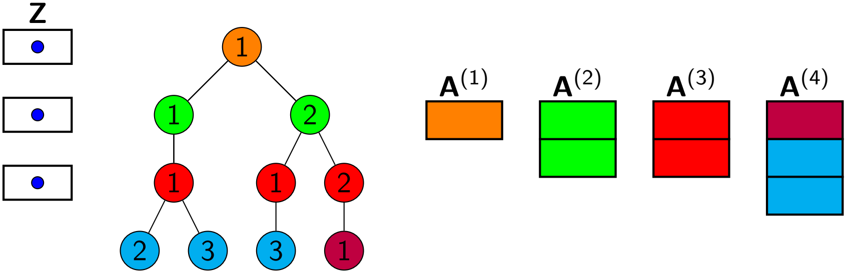
CSF-LEAF

- The traversal continues...



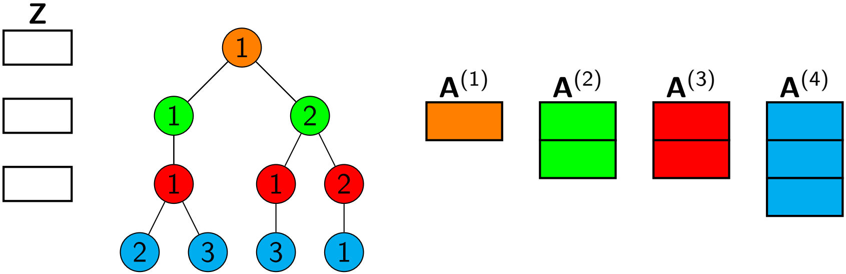
CSF-LEAF

- The traversal continues...



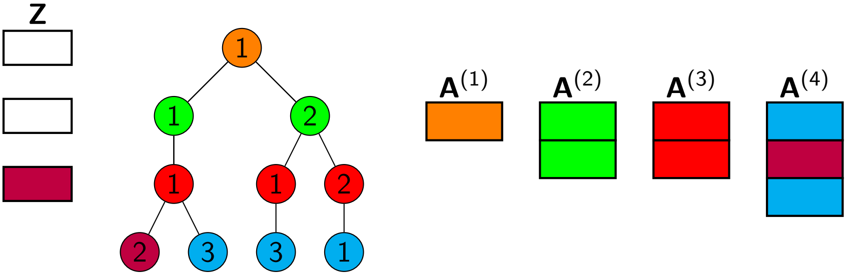
CSF-ROOT

$$\sum_{\mathcal{X}(i,j,:,:)} \mathbf{A}^{(2)}(j,:) * \left[\sum_{\mathcal{X}(i,j,k,:)} \mathbf{A}^{(3)}(k,:) * \left(\sum_{\mathcal{X}(i,j,k,:)} \mathcal{X}(i,j,k,l)\mathbf{A}^{(4)}(l,:) \right) \right]$$



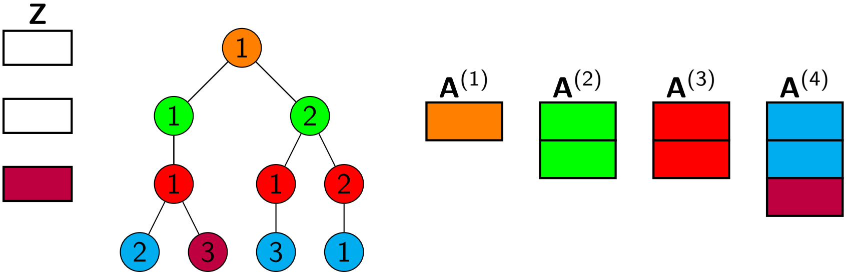
CSF-ROOT

- Inner products are accumulated in a buffer



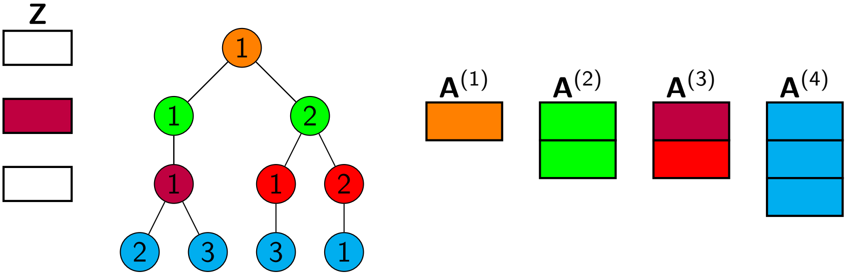
CSF-ROOT

- Inner products are accumulated in a buffer



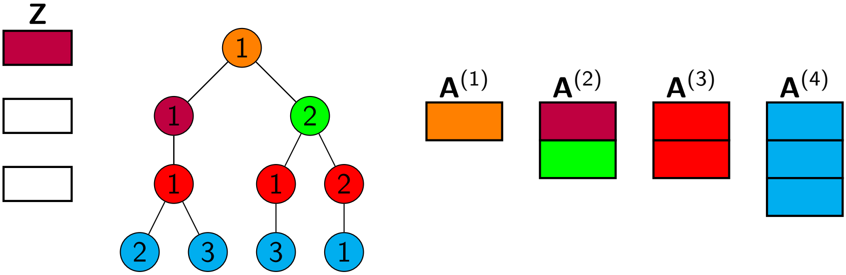
CSF-ROOT

- Hadamard products are then propagated up the CSF tree



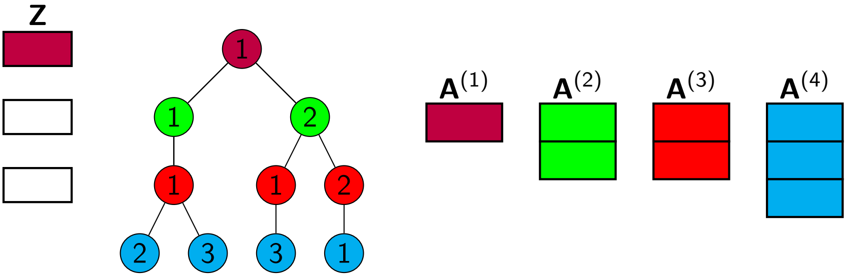
CSF-ROOT

- Hadamard products are then propagated up the CSF tree



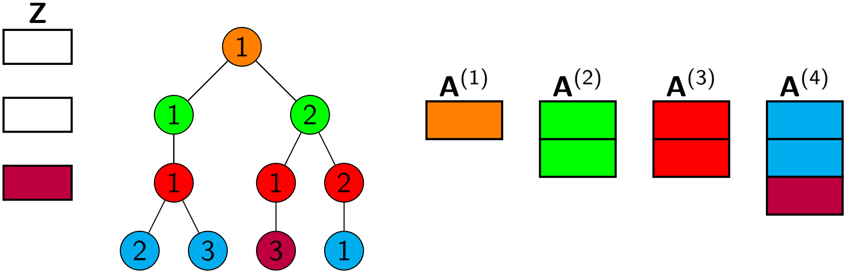
CSF-ROOT

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



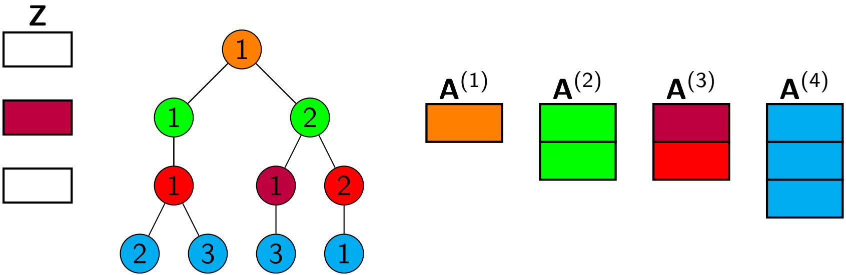
CSF-ROOT

- The traversal continues...



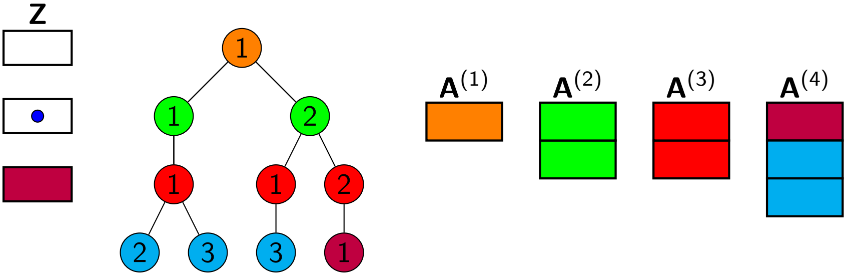
CSF-ROOT

- The traversal continues...



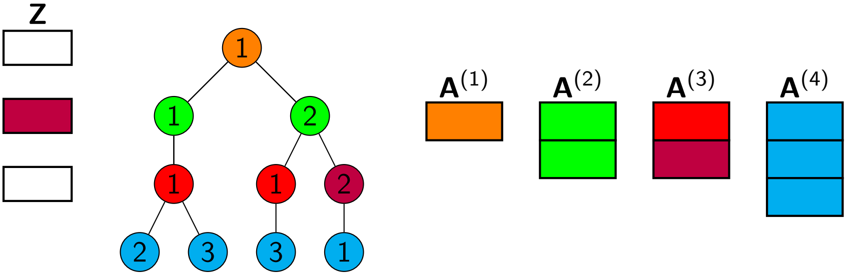
CSF-ROOT

- Partial results are kept in buffer



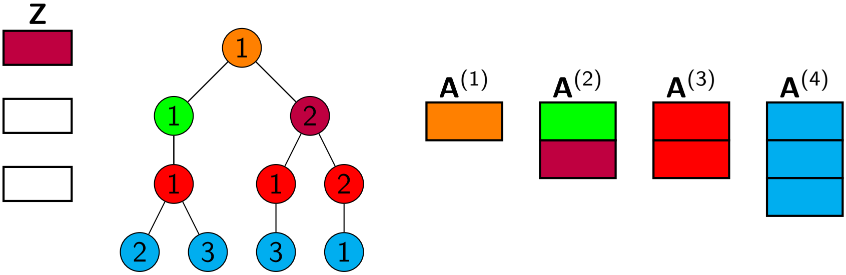
CSF-ROOT

- Inner products are accumulated in a buffer



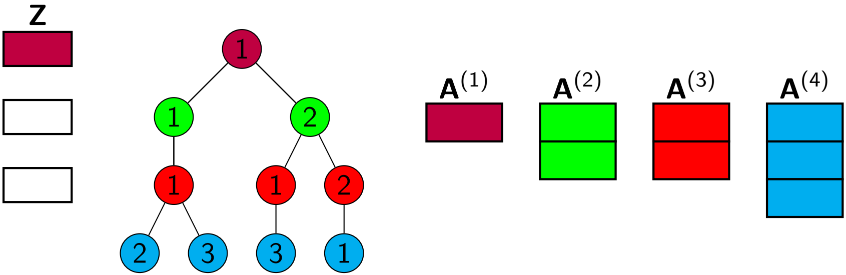
CSF-ROOT

- Inner products are accumulated in a buffer



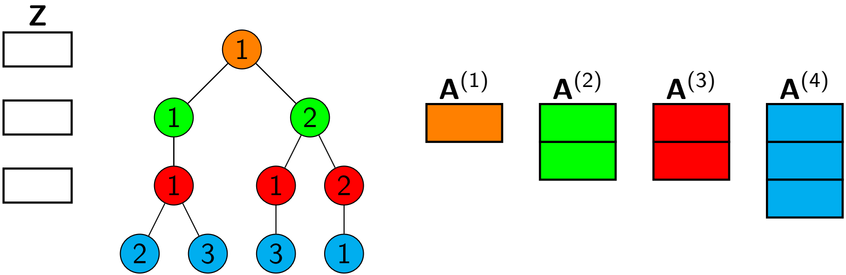
CSF-ROOT

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



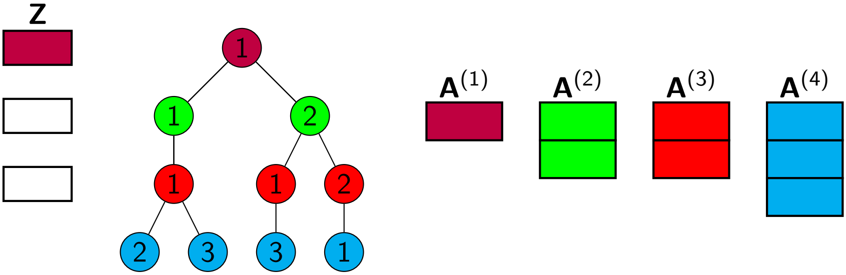
CSF-INTERNAL

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



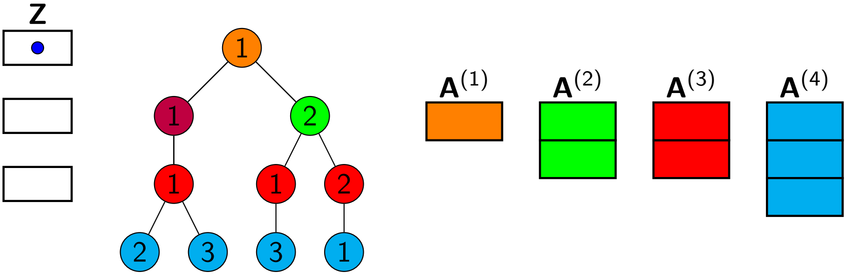
CSF-INTERNAL

- Hadamard products are pushed down to the output level



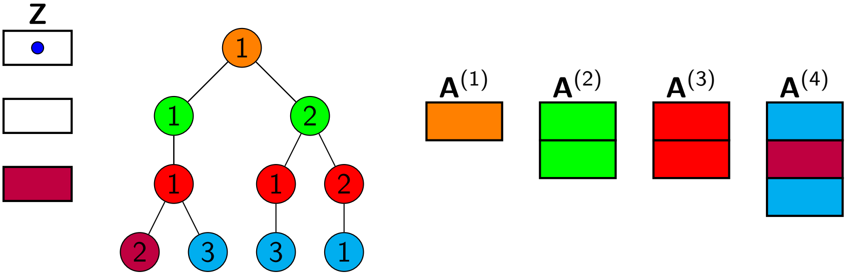
CSF-INTERNAL

- CSF-ROOT next pulls up to the output level



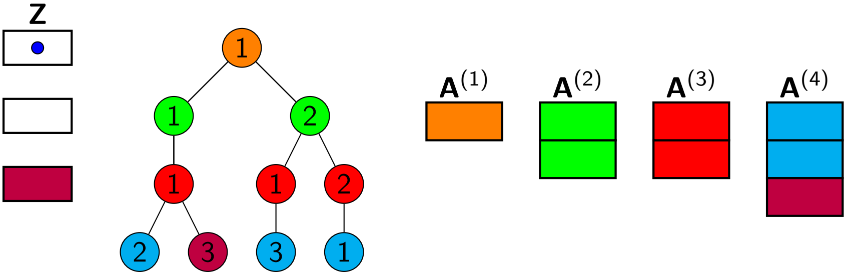
CSF-INTERNAL

- CSF-ROOT next pulls up to the output level



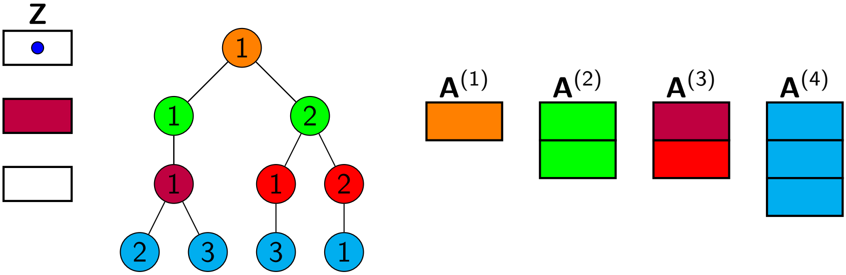
CSF-INTERNAL

- CSF-ROOT next pulls up to the output level



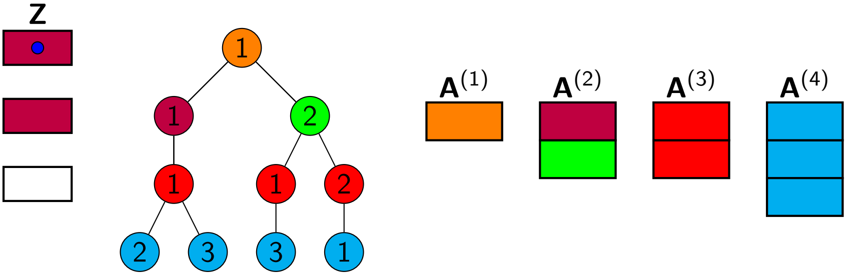
CSF-INTERNAL

- CSF-ROOT next pulls up to the output level



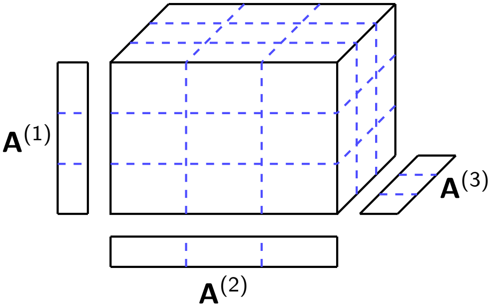
CSF-INTERNAL

- CSF-ROOT next pulls up to the output level



Parallelism – Tiling

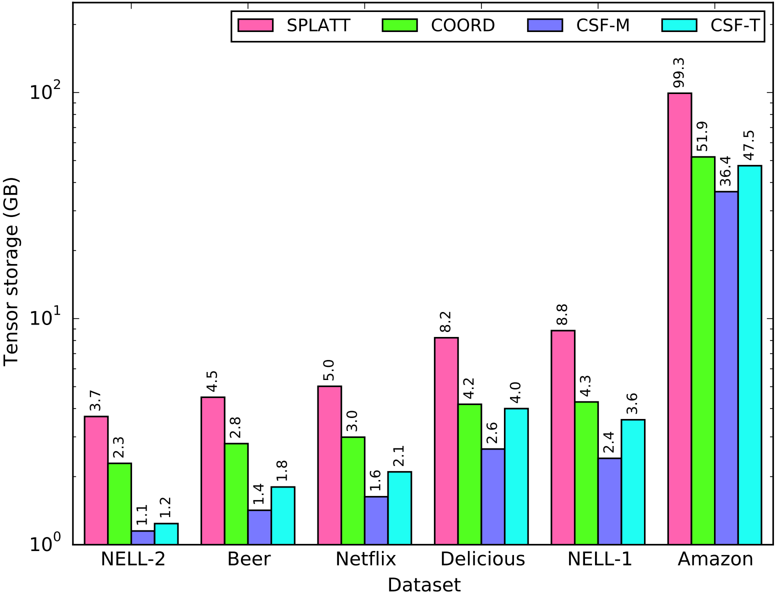
- For p threads, we do a p-way tiling of each tensor mode
- Distributing the tiles allows us to eleminate the need for mutexes



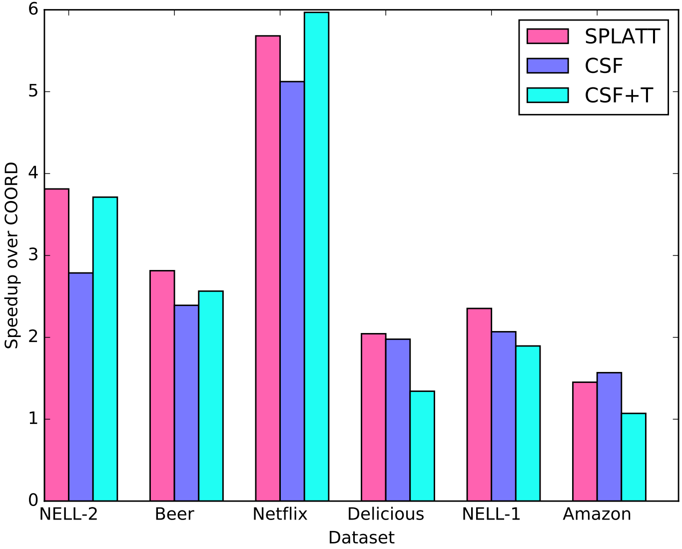
Datasets

Dataset	$\mathbf{I}_1$	$\mathbf{I}_2$	$\mathbf{I}_3$	**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

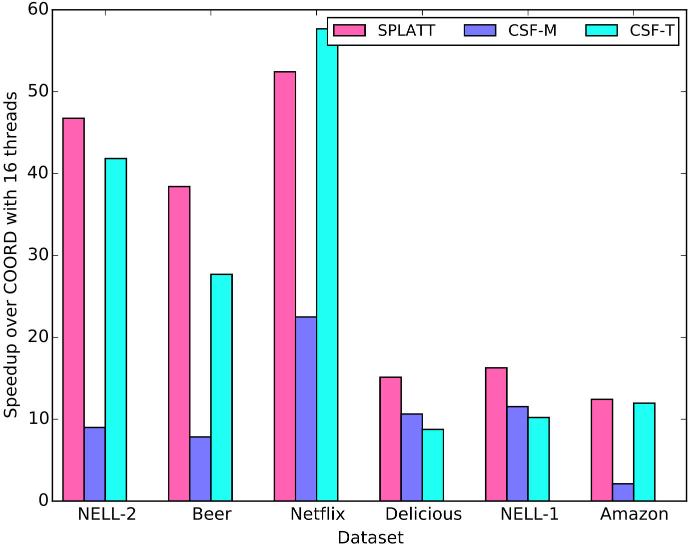
Storage Comparison



Serial Comparison



MTTKRP



SPLATT
CSF-M
CSF-T
Speedup over COORD with 16 threads
0
10
20
30
40
50
60
NELL-2
Beer
Netflix
Delicious
NELL-1
Amazon
Dataset

Conclusions

Compressed Sparse Fiber

- CSF uses 58% less memory than SPLATT while maintaining 81% of its performance
- CSF and related algorithms are now included in SPLATT

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