Accelerating the Tucker Decomposition with Compressed Sparse Tensors

Shaden Smith and George Karypis

Department of Computer Science & Engineering, University of Minnesota
{shaden, karypis}@cs.umn.edu

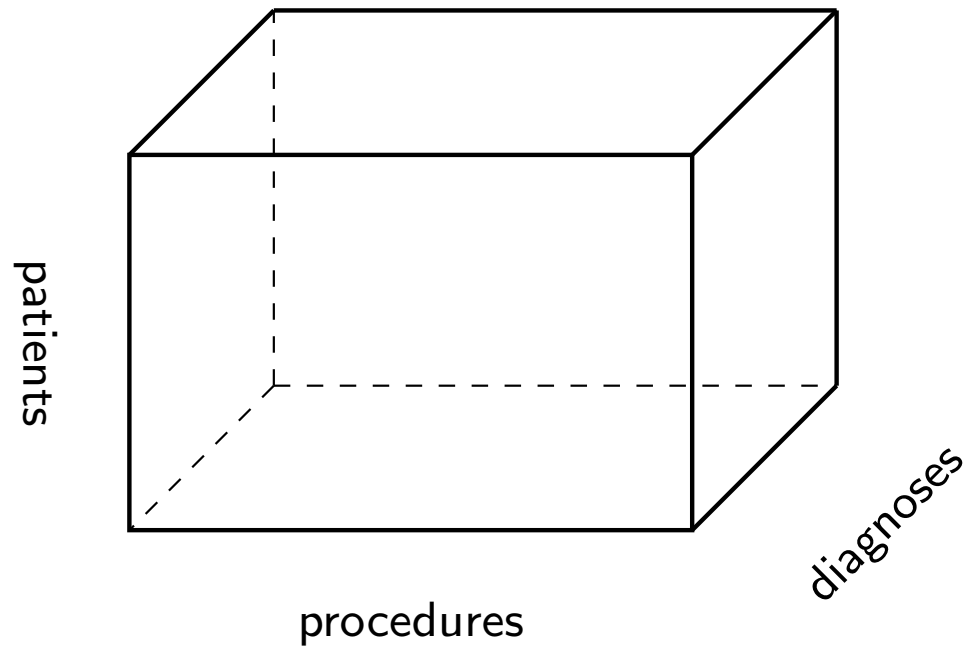
SIAM Conference on Applied Linear Algebra 2018

Table of Contents

Tensors

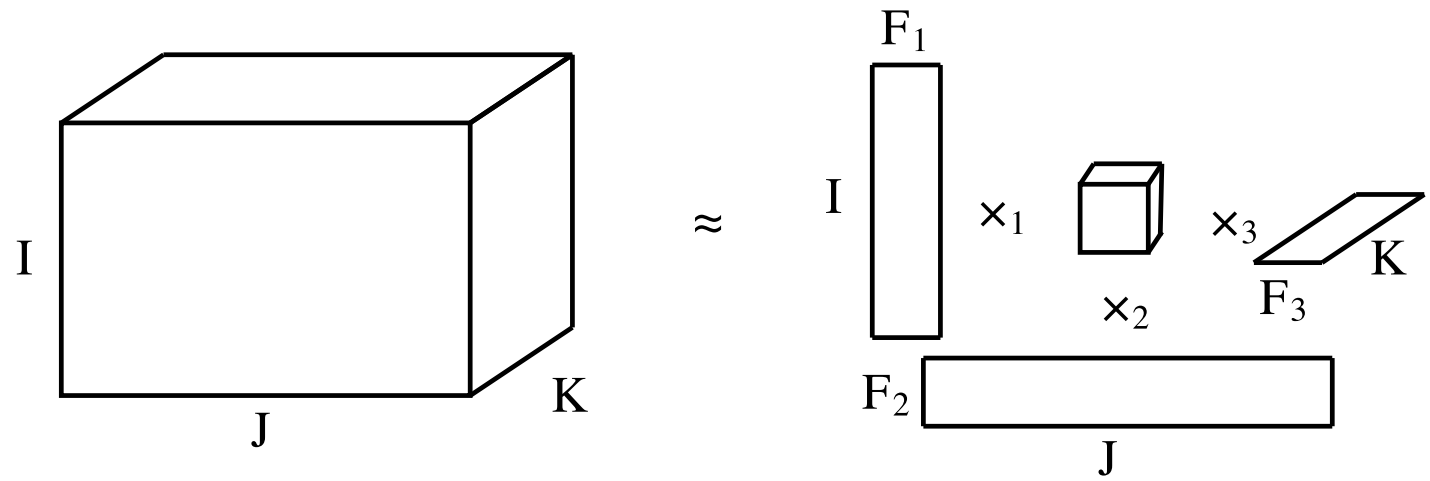
Tensors are the generalization of matrices to higher orders.

- Allow us to represent and analyze multi-dimensional data
- Applications in precision healthcare, cybersecurity, recommender systems, ...



Tucker decomposition

The Tucker decomposition models a tensor $\mathcal{X}$ as a set of orthogonal factor matrices and a core tensor.



Notation

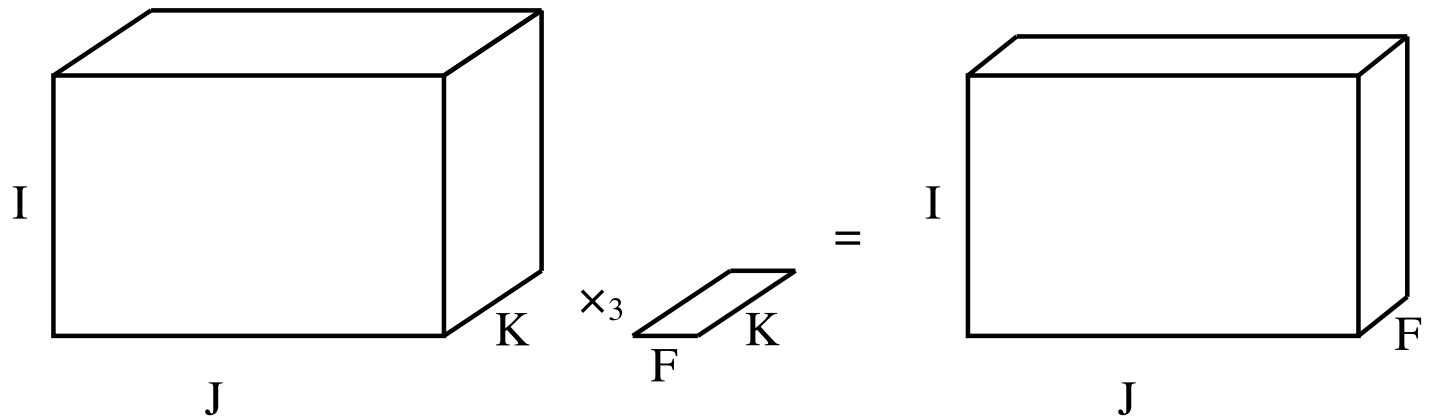
$\mathbf{A} \in \mathbb{R}^{I \times F_1}$, $\mathbf{B} \in \mathbb{R}^{J \times F_2}$, and $\mathbf{C} \in \mathbb{R}^{K \times F_3}$ denote the factor matrices.

$\mathcal{G} \in \mathbb{R}^{F_1 \times F_2 \times F3}$ denotes the core tensor.

Essential operation: tensor-matrix multiplication

Tensor-matrix multiplication (TTM; also called the n-way product)

- Given: tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$ and matrix $\mathbf{M} \in \mathbb{R}^{F \times K}$.
- Operation: $\mathcal{X} \times_3 \mathbf{M}$
- Output: $\mathcal{Y} \in \mathbb{R}^{I \times J \times F}$

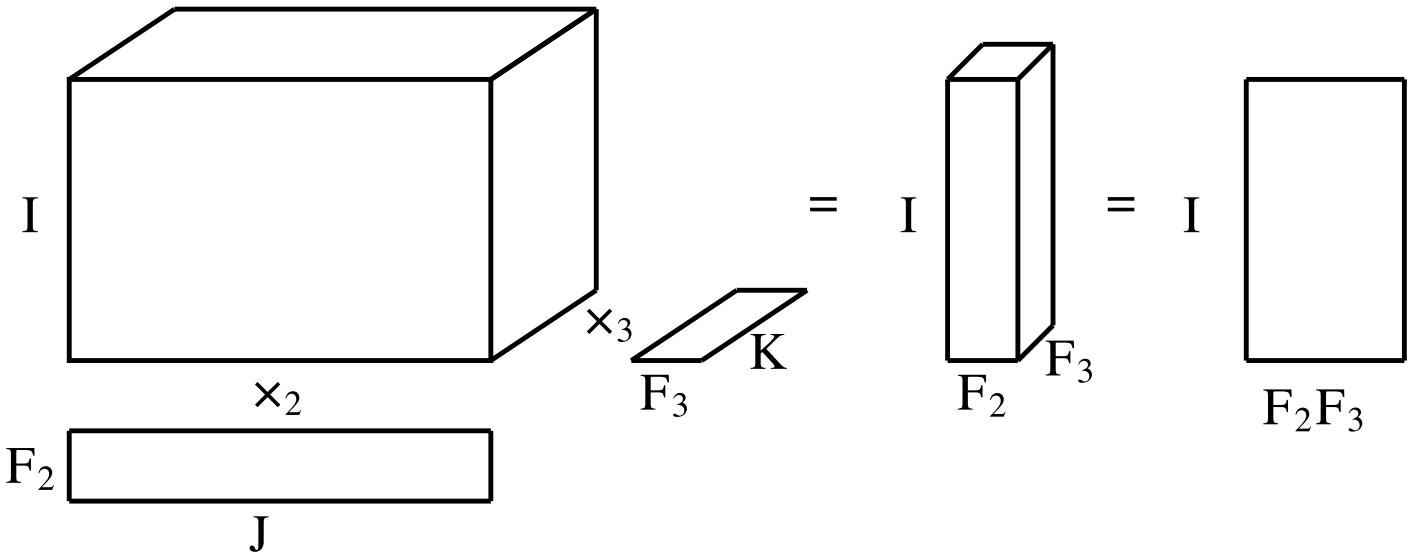


Elementwise:

$$\mathcal{Y}(i,j,f) = \sum_{k=1}^{K} \mathcal{X}(i,j,k)\mathbf{M}(f,k).$$

Chained tensor-matrix multiplication (TTMc)

Tensor-matrix multiplications are often performed in sequence (*chained*).



$$\mathcal{Y}_1 \leftarrow \mathcal{X} \times_2 \mathbf{B}^T \times_3 \mathbf{C}^T$$

Notation

Tensors can be *unfolded* along one mode to matrix form: $\mathbf{Y}_{(n)}$.

- Mode n forms the rows and the remaining modes become columns.

Higher-Order Orthogonal Iterations (HOOI)

Tucker Decomposition with HOOI

while not converged **do**
  $\mathcal{Y}_1 \leftarrow \mathcal{X} \times_2 \mathbf{B}^T \times_3 \mathbf{C}^T$
  $\mathbf{A} \leftarrow F_1$ leading left singular vectors of $\mathbf{Y}_{(1)}$

  $\mathcal{Y}_2 \leftarrow \mathcal{X} \times_1 \mathbf{A}^T \times_3 \mathbf{C}^T$
  $\mathbf{B} \leftarrow F_2$ leading left singular vectors of $\mathbf{Y}_{(2)}$

  $\mathcal{Y}_3 \leftarrow \mathcal{X} \times_1 \mathbf{A}^T \times_2 \mathbf{B}^T$
  $\mathbf{C} \leftarrow F_3$ leading left singular vectors of $\mathbf{Y}_{(3)}$

  $\mathcal{G} \leftarrow \mathcal{X} \times_1 \mathbf{A}^T \times_2 \mathbf{B}^T \times_3 \mathbf{C}^T$
end while

TTMc is the most expensive kernel in the HOOI algorithm.

Table of Contents

TTM & Intermediate memory blowup

A first step is to optimize a single TTM kernel and apply in sequence:

$$\mathcal{Y}_1 \leftarrow \left(\left(\mathcal{X} \times_2 \mathbf{B}^T\right) \times_3 \mathbf{C}^T\right)$$

Challenge:

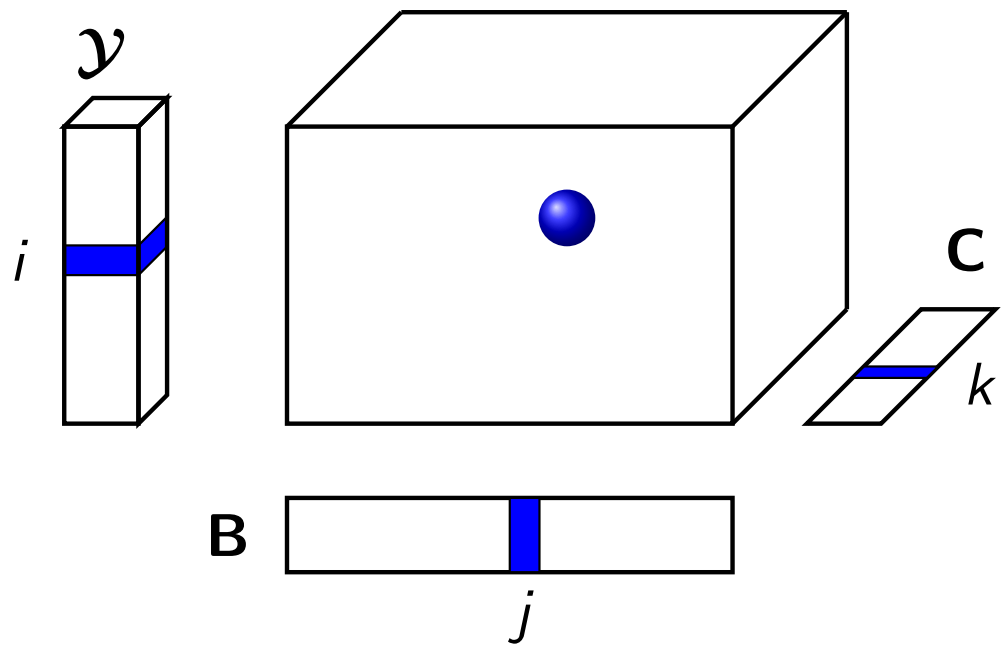
- Intermediate results become more dense after each TTM.
- Memory overheads are dependent on sparsity pattern and factorization rank, but can be several orders of magnitude.

Tamara Kolda and Jimeng Sun. "Scalable tensor decompositions for multi-aspect data mining". In: *International Conference on Data Mining (ICDM)*. 2008.

Elementwise formulation

Processing each non-zero individually has cost $\mathcal{O}(\text{nnz}(\mathcal{X})F_2F_3)$ and $\mathcal{O}(F_2F_3)$ memory overhead.

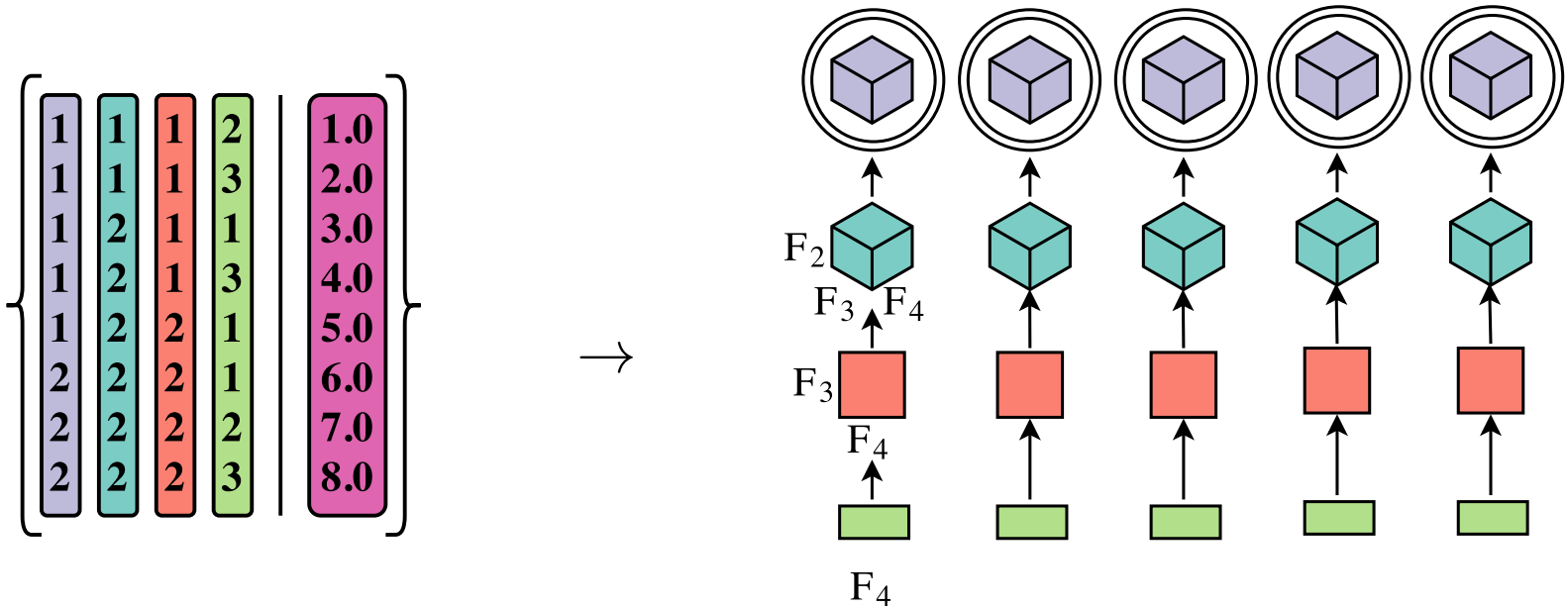
$$\mathcal{Y}_1(i,:,:) \mathrel{+}= \mathcal{X}(i,j,k)\,[\mathbf{B}(j,:) \circ \mathbf{C}(k,:)]$$



Oguz Kaya and Bora Uçar. *High-performance parallel algorithms for the Tucker decomposition of higher order sparse tensors*. Tech. rep. RR-8801. Inria-Research Centre Grenoble–Rhône-Alpes, 2015.

TTMc with coordinate form

The elementwise formulation of TTMc naturally lends itself to a coordinate storage format:

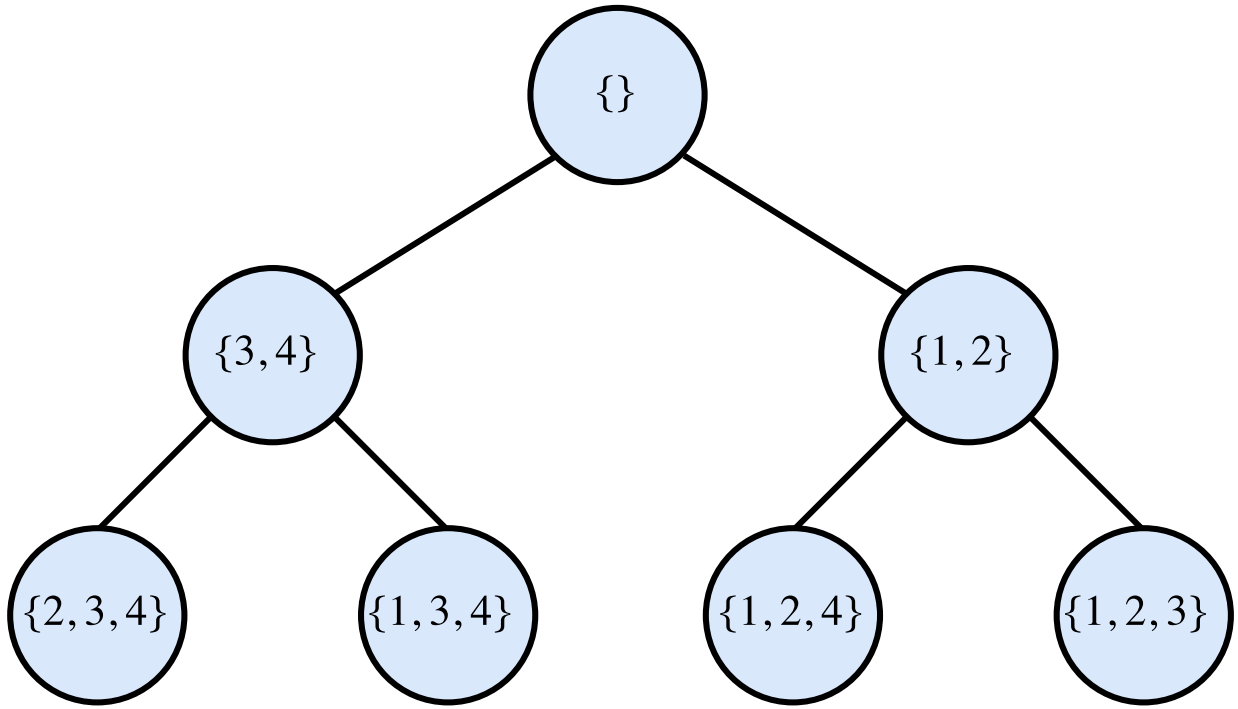


$$\mathcal{Y}_1(i,:,:,:) \mathrel{+}= \mathcal{X}(i,j,k,l)\,[\mathbf{B}(j,:) \circ \mathbf{C}(k,:) \circ \mathbf{D}(l,:)]$$

TTMc with dimension trees

State-of-the-art TTMc:

- Each node in the tree stores intermediate results from a set of modes.



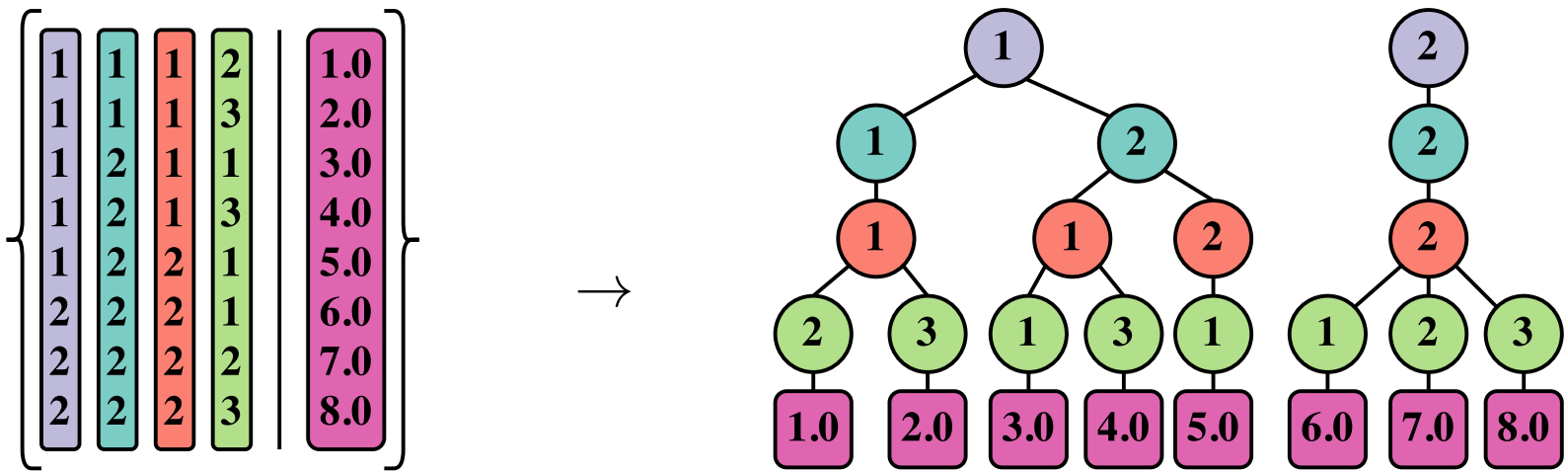
Oguz Kaya and Bora Uçar. *High-performance parallel algorithms for the Tucker decomposition of higher order sparse tensors*. Tech. rep. RR-8801. Inria-Research Centre Grenoble–Rhône-Alpes, 2015.

Table of Contents

Compressed Sparse Fiber (CSF)

CSF encodes a sparse tensor as a forest.

- Each path from root to leaf encodes a non-zero.
- CSF can be viewed as a generalization of CSR.



Shaden Smith and George Karypis. "Tensor-Matrix Products with a Compressed Sparse Tensor". In: *5th Workshop on Irregular Applications: Architectures and Algorithms*. 2015.

Arithmetic redundancies in TTMc

Going back to the non-zero formulation:

$$\mathcal{Y}_1(i,:,:) \mathrel{+}= \mathcal{X}(i,j,k)\,[\mathbf{B}(j,:) \circ \mathbf{C}(k,:)]$$

There are two arithmetic redundancies we can exploit:

1. Distributive outer products
2. Redundant outer products

Distributive outer products

Consider two non-zeros in the same fiber $\boldsymbol{\mathcal{X}}(i, j, :)$

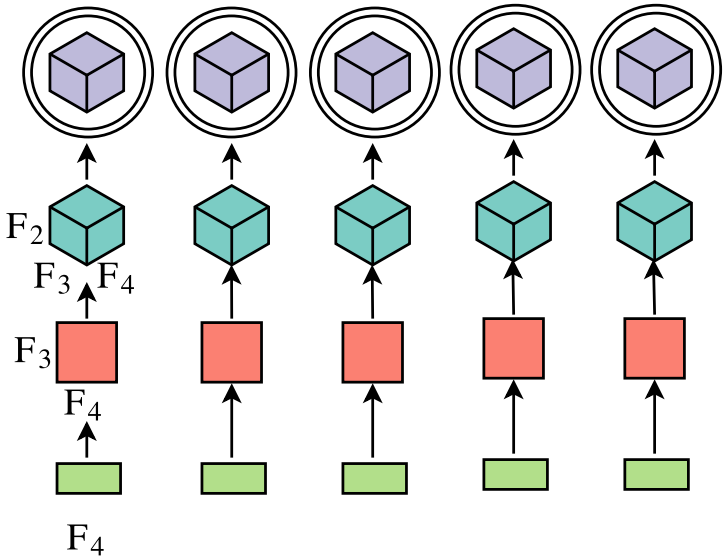
$$\boldsymbol{\mathcal{Y}}_1(i, :, :) \mathrel{+}= \boldsymbol{\mathcal{X}}(i, j, k_1) \left[\mathbf{B}(j, :) \circ \mathbf{C}(k_1, :)\right]$$
$$\boldsymbol{\mathcal{Y}}_1(i, :, :) \mathrel{+}= \boldsymbol{\mathcal{X}}(i, j, k_2) \left[\mathbf{B}(j, :) \circ \mathbf{C}(k_2, :)\right]$$

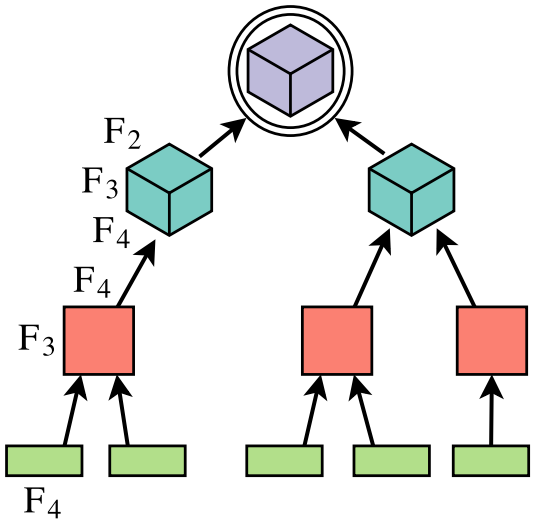
We can factor out $\mathbf{B}(j, :)$

$$\boldsymbol{\mathcal{Y}}_1(i, :, :) \mathrel{+}= \mathbf{B}(j, :) \circ \left[\boldsymbol{\mathcal{X}}(i, j, k_1)\mathbf{C}(k_1, :) + \boldsymbol{\mathcal{X}}(i, j, k_2)\mathbf{C}(k_2, :)\right]$$

Distributive outer products with CSF

Compare to computing with coordinate format:





Savings: Under some mild assumptions, the cost of each non-zero (leaf) is now linear in the rank.

Redundant outer products

Suppose that we are now computing $\mathcal{Y}_3$:

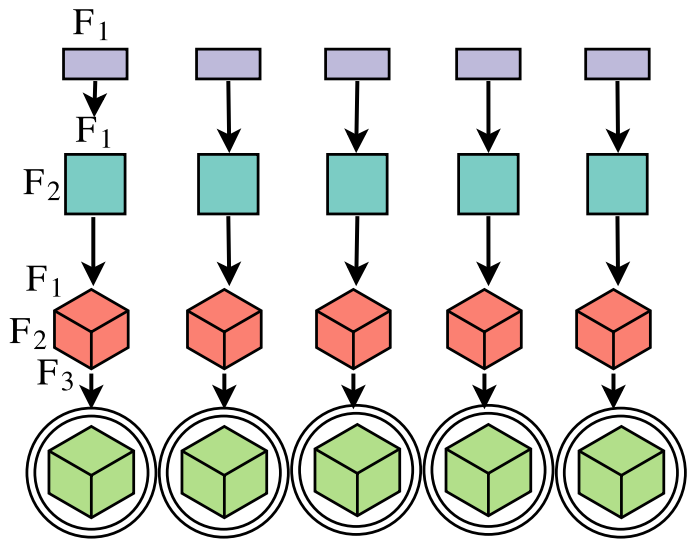
$$\mathcal{Y}_3(:,:,k_1) \mathrel{+}= \mathcal{X}(i,j,k_1)\left[\mathbf{A}(i,:) \circ \mathbf{B}(j,:)\right],$$
$$\mathcal{Y}_3(:,:,k_2) \mathrel{+}= \mathcal{X}(i,j,k_2)\left[\mathbf{A}(i,:) \circ \mathbf{B}(j,:)\right].$$

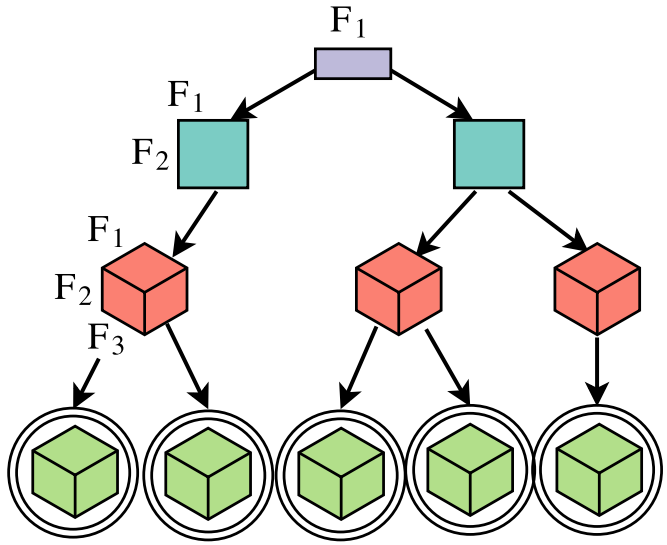
The outer product can be reused:

$$\mathbf{S} \leftarrow \mathbf{A}(i,:) \circ \mathbf{B}(j,:)$$
$$\mathcal{Y}_3(:,:,k_1) \mathrel{+}= \mathcal{X}(i,j,k_1) \cdot \mathbf{S}$$
$$\mathcal{Y}_3(:,:,k_2) \mathrel{+}= \mathcal{X}(i,j,k_2) \cdot \mathbf{S}$$

Redundant outer products with CSF

Compare to computing with coordinate format:

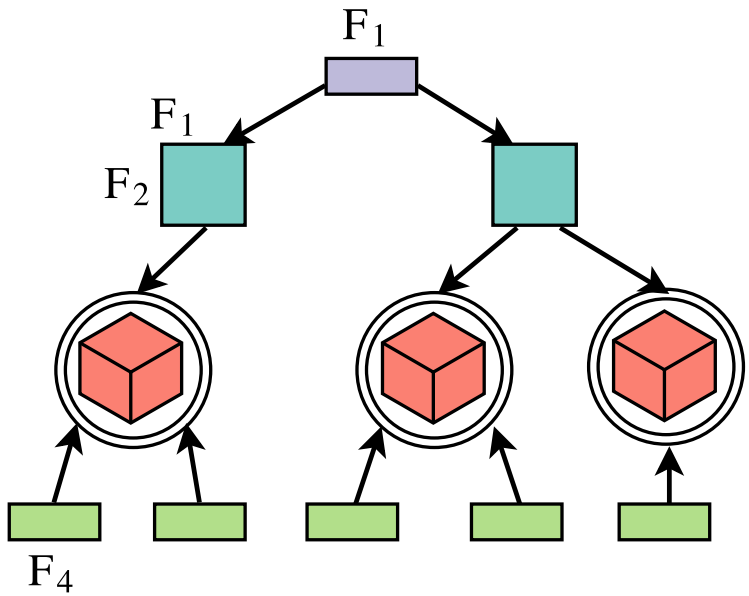




Savings: Outer products are constructed less often, but non-zeros still have the same asymptotic cost as coordinate form.

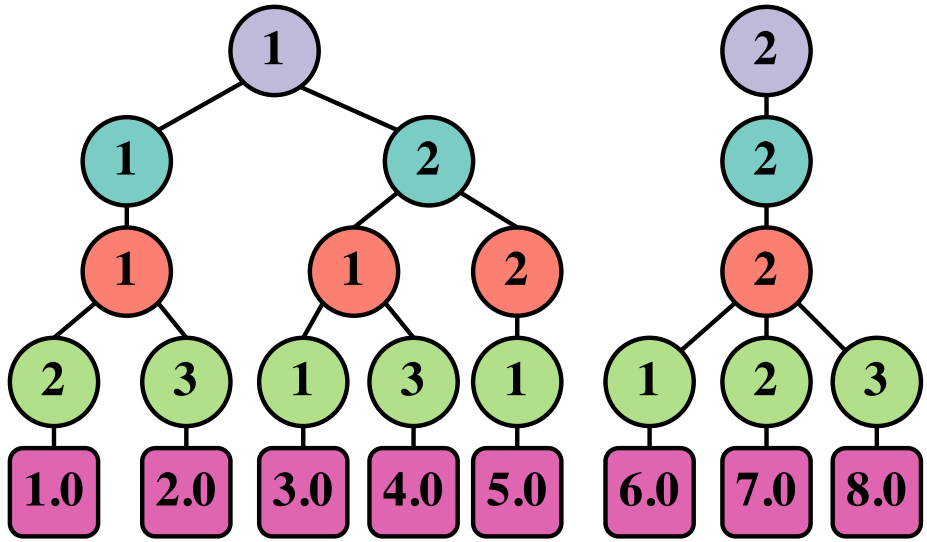
Putting it all together

These two optimizations can be combined within the same kernel.



Parallelism with CSF

We distribute trees to threads and use dynamic load balancing.



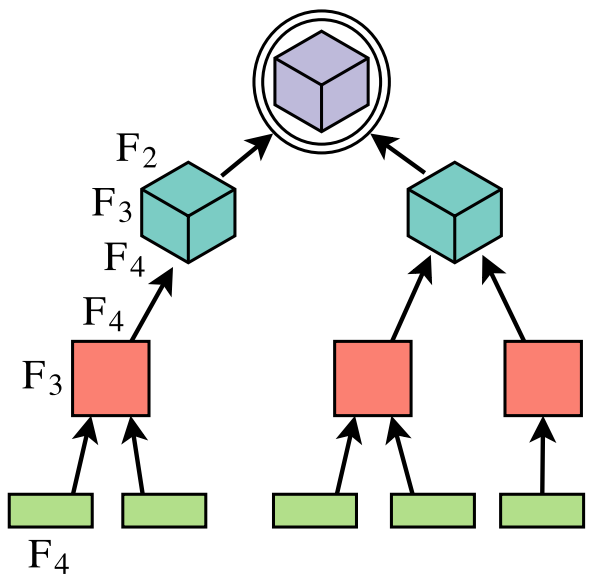
Race conditions are dependent on the mode of interest:

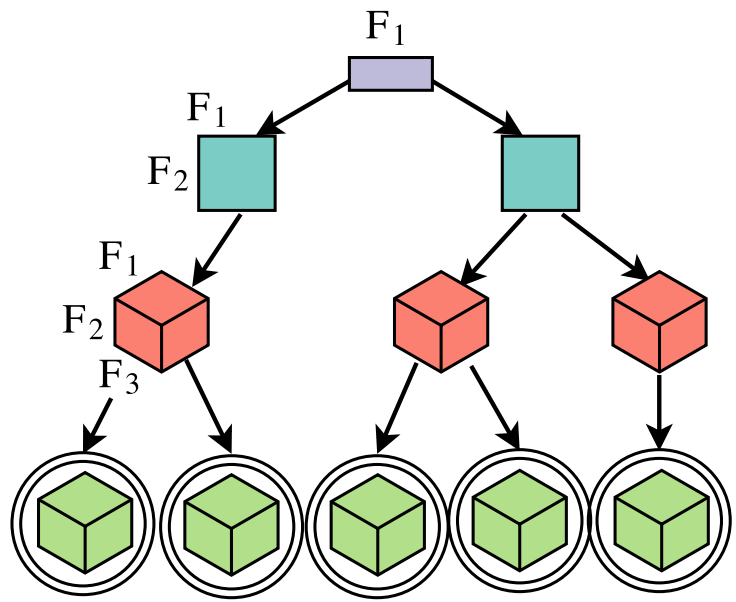
- Root nodes are unique, so no race conditions
- Otherwise, use a mutex to lock the slice of $\mathcal{Y}$

Limitation

TTMc is significantly more expensive when computing for the lower-level modes.

- This is due to FLOPs and synchronization costs.





Multiple CSF representations

We can reorder the modes of $\mathcal{X}$ and store additional copies of the tensor.

- Selectively place some modes near the top which were previously expensive.

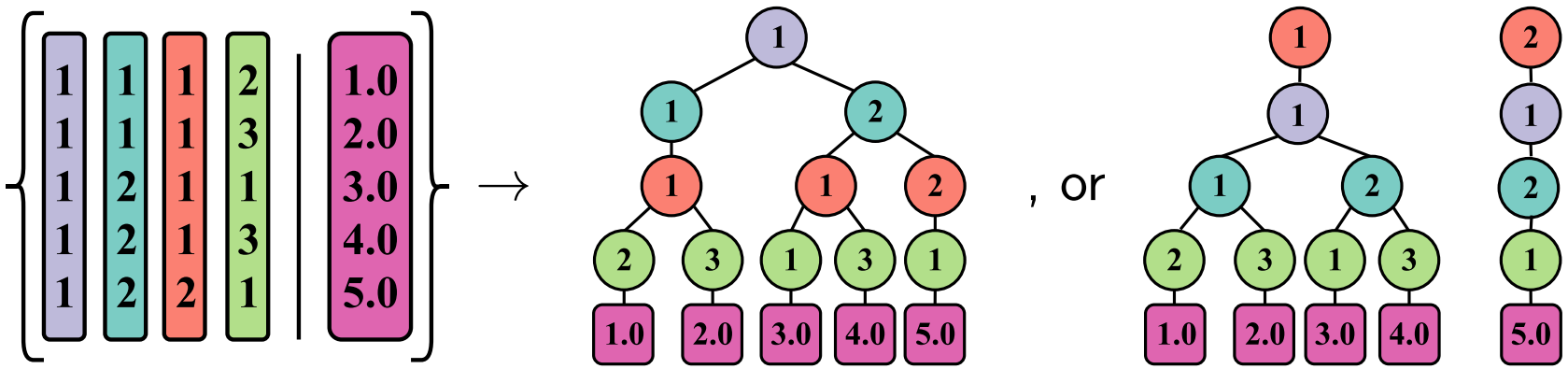


Table of Contents

Experimental Setup

Source code:

- Part of the **S**urprisingly **P**aralle**L** sp**A**rse **T**ensor **T**oolkit[1]
- Written in C and parallelized with OpenMP
- Compiled with `icc` v16.0.3 and linked with Intel MKL

HyperTensor

- Implements dimenson tree-based methods
- Written in C++ and parallelized with OpenMP

Machine specifications:

- 2x 12-core Intel E5-2680v3 processors (Haswell)
- Double-precision floats and 32-bit integers

[1] SPLATT: `https://github.com/ShadenSmith/splatt`

Datasets

Most datasets are available from FROSTT:[2]

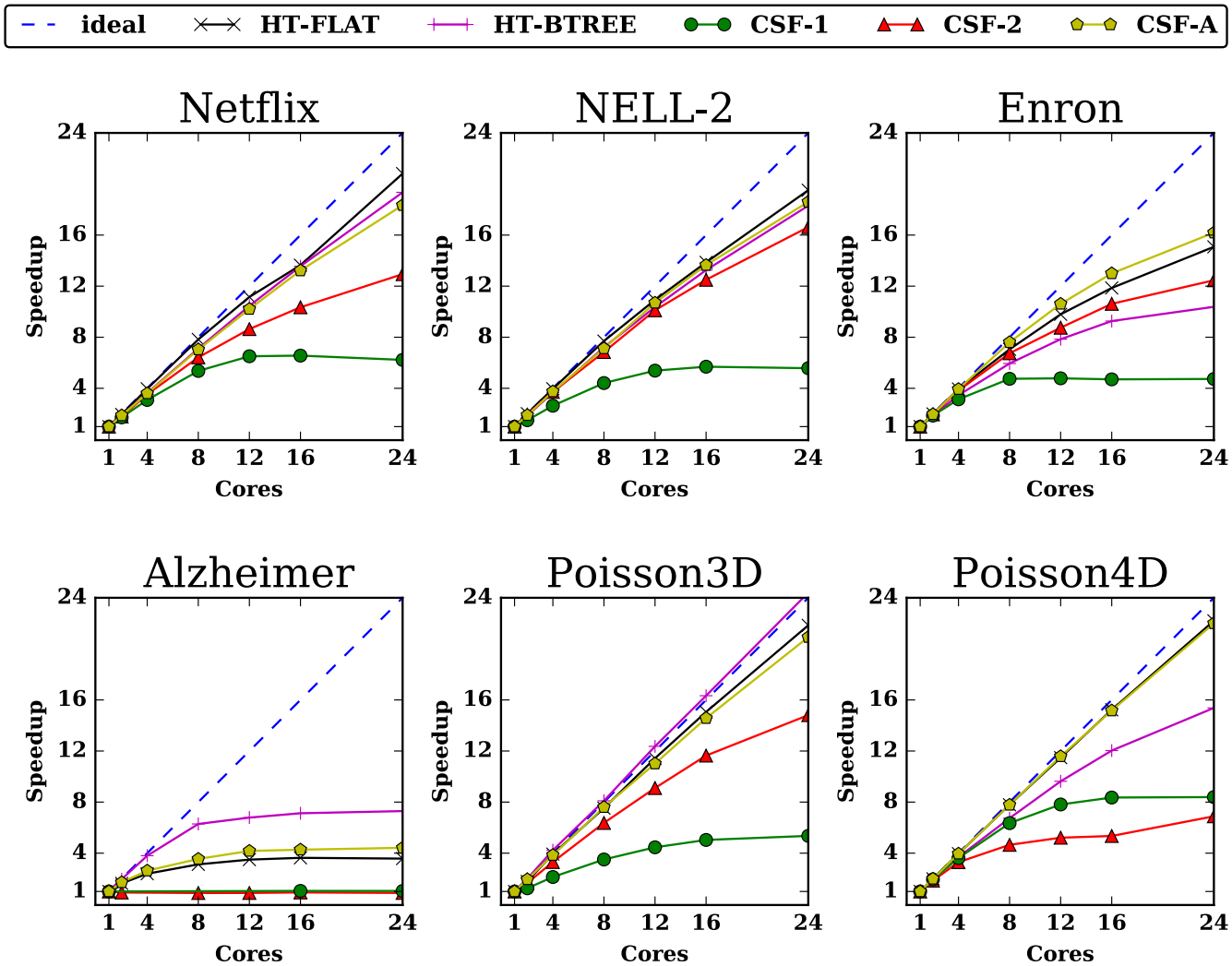
Dataset	Non-zeros	Modes	Dimensions
`NELL-2`	77M	3	12K, 9K, 29K
`Netflix`	100M	3	480K, 18K, 2K
`Enron`	54M	4	6K, 6K, 244K, 1K
`Alzheimer`	6.27M	5	5, 1K, 156, 1K, 396
`Poisson3D`, `Poisson4D`	100M	3,4	3K, . . . , 3K

K and **M** stand for thousand and million, respectively.

[2] FROSTT: `http://frostt.io/`

Parallel Scalability

Adding CSF representations improves scalability due to fewer and smaller critical regions.



Performance tradeoffs

Selecting the number of CSFs provides tuning for memory vs. speed.

- CSF always provides the options for the smallest and fastest executions.

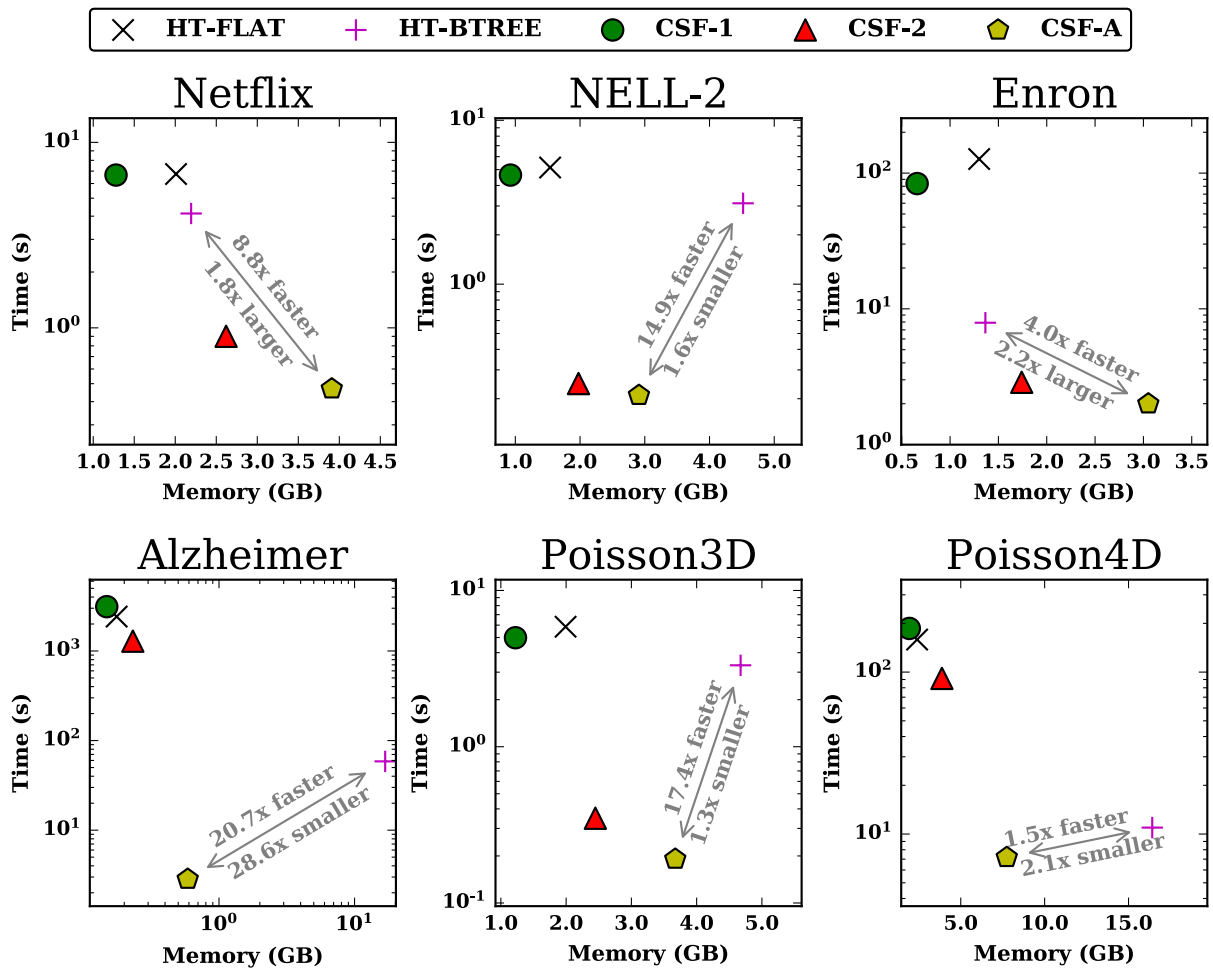


Table of Contents

Wrapping up

Contributions:

- We optimized TTMc kernels via a compressed tensor representation
  - CSF naturally exposes arithmetic redundancies in TTMc
- Multiple CSF tensors can further accelerate computation
  - Up to 20× speedup over the state-of-the-art while using 28× less memory
  - Choosing the number of data copies offers *tunable* computation/memory tradeoff