BERKELEY LAB

Distributed-Memory Randomized Algorithms for Sparse Tensor CP Decomposition

SPAA 2024

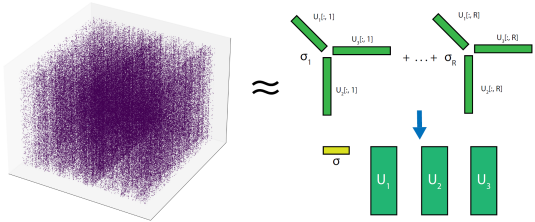
June 18, 2024

Vivek Bharadwaj [1, 2], Osman Asif Malik [3], Riley Murray [4], Aydın Buluç [2, 1], James Demmel [1]

[1] University of California, Berkeley
[2] Lawrence Berkeley National Laboratory
[3] Encube Technologies
[4] Sandia National Laboratories

Sparse Tensor CP Decomposition

Our Goal: Compute an approximate rank-R Candecomp / PARAFAC decomposition of an N-dimensional $I \times ... \times I$ sparse tensor $\mathcal{T}$.



Rows of factor matrices are **embedding vectors**, can be used for pattern / trend identification [Smi+18], anomaly detection [Mao+14], social network analysis [LK22].

Parallel Sparse CP is Well-Studied

Software	Source	Notes
SPLATT	[Smi+15]	CA-Distributed Algorithms, CSF Data Structure
BIGTensor	[Par+16]	Hadoop MapReduce
ParTI!	[LMV18]	GPU Support, HiCOO Data Structure
Genten	[PK19]	Kokkos Parellelism and Performance Portability

Iterative algorithms for sparse CP decomposition are typically:

- *Based on overdetermined linear least-squares* with very tall design matrices.
- *Communication-bound* on distributed-memory clusters.
- *Memory-bandwidth bound* locally on each processor.
- *Difficult to performance-model* without assumptions on the sparse tensor.

Randomized Algorithms for CP Decomposition

Algorithm	Source	Round Complexity ($\tilde{O}$ notation)
CP-ALS	[KB09]	$N(N+I)I^{N-1}R$
CP-ARLS-LEV	[LK22]	$N(R+I)R^N/(\varepsilon\delta)$
TNS-CP	[Mal22]	$N^3IR^3/(\varepsilon\delta)$
GTNE	[MS22]	$N^2(N^{1.5}R^{3.5}/\varepsilon^3 + IR^2)/\varepsilon^2$
STS-CP	[Bha+23]	$N(NR^3\log I + IR^2)/(\varepsilon\delta)$

Table: Complexity of Randomized CP Methods for Dense Tensors

- Several recent works use *randomization* to reduce CP decomposition runtime.
- Prototype implementations **cannot compete** with existing optimized, non-randomized software packages. Could they be made competitive in practice?

Contributions: Randomized CP at Scale

- We give high-performance implementations of STS-CP and CP-ARLS-LEV in the distributed-memory parallel setting.
- Up to 11x speedup over SPLATT on hundreds of MPI ranks / thousands of CPU cores.
- To achieve this, we optimize both communication and computation at several stages the iterative algorithm for CP decomposition.

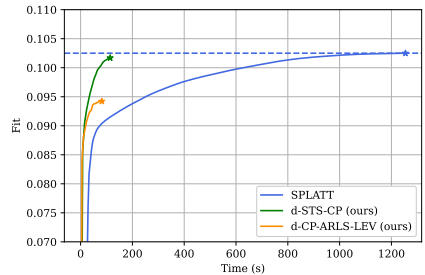


Figure: Accuracy vs. time, Reddit tensor, $R = 100$, 512 cores / 4 Perlmutter CPU nodes, 4.7B NNZ.

Preliminaries and Base Data Distribution

Alternating Least-Squares CP Decomposition

- ALS procedure: Randomly initialize factors $U_1, ..., U_N$, iteratively optimize one factor at a time while keeping others constant.

- Optimal value for U_j:

$$\text{argmin}_X \|AX - B\|_F$$

where

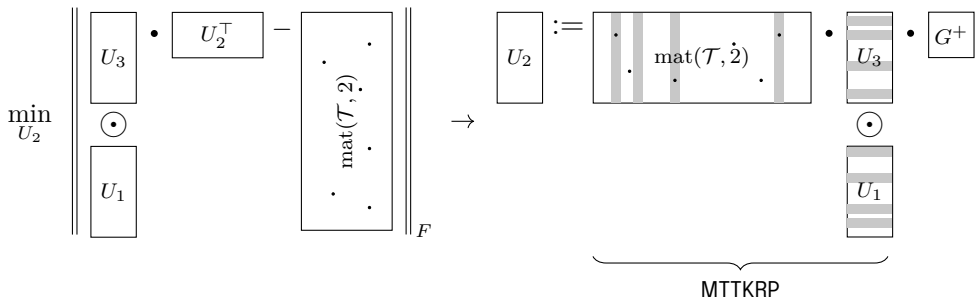
- $A = U_N \odot ... \odot U_{j+1} \odot U_{j-1} \odot ... \odot U_1$ is a **Khatri-Rao product**

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \odot \begin{bmatrix} w & x \\ y & z \end{bmatrix} = \begin{bmatrix} aw & bx \\ cw & dx \\ ay & bz \\ cy & dz \end{bmatrix}$$

- $B = \text{mat}(\mathcal{T}, j)^\top$

Illustrated Least-Squares Problem

$$\min_{U_j} \left\| \left[\bigodot_{k \neq j} U_k \right] \cdot U_j^\top - \mathrm{mat}(\mathcal{T}, j)^\top \right\|_F$$



Randomized Linear Least-Squares

- Apply sketching operator S to both A and B, solve reduced problem

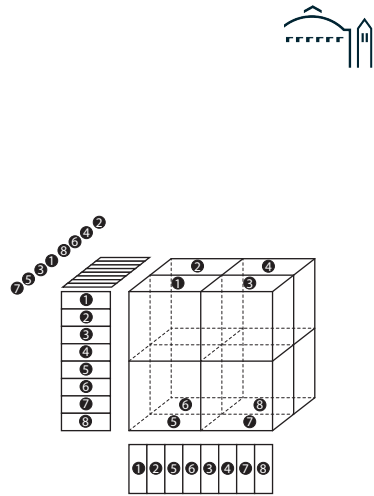
$$\min_{\tilde{X}} \left\| SA\tilde{X} - SB \right\|_F$$

- Want an (ε, δ) guarantee on solution quality: with high probability $(1 - \delta)$,

$$\left\| A\tilde{X} - B \right\|_F \leq (1 + \varepsilon) \min_X \|AX - B\|$$

- Here, restrict S to be a *sampling* matrix: selects and reweights rows from A and B. Preserves the sparsity of the matricized tensor.

Factor and Sparse Matrix Layout



- Processors arranged in N-dimensional hypercube.
- Factor matrices $U_1, ..., U_N$ distributed by block rows. Assume that all processors redundantly compute $U_j^\top U_j$ for all j, use identity

$$G = A^\top A = \circledast_{j=1}^{N} U_j^\top U_j$$

- Each processor owns a block of the sparse tensor. Randomly permute modes to load-balance.

Bulk-Synchronous Randomized ALS Update

1. **Sampling Identification:** Draw rows from $U_{\neq j}$ that are "influential".

2. **Distributed MTTKRP Communication:** Communicate rows of $U_{\neq j}$ to processors who require them. After the local computation, reduce the output if necessary.

3. **Local MTTKRP Computation:** Use the nonzeros of the sparse tensor and gathered rows to perform a modified sparse-dense matrix multiplication on each processor.

Technical Contributions

Distributed Sampling Algorithms

We devise **two** distinct sampling strategies with nearly ideal communication scaling.

Sampler	Compute	Messages	Words Communicated
d-CP-ARLS-LEV	JN/P	JN/P	P
d-STS-CP	$JNR^2 \log I/P$	$NP \log P$	$JNR \log P/P$

- d-CP-ARLS-LEV allows processors to sample independently from each factor matrix block row. Cheap, but lower final accuracy.
- d-STS-CP performs a random walk down a binary tree to locate each sample index. We distribute both the tree and the random walk.

d-CP-ARLS-LEV Parallelization Scheme

- **Key Idea:** Sample independently from each factor matrix block row according to a *leverage score distribution*.

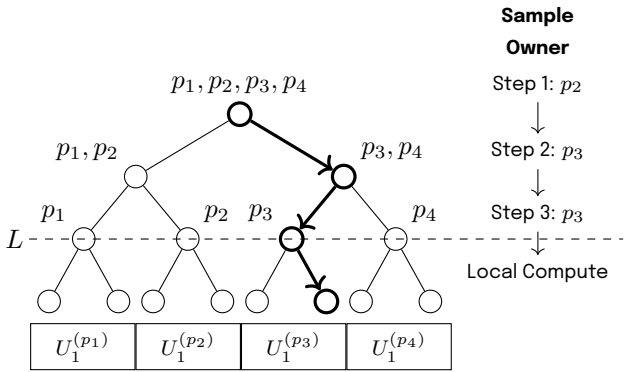
- Let $U_i^{(p_j)}$ be the block row of U_i owned by processor p_j. Leverages scores of this block given by

$$\operatorname{diag}\left(U_i^{(p_j)} G^{+} U_i^{(p_j)\top}\right)$$

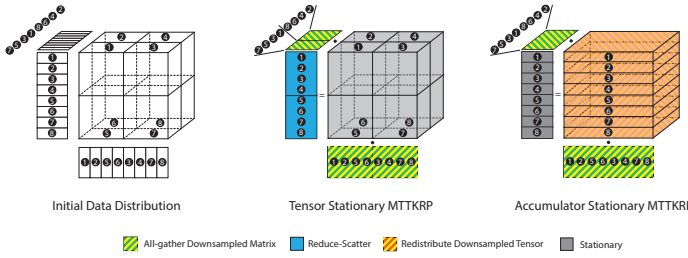
- Computed locally on each processor without communication. Sampling requires (in expectation) only a small constant number of words communicated, followed by drawing samples from the local weights.

- **Drawback:** Accuracy degrades for high N or R.

d-STS-CP Parallelization Scheme



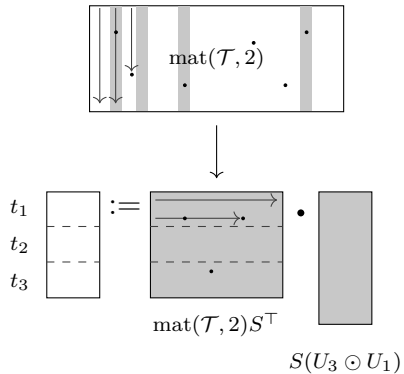
Optimizing MTTKRP Communication



Schedule	Communication
Non-Randomized TS	$2IRN/P^{1/N}$
Sampled TS	$IRN/P^{1/N}$
Sampled AS	$JRN(N-1)$

Variable	Range
I	$\sim 10^5 - 10^7$
J	$\sim 10^3 - 10^5$
R	$\sim 10^1 - 10^2$
N	$3 - 6$

Sparse Transpose in Local MTTKRP



Switch between modified CSC format for nonzero selection, CSR format for kernel execution.

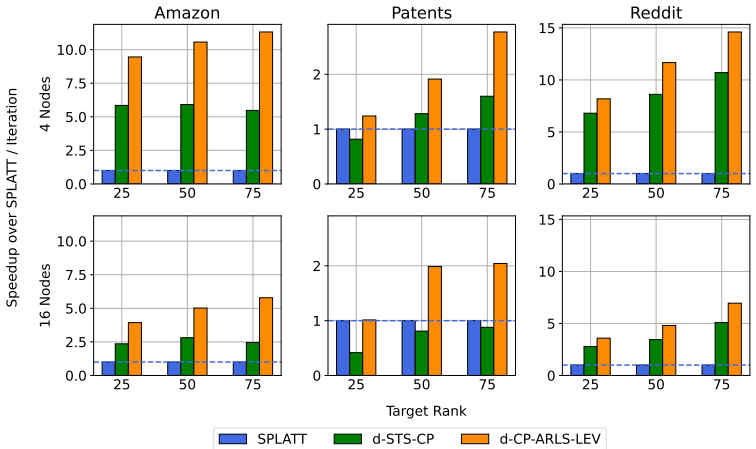
Experiments

Experimental Platform

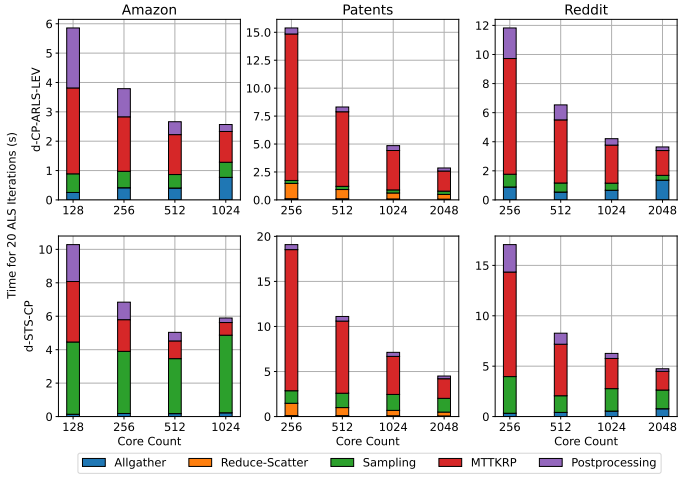
- Experiments conducted on 2048 CPU cores on NERSC Perlmutter at LBNL.
- Baseline : SPLATT, an optimized non-randomized decomposition library. Selected three **largest** tensors from the FROSTT collection to benchmark.

Tensor	Dimensions	NNZ
Amazon	$4.8M \times 1.8M \times 1.8M$	$1.7B$
Patents	$46 \times 239K \times 239K$	$3.6B$
Reddit	$8.2M \times 177K \times 8.1M$	$4.7B$

Speedup over SPLATT



Strong Scaling



Conclusions and Summary

- We provided distributed-memory algorithms for sparse CP decomposition with robust theoretical guarantees and a distributed-memory communication analysis.
- We demonstrated that our algorithms are **practical** on billion-scale real-world datasets.
- Our work serves as a case study in combining **randomness**, **sparsity**, and **distributed-memory parallel computing**.

Thank you, questions welcome.

References I

[Bha+23] Vivek Bharadwaj, Osman Asif Malik, Riley Murray, Laura Grigori, Aydin Buluc, and James Demmel. "Fast Exact Leverage Score Sampling from Khatri-Rao Products with Applications to Tensor Decomposition". In: *Thirty-seventh Conference on Neural Information Processing Systems*. 2023.

[KB09] Tamara G. Kolda and Brett W. Bader. "Tensor Decompositions and Applications". In: *SIAM Review* 51.3 (Aug. 2009). Publisher: Society for Industrial and Applied Mathematics, pp. 455–500. ISSN: 0036-1445. DOI: 10.1137/07070111X. (Visited on 09/12/2022).

[LK22] Brett W. Larsen and Tamara G. Kolda. "Practical Leverage-Based Sampling for Low-Rank Tensor Decomposition". In: *SIAM J. Matrix Analysis and Applications* (June 2022). accepted for publication. eprint: 2006.16438.

[LMV18] Jiajia Li, Yuchen Ma, and Richard Vuduc. *ParTI! : A Parallel Tensor Infrastructure for multicore CPUs and GPUs*. Last updated: Jan 2020. Oct. 2018. URL: http://parti-project.org.

[Mal22] Osman Asif Malik. "More Efficient Sampling for Tensor Decomposition With Worst-Case Guarantees". en. In: *Proceedings of the 39th International Conference on Machine Learning*. ISSN: 2640-3498. PMLR, June 2022, pp. 14887–14917.

References II

[Mao+14] Hing-Hao Mao, Chung-Jung Wu, Evangelos E. Papalexakis, Christos Faloutsos, Kuo-Chen Lee, and Tien-Cheu Kao. "MalSpot: Multi2 Malicious Network Behavior Patterns Analysis". en. In: *Advances in Knowledge Discovery and Data Mining*. Ed. by Vincent S. Tseng, Tu Bao Ho, Zhi-Hua Zhou, Arbee L. P. Chen, and Hung-Yu Kao. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2014, pp. 1–14. ISBN: 978-3-319-06608-0. DOI: 10.1007/978-3-319-06608-0_1.

[MS22] Linjian Ma and Edgar Solomonik. "Cost-efficient Gaussian tensor network embeddings for tensor-structured inputs". In: *Advances in Neural Information Processing Systems*. Vol. 35. Curran Associates, Inc., 2022, pp. 38980–38993.

[Par+16] Namyong Park, ByungSoo Jeon, Jungwoo Lee, and U Kang. "BIGtensor: Mining Billion-Scale Tensor Made Easy". In: *ACM International Conference on Conference on Information and Knowledge Management (CIKM)*. 2016.

[PK19] Eric T Phipps and Tamara G Kolda. "Software for sparse tensor decomposition on emerging computing architectures". In: *SIAM Journal on Scientific Computing* 41.3 (2019), pp. C269–C290.

References III

[Smi+15] Shaden Smith, Niranjay Ravindran, Nicholas D. Sidiropoulos, and George Karypis. “SPLATT: Efficient and Parallel Sparse Tensor-Matrix Multiplication”. In: *2015 IEEE International Parallel and Distributed Processing Symposium*. ISSN: 1530-2075. May 2015, pp. 61–70. DOI: 10.1109/IPDPS.2015.27.

[Smi+18] Shaden Smith, Kejun Huang, Nicholas D. Sidiropoulos, and George Karypis. “Streaming Tensor Factorization for Infinite Data Sources”. In: *Proceedings of the 2018 SIAM International Conference on Data Mining (SDM)*. SIAM, 2018, pp. 81–89. DOI: 10.1137/1.9781611975321.10.

Backup Slides

Technical Challenges

The arithmetic intensity of an algorithm is the ratio of FLOPs / data words moved between processors. We face dual challenges to parallel scaling:

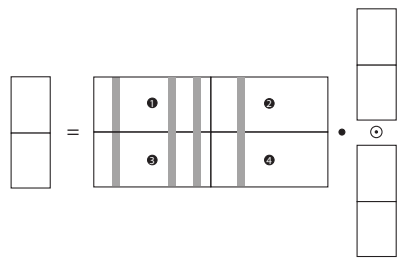
1. Sparse linear algebra primitives have far lower intensity than their dense operations.
2. Randomization may decrease computation costs while keeping communication unchanged.

We achieve both scaling and speedup through three key optimizations:

Fully Distributed Random Sampling	Randomization-Tailored MTTKRP	Sparse Transpose for Local Computation

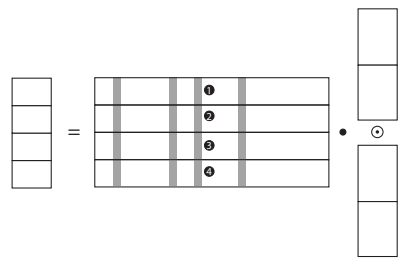
Tensor-Stationary MTTKRP Load Balance

- We use random permutations of each tensor mode to evenly distribute nonzeros & samples to processors.
- Theoretical model: each sampled column has q nonzeros with row i.i.d. uniform.
- TS Load Balance: J balls into $P^{1-1/N}$ bins (each ball here is a column).



Accumulator-Stationary MTTKRP Load Balance

- AS Load Balance: Jq balls into P bins.
- Here, each ball is a nonzero entry. This distibution has better load balance when q is high.

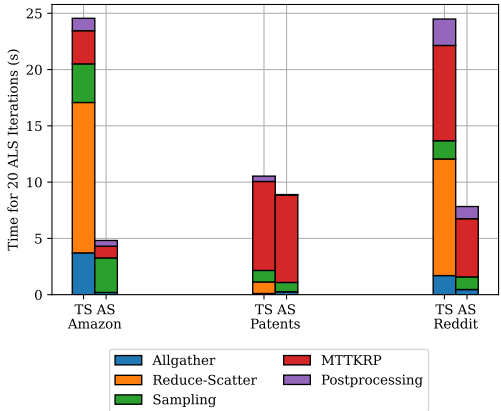


Randomized Algorithm Accuracy

Tensor	R	d-CP-ARLS-LEV	d-STS-CP	Exact
Amazon	25	0.338	0.340	0.340
	50	0.359	0.366	0.366
	75	0.368	0.381	0.382
Patents	25	0.451	0.451	0.451
	50	0.467	0.467	0.467
	75	0.475	0.475	0.476
Reddit	25	0.0583	0.0592	0.0596
	50	0.0746	0.0775	0.0783
	75	0.0848	0.0910	0.0922

Table: Average Fits, $J = 2^{16}$, 32 MPI Ranks, 4 Nodes

Comparison of Communication Schedules



Load Balance