High-Performance Star-M SVD for Big Data Compression

Md Taufique Hussain*, Grey Ballard*, Aditya Devarakonda*, Srinivas Eswar†, Naman Pesricha‡, Vishwas Rao†
*Wake Forest University, Winston-Salem, NC, USA
†Argonne National Laboratory, Lemont, IL, USA
‡Indian Institute of Science, Bengaluru, India
{hussaint, ballard, devaraa}@wfu.edu, {seswar, vhebbur}@anl.gov, namanp@iisc.ac.in

***Abstract*—In the era of big data, effectively compressing large datasets while performing complex mathematical operations is crucial. Tensor-based decomposition methods have shown superior compression capabilities with minimal loss of accuracy compared to traditional matrix methods. Under the $\star_{\mathrm{M}}$ tensor framework, tensors can be decomposed in a matrix-mimetic way, including using the $\star_{\mathrm{M}}$ SVD. This tensor SVD has optimality guarantees and has shown exceptional performance on specific types of data, but software implementations have been mostly limited to productivity-oriented languages. In this work, we present our development of a shared-memory parallel, high-performance solution designed to efficiently implement the underlying algorithms. This software will enable optimal compression of extensive scientific datasets, paving the way for enhanced data analysis and insights.**

I. Introduction

Truncated tensor SVDs built on the $\star_{\mathrm{M}}$ tensor-tensor product mimic the matrix SVD [1]: an input tensor is decomposed into three factor tensors with properties of orthogonality and structure analogous to the matrix case, and that decomposition can be truncated for compression given either an error tolerance or specified rank parameters. Unlike established tensor decompositions such as CP, Tucker, and Tensor Train, this truncation is provably optimal, owing to guarantees for the $\star_{\mathrm{M}}$ algebra that generalize the Eckart–Young theorem [2]. Yet implementations of the $\star_{\mathrm{M}}$ t-SVD exist only in productivity-oriented languages [3]–[5], while high-performance parallel software is well established for those other formats [6]–[8]. We close this gap with `pystarm`, a shared-memory parallel implementation of the $\star_{\mathrm{M}}$ t-SVD algorithms for general multiway data, enabling its use on datasets that were otherwise too large to handle.

II. Background

Given an invertible matrix $\mathbf{M}$, the $\star_{\mathrm{M}}$ product of two tensors is defined in three steps: transform both operands along mode 3 by $\mathbf{M}$, multiply the corresponding pairs of frontal slices independently, and map the result back by $\mathbf{M}^{-1}$. The $\star_{\mathrm{M}}$ t-SVD follows directly: after the transform the frontal slices are mutually decoupled, so each can be factored by an independent thin matrix SVD. The decomposition therefore reduces to two kernels—a tensor-times-matrix (TTM) product and a batch of independent slice SVDs—and for d-way tensors $\mathbf{M}$ is applied to every mode $k > 2$. Two truncation variants exist. t-SVDM-I fixes a single rank k for every slice. t-SVDM-II instead takes a relative error tolerance, derives one global threshold from the singular values of all slices, and truncates slice i at the rank ρ_i that threshold implies.

III. Design and Implementation

We design and implement both algorithms in C++ on top of Intel MKL, with Python bindings via pybind11. Data is laid out so that frontal slices occupy contiguous memory and can be passed to a matrix routine without copying or transposition.

A. Tensor-Times-Matrix

The transform must be applied to every mode beyond the second. For the last mode the whole tensor can be treated as a single matrix, so the transform is one GEMM. For a middle mode the values each multiplication needs are strided in memory, so the transform instead becomes a large collection of independent, identically shaped GEMMs. Middle modes arise only for tensors with more than three modes, but when they do they account for most of the transform work. We consider three ways to parallelize that collection: issue the GEMMs one after another, each using all threads; issue them simultaneously, each restricted to one thread; or hand the whole collection to a single batched GEMM call. In our experiments batched GEMM was the best choice at every mode and thread count.

B. Slice-wise SVD

After the transform, each slice's SVD is independent of the others. Two strategies are available: process the slices one at a time and let each SVD use all threads, or process the slices in parallel with each SVD restricted to a single thread. The first suits a few large slices, the second many small ones. In our experiments the second scaled almost linearly up to 64 threads, while the first did not scale.

IV. Application on Climate Data

We evaluate on ten years of NCEP/NCAR reanalysis air temperature [9], represented as a 6-way tensor of shape $73 \times 144 \times 17 \times 4 \times 365 \times 10$ (roughly 20 GB), using a single ALCF Aurora node with dual-socket Intel Xeon Sapphire Rapids 9470C processors (104 cores).

Both tensor methods compress better than Empirical Orthogonal Functions [10], the standard matrix-based method in

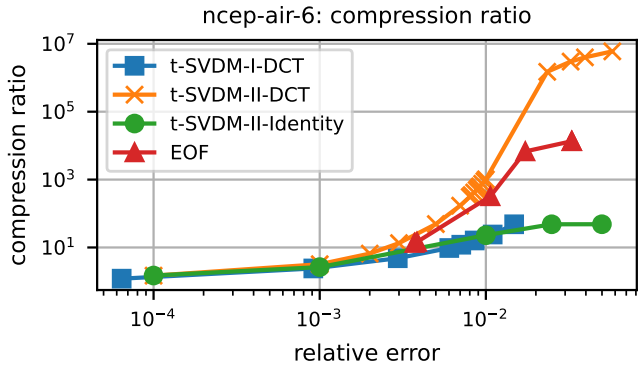


Fig. 1. Compression ratio versus relative error.

climate science, and t-SVDM-II beats t-SVDM-I at every error level because variable per-slice ranks allow it to discard many slices entirely; the DCT in turn beats the identity transform (Fig. 1). At a matched compression ratio near $48\times$, t-SVDM-I scales $42\times$ from 1 to 64 threads (147 s to 3.5 s) and t-SVDM-II $27\times$ (Fig. 2); the gap arises because the threshold calculation in t-SVDM-II is sequential and consumes a quarter of the runtime at 64 threads.

V. Conclusions and Future Work

We present the first high-performance implementation of tensor decompositions in the $\star_{\mathrm{M}}$ framework, scaling efficiently to 64 threads and running up to $1000\times$ faster than existing MATLAB code. The software is available at https://github.com/Streaming-Tensors/pystarm/. GPU support and distributed-memory algorithms are immediate next steps.

References

[1] M. E. Kilmer, L. Horesh, H. Avron, and E. Newman, "Tensor-tensor algebra for optimal representation and compression of multiway data," *Proceedings of the National Academy of Sciences*, vol. 118, no. 28, p. e2015851118, 2021.

[2] C. Eckart and G. Young, "The approximation of one matrix by another of lower rank," *Psychometrika*, vol. 1, no. 3, pp. 211–218, 1936.

[3] C. Lu, *Tensor-Tensor Product Toolbox*, Carnegie Mellon University, June 2018, https://github.com/canyilu/tproduct.

[4] E. Newman and K. Keegan, "Optimal matrix-mimetic tensor algebras via variable projection," *SIAM Journal on Matrix Analysis and Applications*, vol. 46, no. 3, pp. 1764–1790, 2025. [Online]. Available: https://doi.org/10.1137/24M1702635

[5] K. Keegan, T. Vishwanath, and Y. Xu, "A tensor svd-based classification algorithm applied to fmri data," 2021. [Online]. Available: https://arxiv.org/abs/2111.00587

[6] A. E. Helal, J. Laukemann, F. Checconi, J. J. Tithi, T. Ranadive, F. Petrini, and J. Choi, "ALTO: adaptive linearized storage of sparse tensors," in *Proceedings of the 35th ACM International Conference on Supercomputing*, ser. ICS '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 404?416. [Online]. Available: https://doi.org/10.1145/3447818.3461703

[7] G. Ballard, A. Klinvex, and T. G. Kolda, "TuckerMPI: A parallel C++/MPI software package for large-scale data compression via the tucker tensor decomposition," *ACM Transactions on Mathematical Software*, vol. 46, no. 2, Jun. 2020. [Online]. Available: https://dl.acm.org/doi/10.1145/3378445

[8] H. Al Daas, G. Ballard, and P. Benner, "Parallel algorithms for tensor train arithmetic," *SIAM Journal on Scientific Computing*, vol. 44, no. 1, pp. C25–C53, 2022. [Online]. Available: https://doi.org/10.1137/20M1387158

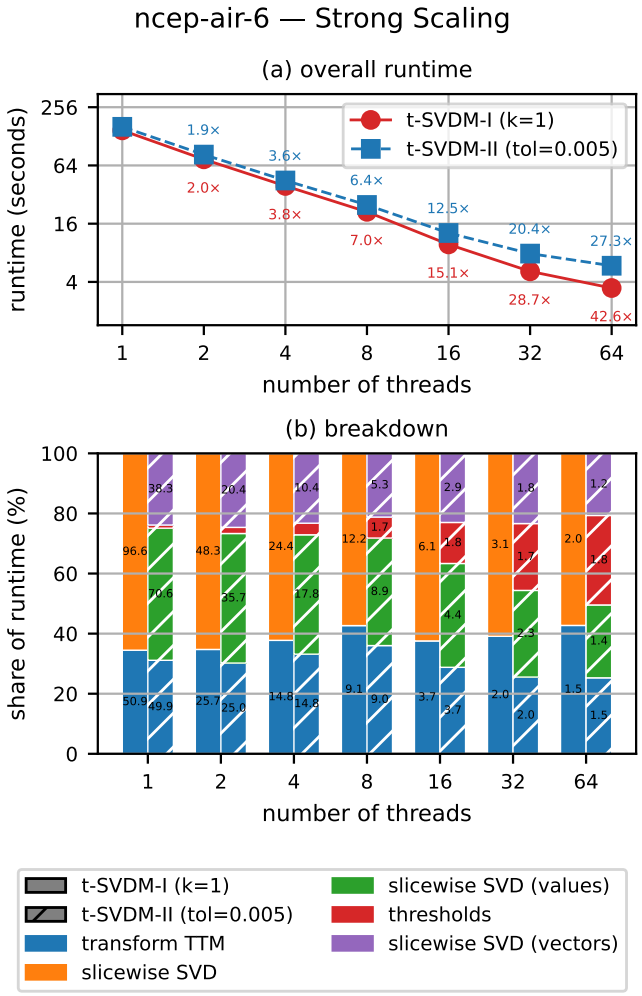


Fig. 2. Strong scaling with the DCT transform.

[9] E. Kalnay, M. Kanamitsu, R. Kistler, W. Collins, D. Deaven, L. Gandin, M. Iredell, S. Saha, G. White, J. Woollen, Y. Zhu, M. Chelliah, W. Ebisuzaki, W. Higgins, J. Janowiak, K. C. Mo, C. Ropelewski, J. Wang, A. Leetmaa, R. Reynolds, R. Jenne, and D. Joseph, "The NCEP/NCAR 40-year reanalysis project," *Bulletin of the American Meteorological Society*, vol. 77, no. 3, pp. 437–472, 1996.

[10] A. Hannachi, I. T. Jolliffe, and D. B. Stephenson, "Empirical Orthogonal Functions and Related Techniques in Atmospheric Science: A Review," *International Journal of Climatology*, vol. 27, no. 9, pp. 1119–1152, 2007.