High-Performance Star-M SVD for Big Data Compression

Md Taufique Hussain[1] Grey Ballard[1] Aditya Devarakonda[1] Srinivas Eswar[2] Naman Pesricha[3] Vishwas Rao[2]

[1]Wake Forest University [2]Argonne National Laboratory [3]Indian Institute of Science

Tensor Decomposition Landscape

Method	Optimality	HPC Ready
Matrix SVD	Exact (Eckart–Young)	✓
CP	None	✓
Tucker	$\sqrt{d}\times$ optimal	✓
Tensor Train	$\sqrt{d}\times$ optimal	✓
$\star_{\mathrm{M}}$ t-SVD	Exact (Eckart–Young)	✗

Goal: compress GB–TB scientific data with provable optimality. $\star_{\mathrm{M}}$ t-SVD is exactly optimal, yet has no HPC implementation.

The Software Gap for $\star_{\mathrm{M}}$ t-SVD

Existing
MATLAB & pure Python

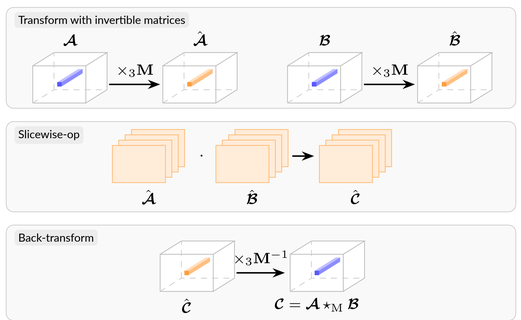
- ✗ Single-threaded
- ✗ Limited parallelism
- ✗ GB–TB data infeasible
- ✗ Small experiments only

`pystarm`
C++ with Intel MKL + Python

- ✓ Shared-memory parallel
- ✓ Utilizes all available cores
- ✓ GB–TB datasets
- ✓ Python interface

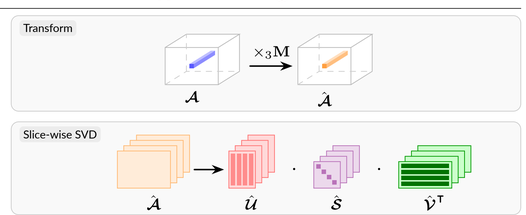
1000× faster than currently available solutions on a ≈3 GB Computational Fluid Dynamics (CFD) tensor

$\mathcal{A} \star_{\mathrm{M}} \mathcal{B}$: The $\star_{\mathrm{M}}$ Framework [1]



The choice of $\mathbf{M}$ shapes the transform domain — a good $\mathbf{M}$ concentrates energy and improves compression.

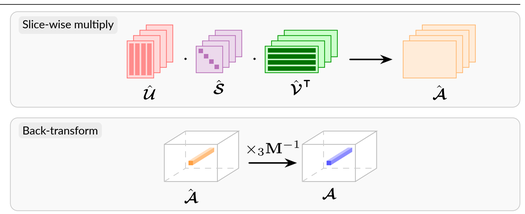
The $\star_{\mathrm{M}}$ t-SVD Algorithm



$\star_{\mathrm{M}}$ t-SVD = TTM + n **independent** matrix SVDs

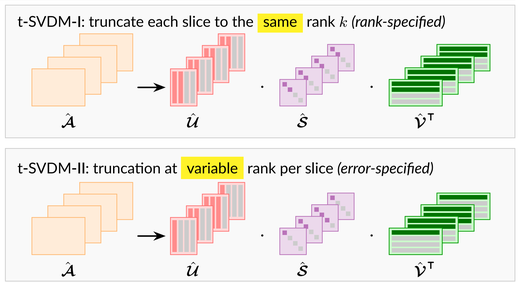
$$\hat{A}^{(i)} = \hat{U}^{(i)}\,\hat{S}^{(i)}\,\hat{V}^{(i)\mathsf{T}}, \quad i = 1, \ldots, n$$

$\star_{\mathrm{M}}$ t-SVD Reconstruction



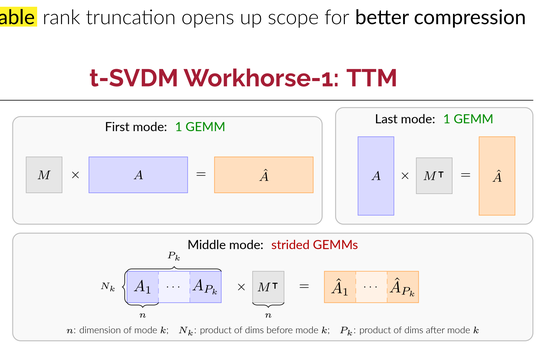
For d-way tensors: all modes beyond the 2nd are transformed — either mode by mode via individual TTMs, or equivalently via a single TTM with a Kronecker product transform. Slices are then treated identically.

Two Truncation Strategies



Variable rank truncation opens up scope for better compression

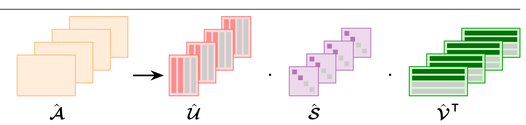
t-SVDM Workhorse-1: TTM



Possible designs (all three call Intel MKL)

- *loop* — loop over multithreaded GEMM
- *parfor* — parallel-for over sequential GEMM
- *batched* — MKL batched strided GEMM

t-SVDM Workhorse-2: Slice-wise SVD

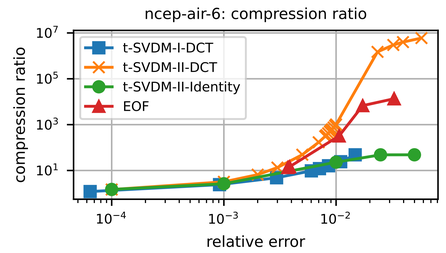


Possible designs

- **parallel slices, sequential SVD** — embarrassingly parallel, no dependencies between slices
- **sequential slices, parallel SVD**
- **a mix** — batched SVD, available in recent MKL but not as reliable

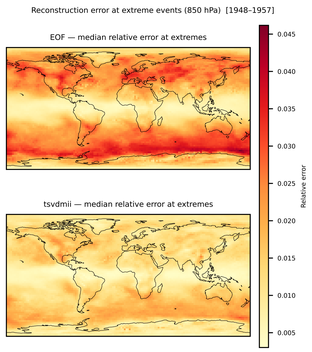
Compression: Climate Reanalysis Data

`ncep-air-6`: $73 \times 144 \times 17 \times 4 \times 365 \times 10$ (≈ 19.5 GB), $\mathbf{M}$: DCT



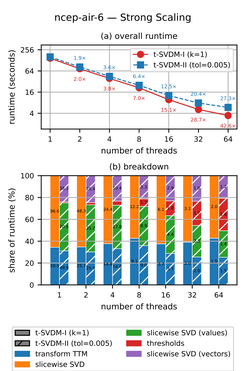
- **Tensor vs. matrix** — tensor methods outperform EOF (Empirical Orthogonal Functions), the standard in climate science
- **t-SVDM-II vs. t-SVDM-I** — variable-rank truncation gives higher compression at every error level
- **DCT vs. identity** — DCT concentrates energy into fewer singular values per slice

Extreme Event Preservation: t-SVDM-II vs. EOF



Median pointwise relative error at extreme temperature events (top/bottom 0.5% at 850 hPa) at a **matched compression ratio of** ≈320×. t-SVDM-II-DCT achieves **lower error across most of the globe** — EOF is known to smooth extremes.

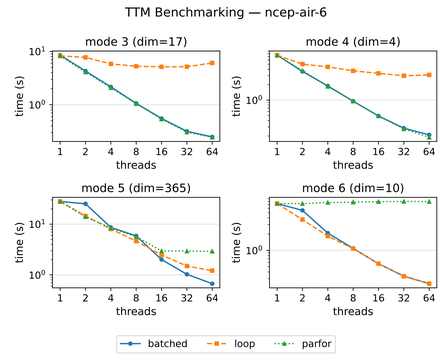
Strong Scaling: `ncep-air-6`



t-SVDM-II at tol 0.005 (CR ≈ 48.9) and t-SVDM-I at rank $k = 1$ (CR ≈ 48.2). (a) total runtime vs. thread count, with speedup annotated; (b) per-stage breakdown as a percentage of the total.

TTM Benchmark Results

`ncep-air-6`: $73 \times 144 \times 17 \times 4 \times 365 \times 10$

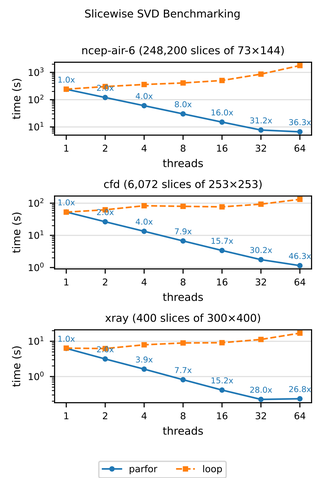


All variants call MKL; gaps follow from the batch count P_k:

- **mode 3:** 14,600 GEMMs of $10{,}512 \times 17$ — too small for multithreaded GEMM (*loop*)
- **mode 5:** $P_k = 10$ — *parfor* uses ≤10 threads
- **mode 6:** $P_k = 1$ — *parfor* is one sequential GEMM

Batched GEMM is the only variant close to the best in every mode.

Slice-wise SVD Benchmark Results



- **parallel slices, sequential SVD** — near-linear scaling up to 64 threads
- **sequential slices, parallel SVD** — does not scale well, especially for small slices

Conclusions and Future Work

- 64-thread speedup: t-SVDM-I 42×, t-SVDM-II 27×
- Tensor methods beat EOF and better preserve climate extremes
- Next: parallel t-SVDM-II threshold (≈25% of runtime at 64 threads), GPU, distributed memory
- Code: `https://github.com/Streaming-Tensors/pystarm/`

[1] M. E. Kilmer, L. Horesh, H. Avron, and E. Newman. Tensor-tensor algebra for optimal representation and compression of multiway data. *PNAS*, 118(28):e2015851118, 2021.

AI use: figures, plots, and layout assisted by Claude Opus 4.7 (Anthropic).