Parallel Rank-Adaptive Higher Order Orthogonal Iteration

PP26

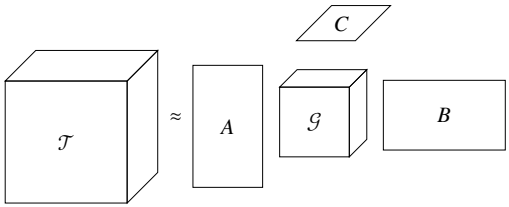
J Pinheiro[1], **Aditya Deverakonda**[2], **Grey Ballard**[2]

[1]**Purdue University**, [2]**Wake Forest University**

March 4, 2026

PURDUE UNIVERSITY | Department of Computer Science

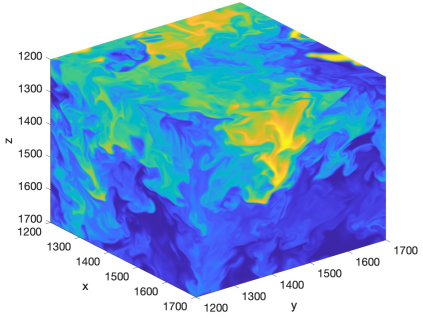
Tucker Compresses Datasets on Regular Grids



Tucker-Tensor Decomposition

d-way notation:

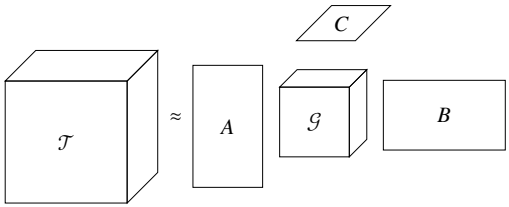
$$\mathcal{T} \approx \mathcal{G} \times_1 \mathbf{U_1} \times_2 \mathbf{U_2} \times_3 \cdots \times_d \mathbf{U}_d$$



Miranda Dataset - 3D Fluid Flow Simulation Snapshot

[**JP**, Deverakonda, Ballard, ACM SC 2025 Parallel Rank-Adaptive Higher Order Orthogonal Iteration]

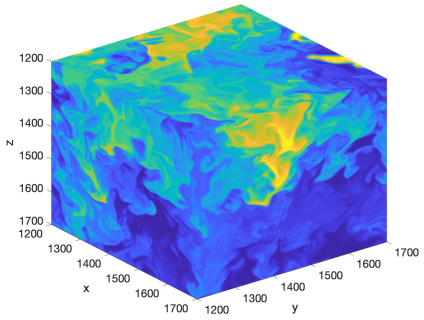
Tucker Compresses Datasets on Regular Grids



Tucker-Tensor Decomposition

d-way notation:

$$\mathcal{T} \approx \mathcal{G} \times_1 \mathbf{U_1} \times_2 \mathbf{U_2} \times_3 \cdots \times_d \mathbf{U}_d$$



Miranda Dataset - 3D Fluid Flow Simulation Snapshot

- In **rank-specified** formulation, user specifies core size and compression is known beforehand
- In **error-specified** formulation, user specifies maximum relative error threshold

[**JP**, Deverakonda, Ballard, ACM SC 2025 Parallel Rank-Adaptive Higher Order Orthogonal Iteration]

Tucker Algorithms

- The **Sequentially Truncated Higher-Order Singular Value Decomposition (STHOSVD)** is an algorithm to compute the Tucker approximation of a tensor.
  - **Higher-Order Orthogonal Iteration (HOOI)** was used only to refine the result given by STHOSVD.

[**JP**, Deverakonda, Ballard, ACM SC 2025 Parallel Rank-Adaptive Higher Order Orthogonal Iteration]

Tucker Algorithms

- The **Sequentially Truncated Higher-Order Singular Value Decomposition (STHOSVD)** is an algorithm to compute the Tucker approximation of a tensor.
  - **Higher-Order Orthogonal Iteration (HOOI)** was used only to refine the result given by STHOSVD.

- STHOSVD can be either rank-specified or error-specified whereas regular HOOI is only rank-specified.

[**JP**, Deverakonda, Ballard, ACM SC 2025 Parallel Rank-Adaptive Higher Order Orthogonal Iteration]

Tucker Algorithms

- The **Sequentially Truncated Higher-Order Singular Value Decomposition (STHOSVD)** is an algorithm to compute the Tucker approximation of a tensor.
  - **Higher-Order Orthogonal Iteration (HOOI)** was used only to refine the result given by STHOSVD.
- STHOSVD can be either rank-specified or error-specified whereas regular HOOI is only rank-specified.
- We argue that HOOI can be better than STHOSVD when ranks are small
  - This requires a new technique for **rank adaptivity** to satisfy error tolerance
  - Also requires performance optimization using **memoization** and **iterative SVD**

[**JP**, Deverakonda, Ballard, ACM SC 2025 Parallel Rank-Adaptive Higher Order Orthogonal Iteration]

Tucker Algorithms

- The **Sequentially Truncated Higher-Order Singular Value Decomposition (STHOSVD)** is an algorithm to compute the Tucker approximation of a tensor.
  - **Higher-Order Orthogonal Iteration (HOOI)** was used only to refine the result given by STHOSVD.
- STHOSVD can be either rank-specified or error-specified whereas regular HOOI is only rank-specified.
- We argue that HOOI can be better than STHOSVD when ranks are small
  - This requires a new technique for **rank adaptivity** to satisfy error tolerance
  - Also requires performance optimization using **memoization** and **iterative SVD**
- We demonstrate speed and compression improvements for physical simulation datasets
  - HOOI converges to comparable accuracy of STHOSVD in 1-2 iterations
  - At times it can achieve more compact Tucker approximations satisfying error tolerance

[**JP**, Deverakonda, Ballard, ACM SC 2025 Parallel Rank-Adaptive Higher Order Orthogonal Iteration]

Tucker Algorithms

Algorithm STHOSVD

1: **function** $[\mathcal{G}, \{\mathbf{U}_i\}]$ =STHOSVD($\mathcal{T}, \mathbf{r}$)
2: $\quad \mathcal{G} = \mathcal{T}$
3: $\quad$ **for** $j = 1 : d$ **do**
4: $\quad\quad \mathbf{U}_j$ = leading r_j left sing. vecs. of $\mathbf{G}_j$
5: $\quad\quad \mathcal{G} = \mathcal{G} \times_j \mathbf{U}_j'$
6: $\quad$ **end for**
7: **end function**

- easily adapted to error specified case
- direct algorithm giving quasi-optimal solution
- dominant flop cost comes from SVD of $\mathbf{T}_1$

Algorithm HOOI

1: **function** $[\mathcal{G}, \{\mathbf{U}_j\}]$ = HOOI($\mathcal{T}, \mathbf{r}$)
2: $\quad$ initialize $\{\mathbf{U}_j\}$ randomly
3: $\quad$ **repeat**
4: $\quad\quad$ **for** $j = 1 : d$ **do**
5: $\quad\quad\quad \mathcal{Y} = \mathcal{T} \times_{-j} \{\mathbf{U}_k'\}$ ▷ Multi-TTM
6: $\quad\quad\quad \mathbf{U}_j$ = leading r_j left sing. vecs. of $\mathbf{Y}j$
7: $\quad\quad$ **end for**
8: $\quad$ **until** convergence
9: $\quad \mathcal{G} = \mathbf{Y} \times_d \mathbf{U}_d'$
10: **end function**

- originally designed for rank-specified case
- iterative algorithm converging to local min
- dominant flop cost from Multi-TTM

[**JP**, Deverakonda, Ballard, ACM SC 2025 Parallel Rank-Adaptive Higher Order Orthogonal Iteration]

Tucker Algorithms

Algorithm STHOSVD

function $[\mathcal{G}, \{\mathbf{U}_i\}]$ =STHOSVD($\mathcal{T}$, $\mathbf{r}$)
  $\mathcal{G} = \mathcal{T}$
  for $j = 1 : d$ **do**
    $\mathbf{U}_j$ = leading r_j left sing. vecs. of $\mathbf{G}_j$
    $\mathcal{G} = \mathcal{G} \times_j \mathbf{U}_j'$
  end for
end function

Algorithm HOOI

function $[\mathcal{G}, \{\mathbf{U}_j\}]$ = HOOI($\mathcal{T}$, $\mathbf{r}$)
  initialize $\{\mathbf{U}_j\}$ randomly
  repeat
    for $j = 1 : d$ **do**
      $\mathcal{Y} = \mathcal{T} \times_{-j} \{\mathbf{U}_k'\}$ ▷ Multi-TTM
      $\mathbf{U}_j$ = leading r_j left sing. vecs. of $\mathbf{Y}j$
    end for
  until convergence
  $\mathcal{G} = \mathbf{Y} \times_d \mathbf{U}_d'$
end function

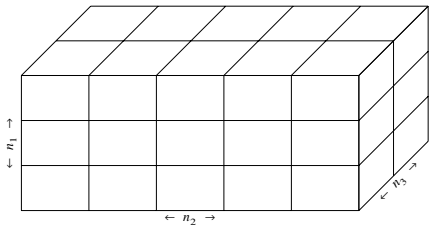
- Observation/Motivation: the cost of a HOOI iteration is less than STHOSVD when ranks are small
  - Dominant cost of STHOSVD (SVD of $\mathbf{T}_{(1)}$) is $O(n^{d+1})$ to compute the Gram matrix
  - Dominant cost of HOOI (Multi-TTMs) is $O(4n^d/r)$ (spoiler: this is with the *dimension trees* optimization)
  - This implies that 2 iterations of HOOI is cheaper when $n/r > 8$

[**JP**, Deverakonda, Ballard, ACM SC 2025 Parallel Rank-Adaptive Higher Order Orthogonal Iteration]

TuckerMPI and Parallel Tensor Distribution

- Existing C++/MPI library
- Implements deterministic STHOSVD
- Has efficient sequential and parallel kernels for SVD and TTM

For d-way tensor, we use d-way processor grid with Cartesian block distribution



Example: $p_1 \times p_2 \times p_3 = 3 \times 5 \times 2$

Local tensor size: $\frac{n_1}{p_1} \times \frac{n_2}{p_2} \times \frac{n_3}{p_3}$

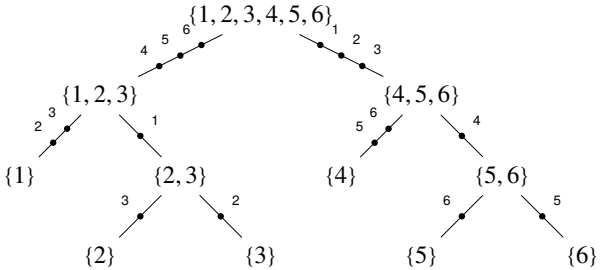
[**JP**, Deverakonda, Ballard, ACM SC 2025 Parallel Rank-Adaptive Higher Order Orthogonal Iteration]

Optimzation 1: Memoization of Multi-TTMs using Dimension Trees

Multi-TTMs in 4-way case

$$
\begin{aligned}
\mathcal{Y}^{(1)} &= \mathcal{T} \times_2 \mathbf{U}_2 \times_3 \mathbf{U}_3 \times_4 \mathbf{U}_4 \\
\mathcal{Y}^{(2)} &= \mathcal{T} \times_1 \mathbf{U}_1 \times_3 \mathbf{U}_3 \times_4 \mathbf{U}_4 \\
\mathcal{Y}^{(3)} &= \mathcal{T} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \times_4 \mathbf{U}_4 \\
\mathcal{Y}^{(4)} &= \mathcal{T} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \times_3 \mathbf{U}_3
\end{aligned}
$$

Dimension Tree Example For 6-way case



Using dimension tree memoization reduces leading-order flop cost of Multi-TTMs from $2drn^d$ to $4rn^d$
a reduction of approximately $\frac{d}{2}$

Optimzation 2: Subspace Iteration to Reduce Flops and Increase Parallelism

- LLSV operation is done per dimension for STHOSVD, and per dimension per iteration for HOOI

function U = EIG_LLSV(Y, r or ϵ)
  $S = Y \cdot Y^\top$ ▷ MM
  $[U, \Lambda]$ = EIG(S) ▷ Sequential EIG
  return $U(:, 1:r)$
end function

- Cost of sequential EVD is $O(n_j^3)$ per iteration, which becomes a bottleneck when n_j is large

function U = SI_LLSV($\mathcal{Y}, U, n$)
  $\mathcal{G} = \mathcal{Y} \times_n U^\top$ ▷ TTM
  U = Contract($\mathcal{Y}, \mathcal{G}, n$) ▷ TTT
  $[U, \sim]$ = QR(U) ▷ With Column Pivoting
end function

- Using subspace iteration to compute the approximate principal subspace using matrix multiplications reduces the cost to $O(n_j^2 r_j)$ per iteration

[**JP**, Deverakonda, Ballard, ACM SC 2025 Parallel Rank-Adaptive Higher Order Orthogonal Iteration]

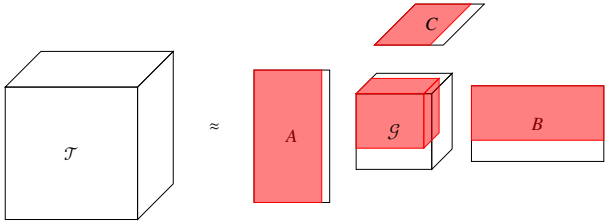
Optimzation 3: Transforming HOOI in a Rank-Adaptive Algorithm

- Key Facts we will take the advantage off
  - For orthonormal factor matrices, $\|\mathcal{T} - \mathcal{G} \times_1 \mathbf{U}_1 \times_2 \cdots \times_d \mathbf{U}_d\|^2 = \|\mathcal{T}\|^2 - \|\mathcal{G}\|^2$
  - Mass of core tensor $\mathcal{G}$ is concentrated in 1^{st} corner when factor matrices are ordered

[**JP**, Deverakonda, Ballard, ACM SC 2025 Parallel Rank-Adaptive Higher Order Orthogonal Iteration]

Optimzation 3: Transforming HOOI in a Rank-Adaptive Algorithm

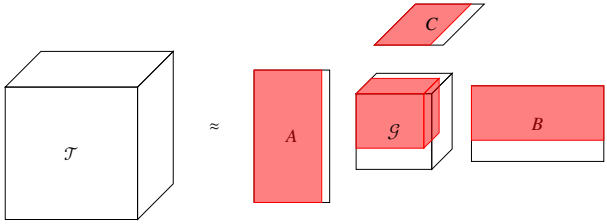
- Key Facts we will take the advantage off
  - For orthonormal factor matrices, $\|\mathcal{T} - \mathcal{G} \times_1 \mathbf{U}_1 \times_2 \cdots \times_d \mathbf{U}_d\|^2 = \|\mathcal{T}\|^2 - \|\mathcal{G}\|^2$
  - Mass of core tensor $\mathcal{G}$ is concentrated in 1^{st} corner when factor matrices are ordered
- When current approximation is sufficiently accurate (e.g. at the end of a HOOI iteration)
  - We contract the ranks to the smallest possible ranks that still satisfies error tolerance (as seen below)
  - This requires a multidimensional prefix sum to compute all errors efficiently



[**JP**, Deverakonda, Ballard, ACM SC 2025 Parallel Rank-Adaptive Higher Order Orthogonal Iteration]

Optimzation 3: Transforming HOOI in a Rank-Adaptive Algorithm

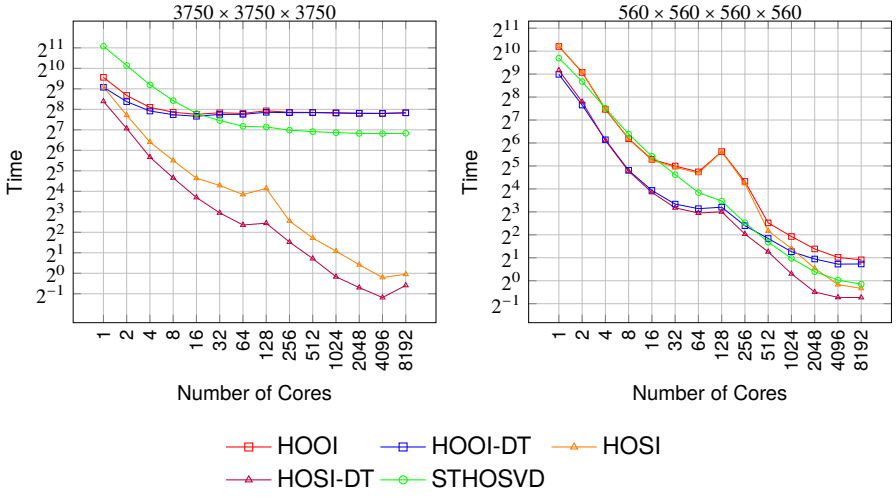
- Key Facts we will take the advantage off
  - For orthonormal factor matrices, $\|\mathcal{T} - \mathcal{G} \times_1 \mathbf{U}_1 \times_2 \cdots \times_d \mathbf{U}_d\|^2 = \|\mathcal{T}\|^2 - \|\mathcal{G}\|^2$
  - Mass of core tensor $\mathcal{G}$ is concentrated in 1^{st} corner when factor matrices are ordered
- When current approximation is sufficiently accurate (e.g. at the end of a HOOI iteration)
  - We contract the ranks to the smallest possible ranks that still satisfies error tolerance (as seen below)
  - This requires a multidimensional prefix sum to compute all errors efficiently



- If current approximation is not sufficiently accurate, we increase ranks by some factor for the next HOOI iteration

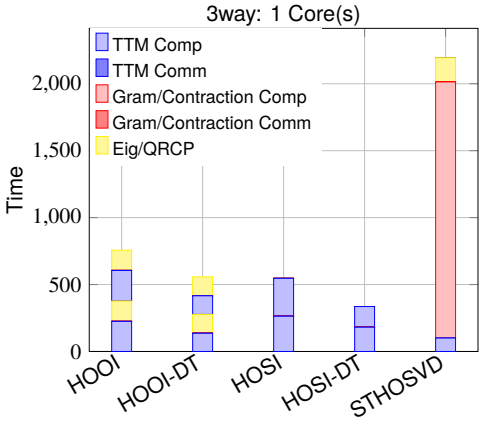
[**JP**, Deverakonda, Ballard, ACM SC 2025 Parallel Rank-Adaptive Higher Order Orthogonal Iteration]

Parallel Strong Scaling on Synthetic Tensors



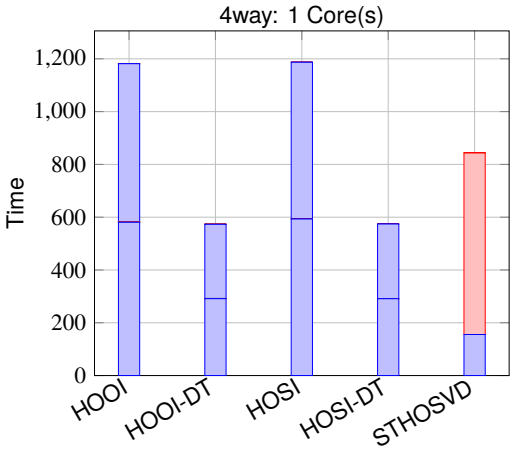
[**JP**, Deverakonda, Ballard, ACM SC 2025 Parallel Rank-Adaptive Higher Order Orthogonal Iteration]

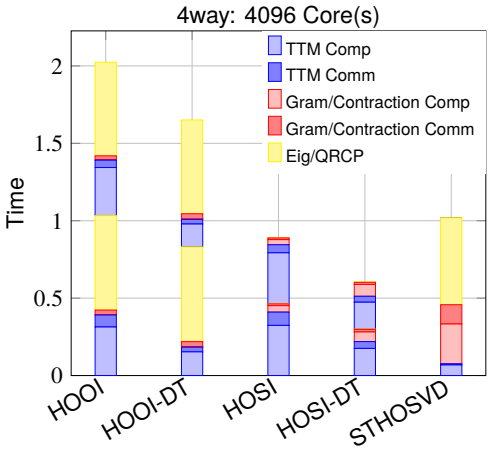
Breakdown of 3-way Synthetic Tensors



[**JP**, Deverakonda, Ballard, ACM SC 2025 Parallel Rank-Adaptive Higher Order Orthogonal Iteration]

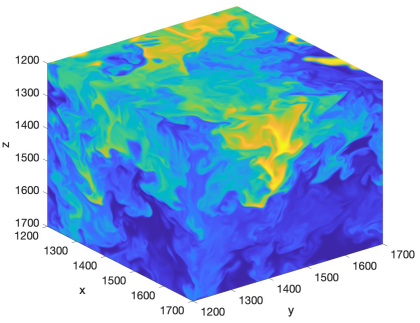
Breakdown of 4-way Synthetic Tensors



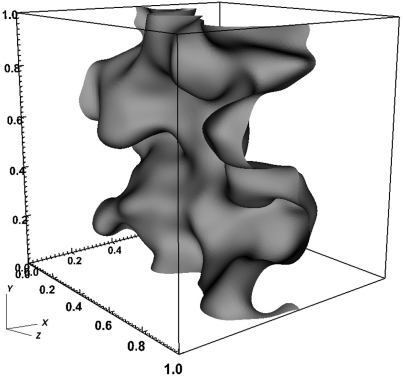


[**JP**, Deverakonda, Ballard, ACM SC 2025 Parallel Rank-Adaptive Higher Order Orthogonal Iteration]

Real Dataset Experiments



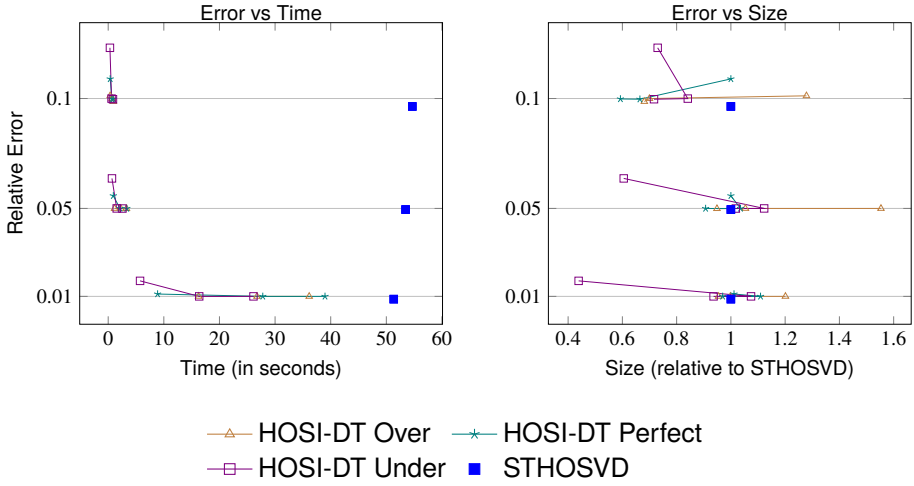
Miranda Dataset - 3D Fluid Flow Simulation Snapshot - $3072 \times 3072 \times 3072$ - 115GB



Stats Planner Dataset - 5D Statistically Planar Combustion Simulation Snapshot - $500 \times 500 \times 500 \times 11 \times 400$ - 4.4TB

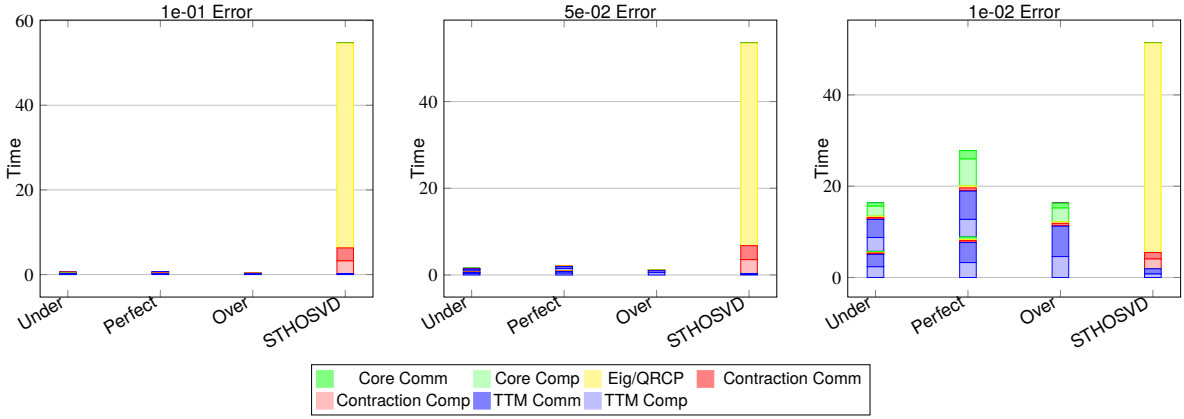
[**JP**, Deverakonda, Ballard, ACM SC 2025 Parallel Rank-Adaptive Higher Order Orthogonal Iteration]

Miranda Performance using 1024 Cores



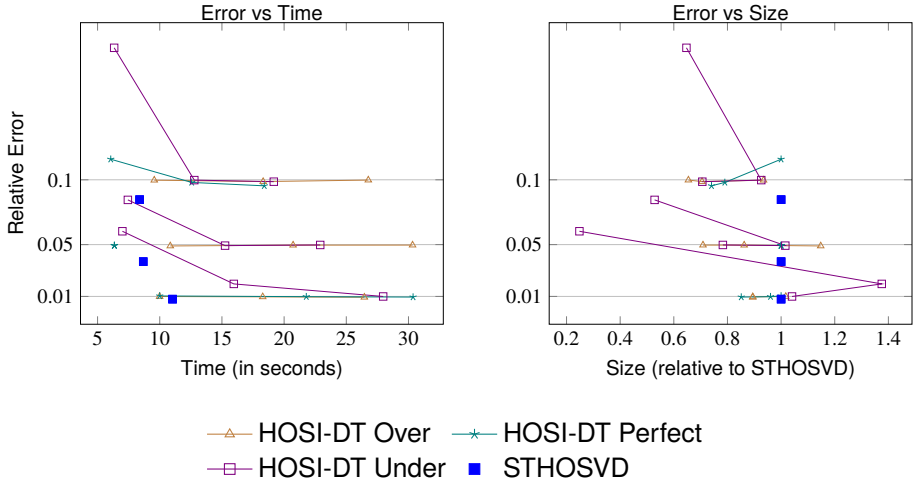
[**JP**, Deverakonda, Ballard, ACM SC 2025 Parallel Rank-Adaptive Higher Order Orthogonal Iteration]

Miranda Performance using 1024 Cores



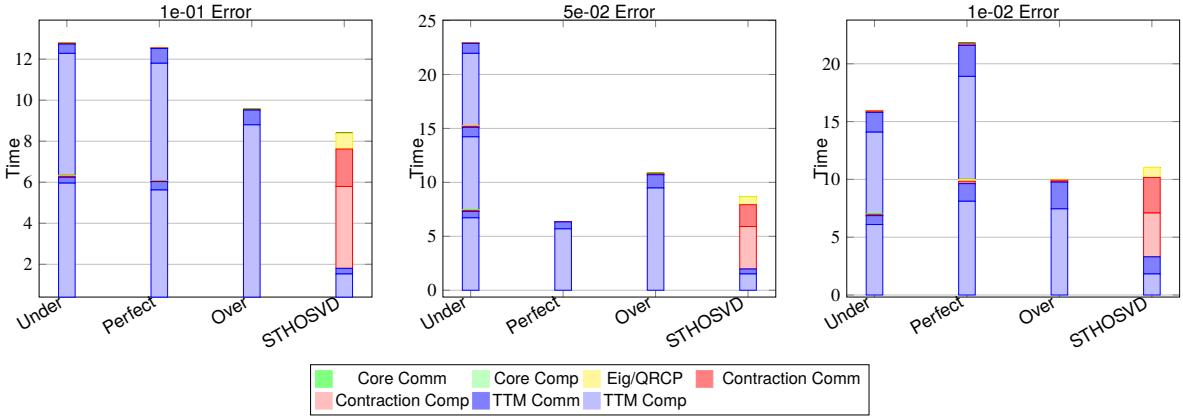
[**JP**, Deverakonda, Ballard, ACM SC 2025 Parallel Rank-Adaptive Higher Order Orthogonal Iteration]

SP Performance using 2048 Cores



[**JP**, Deverakonda, Ballard, ACM SC 2025 Parallel Rank-Adaptive Higher Order Orthogonal Iteration]

SP Performance using 2048 Cores



[**JP**, Deverakonda, Ballard, ACM SC 2025 Parallel Rank-Adaptive Higher Order Orthogonal Iteration]

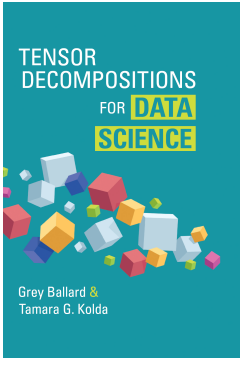
Conslusions and Shoutouts

- Adaptive-HOSI-DT combines three optimzations
  - Dimension Trees optimizes TTM operations
  - Subspace Iteration optimizes LLSV operations
  - AdaptiveHOOI removes rank-specified constraint
- We expect speedup over STHOSVD when $n/r > 8$
- Tighter compression is often observed

[**JP**, Deverakonda, Ballard, ACM SC 2025 Parallel Rank-Adaptive Higher Order Orthogonal Iteration]

Conslusions and Shoutouts

- Adaptive-HOSI-DT combines three optimzations
  - Dimension Trees optimizes TTM operations
  - Subspace Iteration optimizes LLSV operations
  - AdaptiveHOOI removes rank-specified constraint
- We expect speedup over STHOSVD when $n/r > 8$
- Tighter compression is often observed



tensortextbook.com

[**JP**, Deverakonda, Ballard, ACM SC 2025 Parallel Rank-Adaptive Higher Order Orthogonal Iteration]

Extra Slide

Tol	Miranda	HCCI	SP
0.1	$36 \times 15 \times 6$	$30 \times 25 \times 8 \times 10$	$5 \times 8 \times 6 \times 1 \times 2$
0.05	$130 \times 95 \times 50$	$50 \times 45 \times 12 \times 19$	$7 \times 13 \times 12 \times 2 \times 4$
0.01	$516 \times 497 \times 381$	$111 \times 105 \times 22 \times 46$	$15 \times 24 \times 24 \times 2 \times 9$
0	$3072 \times 3072 \times 3072$	$672 \times 672 \times 33 \times 626$	$500 \times 500 \times 500 \times 11 \times 400$

[**JP**, Deverakonda, Ballard, ACM SC 2025 Parallel Rank-Adaptive Higher Order Orthogonal Iteration]