OptiCPD: Optimization For The Canonical Polyadic Decomposition Algorithm on GPUs

Srinivasan Subramaniyan, Xiaorui Wang

The Ohio State University

May 15, 2023

Overview

What are Tensors?

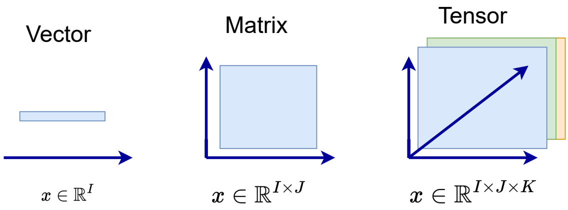


Figure 1: Tensor Representation across different modes

- Tensors are representations of multidimensional array.
- A first-order tensor is a vector.

What are Tensors?

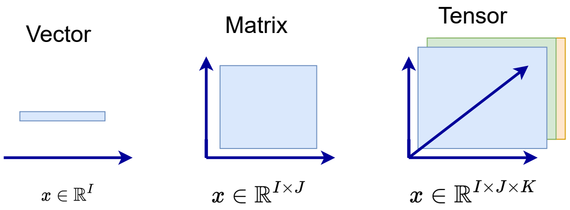


Figure 1: Tensor Representation across different modes

- Tensors are representations of multidimensional array.
- A first-order tensor is a vector.
- A second-order tensor is a matrix.
- Tensors of order three or higher are called higher-order tensors.

Where are Tensors used?

- Used in many applications like
  - Machine Learning
  - Recommend-er systems
  - Neural networks
  - Psychometric
  - Chemo-metrics & Fluid Mechanics

Tensor Annotations

Table 1: Tensor Elucidations

Representation	Elucidation
X	Tensor
M	Matrix
R	Rank
N	Tensor Order
v	Vector
X_{ijk}	Tensor in i,j,k dimensions
S	Slices
F	Fibres

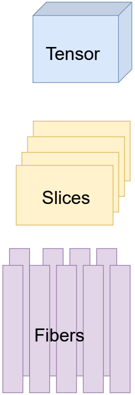


Figure 2: Representation of a Tensor across different modes.

Matricization

- Matricization, also known as unfolding or flattening, is the process of reordering the elements of an n-dimensional array into a matrix.
- For instance, a $2 \times 3 \times 4$ tensor can be arranged as a 6×4 matrix or a 3 x 8 matrix.
- The mode n matricization of a tensor $X \in R^{I1 \times I2 \times I3}$ is represented as X_n.

$$X(::1) = \begin{bmatrix} 2 & 4 \\ 3 & 5 \end{bmatrix}$$

$$X(::2) = \begin{bmatrix} 6 & 8 \\ 7 & 9 \end{bmatrix}$$

$$X_1 = \begin{bmatrix} 2 & 4 & 6 & 8 \\ 3 & 5 & 7 & 9 \end{bmatrix}$$

$$X_2 = \begin{bmatrix} 2 & 3 & 6 & 7 \\ 4 & 5 & 8 & 9 \end{bmatrix}$$

$$X_3 = \begin{bmatrix} 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 \end{bmatrix}$$

Kronecker Product

- The Kronecker product of matrices $A \in \mathbb{R}^{I \times J}$ and $B \in \mathbb{R}^{K \times L}$ is denoted by $A \otimes B$. The resultant matrix is of the size $(IK) \times (JL)$.

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \dots & a_{1J}B \\ a_{21}B & a_{22}B & \dots & a_{2J}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{I1}B & a_{I2}B & \dots & a_{IJ}B \end{bmatrix}$$

or equivalently,

$$A \otimes B = \begin{bmatrix} a_1 \times b_1 & a_1 \otimes b_2 & \dots & a_J \otimes b_{L-1} & a_J \otimes b_L \end{bmatrix}$$

Example of Kronecker Product

Given two matrices A and B the Kronecker Product for them is defined below.

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \quad B = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$$

$$A \otimes B = \begin{bmatrix} 1 \cdot 5 & 1 \cdot 6 & 2 \cdot 5 & 2 \cdot 6 \\ 1 \cdot 7 & 1 \cdot 8 & 2 \cdot 7 & 2 \cdot 8 \\ 3 \cdot 5 & 3 \cdot 6 & 4 \cdot 5 & 4 \cdot 6 \\ 3 \cdot 7 & 3 \cdot 8 & 4 \cdot 7 & 4 \cdot 8 \end{bmatrix}$$

Hadamard Product

- The Hadamard product is the element-wise matrix product. Given matrices A and B, both of size $I \times J$, their Hadamard product is denoted by $A \odot B$.

$$A \odot B = \begin{bmatrix} a_{11}b_{11} & a_{12}b_{12} & \dots & a_{1J}b_{1J} \\ a_{21}b_{21} & a_{22}b_{22} & \dots & a_{2J}b_{2J} \\ \vdots & \vdots & \ddots & \vdots \\ a_{I1}b_{I1} & a_{I2}b_{I2} & \dots & a_{IJ}b_{IJ} \end{bmatrix}$$

Example for Hadamard Product

Suppose we have matrices A and B, where:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}, \; B = \begin{bmatrix} 7 & 8 & 9 \\ 10 & 11 & 12 \end{bmatrix}$$

$$A \odot B = \begin{bmatrix} 1 \cdot 7 & 2 \cdot 8 & 3 \cdot 9 \\ 4 \cdot 10 & 5 \cdot 11 & 6 \cdot 12 \end{bmatrix} = \begin{bmatrix} 7 & 16 & 27 \\ 40 & 55 & 72 \end{bmatrix}$$

Khatri Rao Product

- The Khatri-Rao product is the "matching column-wise" Kronecker product.
- If a and b are vectors, then the Khatri-Rao and Kronecker products are identical $a \otimes b = a \odot b$.
- Given matrices $A \in \mathbb{R}^{I \times K}$ and $B \in \mathbb{R}^{J \times K}$, their Khatri-Rao product is denoted by $A \odot B$. The result is a matrix of size $(IJ) \times K$ and defined by $\begin{bmatrix} a_1 \otimes b_1 \\ a_2 \otimes b_2 \\ \vdots \\ a_K \otimes b_K \end{bmatrix}$

Example of Khatri Rao Product

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \quad B = \begin{bmatrix} 7 & 8 \\ 9 & 10 \\ 5 & 6 \end{bmatrix}$$

$$A \odot B = \begin{bmatrix} 7 \cdot 1 & 8 \cdot 2 \\ 7 \cdot 3 & 8 \cdot 4 \\ 9 \cdot 1 & 10 \cdot 2 \\ 9 \cdot 3 & 10 \cdot 4 \\ 5 \cdot 1 & 6 \cdot 2 \\ 5 \cdot 3 & 6 \cdot 4 \end{bmatrix} = \begin{bmatrix} 7 & 16 \\ 21 & 32 \\ 9 & 20 \\ 27 & 40 \\ 5 & 12 \\ 15 & 24 \end{bmatrix}$$

MTTKRP

- Mode-0 MTTKRP: $G_{i,r} = \sum_{j=1}^{J} \sum_{k=1}^{K} X_{ijk} V_{jr} W_{kr}$
- Mode-1 MTTKRP: $G_{j,r} = \sum_{i=1}^{I} \sum_{k=1}^{K} X_{ijk} U_{ir} W_{kr}$
- Mode-2 MTTKRP: $G_{k,r} = \sum_{i=1}^{I} \sum_{j=1}^{J} X_{ijk} U_{ir} V_{jr}$
- Here R is the rank of the matrix $1 \leq r \leq R$, and U_{ir}, V_{jr}, and W_{kr} are the factor matrices for mode-0, mode-1, and mode-2, respectively.

HIP Graphs

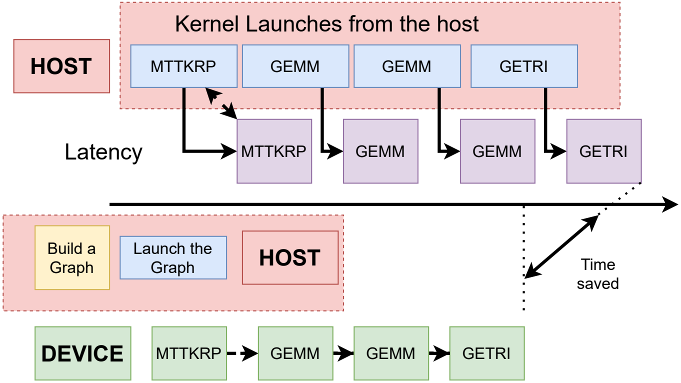


Figure 3: Comparision of Hip-Graphs vs Regular kernel launches.

- Graph launch submits all work at once, reducing CPU cost.
- Release CPU Time For Lower Power, or Running Other Work
- Efficient way to express dependency
- Reduce Launch latency

Overview

CPD-ALS

- CPD (Canonical Polyadic Decomposition) is different from other decomposition's.
- SVD (Singular Value Decomposition) can only be used if tensors are flattened to a matrix
- NMF (Non-negative matrix Factorization (NMF))is used for decomposing matrices and show a significant performance improvement for smaller matrices.
- CPD-PARAFAC ALS has the ability to perform decomposition even if some data samples are absent.

CPD-ALS (contd)

Algorithm 1 CPD-ALS Algorithm

Input Tensor: $X \in \mathbb{R}^{I \times J \times K}$

Dense Matrices :$A, B, C \in \mathbb{R}$

for *iter* $\leftarrow 1$ n **do**

$$\hat{A} = X_1(C \odot B)(B^T B * C^T C)^{\dagger}$$

$$\hat{B} = X_2(A \odot C)(A^T A * C^T C)^{\dagger}$$

$$\hat{C} = X_3(A \odot B)(A^T A * B^T B)^{\dagger}$$

Convergence of $\hat{A}$, $\hat{B}$ and $\hat{C}$.

- The CPD decomposes an Nth-order tensor into a sum of R rank-one tensors.
- The tensors can be decomposed as $X \approx \lambda_r a_r (\circ) b_r (\circ) c_r = [[\lambda; A, B, C]]$
- We compute the difference between the original tensor and the approximate value $||X - \hat{X}||$ in each iteration
- Continued till convergence or max iterations.

Mode 0 Analysis

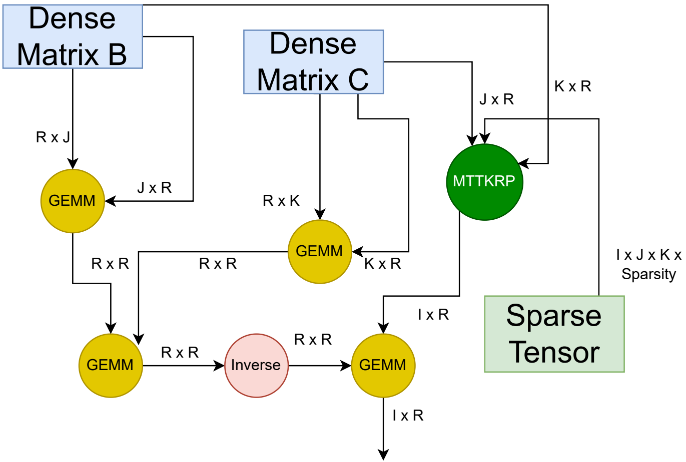
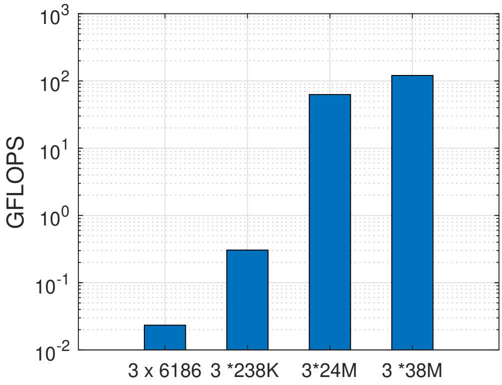


Figure 4: Mode 0 analysis of the CPD Decomposition.

- $T\_total = T\_mttkrp + T\_Gemm + T\_inverse$.
- Matrix multiplications:GEMMs.
- Inverse : LU Decomposition.
- MTTKRP: Custom Cuda kernel

Are GEMMs a bottleneck?



- GEMMs are usually performed by Vendor specific BLAS Libraries
- High GFLOPS $\leftarrow$ Regular matrix.
- Poor Performance for tall and wide matrices $A \in R^{I \times J}$ $I >> J$ or $J << I$
- There is no optimization specifically designed for different architectures.

Baseline

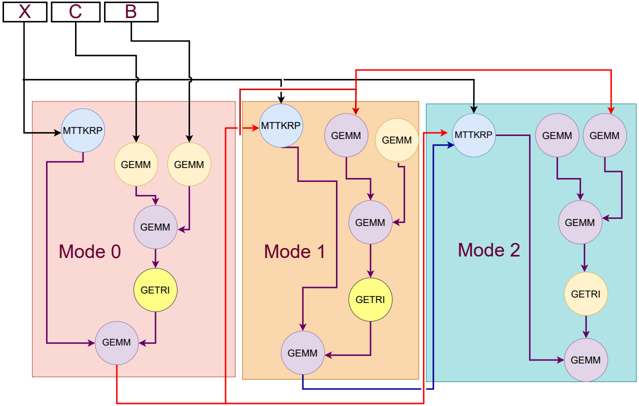


Figure 5: Baseline: Dataflow representation of the CPD/PARAFAC-ALS Algorithm using a third order tensor for a single iteration.

- Two GEMM operations must be computed for each mode.
- No reuse of partially computed GEMMs.
- For n iterations for a 3rd order tensor $\leftarrow 2n \times n$ GEMMs .

Optimization 1 & Optimization 2

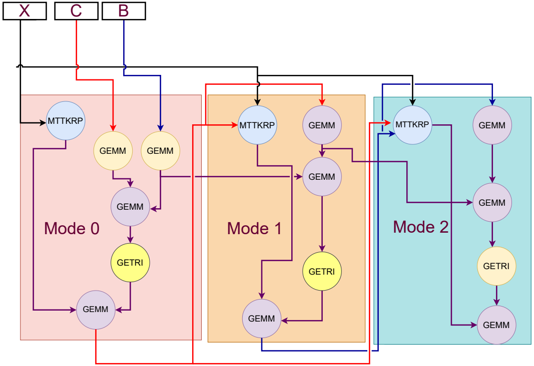


Figure 6: Optimization I

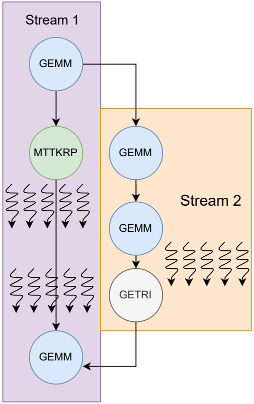


Figure 7: Optimization *II*

OptiCPD

Algorithm 2 OptiCPD algorithm

Input Tensor: $X \in \mathbb{R}^{I \times J \times K}$

for tensor in Dataset **do**

  if $\alpha >> 2.5$ **then** $(Optimization2)$;

    if $\alpha << 0.5$ **then** $(Optimization1)$;

      if $\alpha << 2.5$ and $\alpha >> 0.5$ **then**

        if $I >> J * K$ or $J >> I * K$ or $K >> J * I$ **then** $(Optimization2)$;

        else $(Optimization1)$;

- The choice of α was device specific and was specific to the device and the BLAS libraries.
- The value of alpha was determined by performing regression analysis on multiple tensors under various conditions.

Overview

Benchmark Tensors

Serial No	Tensor Name	Dimension	Size (GB)
I	Chicago	6186 x 24 x77 x 32	0.077
II	Enron	6066 x 5699 x 244268 x 1176	1.2
III	Nell-1	2902330 x 2143368 x 25495389	3.8
IV	Nell-2	12092 x 9184 x 28818	1.5
V	Nips	2482 x 2862 x 14036 x 17	0.057
VI	Darpa	22476 x 22476 x 23776223	0.575
VII	Freebase_music	23344784 x 23344784 x 166	2.0
VIII	Freebase_sampled	38955429 x 38955429 x 532	2.9
IX	Uber	183 x 24 x 1140 x 1717	0.052
X	Synthetic 1	200K x 80K x 16K	9.0
XI	Synthetic 2	400K x 80K x 8K	9.0
XII	Synthetic 3	800K x 40K x 8K	9.0
XIII	Synthetic 4	800K x 20K x 16K	9.0

Experimental Setup

- Intel(R) Xeon(R) Gold 5215 CPU running at 2.20GHz with the MI-100 GPU.
- ROCM stack 5.3.0
- The FROSTT benchmarks Smith et al. (2017), the tensors from the Haten dataset Jeon et al. (2015) Jeon et al. (2016) and certain synthetic tensors were used for the experiments.
- The synthetic tensors a generated using the Gaussian random process with a zero mean and variance one.
- The MI-100 GPU has a maximum DRAM capacity of 32 GB.
- The number of iterations was set to 5.

Experiment 1: Variation of execution time for all the techniques

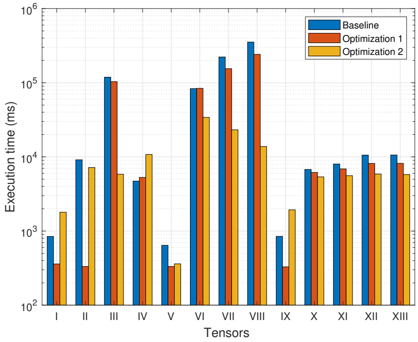


Figure 8: Variation of the overall execution time of the benchmark tensors for the CPD/PARAFAC-ALS for the baseline and the proposed optimization techniques.

- Optimization 1 shows good performance for benchmark tensors *I*, *II* *V* and *IX*.
- The use of hip-graphs allows for fine-grained task scheduling and parallelism.
- The delay caused by GEMM operations is masked by dividing the workload into smaller tasks and using dedicated streams for computation.

Experiment 2 :A detailed breakdown of execution time

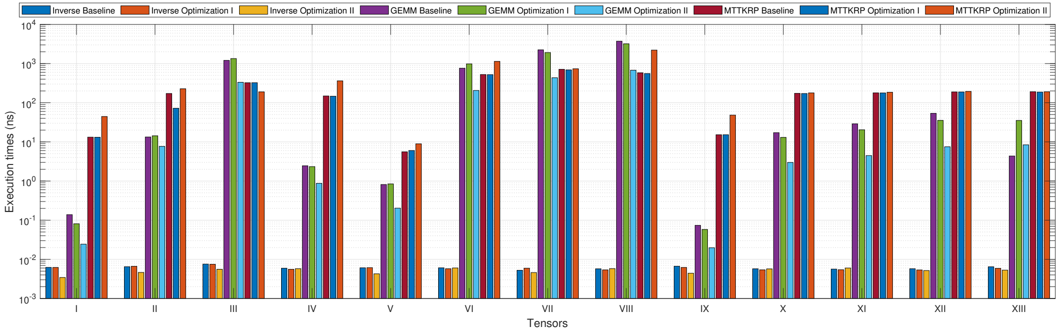


Figure 9: Variation of the execution time of the benchmark tensors for the three design techniques. The bar plot contains the split-up time for the Inverse, GEMM, and MTTKRP operation in the CPD/PARAFAC-ALS toolchain.

- It is to be noted that the GEMM operations are consuming a lot of GPU resources.
- GEMMs Performed using optimization 2 show less latency.

Experiment 3 :Performance Analysis of OptiCPD

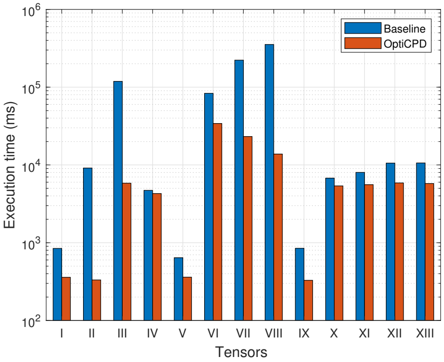


Figure 10: Variation of the overall execution time of the benchmark tensors for the baseline implementation and OptiCPD.

- OptiCPD achieves a speedup of more than 2.35x for tensor benchmark I,20.37x for tensor benchmark II.
- For Large tensors OptiCPD uses Optimization 2 to mask the latency caused by GEMM operation.
- For Small tensors OptiCPD performs better because less time is spent on the synchronization wait and the overhead caused by small streams.

Overview

Conclusion & Future Work

- OptiCPD achieved an average speedup of 7.5x.
- Planning to work on architectural optimization for improving CPD-ALS.
- Will investigate the division of work for CPD-ALS to CPUs and GPUs.

References I

Inah Jeon, Evangelos E. Papalexakis, U Kang, and Christos Faloutsos. Haten2: Billion-scale tensor decompositions. In *IEEE International Conference on Data Engineering (ICDE)*, 2015.

Inah Jeon, Evangelos E. Papalexakis, Christos Faloutsos, Lee Sael, and U Kang. Mining billion-scale tensors: Algorithms and discoveries. In *The International Journal on Very Large Data Bases (VLDB)*, 2016.

Shaden Smith, Jee W. Choi, Jiajia Li, Richard Vuduc, Jongsoo Park, Xing Liu, and George Karypis. FROSTT: The formidable repository of open sparse tensors and tools, 2017. URL http://frostt.io/.