PLANC: Parallel Low Rank Approximations with Non-negativity Constraints

Ramakrishnan Kannan, Michael Matheson, Grey Ballard, Srinivas Eswar, Koby Hayashi, Haesun Park

Ph.D student, School of CSE
Georgia Institute of Technology
Advisors: Rich Vuduc and Haesun Park

January 26, 2019

Workshop on Compiler Techniques for Sparse Tensor Algebra

Acknowledgement: This work was partly sponsored by NSF, Sandia and ORNL
Summary

- PLANC is an open source, scalable and flexible software package to compute Non-negative Tensor Factorisation.
Summary

- PLANC is an open source, scalable and flexible software package to compute Non-negative Tensor Factorisation.
- Implements state of the art communication avoiding algorithm for matricised-tensor times Khatri-Rao product (MTTKRP).

NTF is an important contributor towards explainable AI with a wide range of applications like spectral unmixing, scientific visualization, healthcare analytics, topic modelling etc.
PLANC is an open source, scalable and flexible software package to compute Non-negative Tensor Factorisation.

- Implements state of the art communication avoiding algorithm for matricised-tensor times Khatri-Rao product (MTTKRP).
- Popular optimisation methods like Block Principal Pivoting, Alternating Direction Method of Multipliers, first order Nesterov methods etc. for Non-negative Least Squares are included.
PLANC is an open source, scalable and flexible software package to compute Non-negative Tensor Factorisation.

Implements state of the art communication avoiding algorithm for matricised-tensor times Khatri-Rao product (MTTKRP).

Popular optimisation methods like Block Principal Pivoting, Alternating Direction Method of Multipliers, first order Nesterov methods etc. for Non-negative Least Squares are included.

NTF is an important contributor towards explainable AI with a wide range of applications like spectral unmixing, scientific visualization, healthcare analytics, topic modelling etc.
This is known as the CANDECOMP or PARAFAC or canonical polyadic or **CP decomposition**. It approximate tensors as sum of outer products or rank-1 tensors. **NNCP** imposes **non-negativity** constraints on the factor matrices to aid interpretability.
The MTTKRP is the major bottleneck for NNCP.

\[ M^{(1)} = X^{(1)}(W \odot V) \]

\[ m_{ir} = \sum_{j=1}^{J} \sum_{k=1}^{K} x_{ijk} v_{jr} w_{kr} \]

Standard approach is to explicitly matricise the tensor and form the Khatri-Rao product before calling DGEMM.

Can we do better? Avoid matricisation of the tensor and full Khatri-Rao products ...
Following the nested arrays lower bounds [BKR18].

**Theorem**

Any parallel MTTKRP algorithm involving a tensor with \( l_k = l^{1/N} \) for all \( k \) and that evenly distributes one copy of the input and output performs at least

\[
\Omega \left( \left( \frac{NIR}{P} \right)^{\frac{N}{2N-1}} + NR \left( \frac{l}{P} \right)^{1/N} \right)
\]

sends and receives. (Either term can dominate.)

- **Key Assumptions:** algorithm is not allowed to pre-compute and re-use temporary values.
- \( \Omega \left( NR \left( \frac{l}{P} \right)^{1/N} \right) \) is the most frequently occurring case for relatively small \( P \) or \( R \).
Reusable computations across MTTKRP.

\[ M^{(1)} = X_{(1)}(U^{(3)} \odot U^{(2)}) \quad M^{(2)} = X_{(2)}(U^{(3)} \odot U^{(1)}) \]

Utilise a “dimension tree” to store and reuse partial products [PTC13, LKL+17, HBJT18].

\[ \{1, 2, 3\} \]
\[ \{1, 2\} \]
\[ \text{mTTV/} \quad \text{mTTV} \]
\[ \text{PM/} \quad \text{PM} \]
\[ \text{PM} \]
\[ M^{(1)} \quad M^{(2)} \]
\[ M^{(3)} \]

PM = partial MTTKRP

mTTV = multi-Tensor-Times-Vector
Distributed Memory Optimisation - Communication Avoiding

Each processor

1. Starts with one subtensor and subset of rows of each input factor matrix

2. All-Gathers all the rows needed from $U^{(1)}$

3. All-Gathers all the rows needed from $U^{(3)}$

4. Computes its contribution to rows of $M^{(2)}$ (local MTTKRP)

5. Reduce-Scatters to compute and distribute $M^{(2)}$ evenly

6. Solve local NLS problem using $M^{(2)}$ and $U^{(2)}$
Distributed Memory Optimisation - Communication Avoiding

Each processor
1. Starts with one subtensor and subset of rows of each input factor matrix
2. All-Gathers all the rows needed from $U^{(1)}$
3. All-Gathers all the rows needed from $U^{(3)}$
4. Computes its contribution to rows of $M^{(2)}$ (local MTTKRP)
5. Reduce-Scatters to compute and distribute $M^{(2)}$ evenly
6. Solve local NLS problem using $M^{(2)}$ and $U^{(2)}$
Distributed Memory Optimisation - Communication Avoiding

Each processor

1. Starts with one subtensor and subset of rows of each input factor matrix
2. All-Gathers all the rows needed from $U^{(1)}$
3. All-Gathers all the rows needed from $U^{(3)}$

- $U^{(1)}$
- $U^{(3)}$
- $M^{(2)}$
Each processor

1. Starts with one subtensor and subset of rows of each input factor matrix
2. All-Gathers all the rows needed from \( U^{(1)} \)
3. All-Gathers all the rows needed from \( U^{(3)} \)
4. Computes its contribution to rows of \( M^{(2)} \) (local MTTKRP)
Distributed Memory Optimisation - Communication Avoiding

Each processor

1. Starts with one subtensor and subset of rows of each input factor matrix

2. All-Gathers all the rows needed from $U^{(1)}$

3. All-Gathers all the rows needed from $U^{(3)}$

4. Computes its contribution to rows of $M^{(2)}$ (local MTTKRP)
Distributed Memory Optimisation - Communication Avoiding

Each processor

1. Starts with one subtensor and subset of rows of each input factor matrix
2. All-Gathers all the rows needed from $U^{(1)}$
3. All-Gathers all the rows needed from $U^{(3)}$
4. Computes its contribution to rows of $M^{(2)}$ (local MTTKRP)
5. Reduce-Scatters to compute and distribute $M^{(2)}$ evenly
Distributed Memory Optimisation - Communication Avoiding

Each processor
1. Starts with one subtensor and subset of rows of each input factor matrix
2. All-Gathers all the rows needed from $U^{(1)}$
3. All-Gathers all the rows needed from $U^{(3)}$
4. Computes its contribution to rows of $M^{(2)}$ (local MTTKRP)
5. Reduce-Scatters to compute and distribute $M^{(2)}$ evenly
6. Solve local NLS problem using $M^{(2)}$ and $U^{(2)}$
Can achieve **nearly linear** scaling since NNCP is compute bound.
Performance Plots - CPU vs GPU

Figure: CPU

Figure: GPU

4D synthetic tensor of dimensions $384 \times 384 \times 384 \times 384$ on 81 Titan nodes as a $3 \times 3 \times 3 \times 3$ grid with varying low rank.

Offloading DGEMM calls to GPU can provide 7X speedup.
Compiler Challenges and Extensions to the Sparse Setting

1. Dimension Tree ordering.
   - Combinatorial explosion for sparse case (contrasted with the single split choice for the dense case).
   - Sparse case involves growth in intermediate values.

2. Communication Pattern establishment and load balancing.
   - Automatic communicator setup given a processor grid and tensor operation.
   - Automatic data distribution using the communication-avoiding loop optimisation [Kni15, DR16].

   - Active Set orderings can be grouped in an embarrassingly parallel call.
   - Sparse case with masking matrix has a similar RHS pattern.

4. Binary Bloat
   - Separate binaries for GPU/CPU and Sparse/Dense.
Summary

- PLANC is an open source, scalable and flexible software package to compute Non-negative Tensor Factorisation.
- Implements state of the art communication avoiding algorithm for matricised-tensor times Khatri-Rao product (MTTKRP).
- Popular optimisation methods like Block Principal Pivoting, Alternating Direction Method of Multipliers, first order Nesterov methods etc. for Non-negative Least Squares are included.
- NTF is an important contributor towards explainable AI with a wide range of applications like spectral unmixing, scientific visualization, healthcare analytics, topic modelling etc.
Grey Ballard, Nicholas Knight, and Kathryn Rouse.  
Communication lower bounds for matricized tensor times khatri-rao product.  
abs/1708.07401:Accepted, 2018.

James Demmel and Alex Rusciano.  
Parallelepipedes obtaining HBL lower bounds.  

Koby Hayashi, Grey Ballard, Yujie Jiang, and Michael J. Tobia.  
Shared-memory parallelization of MTTKRP for dense tensors.  

Nicholas Sullender Knight.  
Communication-Optimal Loop Nests.  

Athanasios P Liavas, Georgios Kostoulas, Georgios Lourakis, Kejun Huang, and Nicholas D Sidiropoulos.  
Nesterov-based parallel algorithm for large-scale nonnegative tensor factorization.  

Anh-Huy Phan, Petr Tichavsky, and Andrzej Cichocki.  
Fast alternating LS algorithms for high order CANDECOMP/PARAFAC tensor factorizations.  
For given rank $R$ we formulate NNCP as the following optimisation problem,

$$\min_{u, v, w \geq 0} \left\| X - \sum_{r=1}^{R} \lambda_r u_r \odot v_r \odot w_r \right\|$$

- Non-linear and non-convex problem.
- Solve via Alternating Non-negative Least Squares in an iterative manner using Block Coordinate Descent.
Alternating Non-negative Least Squares (ANLS)

- Fixing all factor matrices but one results in a linear NLS problem,

\[
\min_{U \geq 0} \left\| X - \sum_{r=1}^{R} u_r \odot \hat{v}_r \odot \hat{w}_r \right\|
\]

or equivalently,

\[
\min_{U \geq 0} \left\| X(1) - U(\hat{W} \odot \hat{V})^T \right\|_F
\]

- \( \odot \) is the Khatri-Rao product (column-wise Kronecker Product) of the factor matrices.

- Utilising the identity \((A \odot B)^T(A \odot B) = A^T A \ast B^T B\), we can cast the above problem as NLS solves via normal equations.

\[
\min_{U \geq 0} \left\| X(1)(\hat{W} \odot \hat{V}) - U \left( \hat{W}^T \hat{W} \ast \hat{V}^T \hat{V} \right) \right\|_F
\]

where \( \ast \) is the Hadamard Product.