Design and Implementation of Module of Tensor Decompositions based on OpenCL

ZHOU Qi, CHAI Xiao Li, MA Ke Jie, YU Ze Ren
(The 32nd Research Institute of China Electronics Technology Group Corporation. Shanghai Jiading, 201808)

Abstract. Tensor is a data aggregate that can expand data linear relationship from low order to high order. Tensor don’t need to dimension reduction when tensor was analyzed. In this way, Tensor, this new concept, was widely used in the field of computer science in recent years. Tensor decompositions, as the mainly diagnostic tool of tensor, is widely applied in data mining, machine learning, etc. There are two basic tensor decompositions: CANDECOMP/PARAFAC Decompositions (abbreviated as CP Decompositions) and Tucker Decompositions. They all have a lot of iterations and matrix multiplications, which is not suitable to operator on CPU platform. In this paper, we design a module for Tensor Decompositions based on OpenCL. Considering that OpenCL platform is good at iterations and matrix multiplications. This module will highly improve the effectivity of Tensor Decompositions.

1. Introductions
Tensor is originally a notion from mechanical system. It is used to represent stress and strain state in nonlinear continuum. Now, It is a data aggregate that show data linear relationship. Because it can expends from two orders to high order. It has been used for describing high order data in the field of machine learning, image processing, data mining, etc. Before that, these data have to do dimension reduction before being analyzed, which preserves the integrality of data.

Tensor decompositions are the main methods of analysis. It consists two basic tensor decompositions: CANDECOMP/PARAFAC Decompositions (abbreviated as CP Decompositions) and Tucker Decompositions. However, the process of tensor decompositions includes a lot of iterations and matrix multiplications. If the scale of matrix multiplication operation begins to reach a certain scale (such as the number of rows or columns exceeding 1000), the efficiency of tensor decomposition using CPU will be greatly reduced, which limits the application of Tensor decomposition in large-scale data.

In this paper, we use cBLAS, the BLAS library based on OpenCL, to design Tensor decompositions module. OpenCL is a framework that is used programming on heterogeneous system. This heterogeneous system consists of CPU, GPU, DPS, FPGA and other processors or hardware accelerators. We use cBLAS to write module because we need a Tensor decompositions module without platform constraints.

2. Introductions of Tensor

2.1 Notation and Preliminaries
Tensor & Tensor’s expression: Tensor is a multistage sequence composed of multiple vectors. This concept is different from the tensor in mechanical systems, where the tensor refers to the tensor in mathematical concepts. In this paper, Tensor's expression is written in a capital $X$. The element in tensor is represented by a lowercase $x$. The position of elements is represented by subscript. Assuming the three-dimensional tensor $X \in R^{i \times j \times k}$, the element expression in the tensor is $X_{ijk}$.

Tensor Unfolding: Unfolding, also known as flattening, is the process of transforming from an N-order tensor into a matrix.

Norm: The norm of a tensor $X \in R^{i \times j \times l \times m}$ is the square root of the sum of the squares of all its elements.

Tensor multiplication: Tensor multiplication, another name is the n-mode product, are much more complex than for matrices multiplication. If exist an N-way tensor $X \in R^{i \times j \times k}$ and a matrix $U \in R^{k \times l}$. The formula for the tensor multiplication is $X \times U$. At the same time, the scale of the result of tensor multiplication $X \times U$ is $I_1 \times I_2 \times \cdots \times I_n \times J_1 \times J_2 \times \cdots \times J_k$.

Kronecker product: If exists matrix $A$ and $B$. Their size are all $I \times J$. The Kronecker product is denoted by $A \otimes B$. The result is a matrix of size $(IK) \times (JL)$ like:

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \cdots & a_{1j}B \\ a_{21}B & a_{22}B & \cdots & a_{2j}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{i1}B & a_{i2}B & \cdots & a_{ij}B \end{bmatrix}$$

Khatari-Rao product: The Khatri–Rao product is the Kronecker product in special cases. Two of the matrix of this product have the same column. If exists $A \in R^{i \times j} \text{ and } B \in R^{k \times l}$. The Khatri–Rao product is denoted by $A \odot B$. The result is a matrix of size $(IJ) \times K$. The expression is $A \odot B = [a_1 \odot b_1 \ a_2 \odot b_2 \ \cdots \ a_i \odot b_j]$.

Hadamard product: If exists matrix $A$ and $B$. Their size are all $I \times J$. The Khatri–Rao product is denoted by $A \ast B$. The result is a matrix of size $I \times J$ like:

$$A \ast B = \begin{bmatrix} a_{11}b_{11} & a_{12}b_{12} & \cdots & a_{ij}b_{ij} \\ a_{12}b_{12} & a_{22}b_{22} & \cdots & a_{2j}b_{2j} \\ \vdots & \vdots & \ddots & \vdots \\ a_{ij}b_{ij} & a_{ij}b_{ij} & \cdots & a_{ij}b_{ij} \end{bmatrix}$$

There is also an equation related to tensor multiplication. If tensor $X \in R^{i_1 \times i_2 \times \cdots \times i_N}$ and $A_{(n)} \in R^{J_n \times I_n}$. For any $n \in \{1, \ldots, N\}$, there is a formula of equivalence:

$$Y = X \times A^{(1)} \times A^{(2)} \cdots x_N A^{(N)} \iff Y_{(n)} = A^{(n)} X_{(n)} (A^{(N)} \otimes \cdots \otimes A^{(n+2)} \otimes A^{(n-1)} \otimes \cdots \otimes A^{(1)})$$

3. The Algorithm of Tensor Decompositions
3.1 Definition of Tensor Decompositions

Before the introduction of tensor decomposition, we need to understand the mathematical expression of tensor decomposition. For the definition of tensor decomposition cannot be used in the algorithm in practice.

The tensor decomposition algorithm used in practice is different from the definition of tensor decomposition itself, but it must be changed based on the tensor decomposition’s definition.

3.1.1 CP Decompositions

Since rank-1 decomposition is the basis of tensor CP decomposition. So we the definition of rank-1 decomposition.

If exists an N-order tensor can be written as an outer product of N vectors. The expression will be

\[ X = a^{(1)} \circ a^{(2)} \circ \cdots \circ a^{(N)}. \]

For the three-order tensor \( X \), the expression of tensor is \( X = a \circ b \circ c \). So, we know that three-order CP Decompositions can be expressed in

\[ X \approx \sum_{r=1}^{R} a_r \circ b_r \circ c_r. \]

3.1.2 Tucker Decompositions

If exists an N-order tensor can be written as an core tensor and N factor matrices. The expression can be written in

\[ X = G \times_1 A^{(1)} \times_2 A^{(2)} \cdots \times_{(N)} A^{(N)}. \]

For the three-order tensor \( X \), the expression of tensor is \( X \approx G \times_1 A \times_2 B \times_3 C \).

As CP Decompositions, \( A \in \mathbb{R}^{I \times P} \), \( B \in \mathbb{R}^{J \times Q} \), \( C \in \mathbb{R}^{K \times R} \) are factor matrices. It is the principal component of a tensor in a corresponding module as well. Tensor \( G \in \mathbb{R}^{P \times Q \times R} \) is called core tensor. It shows the degree of interaction between different components.

3.2 Notation about Tensor Decompositions

After the introduction of algorithm about tensor decomposition, we need to understand two concepts related to tensor decomposition: rank of tensor and uniqueness of tensor decomposition.

3.2.1 Rank of tensor

The rank of tensor has difference definition in difference tensor decomposition.

In CP decompositions, the rank of tensor (written in \( \text{rank} \left( X \right) \)) is the smallest number of the tensors of the sum of the rank 1 decomposition. Unlike the rank of a matrix, no algorithm can directly calculate the rank of a given tensor. How to get the rank of tensors is a HP-Hard problem. In the actual use of the algorithm, the rank of the tensor is given before the algorithm is calculated.

In Tucker decompositions, a tensor has several ranks. The number of the tensor ranks (written in \( \text{rank}_n \left( X \right) \)) is the same as the tensor dimension. It’s the number of column of factor matrices. Each dimension has its own rank. In this way, we can say that the N-order tensor \( X \) is a tensor of \( \text{rank} = \left( R_1, R_2, \ldots, R_N \right) \), and its rank n corresponds to \( R_n = \text{rank}_n \left( X \right) \).

3.2.2 Uniqueness

In difference tensor decompositions, the meaning of Uniqueness is not the same.

In CP decompositions, Uniqueness means that this is the only possible combination of rank-one tensors that sums to \( X \). Of cause, CP decompositions is not unique under some circumstance. The result on uniqueness depends on the concept of k-rank, which is the maximum value such that columns are linearly independent.
In Tucker decompositions, Uniqueness is not available. This means that the core tensor can be modified by adjusting the factor matrices without affecting the fit. The constraint condition of Tucker decomposition is that the factor matrices are orthogonal.

3.3 Algorithm in practice
The definition of tensor decompositions cannot be used in practice. The definition expressed in mathematical formula should be converted to a formula that can be used in practical algorithms.

3.3.1 CP Decompositions
In 2.1.1, we know that CP Decompositions is the processes tensor X is decomposed into the sum of R rank 1 decomposition. So, the solution formula of factor matrices (Here is A for example) is

\[ X_{(1)} \approx \hat{A}(C \otimes B)^T. \]

Here \( \hat{A} = \text{Adiag}(\lambda) \).

The mathematical formula can be converted in

\[ \hat{A} = X_{(1)}(C \otimes B)^T. \]

Here the right upper corner symbol "." is the pseudo inverse matrix.

For the formula about Khatri-Rao product

\[ (A \otimes B)^T = (A^T A \ast B^T B)^T(A \otimes B)^T \]

We can transform the pseudo inverse matrix of \((IJ)^T \times K\) to the inverse matrix of \(R \times R\). In this way, we can get the solution formula of factor matrices

\[ \hat{A} = (X_{(1)}(C \otimes B)(C^T C \ast B^T B)^{-1}) \]

The acquisition of factor matrices for CP decomposition is to use transformation formula to modify A through B and C, B through C and A, C through B and A. The process of modifying factor matrices by two factor matrices is repeated until the convergence criteria fits.

The convergence criterion is

\[ \min \|X - \hat{X}\|. \]

Here \( X \) is the original tensor. And \( \hat{X} \) is the tensor calculated from three factor matrices.

According to the above description, the algorithm of CP decomposition is:

1: Input: \( X, R \)
2: Initialize factor matrices \( B \) and \( C \)
3: Repeat
4: \[ M_{\text{pre}} = M_{\text{now}} \text{ Or } M_{\text{pre}} = 0 \] (If it’s first time through the loop)
5: \[ \hat{A} = (X_{(1)}(C \otimes B)(C^T C \ast B^T B)^{-1}) \]
6: \[ \hat{B} = (X_{(2)}(C \otimes A)(C^T C \ast A^T A)^{-1}) \]
7: \[ \hat{C} = (X_{(3)}(B \otimes A)(B^T B \ast A^T A)^{-1}) \]
8: Normalize columns \( A, B, C \), Get \[ \hat{\lambda}_1 \ldots \hat{\lambda}_R \]
9: \[ M_{\text{now}} = \|X - \hat{X}\| \]
10: \[ \Delta = M_{\text{pre}} - M_{\text{now}} \]
11: Until: The \( \Delta \) is minimum
12: Out \[ \hat{\lambda}_1 \ldots \hat{\lambda}_R, A, B, C \]

3.3.2 Tucker Decompositions
The founder of Tucker decomposition have mentioned the method of Tucker decomposition. Its basic idea of the algorithm is to find N-order correspondence to find the best factor matrices, but the algorithm is not described in detail. It’s called higher-order Singular Value Decomposition (Abbreviated as HOSVD). When \( R_n < \text{rank}(X) \), the decomposition is called the truncated HOSVD. Based on the alternating least squares algorithm, Higher-order Orthogonal Iteration is proposed (abbreviated as HOOI).

The approximation condition of HOOI is to obtain the minimum of \( \|X - \hat{X}\| \), which is the same as CP Decompositions. However, According to the properties of the Tucker decomposition and the Norm Computation, the minimum of \( \|X - \hat{X}\| \) can be convert to the maximum of the norm of core tensor \( \|G\| \).

Because of the non-uniqueness of Tucker decomposition algorithm, a constraint should be given in the algorithm. The common constraint is that the column vectors of a sequence of factor matrices should be orthogonal.

Although HOOI is a very common algorithm of Tucker decomposition. In practice, it has to take very expensive system resources and time to calculate tensor multiplication, because of the frequent matrices multiplication and tensor unfolding & folding. So, a matricized version of algorithm is needed. According to the equation related to tensor multiplication, in the three-order tensor, the expression can be converted into:

\[
Y = X \times_1 B^T \times_2 C^T \Leftrightarrow Y = X(C \otimes B)
\]

In this way, the cost of system resources and time is obviously declined. So, the algorithm process of HOOI in three-order can be changed in:

| Step | Description |
|------|-------------|
| 1    | Input \( X, R_p, R_q, R_r \) |
| 2    | Initialize factor matrices \( B \) and \( C \) |
| 3    | Repeat |
| 4    | \( Y_p = X(1)(C \otimes B) \), \( A \leftarrow R_a \) Leading left singular vectors of \( Y_p \) |
| 5    | \( Y_q = X(2)(C \otimes A) \), \( B \leftarrow R_b \) Leading left singular vectors of \( Y_q \) |
| 6    | \( Y_r = X(3)(B \otimes A) \), \( C \leftarrow R_c \) Leading left singular vectors of \( Y_r \) |
| 7    | \( G(3) = C^T X(3)(B \otimes A) \) |
| 8    | Until the maximum of \( \|G(3)\| \) (Here \( \|G\| = \|G(3)\| \) ) |
| 9    | Out \( G, A, B, C \) |

Here \( G(3) \) is used to calculate the norm of core tensor, because they can get the same result.

4. Experiments and Analysis

4.1 Introductions of clBLAS and clBLAST

Although OpenCL has advantages in computational efficiency, it is better to be as efficient as possible in linear algebra. Before the experiment, we need to introduce the clBLAS, the library is specially used for linear algebra under OpenCL.

BLAS (Basic Linear Algebra Subprograms) are routines that provide standard building blocks for
performing basic vector and matrix operations. The clBLAS is a software library containing the complete set of BLAS on OpenCL. The goal of clBLAS is to make it easier for developers to utilize the inherent performance and power efficiency benefits of heterogeneous computing. clBLAS interfaces do not hide nor wrap OpenCL interfaces, but rather leaves OpenCL state management to the control of the user to allow for maximum performance and flexibility.

CLBlast is a tunable OpenCL BLAS library written in C++11. It is designed to leverage the full performance potential of a wide variety of OpenCL devices from different vendors, including desktop and laptop GPUs, embedded GPUs, and other accelerators. In this way, we use CLBlast for

4.2 Experiments On clBLAS

For text the module of tensor decompositions designed by clBLAS, We have prepared a test platform. It's configured as follows:

|                     |                   |
|---------------------|-------------------|
| CPU                 | Intel® Core™ i5-5200U |
| GPU                 | Nvidia GeForce 920M |
| Memory              | 4G                |
| Hard Disk           | 500G              |
| Windows Version     | Windows 7        |
| CUDA Version        | 8.0               |
| Program platform    | Visual Studio 2017 |

The following table shows the time of different tensor decomposition algorithm based on clBLAS.

| Tensor Scale | CP    | Tucker |
|--------------|-------|--------|
| 512*512*512  | 43.513| 62.308 |
| 768*768*768  | 43.325| 61.741 |
| 1024*1024*1024| 43.118| 62.487 |
| 1280*1280*1280| 43.701| 62.893 |

It can be seen that the calculation time of the tensor decomposition module using clBLAST is independent of the Tensor Scale, and it is basically stable in the same time. It can be seen that the module using clBLAST is not only versatile, but also has little to do with the efficiency of module calculation, which is very useful in the process of large-scale tensor decomposition.

5. Summary and Conclusion

This paper summarizes the current research status of tensor decomposition algorithm as, and designs a tensor decomposition module based on clBLAST. By using clBLAST, far more than other acceleration can be achieved on a single GPU, and the efficiency of tensor decomposition can be improved.

Meanwhile, this module also shows some limitations. For example, the decomposition-oriented tensor of this module is a general tensor, and the decomposition of large sparse tensors are not accelerated accordingly. At the same time, we do not discuss the decomposition of tensors with special requirements (for example, non-negative tensor). These problems are the focus of follow-up research, so as to expand the application scope of decomposition.

References:

[1] Kolda, T. G., & Bader, B. W. (2009). Tensor decompositions and applications. Siam Review, 51(3), 455-500.

[2] Tucker, L. R. (1966). Some mathematical notes on three-mode factor
analysis. Psychometrika, 31(3), 279-311.

[3] Lathauwer, L. D., Moor, B. D., & Vandewalle, J. (2000). A multilinear singular value decomposition. Siam J. matrix Anal. appl, 21(4), 1253-1278.

[4] Lathauwer, L. D., Moor, B. D., & Vandewalle, J. (2013). On the best rank-1 and rank-(r1, r2,...,rn) approximation of higher-order tensor. Siam J. matrix Anal. appl, 21(4), 1324-1342.

[5] Bro, R., & Andersson, C. A. (1998). Improving the speed of multiway algorithms: part ii: Tucker3. Chemometrics & Intelligent Laboratory Systems, 42(1–2), 93-103.

[6] William, H. (2003). Numerical recipes in C++. Pub. House of Electronics Industry.

[7] Chakaravarthy, V. T., Choi, J. W., Joseph, D. J., Liu, X., Murali, P., & Sabharwal, Y., et al. (2017). On Optimizing Distributed Tucker Decomposition for Dense Tensors. Parallel and Distributed Processing Symposium (pp.1038-1047). IEEE.

[8] Kroonenberg, P. M., & Leeuw, J. D. (1980). Principal component analysis of three-mode data by means of alternating least squares algorithms. Psychometrika, 45(1), 69-97.