This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2018.2824303, IEEE Transactions on Big Data

A Multi-order Distributed HOSVD with its Incremental Computing for Big Services in Cyber-Physical-Social Systems

Xiaokang Wang, Laurence T. Yang, Xingyu Chen, Lizhe Wang, Rajiv Ranjan, Xiaodao Chen, and M. Jamal Deen

Abstract—Big service is an extremely important application of service computing to provide predictive and needed services to humans. To operationalize big services, the heterogeneous data collected from Cyber-Physical-Social Systems (CPSS) must be processed efficiently. However, because of the rapid rise in the volume of data, faster and more efficient computational techniques are required. Therefore, in this paper, we propose a multi-order distributed high-order singular value decomposition method (MDHOSVD) with its incremental computational algorithm. To realize the MDHOSVD, a tensor blocks unfolding integration regulation is proposed. This method allows for the efficient analysis of large-scale heterogeneous data in blocks in an incremental fashion. Using simulation and experimental results from real-life, the high efficiency of the proposed data processing and computational method, is demonstrated. Further, a case study about cyber-physical-social system data processing is illustrated. The proposed MDHOSVD method speeds up data processing, scales with data volume, improves the adaptability and extensibility over data diversity and converts low-level data into actionable knowledge.

Index Terms—Big Data; Tensor; MDHOSVD; MIHOSVD; Distributed Computing; Incremental Computing; CPSS.

1 Introduction

Big services are based on the efficient processing of large amount of heterogeneous collected data. The Internet of Things (IoT) provides such data generated from smart devices [1] and integrates the cyber space and physical space together, into the so-called Cyber-Physical Systems (CPS) [2]. The new paradigm, Cyber-Physical-Social Systems (CPSS), combining the social space with the Internet of Things [3], [4], [5], have the potential to provide the valuable information on the behavior of and predictive services to humans [6]. For this potential to be realized, heterogeneous data, the common element in cyber, physical and social spaces - three components of CPSS, must be collected, processed and converted into actionable information [7].

The source of big data is the billions of bytes collected every second on cyber, physical and social systems. These heterogeneous data encompasses a variety of complex, large-scale information which is presently beyond the capabilities of conventional software and hardware platforms. Therefore, new methods to efficiently process the big, heterogeneous data are needed so that efficient, timely and high-quality services are provided to humans. A schematic representation of the relationships among the components of CPS, CPSS, big data, and big service is shown in Fig. 1. This diagram shows the need for efficient big data processing as a key component of big services.

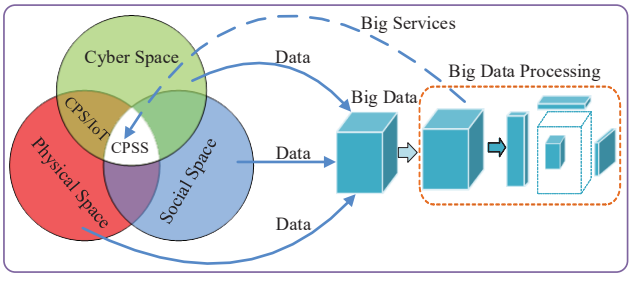


Fig. 1. Relationships among Cyber-Physical Systems (CPS), Cyber-Physical-Social Systems (CPSS), Big Data and Big Services.

To efficiently process big data, the characteristics of the collected data must be carefully considered. Firstly, we

- *X. Wang, and X. Chen are with School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China.*
- *L.T. Yang is with School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China, and Department of Computer Science, St. Francis Xavier University, Antigonish, Canada.*
- *L. Wang is with School of Computer Science, China University of Geosciences, Wuhan, China, and the Institute of Remote Sensing and Digital Earth, Chinese Academy of Sciences, Beijing, China.*
- *R. Ranjan is with School of Computing Science, Newcastle University, Newcastle upon Tyne, United Kingdom.*
- *X. Chen is with School of Computer Science, China University of Geosciences, Wuhan, China.*
- *M.J. Deen is with Department of Electrical and Computer Engineering, McMaster University, Hamilton, Canada.*
  L.T. Yang is the corresponding author, E-mail: ltyang@ieee.org.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2018.2824303, IEEE Transactions on Big Data

need to consider that the collected data are heterogeneous from a variety of independent devices and could be video, image, audio, and text etc. Secondly, these big collected data are large scale and must be processed rapidly or even in real-time [8], [9]. For the former, Kuang *et al.* [10] proposed a tensor-based big data representation model to represent the heterogeneous CPSS big data. This tensor representation for big data included unstructured data, semi-structured data and structured data gave promising results. For the second characteristic, once the representation is decided, then big data processing is required to analyze big data by reducing noise and redundancies, thereby producing high-quality big data.

Currently, Singular Value Decomposition (SVD), is one of the main data processing methods studied [8], [9], [10]. For SVD, two main computational methods for two-order data (matrix), namely the Golub-Kahan SVD method and the Jacobi-based SVD method, were proposed in [11]. Also, incremental SVD computation [12], [13], SVD computation of two-order matrix stream [14], and parallel SVD computation [15] have been proposed.

An extension of two-order SVD, the High-Order Singular Value Decomposition (HOSVD), is considered as an efficient, scalable, and practical big data processing method [8], [9], [10]. HOSVD is based on the tensor representation, and was used in data processing [16], [17], noise reduction [18], and human motion recognition [19]. An extension to HOSVD, the distributed High-Order Singular Value Decomposition (DHOSVD) with its incremental computation (IHOSVD) was proposed in [8]. DHOSVD is a distributed decomposition of tensor used for processing big data in high-order space.

Notations	Descriptions
$\mathcal{A}\in R^{I_1\times I_2\times\cdots\times I_n\times\cdots\times I_N}$	The Nth-order tensor $\mathcal{A}$
I_n	The dimensionality of the n-th order
$A_{(n)}$	The unfolding matrix of tensor $\mathcal{A}$ along the n-th order
$\mathcal{A}^{(a_1,a_2,\cdots,a_n,\cdots,a_N)}$	The tensor $\mathcal{A}$ is divided into a_n part along the n-th order, $1\leq n\leq N$
$\mathcal{A}^{(i_1,i_2,\cdots,i_n,\cdots,i_N)}$	The i_n-th sub-tensor along the n-th order, $1\leq i_n\leq a_n$, $1\leq n\leq N$
$A_{(n)}^{(i_1,i_2,\cdots,i_n,\cdots,i_N)}$	The unfolding matrix of tensor $\mathcal{A}^{(i_1,i_2,\cdots,i_n,\cdots,i_N)}$ along the n-th order
$\mathcal{A}^{(i_1,i_2,\cdots,i_n,\cdots,i_{N-1},:)}$	The integration of a_N sub-tensors along the N-th order is finished
$A_{(n)}^{(i_1,i_2,\cdots,i_n,\cdots,i_{N-1},:)}$	The unfolding matrix of tensor $\mathcal{A}^{(i_1,i_2,\cdots,i_n,\cdots,i_{N-1},:)}$ along the n-th order
$\mathcal{A}^{(i_1,i_2,\cdots,i_n,:,\cdots,:)}$	The tensor integration from $(n{+}1)$-th order to N-th order is finished
$A_{(n)}^{(i_1,i_2,\cdots,i_n,:,\cdots,:)}$	The unfolding matrix of tensor $\mathcal{A}^{(i_1,i_2,\cdots,i_n,:,\cdots,:)}$ along the n-th order
$\mathcal{A}^1\|_j\mathcal{A}^2$	Tensors $\mathcal{A}^1$, $\mathcal{A}^2$ are integrated along the j-th order
$\Sigma_{i_j=1}^{a_j}\|_j\mathcal{A}^{(i_1,i_2,\cdots,i_j,\cdots,i_N)}$	a_j tensors $\mathcal{A}^{(i_1,i_2,\cdots,i_j,\cdots,i_N)}$, $1\leq i_j\leq a_j$ are integrated along the j-th order
$\Sigma_{i_j=1}^{a_j}\|_j A_{(k)}^{(i_1,i_2,\cdots,i_j,\cdots,i_N)}$	The unfolding matrix of tensor $\Sigma_{i_j=1}^{a_j}\|_j\mathcal{A}^{(i_1,i_2,\cdots,i_j,\cdots,i_N)}$ along the k-th order

TABLE 1
The Necessary Notations Used in This Paper

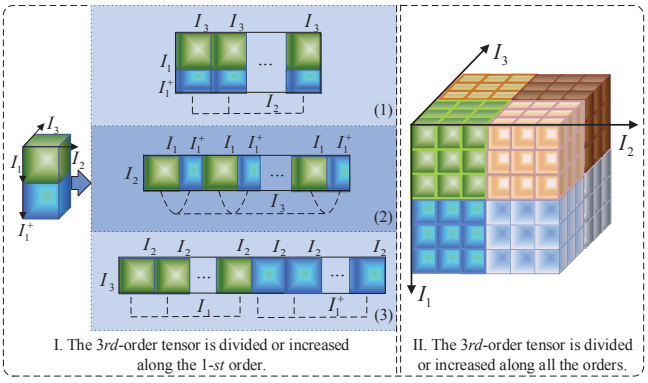


Fig. 2. Tensor Division or Increasing.

The DHOSVD and IHSOVD methods proposed in [8] are appropriated for special distributed or incremental computational condition in which the tensor is divided or increases along a certain order, for example, as a function of time (such as the 1-st order I_1) as shown in Fig. 2 (I). Normally, the heterogeneous CPSS big data increase from the aspects of multi-orders. This means that the CPSS big data after tensor representation should be divided or increased along multi-orders, for example, as an function of time (such as the 1-st order I_1) or the number of people (such as the 2-ed order I_2) as shown in Fig. 2 (II). Since the unfolding process is essential for HOSVD [8], then the tensor is divided or increased along multi-orders, resulting in a very complex unfolding. For example, two key questions to be addressed are - how to integrate the unfolding matrices of all sub-tensors, and what is the full-precision position of each element in the integrated unfolding matrix? Therefore, MDHOSVD is a significant challenge to be solved for the efficient processing of big data.

Previously, the computational methods for DHOSVD and IHOSVD proposed in [8] were based on the RoundRobin process. In this process, each sub-data block must be transferred to every available node in distributed systems, resulting in a huge communication overhead and inefficient big data processing. Furthermore, if any employed node has a computational problem, then the whole RoundRobin process must be repeated. The result of these limitations is that the DHOSVD and IHOSVD proposed in [8] lack robustness and flexibility when dealing with big data processing.

To tackle aforementioned challenges related to integrating the unfolding matrices and position of its elements,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2018.2824303, IEEE Transactions on Big Data

we propose a tree-based Multi-order Distributed High Order Singular Value Decomposition (MDHOSVD) with its incremental computation (MIHOSVD) in this paper. The main contributions of this paper are the following. Firstly, we improve the mathematical representation of the tensor unfolding method. Then, tensor blocks unfolding integration regulation is proposed to provide the precise position of each element in the integrated unfolding matrix. Secondly, we present a tree-based MDHOSVD (and MIHOSVD) method, in which the tensor can be divided or increased along several orders or even all orders at same time.

The remainder of this paper is organized as follows. In Section II, we briefly review related background information on Big Data and tensor decomposition. Then, in Section III, the improved unfolding method and the tensor blocks unfolding integration regulation, the MDHOSVD and the MIHOSVD, are described. The experimental and simulation results of the proposed algorithms are discussed in Section IV. In Section V, to demonstrate the high efficiency of proposed algorithm, a case study of big service is presented. Finally, in Section VI, the conclusions are given.

2 Background

In this section, relevant background information about tensors, tensor unfolding and Singular Value Decomposition are presented. Also, notations used in this paper are listed in TABLE 1. This will enable the reader to easily follow the MDHOSVD and MIHOSVD methods presented in Section III.

Tensor. Tensors, being higher order generalizations of matrices, are multi-dimensional arrays widely used to represent higher order relationships present in Cyber-Physical-Social Systems [10], [20]. Formally, $\mathcal{A}\in R^{I_1\times I_2\times\cdots\times I_n\times\cdots\times I_N}$ is an Nth-order tensor, where, I_n $(1\leq n\leq N)$ is the dimensionality of the n-th order [21]. Tensors have been used in many domains such as the analysis of images [17], brain data [22], de-noising [23], clustering [24], and even human motion recognition [19].

Tensor unfolding. The process of extracting the elements of a high-order tensor into a matrix is tensor unfolding [20][21]. The unfolding matrix along the n-th order of an Nth-order tensor $\mathcal{A}\in R^{I_1\times I_2\times\cdots\times I_n\cdots\times I_N}$ is represented as $A_{(n)}\in R^{I_n\times(I_{n+1}I_{n+2}\cdots I_N I_1 I_2\cdots I_{n-1})}$ [21]. Also, an example on a three-order tensor unfolding into a matrix was schematically illustrated in [8], [21].

The unfolding of the integrated tensor including the sequential incremental form and interleaving incremental form was discussed in [8]. The former include the increase along the row in order or increase along the column in order. In this unfolding, the order of data from the original tensor does not change, such as the illustrated by (1) and (3) in Fig. 2 (I) [8]. On the other hand, the incremental part in the interleaving form changes the column order of the original tensor, as shown in (2) in Fig. 2 (I).

Singular Value Decomposition (SVD). SVD has been implemented using two computational methods, the QR-based method and Jacobi-based method [8], [11]. The process of Jacobi-based SVD method for a given matrix $A\in R^{m\times n}$ is described as follows [8], [11].

Firstly, the orthogonalization of any column pair of matrix A is implemented by the Jacobi rotation $J(i,j)$. Algorithm 1 below shows the detailed computational process of the Jacobi rotation $J(i,j)$ for realizing the orthogonalization of the i-th and j-th columns of matrix A [8][25].

Secondly, the computation of the right singular matrix V is implemented by the product of a series of Jacobi rotations in order $V_{n\times n}=\Pi_i\Pi_{j,j>i}J(i,j)$ [8][25].

Thirdly, another orthogonal matrix $B_{m\times n}$ is computed as $B=AV=[b_1,b_2,\cdots,b_n]$, $b_i^T b_j=0$, $i\neq j$. Then, the singular value matrix Σ is computed as $\sigma_i=|B(:,i)|=|b_i|$, $\Sigma=diag[\sigma_1,\sigma_2,\cdots,\sigma_n]$ [8][25].

Fourthly, each column of the left singular matrix computation $U_{m\times n}$ is obtained as $U(:,i)=\frac{B(:,i)}{\sigma_i}$. Then, Jacobi-based SVD method of matrix $A_{m\times n}=U_{m\times n}\Sigma_{n\times n}V_{n\times n}$ is completed [8][25].

Algorithm 1 The Jacobi rotation

1: **Input:** Two columns a_i,a_j;
2: **Output:** Jacobi rotation $J(i,j)$.
3: $\lambda=\frac{\|a_j\|^2-\|a_i\|^2}{2(a_i)^T a_j}$;
4: $t=sgn\lambda/(|\lambda|+\sqrt{1+\lambda^2})$;
5: $c=\frac{1}{\sqrt{1+t^2}}$;
6: $s=ct$;
7: $J(i,j)=\begin{pmatrix} c & -s \\ s & c \end{pmatrix}$;

Finally, High Order Singular Value Decomposition (HOSVD) of an Nth-order tensor $\mathcal{A}\in R^{I_1\times I_2\times\cdots\times I_N}$ is given as [20][21]:

$$\mathcal{S}=\mathcal{A}\times_1 U_1^T\times_2 U_2^T\cdots\times_N U_N^T, \tag{1}$$

$$\widehat{\mathcal{A}}=\mathcal{S}\times_1 U_1\times_2 U_2\cdots\times_N U_N, \tag{2}$$

where, U_n, $1\leq n\leq N$ is the left singular matrix of the $A_{(n)}$, and $\mathcal{S}$ is the core tensor [8], [21]. The state-of-the-art of SVD/HOSVD was detailed in [8]. In this paper, we improve on the work [8] and present the multi-order distributed HOSVD with its incremental computing for CPSS big data processing.

3 Multi-order HOSVD Algorithms

In this section, the improved mathematical representation of tensor unfolding method and tensor blocks unfolding integration regulation are discussed. This will allow us to accurately present the position of any element from different tensor blocks in the integrated unfolding matrix. In addition, a tree-based Multi-order Distributed HOSVD (MDHOSVD) algorithm with its incremental computing (MIHOSVD) is proposed.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2018.2824303, IEEE Transactions on Big Data

3.1 Improved tensor unfolding method and Regulation:

The mathematical representation of the tensor unfolding method was presented in [21]. However, to present the method in a better understood way, we supplement the description in [21] by explicitly presenting expressions for the 1-st order and N-th order of an Nth-order tensor.

Improved mathematical representation of tensor unfolding method: Formally, the unfolding matrix $A_{(n)} \in R^{I_n \times (I_{n+1} I_{n+2} \cdots I_N I_1 I_2 \cdots I_{n-1})}$ contains the element $a_{i_1 i_2 \cdots i_n \cdots i_N}$, at the position with row number i_n. The column number will be improved as follows:

If $n = 1$ (1-st order), then the column number equals to,

$$(i_2 - 1) I_N I_{N-1} \cdots I_{n+2} + (i_3 - 1) I_N I_{N-1} \cdots I_{n+3} + \cdots + (i_{N-1} - 1) I_N + i_n,$$

if $n = N$ (N-th order), the column number equals to,

$$i_{N-1} + (i_{N-2} - 1) I_{N-1} + (i_{N-3} - 1) I_{N-2} I_{N-1} + (i_{N-4} - 1) I_{N-3} I_{N-2} I_{N-1} + \cdots + (i_3 - 1) I_4 I_5 \cdots I_{N-1} + (i_2 - 1) I_3 I_4 I_5 \cdots I_{N-1} + (i_1 - 1) I_2 I_3 I_4 I_5 \cdots I_{N-1}.$$

For other values of n, the column number is the same as presented in [21],

$$(i_{n+1} - 1) I_{n+2} I_{n+3} \cdots I_N I_1 I_2 \cdots I_{n-1} + (i_{n+2} - 1) I_{n+3} \cdots I_N I_1 I_2 \cdots I_{n-1} + \cdots + (i_N - 1) I_1 I_2 \cdots I_{n-1} + (i_1 - 1) I_2 I_3 \cdots I_{n-1} + (i_2 - 1) I_3 I_4 \cdots I_{n-1} + \cdots + i_{n-1}.$$

Tensor blocks unfolding integration regulation: Large-scale data cannot be processed by a single core system, so it should be cut into many blocks and then processed using a block-based method. Also, the sub-tensors cut from the large scale data should be processed in a distributed system, which is more efficient than the traditional single core system. Meanwhile, tensor unfolding is necessary for HOSVD [21]. However, how to integrate the sub-tensor blocks unfolding result to obtain the unfolding matrix of the integrated tensor is challenging. To address this challenge, tensor blocks unfolding integration regulation is proposed (see Fig. 3).

In Fig. 3, the horizontal direction represents "increase along the n-th order", $1 \leq n \leq N$. The vertical direction means "unfolding along the n-th order", $1 \leq n \leq N$. To explain it in more detail, we take the cell in the red circle in Fig. 3 as an example. If sub-tensors integrate along the third order, the unfolding matrix of the integrated tensor along the second order increases along the column in order.

Tensor blocks unfolding integration regulation is now illustrated with an example. Consider two $3rd$-order sub-tensors, $\mathcal{A}$ and $\mathcal{B}$ shown at the bottom of Fig. 4. When combined along second order, we obtain the integrated tensor $\mathcal{C}$ shown at the top of Fig. 4. The unfolding matrix of the integrated tensor $\mathcal{C}$ along the first order $C_{(1)}$ is composed by the $A_{(1)}$ and $B_{(1)}$ along the column in order as shown in (3) form of Fig. (2) (I). In the same way,

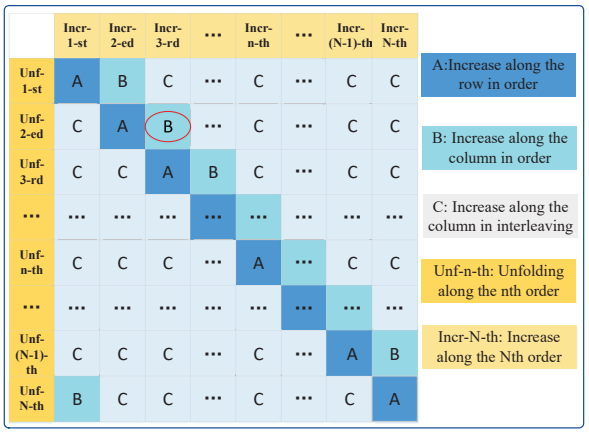


Fig. 3. Tensor Blocks Unfolding Integration Regulation.

$C_{(2)}$ is composed by the $A_{(2)}$ and $B_{(2)}$ along the row in order as shown in (1) form of Fig. (2) (I). Similarly, $C_{(3)}$ is composed by the $A_{(3)}$ and $B_{(3)}$ along the column in interleaving order as shown in the (2) form of Fig. (2) (I).

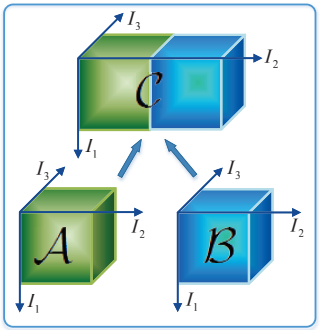


Fig. 4. An Example about Two $3rd$-tensors Integrates along the 2-nd Order.

3.2 Multi-order Distributed HOSVD with its Incremental Computing Algorithms

From above, the unfolding matrix of the integrated tensor presents two main computational challenges due to its increase along the row (such as the (1) form in Fig. 2 (I)) or along the column (such as the (2) and (3) forms in Fig. 2 (I)). Here, we propose an algorithm for integration (Algorithm 2) to address these two challenges. Further, two new tree-based algorithms are proposed to implement MDHOSVD and MIHOSVD.

Algorithm 2 for Integration: Integration of the unfolding matrices including integration along the column or the row is now described. Since the Jacobi-based orthogonalization method is independent on the position of the column [8], then we propose Algorithm 2 which uses it to address the challenge of increase along the column including the (2) and (3) forms shown previously in Fig. 2 (I). Two matrices $A_1 \in R^{I_{row} \times I_{col_1}}$, and $A_2 \in R^{I_{row} \times I_{col_2}}$, with their SVD results, were integrated together in the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2018.2824303, IEEE Transactions on Big Data

Algorithm 2 The Algorithm for Integration

1: **Input:** Two matrices $A_1{=}U_1\Sigma_1V_1^T$, $A_2{=}U_2\Sigma_2V_2^T$.
2: **Output:** SVD result (U, Σ, and V) of matrix A, integrated from A_1 and A_2.
3: **if** Two matrices A_1 and A_2 integrate along the column ($caseB$ or $caseC$) **then**
4: Suppose $B_1{=}U_1 \times \Sigma_1$ and $B_2{=}U_2 \times \Sigma_2$;
5: Establish the matrix $B{=}[B_1|B_2]{=}$ $[b_1, b_2, \cdots, b_i, \cdots, b_j, \cdots, b_{col}]$;
6: Establish the matrix $V = \begin{bmatrix} V_1 & 0 \\ 0 & V_2 \end{bmatrix}$;
7: **else**
8: Suppose $B_1{=}V_1 \times \Sigma_1$ and $B_2{=}V_2 \times \Sigma_2$;
9: Establish the matrix $B = [B_1|B_2]{=}$ $[b_1, b_2, \cdots, b_i, \cdots, b_j, \cdots, b_{col}]$;
10: Establish the matrix $V = \begin{bmatrix} U_1 & 0 \\ 0 & U_2 \end{bmatrix}$;
11: **end if**
12: **for** $i = 1$ to $col{-}1$ **do**
13: **for** $j = i + 1$ to col **do**
14: **if** $b_i^T b_j^T > \tau$ **then**
15: Execute the Algorithm 1 on two columns b_i and b_j and obtain $J(i, j)$;
16: $[b_i \quad b_j]{=}[b_i \quad b_j]J(i, j)$;
17: **end if**
18: **end for**
19: **end for**
20: $C{=}\Pi_i\Pi_{j,j>i}J(i, j)$;
21: $B{=}B \times C$;
22: $\sigma^i{=}|B(:, i)|$;
23: **if** Two matrices A_1 and A_2 integrate along the column ($caseB$ or $caseC$) **then**
24: $V{=}V \times C$;
25: $U(:, i){=}\frac{B(:,i)}{\sigma^i}$;
26: $A{=}U\Sigma V^T$;
27: **else**
28: $V{=}V \times C$;
29: $U(:, i){=}\frac{B(:,i)}{\sigma^i}$;
30: $A{=}V\Sigma U^T$;
31: **end if**

(2) and (3) forms in Fig. 2 (I) to produce the matrix $A{\in}R^{I_{row}\times(I_{col_1}+I_{col_2})}$. The proposed Algorithm 2 to compute the SVD result of matrix $A{\in}R^{I_{row}\times(I_{col_1}+I_{col_2})}$, according to the result of A_1 and A_2 is now described.

In the algorithm, from lines 3 to 5, the description of two matrices $B_1{\in}R^{I_{row}\times I_{col1}}$ and $B_2{\in}R^{I_{row}\times I_{col2}}$ that integrate together along the column and obtain the matrix $B{\in}R^{I_{row}\times I_{col}}$, where $I_{col}{=}I_{col1}{+}I_{col2}$ is given. In line 6, matrix V is established according to the SVD results of A_1 and A_2. Then, the Jacobi-based orthogonalization method is used to orthogonalize the matrix B from line 12 to line 22 in the Algorithm 2. According to the convergence condition $\tau{=}B\varepsilon$ proposed in [8], [11], [25], when the orthogonalization is finished, the matrices B and V are updated. Finally, from lines 24 to 26, we show how the SVD result of the produced matrix A is computed.

Alternatively, from lines 7 to 11, the incremental (1) form proposed in Fig. 2 (I) is addressed by the matrix transpose to convert "update along the row" into "update along the column" as shown in Fig. 2 (I). Here, two matrices $A_1{\in}R^{I_{row_1}\times I_{col}}$, and $A_2{\in}R^{I_{row_2}\times I_{col}}$, whose SVD results have been obtained previously, were integrated together in (1) form in Fig. 2 (I) to produce the matrix $A{\in}R^{(I_{row_1}+I_{row_2})\times I_{col}}$. The computational process is similar with the integration along the column. As shown from lines 8 to 10, the matrices B and V are constructed, and from lines 12 to 22, the same orthogonalization as in the previous paragraph, is implemented. Then, from lines 28 to 30, the computation of the produced matrix A is completed.

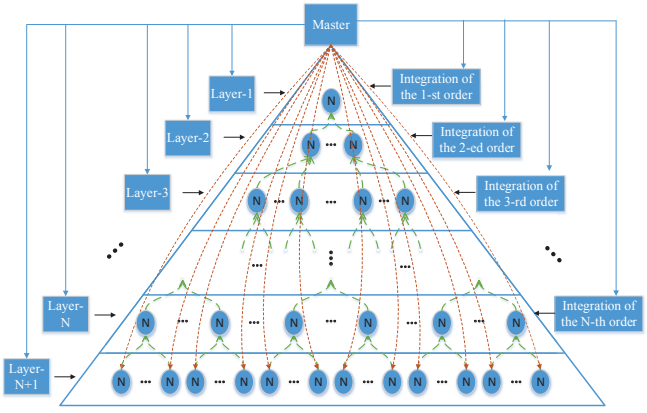


Fig. 5. Tree-based MDHOSVD.

Algorithm 3 for the MDHOSVD: After addressing the two computational challenges related to the unfolding matrix of the integrated tensor, we now describe our proposed tree-based multi-order distributed high-order singular value decomposition (MDHOSVD). Consider an Nth-order tensor $\mathcal{A}{\in}R^{I_1\times I_2\times\cdots\times I_n\times\cdots\times I_N}$ divided into a_n parts along the n-th order, $1{\le}n{\le}N$. Then, in tensor $\mathcal{A}$, a certain sub-tensor is $\mathcal{A}^{(i_1,i_2,\cdots,i_n,\cdots,i_N)}{\in}R^{I_1^{i_1}\times I_2^{i_2}\times\cdots\times I_n^{i_n}\times\cdots\times I_N^{i_N}}$, where i_n means the i_n-th part along the n-th order, and $1{\le}i_n{\le}a_n$, $\Sigma_{i_n=1}^{a_n}I_n^{i_n}{=}I_n$. The position number $(i_1, i_2, \cdots, i_n, \cdots, i_N)$ is named the coordinate of this sub-tensor in tensor $\mathcal{A}$. Now, the main challenge is how to obtain the HOSVD result of the tensor $\mathcal{A}$, according to the HOSVD computational result of each sub-tensor $\mathcal{A}^{(i_1,i_2,\cdots,i_n,\cdots,i_N)}$?

For an Nth-order tensor, the proposed MDHOSVD method can be visualized as an $(N{+}1)$ layer tree. As shown in Fig. 5, the sub-tensors integrate along the n-th order in the n-th layer of the tree, and $1{\le}n{\le}N$. The number of node in the $(n{+}1)$-th layer is $\Pi_{k=1}^{n}a_k$, $1{\le}n{\le}N$. There is only a node in the first layer. Algorithm 3 describes the detailed MDHOSVD process. To provide details on the computational process of MDHOSVD, an example of MDHOSVD of a $3rd$-order tensor is shown in Fig. 6 and is described as follows.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2018.2824303, IEEE Transactions on Big Data

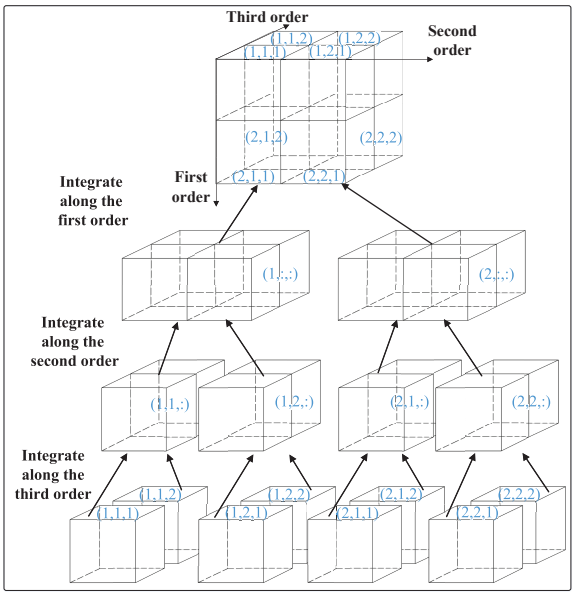


Fig. 6. A Case Study about MDHOSVD of a $3rd$-order Tensor.

Algorithm 3 The Algorithm for MDHOSVD

1: **Input:** Tensor $\mathcal{A}\in R^{I_1\times I_2\times\cdots\times I_n\times\cdots\times I_N}$.
2: **Output:** The high order singular value decomposition of tensor $\mathcal{A}=\mathcal{S}\times_1 U_1\times_2 U_2\times\cdots\times_N U_N$.
3: Divide the tensor $\mathcal{A}$ into a_n parts along the n-th order, $1\leq n\leq N$ and obtain the $\Pi_{n=1}^{N}a_n$ sub-tensors $\mathcal{A}^{(i_1,i_2,\cdots,i_n,\cdots,i_N)}$;
4: Distribute the every sub-tensors to a certain node of the $(N$+1$)$-th layer of the tree;
5: **for** $k=1$ to N **do**
6: In every node of the $(N$+1$)$-th layer, unfold each sub-tensor $\mathcal{A}^{(i_1,i_2,\cdots,i_n,\cdots,i_N)}$ along the k-th order and obtain the unfolding matrix $A_{(k)}^{(i_1,i_2,\cdots,i_n,\cdots,i_N)}$;
7: Implement the Algorithm 1 on the unfolding matrix $A_{(k)}^{(i_1,i_2,\cdots,i_n,\cdots,i_N)}$ and obtain a series of Jacobi rotates J_k;
8: $V_k^{(i_1,i_2,\cdots,i_n,\cdots,i_N)}=\Pi J_k$;
9: $B_k^{(i_1,i_2,\cdots,i_n,\cdots,i_N)}=A_{(k)}^{(i_1,i_2,\cdots,i_n,\cdots,i_N)}\times V_k^{(i_1,i_2,\cdots,i_n,\cdots,i_N)}$;
10: $\sigma_k^i=|B_k^{(i_1,i_2,\cdots,i_n,\cdots,i_N)}(:,i)|$;
11: $U_k^{(i_1,i_2,\cdots,i_n,\cdots,i_N)}(:,i)=\frac{B(:,i)}{\sigma_k^i}$;
12: **end for**
13: **for** $j=N$ to 1 **do**
14: Integrated tensor is obtained by integrating the sub-tensors along the j-th order $\Sigma_{i_j=1}^{a_j}|_j\mathcal{A}^{(i_1,i_2,\cdots,i_j,:,\cdots,:)}$ in the j-th layer;
15: By table lookups, we can find the unfolding integration regulation $case$ of integrated tensor $\Sigma_{i_j=1}^{a_j}|_j\mathcal{A}^{(i_1,i_2,\cdots,i_j,:,\cdots,:)}$ unfold along the k-th order.
16: **for** $k=1$ to N **do**
17: **if** $case$==B or C **then**
18: Execute the computational method for column integration of Algorithm 2 is carried out on the SVD result of the unfolding matrices of the integrated sub-tensors to acquire the SVD result of $\Sigma_{i_j=1}^{a_j}|_jA_{(k)}^{(i_1,i_2,\cdots,i_j,:,\cdots,:)}$;
19: **else**
20: Execute the computational method for row integration of Algorithm 2 is carried out on the SVD result of the unfolding matrices of the integrated sub-tensors to acquire the SVD result of $\Sigma_{i_j=1}^{a_j}|_jA_{(k)}^{(i_1,i_2,\cdots,i_j,:,\cdots,:)}$;
21: **end if**
22: **end for**
23: **end for**
24: Export the computational result of the node in the first layer and left singular matrices U_i, $1\leq i\leq N$ will be obtained;
25: $\mathcal{S}=\mathcal{A}\times_1 U_1^T\times_2 U_2^T\cdots\times_N U_N^T$;
26: $\hat{\mathcal{A}}=\mathcal{S}\times_1 U_1\times_2 U_2\cdots\times_N U_N$;

(**1**) In Algorithm 3, in lines 3 and 4, the Nth-order tensor is divided into $\Pi_{n=1}^{N}a_n$ sub-tensors along every order. The master node (see Fig. 5) distributes each sub-tensor $\mathcal{A}^{(i_1,i_2,\cdots,i_n,\cdots,i_N)}$ to a specific node in the $(N$+1$)$-th layer of the tree. For example, as shown in Fig. 6, the $3rd$-order tensor is divided into $\mathcal{A}^{(1,1,1)}$, $\mathcal{A}^{(1,1,2)}$, $\mathcal{A}^{(1,2,1)}$, $\mathcal{A}^{(1,2,2)}$, $\mathcal{A}^{(2,1,1)}$, $\mathcal{A}^{(2,1,2)}$, $\mathcal{A}^{(2,2,1)}$, and $\mathcal{A}^{(2,2,2)}$, where a_1=a_2=a_3=2. This assumes that the eight sub-tensors are distributed into eight nodes in the fourth layer.

(**2**) In line 6, in every node of the $(N$+1$)$-th layer, unfold each sub-tensor $\mathcal{A}^{(i_1,i_2,\cdots,i_n,\cdots,i_N)}$ along every order to obtain the unfolding matrices $A_{(1)}^{(i_1,i_2,\cdots,i_n,\cdots,i_N)}$, $A_{(2)}^{(i_1,i_2,\cdots,i_n,\cdots,i_N)}$, $\cdots$, $A_{(k)}^{(i_1,i_2,\cdots,i_n,\cdots,i_N)}$, $\cdots$, $A_{(N)}^{(i_1,i_2,\cdots,i_n,\cdots,i_N)}$. For every unfolding matrix $A_{(k)}^{(i_1,i_2,\cdots,i_n,\cdots,i_N)}$, with $1\leq k\leq N$, Algorithm 1 is used to orthogonalize the columns to produce a series of Jacobi rotates, as shown in line 7. Lines 8 to 11 describes how the singular value decomposition of each unfolding matrix is implemented by the Jacobi-based orthogonalization method [8]. These operations (from lines 5 to 12) about different sub-tensor in different node is implemented in parallel. Taking the sub-tensor $\mathcal{A}^{(1,1,1)}$ as an example, and the procedure described in lines 6 to 11 above, the unfolding matrices $A_{(1)}^{(1,1,1)}$, $A_{(2)}^{(1,1,1)}$ and $A_{(3)}^{(1,1,1)}$ with their SVD computational result are obtained. Furthermore, the operations of step (2) about other sub-tensors $\mathcal{A}^{(1,1,2)}$, $\mathcal{A}^{(1,2,1)}$, $\mathcal{A}^{(1,2,2)}$, $\mathcal{A}^{(2,1,1)}$, $\mathcal{A}^{(2,1,2)}$, $\mathcal{A}^{(2,2,1)}$, and $\mathcal{A}^{(2,2,2)}$ in different node are implemented in parallel.

(**3**) In this third step, by integrating the sub-tensors $\mathcal{A}^{(i_1,i_2,\cdots,i_j,:,\cdots,:)}$, $1\leq i_j\leq a_j$, along the j-th, $1\leq j\leq N$ order in the j-th, $1\leq j\leq N$ layer, the integrated tensor $\Sigma_{i_j=1}^{a_j}|_j\mathcal{A}^{(i_1,i_2,\cdots,i_j,:,\cdots,:)}$ is obtained. Concurrently, $a_1\times a_2\times\cdots\times a_{j-1}$ integration groups about j-th order are realized in parallel. For example, the integrated tensor $\mathcal{A}^{(1,1,:)}=\mathcal{A}^{(1,1,1)}|_3\mathcal{A}^{(1,1,2)}$ is acquired by integrating

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2018.2824303, IEEE Transactions on Big Data

the sub-tensors $\mathcal{A}^{(1,1,1)}$ and $\mathcal{A}^{(1,1,2)}$ along the 3-rd order in the third layer of the tree model. In parallel, $\mathcal{A}^{(1,2,:)}$=$\mathcal{A}^{(1,2,1)}|_3\mathcal{A}^{(1,2,2)}$, $\mathcal{A}^{(2,1,:)}$=$\mathcal{A}^{(2,1,1)}|_3\mathcal{A}^{(2,1,2)}$, and $\mathcal{A}^{(2,2,:)}$=$\mathcal{A}^{(2,2,1)}|_3\mathcal{A}^{(2,2,2)}$ are implemented.

$(\mathbf{4})$ According to the tensor blocks unfolding integration regulation, how to find the SVD result of unfolding matrix of the integrated tensor $\Sigma_{i_j=1}^{a_j}|_j\mathcal{A}^{(i_1,i_2,\cdots,i_j,:,\cdots,:)}$ that is composed of the unfolding matrices of the sub-tensors will be described. Then, the Algorithm 2 is carried out on the SVD result of k-th order unfolding matrices of sub-tensors to obtain the SVD result the integrated matrix $\Sigma_{i_j=1}^{a_j}|_jA_{(k)}^{(i_1,i_2,\cdots,i_j,:,\cdots,:)}$, as shown from lines 16 to 22. In fact, Algorithm 2 is implemented on the SVD result of k-th order unfolding matrices ($U_k^{(i_1,i_2,\cdots,i_j,:,\cdots,:)}$, $\Sigma_k^{(i_1,i_2,\cdots,i_j,:,\cdots,:)}$, $V_k^{(i_1,i_2,\cdots,i_j,:,\cdots,:)}$, $U_k^{(i_1,i_2,\cdots,i_j+1,:,\cdots,:)}$, $\Sigma_k^{(i_1,i_2,\cdots,i_j+1,:,\cdots,:)}$, $V_k^{(i_1,i_2,\cdots,i_j+1,:,\cdots,:)}$) one tensor block by one tensor block along the j-th order in each group. In this way, the SVD result of the unfolding matrix of the integrated tensor along the k-th order will be computed. For $a_1\times a_2\times\cdots\times a_{j-1}$ different integrated tensors (every integrated tensor is integrated by a_j sub-tensors, which can be considered as a group), $\Sigma_{i_j=1}^{a_j}|_j\mathcal{A}^{(i_1,i_2,\cdots,i_j,:,\cdots,:)}$, $1\leq j\leq N$, $1\leq i_j\leq a_j$, the operations described from lines 16 to 22 are implemented in parallel. As an example, we illustrate how the tensor $\mathcal{A}^{(1,1,:)}$ is obtained by integrating tensors $\mathcal{A}^{(1,1,1)}$ and $\mathcal{A}^{(1,1,2)}$. According to the tensor block integration regulation, the SVD result of unfolding matrices of $\mathcal{A}^{(1,1,:)}$ including $A_{(1)}^{(1,1,:)}$, $A_{(2)}^{(1,1,:)}$, and $A_{(3)}^{(1,1,:)}$ will be computed by carrying out the Algorithm 2 on the ($A_{(1)}^{(1,1,1)}$, $A_{(1)}^{(1,1,2)}$), ($A_{(2)}^{(1,1,1)}$, $A_{(2)}^{(1,1,2)}$), and ($A_{(3)}^{(1,1,1)}$, $A_{(3)}^{(1,1,2)}$), respectively. Then, the operations on $\mathcal{A}^{(1,2,:)}$, $\mathcal{A}^{(2,1,:)}$, and $\mathcal{A}^{(2,2,:)}$ are carried out in parallel.

$(\mathbf{5})$ In this step, we implement the integration of sub-tensors along the j-th, $1\leq j\leq N$ order in the j-th, $1\leq j\leq N$ layer as shown in step (4), until j equals 1. Then, the left singular matrix U_i, $1\leq i\leq N$ will be exported when the computation is finished. Then, we will get the HOSVD of tensor $\mathcal{A}$ as shown in lines 25 and 26.

From the above representation of MDHOSVD, the whole process involves two main processes, splitting and orthogonalization of each tensor block as shown from lines 3 to 12, and the integration of sub-tensors as shown from lines 13 to 24.

Algorithm 4 for the MIHOSVD: Here, our proposal of a new multi-order incremental HOSVD method to process the incremental HOSVD of high-order tensor streaming is discussed.

An original Nth-order tensor $\mathcal{A}^0\in R^{I_1\times I_2\times\cdots\times I_n\times\cdots\times I_N}$, whose HOSVD result has been computed, increases along several (and even all) orders at the same time to produce an integrated tensor $\mathcal{A}^{'}\in R^{I_1^{'}\times I_2^{'}\times\cdots\times I_n^{'}\times\cdots\times I_N^{'}}$. As shown in Fig. 2 (II), the original tensor is the green part which increases along all orders at the same time. To quickly compute the HOSVD result of the integrated tensor $\mathcal{A}^{'}$ and avoid having to recompute the original part $\mathcal{A}^0$, the multi-order incremental HOSVD is proposed. For convenience, the irregular incremental part is described as $\mathcal{A}^+$. Algorithm 4 describes the detailed process for computing the MIHOSVD.

Algorithm 4 The Algorithm for MIHOSVD

1: **Input:** Incremental part $\mathcal{A}^+$, integrated tensor $\mathcal{A}^{'}\in R^{I_1^{'}\times I_2^{'}\times\cdots\times I_n^{'}\cdots\times I_N^{'}}$, and the HOSVD result of each original tensor blocks.
2: **Output:**The high order singular value decomposition of integrated tensor $\mathcal{A}^{'}$=$\mathcal{S}^{'}\times_1U_1^{'}\times_2U_2^{'}\times\cdots\times_NU_N^{'}$.
3: Supposing the Nth-order original tensor $\mathcal{A}^0$ and the incremental part $\mathcal{A}^+$ are divided as $(a_1^0,a_2^0,\cdots,a_n^0,\cdots,a_N^0)$, $(a_1^+,a_2^+,\cdots,a_n^+,\cdots,a_N^+)$, respectively. The integrated tensor $\mathcal{A}^{'}$=($\mathcal{A}^0$+$\mathcal{A}^+$) is divided as ($a_1^{'}$=$(a_1^0+a_1^+)$, $a_2^{'}$=$(a_2^0+a_2^+)$,$\cdots$, $a_n^{'}$=$(a_n^0+a_n^+)$,$\cdots$, $a_N^{'}$=$(a_N^0+a_N^+)$).
4: Distribute every block $(\mathcal{A}^+)^{(i_1^+,i_2^+,\cdots,i_n^+,\cdots,i_N^+)}$ to the node of the $(N$+1$)$-th layer;
5: **for** $k=1$ to N **do**
6: In every node of the $(N$+1$)$-th layer, implement the similar operation from line 7 to line 11 of Algorithm 3 on matrix $(A^+)_{(k)}^{(i_1^+,i_2^+,\cdots,i_n^+,\cdots,i_N^+)}$ and obtain its SVD result ($V_k^{(i_1^+,i_2^+,\cdots,i_n^+,\cdots,i_N^+)}$,$U_k^{(i_1^+,i_2^+,\cdots,i_n^+,\cdots,i_N^+)}$, $\Sigma_k^{(i_1^+,i_2^+,\cdots,i_n^+,\cdots,i_N^+)}$);
7: **end for**
8: **for** $j=N$ to 1 **do**
9: Integrated tensor is obtained by integrating the sub-tensors along the j-th order in the j-th layer $\Sigma_{i_j^{'}=1}^{a_j^{'}}|_j\mathcal{A}^{(i_1^{'},i_2^{'},\cdots,i_j^{'},:,\cdots,:)}$;
10: By table lookups, we can find the unfolding integration regulation $case$ of integrated tensor $\Sigma_{i_j^{'}=1}^{a_j^{'}}|_j\mathcal{A}^{(i_1^{'},i_2^{'},\cdots,i_j^{'},:,\cdots,:)}$ unfold along the k-th order.
11: **for** $k=1$ to N **do**
12: The similar operation from line 17 to line 21 of Algorithm 3 is carried out to obtain the SVD result of the unfolding matrices $\Sigma_{i_j^{'}=1}^{a_j^{'}}|_jA_{(k)}^{(i_1^{'},i_2^{'},\cdots,i_j^{'},:,\cdots,:)}$;
13: **end for**
14: **end for**
15: Export the computational result of the node in the first layer and left singular matrices $U_i^{'}$, $1\leq i\leq N$ will be obtained;
16: $\mathcal{S}^{'}=\mathcal{A}^{'}\times_1U_1^{'T}\times_2U_2^{'T}\cdots\times_NU_N^{'T}$;
17: $\hat{\mathcal{A}}^{'}=\mathcal{S}^{'}\times_1U_1^{'}\times_2U_2^{'}\cdots\times_NU_N^{'}$;

$(\mathbf{1})$ In Algorithm 4, line 3 shows that the original tensor $\mathcal{A}^0$ and its incremental part $\mathcal{A}^+$ are divided into several blocks along all orders. For example, the a_n^0 and a_n^+ means the original tensor $\mathcal{A}^0$ and the incremental

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2018.2824303, IEEE Transactions on Big Data

part $\mathcal{A}^+$ are divided into a_n^0, a_n^+ blocks along the n-th order, respectively. With the same block method of $\mathcal{A}^0$ and $\mathcal{A}^+$, the integrated tensor $\mathcal{A}^{'}$ integrates both components together.

$(\mathbf{2})$ Distribute every block $(\mathcal{A}^+)^{(i_1^+,i_2^+,\cdots,i_n^+,\cdots,i_N^+)}$, $1{\le}i_n^+{\le}a_n^+$, $1{\le}n{\le}N$ to the node of the $(N{+}1)$-th layer. In every node of the $(N{+}1)$-th layer, the tensor block $(\mathcal{A}^+)^{(i_1^+,i_2^+,\cdots,i_n^+,\cdots,i_N^+)}$ is unfolded along all the orders and the unfolding matrices $(A^+)_{(k)}^{(i_1^+,i_2^+,\cdots,i_n^+,\cdots,i_N^+)}$, $1{\le}k{\le}N$ are obtained. Then, the Algorithm 1 is carried out on the unfolding matrix $(A^+)_{(k)}^{(i_1^+,i_2^+,\cdots,i_n^+,\cdots,i_N^+)}$ to obtain a series of Jacobi rotates. Next, the SVD result of each unfolding matrix such as $(V^+)_k^{(i_1^+,i_2^+,\cdots,i_n^+,\cdots,i_N^+)}$, $(U^+)_k^{(i_1^+,i_2^+,\cdots,i_n^+,\cdots,i_N^+)}$ is computed.

$(\mathbf{3})$ With the same block method of $\mathcal{A}^0$ and $\mathcal{A}^+$, the integrated tensor $\mathcal{A}^{'}$ is divided, this means that $i_n^{'}{=}i_n^0$ when $i_n{\le}a_n^0$, and $i_n^{'}{=}a_n^0{+}i_n^+$ when $a_n^0{<}i_n{\le}a_n^0{+}a_n^+$. Integrate the sub-tensors $\Sigma_{i_j^{'}=1}^{a_j^{'}}|_j\mathcal{A}^{'(i_1^{'},i_2^{'},\cdots,i_j^{'},:,\cdots,:)}$ in the same way as described in step (3) of Algorithm 3 for MDHOSVD.

$(\mathbf{4})$ According to the tensor blocks unfolding integration regulation, the unfolding matrices of the integrated tensor $\Sigma_{i_j^{'}=1}^{a_j^{'}}|_j\mathcal{A}^{(i_1^{'},i_2^{'},\cdots,i_j^{'},:,\cdots,:)}$ will be found. Then, the Algorithm 2 is carried out on the $\Sigma_{i_j^{'}=1}^{a_j^{'}}|_jA_{(k)}^{(i_1^{'},i_2^{'},\cdots,i_j^{'},:,\cdots,:)}$. This step is similar to the step (4) of Algorithm 3 for MDHOSVD.

$(\mathbf{5})$ Implement the integration of sub-tensors along the k-th order in the k-th layer until the k equals to 1. Then, the left singular matrix $U_i^{'}$, $1{\le}i{\le}N$ will be computed, when the computational process is finished.

Similarly, the proposed MIHOSVD also has two main processes, splitting and orthogonalization of the each tensor block of incremental part as shown from lines 3 to 7 of Algorithm 4. And the second process is the integration of the original tensor and the incremental part as shown from lines 8 to 17 of Algorithm 4.

3.3 Algorithm Analysis

In this sub-section, the computational and communication complexities of the proposed MDHOSVD and MIHOSVD are analyzed.

Computational Complexity for the MDHOSVD: As mentioned before, MDHOSVD involves two processes - splitting and orthogonalization of each tensor block, and integration of sub-tensors. The proposed MDHOSVD is based on the Jacobi orthogonalization. Consider the Jacobi orthogonalization method described in Algorithm 1 for a matrix $A{\in}R^{m\times n}$. For this example, both of the main computational cost are of $O(m)$, for Jacobi rotation $J(i,j)$ of every column pair and for $AJ(i,j)$, respectively.

In the first process of MDHOSVD (splitting and orthogonalization of each tensor block), for a certain tensor block $\mathcal{A}^{(i_1,i_2,\cdots,i_n,\cdots,i_N)}{\in}R^{I_1^{i_1}\times I_2^{i_2}\times\cdots\times I_n^{i_n}\cdots\times I_N^{i_N}}$, the unfolding matrix along the k-th order is $A_{(k)}^{(i_1,i_2,\cdots,i_n,\cdots,i_N)}{\in}R^{I_k^{i_k}\times\Pi_{n=1,n\neq k}^N I_n^{i_n}}$, $1{\le}k{\le}N$. Thus, the number of column pairs for matrix $A_{(k)}^{(i_1,i_2,\cdots,i_n,\cdots,i_N)}$ is $\frac{1}{2}\times\Pi_{n=1,n\neq k}^N I_n^{i_n}(\Pi_{n=1,n\neq k}^N I_n^{i_n}-1)$. Therefore, the main computational cost for the matrix $A_{(k)}^{(i_1,i_2,\cdots,i_n,\cdots,i_N)}$ is $S_k^{(i_1,i_2,\cdots,i_n,\cdots,i_N)}\times(I_k^{i_k}{+}I_k^{i_k})\times\frac{1}{2}\times\Pi_{n=1,n\neq k}^N I_n^{i_n}$ $(\Pi_{n=1,n\neq k}^N I_n^{i_n}{-}1){\approx}S_k^{(i_1,i_2,\cdots,i_n,\cdots,i_N)}I_k^{i_k}(\Pi_{n=1,n\neq k}^N I_n^{i_n})^2$, where, the $S_k^{(i_1,i_2,\cdots,i_n,\cdots,i_N)}$ is the number of sweep in the orthogonalization process of matrix $A_{(k)}^{(i_1,i_2,\cdots,i_n,\cdots,i_N)}$, and $I_k^{i_k}$, $\Pi_{n=1,n\neq k}^N I_n^{i_n}$ are the number of rows and columns of targeted matrix, respectively. Because the orthogonalization computation of each tensor block is parallelly implemented on each node of the $(N{+}1)$-th layer of the tree. The computational cost for an Nth-order tensor block $\mathcal{A}^{(i_1,i_2,\cdots,i_n,\cdots,i_N)}$ is also that of the first process $T_{comp}^{D1}{=}\Sigma_{k=1}^N[S_k^{(i_1,i_2,\cdots,i_k,\cdots,i_n,\cdots,i_N)}\times(I_k^{i_k}{+}I_k^{i_k})\times\frac{1}{2}\times\Pi_{n=1,n\neq k}^N I_n^{i_n}(\Pi_{n=1,n\neq k}^N I_n^{i_n}{-}1)]$.

The second process of MDHOSVD is the integration of sub-tensors, which is implemented with Algorithm 2. According to the integration operation mentioned in MDHOSVD, for an integrated tensor $\Sigma_{i_N=1}^{a_N}|_N\mathcal{A}^{(i_1,i_2,\cdots,i_n,\cdots,i_{N-1},i_N)}{=}\mathcal{A}^{(i_1,i_2,\cdots,i_n,\cdots,i_{N-1},:)}$ in the N-th layer of the tree, we could find that $\mathcal{A}^{(i_1,i_2,\cdots,i_n,\cdots,i_{N-1},:)}{\in}R^{I_1^{i_1}\times I_2^{i_2}\times\cdots I_n^{i_n}\times\cdots I_{N-1}^{i_{N-1}}\times I_N}$ whose unfolding matrices are $A_{(k)}^{(i_1,i_2,\cdots,i_n,\cdots,i_{N-1},:)}{\in}$ $R^{I_k^{i_k}\times\Pi_{n=1,n\neq k}^{N-1}I_n^{i_n}I_N}$, unfolded along the k-th, $k{\neq}N$ order or $A_{(N)}^{(i_1,i_2,\cdots,i_n,\cdots,i_{N-1},:)}{\in}R^{I_N\times\Pi_{n=1}^{N-1}I_n^{i_n}}$, unfolded along the N-th order. According to the tensor block unfolding integration regulation as shown in Fig. 3, we could find the unfolding matrix of the integrated tensor along the N-th order is the $caseA$. Then, its transposed matrix is $(A_{(N)}^{(i_1,i_2,\cdots,i_n,\cdots,i_{N-1},:)})^T{\in}R^{\Pi_{n=1}^{N-1}I_n^{i_n}\times I_N}$. Because the integrated computational operation implemented in the N-th layer of tree is carried out in parallel in different nodes. This means that the computational cost of the integrated tensor $\Sigma_{i_N=1}^{a_N}|_N\mathcal{A}^{(i_1,i_2,\cdots,i_n,\cdots,i_N)}$ is also the computational cost of the N-th layer of tree. The computational cost of the integration tensor in the N-th layer of the tree is approximately equal to $T_{comp}^N{\approx}S_N^{(i_1,i_2,\cdots,i_n,\cdots,i_{N-1},:)}\frac{1}{2}I_N(I_N{-}1)(\Pi_{n=1}^{N-1}I_n^{i_n}{+}$ $\Pi_{n=1}^{N-1}I_n^{i_n}){+}\Sigma_{k=1}^{N-1}[\frac{1}{2}\Pi_{n=1,n\neq k}^{N-1}I_n^{i_n}I_N(\Pi_{n=1,n\neq k}^{N-1}I_n^{i_n}I_N{-}1)$ $(I_k^{i_k}{+}I_k^{i_k})S_k^{(i_1,i_2,\cdots,i_n,\cdots,i_{N-1},:)}]{-}\Sigma_{i_N=1}^{a_N}T_{comp}^{D1}$, where, the sweep number for the orthogonalization of matrices $A_{(N)}^{(i_1,i_2,\cdots i_n,\cdots,i_{N-1},:)}$ and $A_{(k)}^{(i_1,i_2,\cdots i_n,\cdots,i_{N-1},:)}$ are $S_N^{(i_1,i_2,\cdots i_n,\cdots,i_{N-1},:)}$ and $S_k^{(i_1,i_2,\cdots i_n,\cdots,i_{N-1},:)}$, respectively.

In the similar way, the computational cost of the integration operation of the second process in the n_0-th layer of tree is $T_{comp}^{Dn_0}{\approx}\Sigma_{k=1,k\neq n_0}^N[\frac{1}{2}\Pi_{k=1}^{n_0-1}I_k^{i_k}\Pi_{k=n_0+1}^N I_k(\Pi_{k=1}^{n_0-1}I_k^{i_k}$ $\Pi_{k=n_0+1}^N I_k{-}1)(I_{n_0}{+}I_{n_0})S_k^{(i_1,i_2,\cdots,i_{n_0-1},:,\cdots,:)}]{+}\frac{1}{2}I_{n_0}(I_{n_0}$ $-1)(2\Pi_{k=1}^{n_0-1}I_k^{i_k}\Pi_{k=n_0+1}^N I_k)S_{n_0}^{(i_1,i_2,\cdots,i_{n_0-1},:,\cdots,:)}-$ $\Sigma_{i_{n_0}}^{a_{n_0}}T_{comp}^{D(n_0+1)}$. Also, the computational cost of the integration operation of the integration operation is $T_{comp}^{D2}{=}\Sigma_{n_0=1}^N T_{comp}^{Dn_0}$. Thus, the computational complexity

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2018.2824303, IEEE Transactions on Big Data

of MDHOSVD is the summation of that of the first and second process T^{D}_{comp}=T^{D1}_{comp}+T^{D2}_{comp}.

Computational Complexity for the MIHOSVD: As described in Algorithm 4, MIHOSVD is a MDHOSVD-based improved computational method for tensor stream $\mathcal{A}^{'}$. The computational cost of $T^{I\mathcal{A}^{'}}_{comp}$ equals to $T^{I\mathcal{A}^{'}}_{comp}$≈$T^{D\mathcal{A}^{'}}_{comp}$−$T^{D\mathcal{A}}_{comp}$, where $T^{D\mathcal{A}^{'}}_{comp}$, $T^{D\mathcal{A}}_{comp}$ are the computational complexity for the MDHOSVD of tensors $\mathcal{A}^{'}$ and $\mathcal{A}$, respectively.

Communication Complexity for the MDHOSVD: The communication time is t_{comm}=$t_{startup}$+$n_{data}t_{data}$, in which $t_{startup}$ is the startup time, t_{data} is the transmission time of every data word, and the n_{data} is the volume of data words [26]. In the first process, the volume of transmission data is $\Pi^{N}_{k=1}I_k$, and there are $\Pi^{N}_{k=1}a_k$ sub-tensor blocks sent to the nodes of the $(N$+1$)$-th layer. With the broadcast, the communication complexity of the first process is T^{D1}_{comm}=$t_{startup}$+$\Pi^{N}_{k=1}I_k t_{data}$.

There are N times transferring sub-integration processes in the integration process, in which the first one is that transfers the data from the $(N$+1$)$-th layer to the N-th layer. There are $\Pi^{N}_{n=1}a_n$ nodes divided into $\Pi^{N-1}_{n=1}a_n$ groups in the $(N$+1$)$-th layer. In every group, there are a_N nodes and every one processes a sub-tensor blocks $\mathcal{A}^{(i_1,i_2,\cdots,i_N)}$, in which the size of matrix U_k and V_k is $\Pi^{N}_{n=1}I^{i_n}_{n}$, $(\frac{\Pi^{N}_{n=1}I^{i_n}_{n}}{I^{i_k}_{k}})^2$ unfolding along the k-th order, respectively. The nodes in different groups transfer the data in parallel, but the nodes in the same group transfer the data in serial communication form. The communication complexity of the first time of the integration process from the $(N$+1$)$-th layer to the N-th layer is, $T^{D2_1}_{comm}$=$a_N[t_{startup}$+$N\Pi^{N}_{n=1}I^{i_n}_{n}$+$\Sigma^{N}_{k=1}(\frac{\Pi^{N}_{n=1}I^{i_n}_{n}}{I^{i_k}_{k}})^2]t_{data}$. Also, the l-th time transferring, $2\leq l\leq N$ in the integration process transfers the data from the $(N$+2$)-l$ layer to the $(N$+1$)-l$ layer. Furthermore, the size of the matrix U_k is $(\Pi^{(N+1)-l}_{n=1}I^{i_n}_{n}\Pi^{N}_{n=(N+2)-l}I_n)$. And, the size of the matrix V_k is $(\frac{\Sigma^{(N+1)-l}_{n=1}I^{i_n}_{n}\Pi^{N}_{n=(N+2)-l}I_n}{I^{i_k}_{k}})^2$, $1\leq k\leq(N+1)-l$, or $(\frac{\Pi^{(N+1)-l}_{n=1}I^{i_n}_{n}\Pi^{N}_{n=(N+2)-l}I_n}{I_k})^2$, $(N+2)-l\leq k\leq N$. In the similar way, the communication complexity of the l-th time transferring is, $T^{D2_l}_{comm}$=$a_{N+1-l}[t_{startup}$+$N(\Pi^{(N+1)-l}_{n=1}I^{i_n}_{n}\Pi^{N}_{n=(N+2)-l}I_n)$ $+\Pi^{(N+1)-l}_{k=1}(\frac{\Sigma^{(N+1)-l}_{n=1}I^{i_n}_{n}\Pi^{N}_{n=(N+2)-l}I_n}{I^{i_k}_{k}})^2+\Sigma^{N}_{k=(N+2)-l}$ $(\frac{\Pi^{(N+1)-l}_{n=1}I^{i_n}_{n}\Pi^{N}_{n=(N+2)-l}I_n}{I_k})^2]t_{data}$.

The communication time of MDHOSVD is T^{D}_{comm}=T^{D1}_{comm}+$T^{D2_1}_{comm}$+$\Sigma^{N}_{l=2}T^{D2_l}_{comm}$.

Communication Complexity for the MIHOSVD: In the same way, it's straightforward to obtain the communication complexity of the MIHOSVD with T^{I}_{comm}=T^{I1}_{comm}+T^{I2}_{comm}, where the T^{I1}_{comm} and T^{I2}_{comm} are the communication complexity of the first and second process, respectively. The former is related to the incremental part $\mathcal{A}^{+}$ and the latter is related to the integrated tensor $\mathcal{A}^{'}$.

4 Experiment and Simulation

In this section, experiments and simulations are carried out to study the performance of the proposed MDHOSVD and MIHOSVD. The experiments are carried out on a distributed system with 80 usable cores in 20 computers. Each computer, also referred as a node, has an Intel Core i5 CPU with four cores, 8GB memory based on the 64-bit Linux operating system. Also, the data used in the experiments are generated randomly. The simulations are implemented by SimGrid [1], which is a scientific simulation tool to study the behavior of large-scale distributed systems. To study the performance of the MDHOSVD and MIHOSVD, the following evaluation criteria are used - computational error, improvement factor and improvement factor ratio that are explained in sub-section 4.1 and 4.2 below.

4.1 Error Measurement

To measure the accuracy of proposed algorithms, the error is defined as follows: Supposing U_i, $1\leq i\leq N$ and $\mathcal{S}$ are the HOSVD results of an Nth-order tensor $\mathcal{A}\in R^{I_1\times I_2\times\cdots\times I_N}$, then the error is E_{error}=$\|\mathcal{A}-\mathcal{S}\times_1 U_1\times_2 U_2\cdots\times_N U_N\|$ [8], [10], [11]. Without loss of generality, we compute the error on the low order test tensor ($3rd$-order tensor) and the high order test tensor ($4th$-order tensor) and the experimental results are shown in Fig. 7.

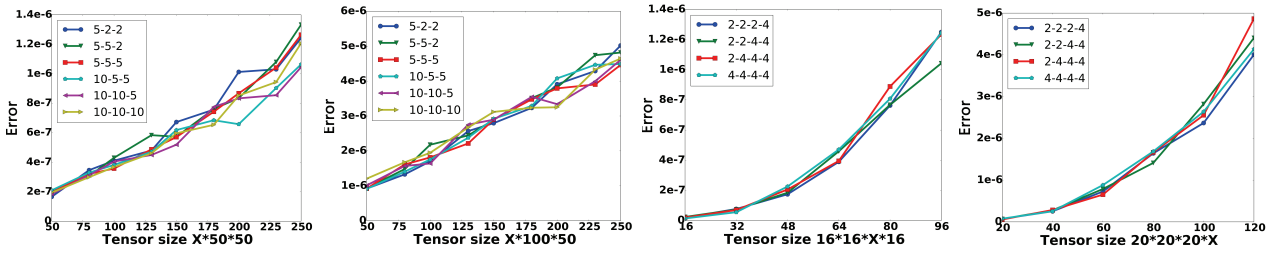
Take the Fig. 7 (a) as an example, the horizontal axis represents the dimensionality of one order represented as "X" for test tensor "X*50*50". For example, the dimensionality of the first order of test tensor "X*50*50" is 50,75,100,125,150,175,200,225,250, respectively. The ordinate axis represents the error obtained from the experiment. Furthermore, the "5-2-2" in the label of the Fig. 7 (a) means that the first order, second order and third order of the test tensor "X*50*50" are cut into "5", "2" and "2" parts, respectively. In other words, the a_1=5, a_2=2 and a_3=2. As shown in Fig. 7, the errors increase with the size of the data set. According to the analysis of Jacobi orthogonalization method, the factors influencing the error are the convergence coefficient ε and the whole data set of test tensor $\mathcal{A}$ [8], [10], [11], [27].

4.2 Improvement Factor and Improvement Factor Ratio Measurement

In this subsection, we quantify the performance of MDHOSVD and MIHOSVD by the improvement factor and improvement factor ratio, respectively. Times t_1 and t_D are the execution times for HOSVD of the Nth-order test tensor $\mathcal{A}$ by the traditional one-sided Jacobi-based SVD method and the MDHOSVD scheme, respectively. Also, times t_2 and t_I are the execution times for the HOSVD of the Nth-order test tensor stream $\mathcal{A}^0$+$\mathcal{A}^+$ by the MDHOSVD and the MIHOSVD schemes. The execution times are collected from the distributed system

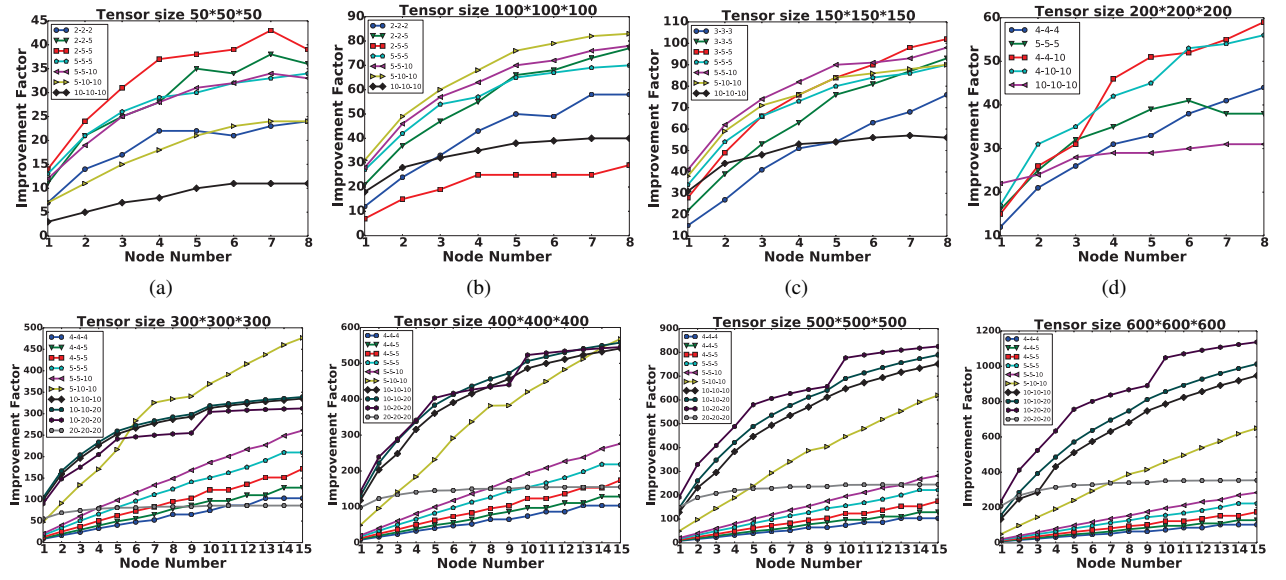
1. http://simgrid.gforge.inria.fr/

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2018.2824303, IEEE Transactions on Big Data



(a) (b) (c) (d)

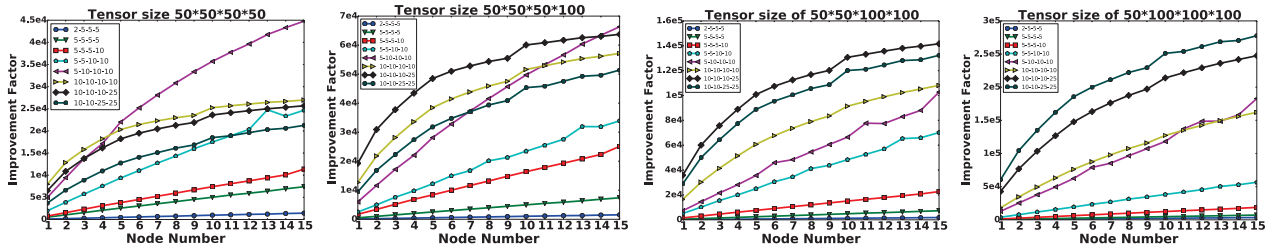
Fig. 7. The Error Measurement



(a) (b) (c) (d)

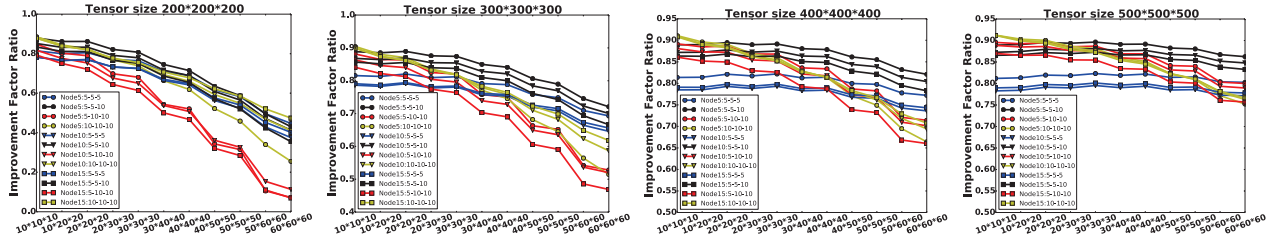
(e) (f) (g) (h)

Fig. 8. The Improvement Factors of $3rd$-order Test Tensors



(a) (b) (c) (d)

Fig. 9. The Improvement Factors of $4th$-order Test Tensors



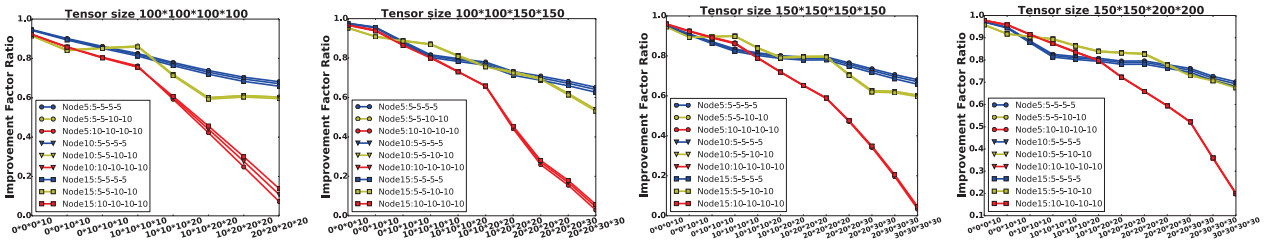
(a) (b) (c) (d)

Fig. 10. The Improvement Factor Ratios of $3rd$-order Test Tensors

and simulations on SimGrid. The ratio $\frac{t_D}{t_1}$ is the MD-HOSVD improvement factor, while the ratio $\frac{t_2-t_I}{t_2}$ is the MIHOSVD improvement factor.

The relationships among test tensor size, improvement factor, node number and blocked methods of the $3rd$-order test tensors and $4th$-order test tensors are shown

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2018.2824303, IEEE Transactions on Big Data



(a) (b) (c) (d)

Fig. 11. The Improvement Factor Ratios of $4th$-order Test Tensors

in Figs. 8 and 9, respectively. The data of the first four sub-figs (Fig. 8 (a), (b), (c) and (d)) are obtained from experiments on the distributed system in which each node has 4 cores. Other data in the Figs. 8 and 9 are collected from simulations on SimGrid, in which each node has a core. The main observations on the performance from Figs. 8 and 9 are as follows: (i) For a given test tensor, the performance initially increases with node number and then tends to saturate as the node number becomes large. (ii) For a given test tensor, with the size of tensor sub-block decreases, the performance factor first increases, but as the sub-block size decreases further, the performance factor now also decreases. (iii) For a given test tensor, there is a optimal blocked method which corresponds to a optimal improvement factor. However, the optimal blocked method for this given test tensor may have fluctuations in the optimal improvement factor, as shown in (e) and (f) of Fig. 8. Taking Fig. 8 (e) as an example, the distributed system with 4 nodes to 24 nodes has the optimal blocked method 10-10-20, then the optimal blocked method becomes 5-5-10 for the distributed system with 24 nodes to 60 nodes. (iv) With the increase in test tensor size, the number of sub-blocks in the tensor from the optimal blocked method becomes larger and larger. For example, the optimal blocked method of a given test tensor with size $300 * 300 * 300$ for the distributed system with 24 nodes to 60 nodes is 5-5-10, then that of a given test tensor with size $600 * 600 * 600$ is 10-20-20.

Following the same analysis method in the previous paragraph, relationships among test tensor size, improvement factor ratio, node number and blocked methods of the $3rd$-order test tensors and $4th$-order test tensors for MIHOSVD are also demonstrated in Figs. 10 and 11, respectively. As shown in Fig. 10 (a), the original test tensor size is $200 * 200 * 200$, the incremental parts are $10 * 10 * 10$, $10 * 20 * 20$, $20 * 20 * 20$, $20 * 30 * 30$, $30 * 30 * 30$, $30 * 40 * 40$, $40 * 40 * 40$, $40 * 50 * 50$, $50 * 50 * 50$, $50 * 60 * 60$, and $60 * 60 * 60$, respectively. Taking the first three incremental parts as an example, the size of updated tensors will be $210 * 210 * 210$, $210*220*220$, and $220*220*220$, respectively. To measure the performance of MIHOSVD, simulations are carried out on the updated tensor using both MDHOSVD and MIHOSVD algorithms, then the main observations from this simulation group are as follows: (i) with the increase of the incremental data size, the improvement factor ratio is trending downwards. (ii) with the same node number, the improvement factor ratio increases at first, but then decreases with the increase of the incremental data size. (iii) with the same incremental data size, the improvement factor ratio increases first and thereafter decreases with increase in the node number.

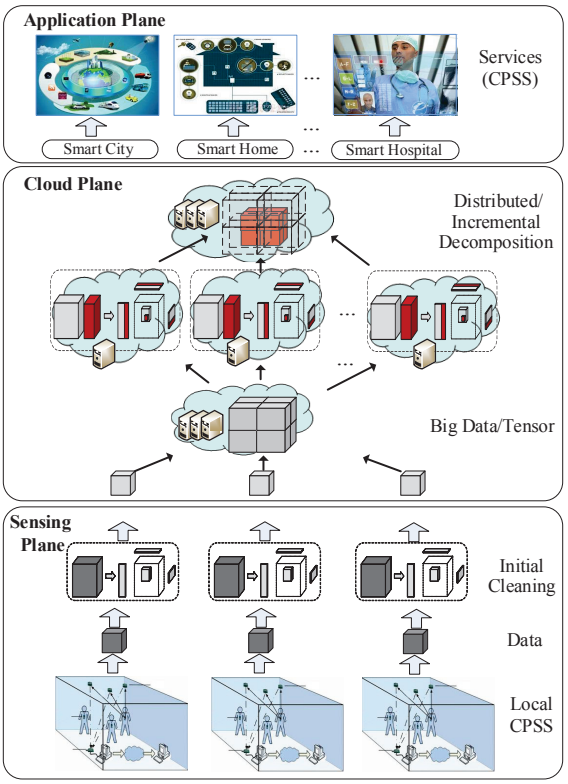
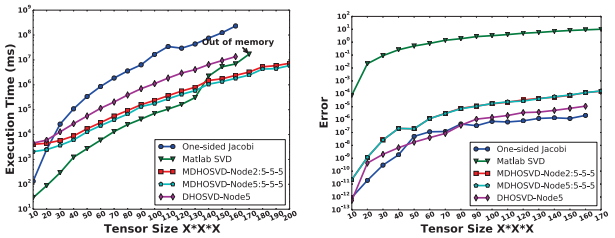


Fig. 12. The Big Data-as-a-Service Framework.

5 A Case Study on CPSS Data

In this section, we proposed a case study about Big Data-as-a-Service framework including the sensing plane, cloud plane and application plane, which is similar to the ones proposed in [9], [28]. In the sensing plane, data are collected and represented from different local CPSS. After initial cleaning, the distributed and incremental computation of the big data are processed in the cloud plane. The extracted high quality data in the cloud plane is then used in the application plane to provide services such as optimal

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2018.2824303, IEEE Transactions on Big Data



(a) (b)

Fig. 13. The Performance Comparison of the CPSS Case Study Data

routing prediction in smart traffic or predicting the medical and evolving living needs of the elderly in a smart home [29], [30].

In this paper, we mainly focused on the distributed and incremental decomposition of the data on the cloud plane, resulting in accelerated services providing in CPSS. Two evaluation factors, the execution time and the error, were selected to compare the performance of one-sided Jacobi SVD, Matlab SVD (including many functions such as "$tenmat$", "ttm" and "$ttensor$"), DHOSVD and MDHOSVD on the test data of several liver cancer patients.

Next, using the same tensorization method proposed in [8], a $3rd$-order tensor is constructed based on the test CPSS data. Taking the CT image data tensor as an example. Using the patient's CT image data, a $3rd$-order tensor $\mathcal{A}\in R^{200*200*200}$ is constructed in which the first order represents the number of CT image. The second and third orders are used to indicate a CT image and the data in this tensor are the gray-scale values of each CT image. We compare the performance metrics - error, execution time of one-sided Jacobi SVD, Matlab SVD, DHOSVD and MDHOSVD in the Fig. 13.

As shown in Fig. 13 (a), the horizontal axis is the size $X*X*X$ of a $3rd$-order tensor, whose three orders are the same. For example, the first number of the horizontal axis, 10, means a $3rd$-order tensor $10\times10\times10$. The execution time of MDHOSVD approaches that of the Matlab SVD as the tensor size increases. When the tensor size is over $140*140*140$, the execution time of MDHOSVD becomes less than that of the Matlab SVD. Further, when the tensor size is over than $170*170*170$, the Matlab SVD cannot be executed because its memory requirements are more than available in the computer. The execution time of the one-sided Jacobi SVD is much larger than either MDHOSVD or the Matlab SVD. And the execution time of the DHOSVD is larger than that of MDHOSVD as well. Similarly, from the Fig. 13 (b), the error performance of MDHOSVD, DHOSVD and one-sided Jacobi SVD are significantly better than that of Matlab SVD. Note that the error in MDHOSVD is larger than that of the one-sided Jacobi and DHOSVD because there is some error accumulations in the computation.

6 Conclusion

In this paper, an improved mathematical representation of tensor unfolding method to make it more convenient and easy to understood was described. Then, a multi-order distributed high order singular value decomposition method (MDHOSVD) and a multi-order incremental high order singular value decomposition method (MIHOSVD) were proposed. The computational and communication complexities of both MDHOSVD and MIHOSVD were analyzed. The performance of MDHOSVD and MIHOSVD were quantified using the criteria of error, improvement factor and improvement factor ratio form detailed measurements. Finally, a case study about service providing in CPSS based on data processing was proposed and discussed using the quantitative metrics of execution time and error performance to highlight the benefits of MDHOSVD and MIHOSVD.

However, from the working process of the proposed MDHOSVD and MIHOSVD, we could find these nodes in the high layers of the tree have heavy workloads, which may bring negative effects for these algorithms. Therefore, how can we reduce the workload of these nodes in the high layers to improve the computational efficiency is one of the main challenges. Also, from the performance of execution time, we could find several other factors such as blocked methods, node number and tensor size play important roles in the execution time. In the same way, how can we provide an optimization model to optimize allocation of these factors is another challenging question to be studied in the future.

References

[1] L. T. Yang, "A Data-as-a-Service Framework for IoT Big Data," in *Proceedings of the 2013 International Conference on Security and Cryptography*, Reykjavik, Iceland, Jul. 29-31 2013, pp. IS–5.

[2] H. Ma, "Internet of Things: Objectives and Scientific Challenges," *Journal of Computer Science and Technology*, vol. 26, no. 6, pp. 919–924, 2011.

[3] Z. Liu, D. Yang, D. Wen, W. Zhang, and W. Mao, "Cyber-Physical-Social Systems for Command and Control," *IEEE Intelligent Systems,*, vol. 26, no. 4, pp. 92–96, 2011.

[4] P. Barnaghi, A. Sheth, V. Singh, and M. Hauswirth, "Physical-Cyber-Social Computing: Looking Back, Looking Forward," *IEEE Internet Computing*, vol. 19, no. 3, pp. 7–11, 2015.

[5] J. Zeng, L. T. Yang, H. Ning, and J. Ma, "A Systematic Methodology for Augmenting Quality of Experience in Smart Space Design," *IEEE Wireless Communications*, vol. 22, no. 4, pp. 81–87, 2015.

[6] X. Liu, M. Dong, K. Ota, P. Hung, and A. Liu, "Sercice Pricing Decision in Cyber-Physical Systems: Insights from Game Theory," *IEEE Transactions on Services Computing*, vol. 9, no. 2, pp. 186–198, 2016.

[7] H. Li, M. Dong, K. Ota, and M. Guo, "Pricing and Repurchasing for Big Data Processing in Multi-clouds," *IEEE Transactions on Emerging Topics in Computing*, vol. 4, no. 2, pp. 266–277, 2016.

[8] X. Wang, W. Wang, L. T. Yang, S. Liao, and D. Yin, "A Distributed HOSVD Method with its Incremental Computation for Big Data in Cyber-Physical-Social Systems," *IEEE Transactions on Computational Social System, DOI: 10.1109/TCSS.2018.2813320*, 2016.

[9] X. Wang, L. T. Yang, H. Liu, and M. J. Deen, "A Big Data-as-a-Service Framework: State-of-the-art and Perspectives," *IEEE Transactions on Big Data*, 2017, DOI10.1109/TBDATA.2017.2757942.

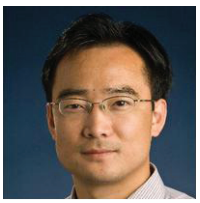
[10] L. Kuang, F. Hao, L. T. Yang, M. Lin, C. Luo, and G. Min, "A Tensor-based Approach for Big Data Representation and Dimensionality Reduction," *IEEE Transactions on Emerging Topics in Computing*, vol. 2, no. 3, pp. 280–291, 2014.

[11] G. Golub and C. V. Loan, *Matrix Computations (4th)*. Johns Hopkins Press, 2012.
[12] P. Symeonidis, A. Nanopoulos, and Y. Manolopoulos, "A Unified Framework for Providing Recommendations in Social Tagging Systems Based on Ternary Semantic Analysis," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 2, pp. 179–192, 2010.
[13] J. Melenchon and E. Martínez, "Efficiently Downdating, Composing and Splitting Singular Value Decompositions Preserving the Mean Information," in *Pattern Recognition and Image Analysis*. Springer, 2007, pp. 436–443.
[14] V. Strumpen, H. Hoffmann, and A. Agarwal, "A Stream Algorithm for the SVD," *Computer Science and Artificial Intelligence Laboratory Technical Report, MIT*, Oct. 22, 2003.
[15] H. Snopce and I. Spahiu, "Parallel Computation of the SVD," in *Proceedings of the 5th European Conference on European Computing Conference*, Stevens Point, USA, 2011, pp. 456–461.
[16] J. Liang, Y. He, D. Liu, and X. Zeng, "Image Fusion Using Higher Order Singular Value Decomposition," *IEEE Transactions on Image Processing*, vol. 21, no. 5, pp. 2898–2909, 2012.
[17] M. A. O. Vasilescu and D. Terzopoulos, "Multilinear Analysis of Image Ensembles:TensorFaces," in *Proceedings of the 2002 European Conference on Computer Vision*, Copenhagen, Denmark, May 28-31, 2002, pp. 1–7.
[18] Z. Zhang, G. Ely, S. Aeron, N. Hao, and M. Kilmer, "Novel Methods for Multilinear Data Completion and De-noising Based on Tensor-SVD," in *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, Columbus, USA, Jun. 23-28, 2014, pp. 3842–3849.
[19] M. A. O. Vasilescu, "Human Motion Signatures: Analysis, Synthesis, Recognition," in *Proceedings of the 16th International Conference on Pattern Recognition*, Quebec, Canada, Aug. 11-15, 2002, pp. 456–460.
[20] T. G. Kolda and B. W. Bader, "Tensor Decompositions and Applications," *SIAM Review*, vol. 51, no. 3, pp. 455–500, 2009.
[21] L. De Lathauwer, B. De Moor, and J. Vandewalle, "A Multilinear Singular Value Decomposition," *SIAM Journal on Matrix Analysis and Applications*, vol. 21, no. 4, pp. 1253–1278, 2000.
[22] S. Pouryazdian, S. Beheshti, and S. Krishnan, "Localization of Brain Activities Using Multiway Analysis of EEG Tensor via EMD and Reassigned TF Representation," in *Proceedings of the 37th IEEE Annual International Conference of Engineering in Medicine and Biology Society*, Milan, Italy, Aug. 25-29, 2015, pp. 113–116.
[23] L. De Lathauwer and B. De Moor, "From Matrix to Tensor: Multilinear Algebra and Signal Processing," *Institute of Mathematics and its Applications Conference Series*, vol. 67, pp. 1–16, 1998.
[24] Q. Zhang, C. Zhu, L.T. Yang, Z. Chen, L. Zhao, L. Peng, "An Incremental CFS Algorithm for Clustering Large Data in Industrial Internet of Things," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 3, pp. 1193–1201, 2017.
[25] M. I. Soliman, S. Rajasekaran, and R. Ammar, "A Block JRS Algorithm for Highly Parallel Computation of SVDs," in *High Performance Computing and Communications*. Springer, 2007, pp. 346–357.
[26] B. Wilkinso and M. Allen, *Parallel Programming Technique and Applications using Networked Workstations and Parallel Computers (2nd)*. Person Education Press, 2004.
[27] J. Demmes, K. Veselic, "Jacobis Method is More Accurate than QR," *SIAM Journal on Matrix Analysis and Applications*, vol. 13, no. 4, pp. 1204–1245, 1992.
[28] X. Wang, L. T. Yang, J. Feng, and X. Chen, "A Tensor-based Big Service Framework for Enhanced Living Environments," *IEEE Cloud Computing*, vol. 3, no. 6, pp. 36–43, 2016.
[29] E. Nemati, M. J. Deen, and T. Mondal, "A Wireless Wearlable ECG Sensor for Long-Term Applications," *IEEE Communications Magazine*, vol. 50, no. 1, pp. 36–43, 2012.
[30] B. Jin, T. Thu, E. Baek, S. Sakong, J. Xiao, T. Mondal, and M. J. Deen, "Walking-Age Analyzer for Healthcare Applications," *IEEE Journal of Biomedical and Health Informatics*, vol. 18, no. 3, pp. 1034–1042, 2014.

Xiaokang Wang received the B.S. degree in Electronic and Information Engineering from Henan Normal University, Xinxiang, China, in 2009, the M.S. degree in Computer Science from Changzhou University, Changzhou, China, in 2012, and the Ph.D degree in Computer System Architecture in Huazhong University of Science and Technology, Wuhan, China, in 2017. Currently, he is a Postdoctoral Research Fellow of Department of Computer Science, St. Francis Xavier University, Canada. His research interests are Cyber-Physical-Social Systems, Big Data, Tensor Computing, and Parallel and Distributed Computing.

Laurence T. Yang received the B.E. degree in Computer Science and Technology from Tsinghua University, China and the Ph.D degree in Computer Science from University of Victoria, Canada. He is a professor in School of Computer Science and Technology, Huazhong University of Science and Technology, China, and in the Department of Computer Science, St. Francis Xavier University, Canada. His research interests include Parallel and Distributed Computing, Embedded and Ubiquitous Computing, and Big Data. His research had been supported by National Sciences and Engineering Research Council and Canada Foundation for Innovation.

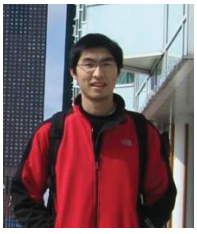
Xingyu Chen received the B.S. degree in Computer Science and Technology from China Three Gorges University, Yichang, China, in 2014, and the M.S. degree in Computer science and Technology from Huazhong University of Science and Technology, Wuhan, China, in 2017. Currently, he is a staff at the Ant Financial Company, Alibaba Group. His research interests include Big data, Parallel and Distributed Computing.



Lizhe Wang received BE and ME degrees from Tsinghua University and the Doctor of Eng degree from the University of Karlsruhe (Magna Cum Laude), Germany. He is a "ChuTian" chair professor at the School of Computer Science, China University of Geosciences (CUG), and a professor at the Institute of Remote Sensing and Digital Earth, Chinese Academy of Sciences (CAS). He serves as an associate editor of the IEEE Transactions on Computers and the IEEE Transactions on Cloud Computing. His main research interests include High-performance Computing, e-Science, and Spatial Data Processing. He is a fellow of the IET and the British Computer Society. He is a senior member of IEEE.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2018.2824303, IEEE Transactions on Big Data

Rajiv Ranjan received the PhD degree. He has been a reader (associate professor) of computing science at Newcastle University since September 1, 2015. He is an internationally renowned researcher in the areas of Cloud Computing, Internet of Things (IoT), and Big Data. He has authored approximately 150 scientific publications, including publications in the IEEE Transactions on Parallel and Distributed Systems, IEEE Transactions on Computers, Journal of Computer and System Sciences, and IEEE/ACM World Wide Web conference.



Xiaodao Chen received the B.E. degree in telecommunication from the Wuhan University of Technology, Wuhan, China, and the M.S. degree in electrical engineering and the Ph.D. degree in computer engineering from Michigan Technological University, Houghton, USA, in 2006, 2009, and 2012, respectively. He is currently an Assistant Professor with the School of Computer Science, China University of Geosciences, Wuhan.

M. Jamal Deen is currently Distinguished University Professor, Senior Canada Research Chair in Information Technology, and President of the Academy of Science, Royal Society of Canada (RSC). His research records include more than 500 peer-reviewed articles and two textbooks “Silicon Photonics-Fundamentals and Devices”, and “Fiber Optic Communications-Fundamentals and Applications”. His current research interests include Nano-/optoelectronics, Nanotechnology, and their emerging applications in health and environment. He was elected to Fellow status in ten national academies and professional societies, including RSC, IEEE, APS, ECS, and AAAS.