Scalable Tensor-Train-Based Tensor Computations for Cyber–Physical–Social Big Data

Huazhong Liu, Laurence T. Yang, *Fellow, IEEE*, Jihong Ding, Yimu Guo, Xia Xie, and Zhi-Jie Wang, *Member, IEEE*

***Abstract*— Tensor-based big data analysis approaches are effectively exploited to handle multisource and heterogeneous cyber–physical–social big data generated from diverse spaces. However, the curse of dimensionality seriously restricts their widespread exploitation, especially under edge/fog computing environments. To alleviate the dilemma, we attempt to present a set of tensor-train (TT)-based tensor operations with their scalable computations and then propose a novel TT-based big data processing framework under edge/fog computing environments. Specifically, in this article, we first summarize and present a set of TT-based tensor operations by converting the original high-order tensor operation to a series of low-order (second- or third-order) TTcore-based operations. Then, we propose a two-layer scalable TT-based computation architecture, including inter-TTcore and intra-TTcore scalable models. Afterward, according to various scalable models, a series of scalable TT-based tensor computations (STT-TCs) with their complexity analysis are proposed in detail. Finally, we propose a novel TT-based big data processing framework to adapt to edge/fog computing environments. We conduct extensive experiments based on both random data sets and real-world ubiquitous bus traffic data sets. Experimental results demonstrate that the proposed STT-TCs can significantly improve computation efficiency and are suitable for edge/fog computing environments.**

***Index Terms*— Big data processing, cyber–physical–social system (CPSS), distributed/parallel computing, edge/fog computing, scalable tensor computations, tensor-train (TT)-based tensor operations.**

I. Introduction

ALONG with the prosperous development of hardware infrastructure and mobile networks, widespread Internet of Things (IoT), such as smartphones, embedded devices, sensors, switches, routers, and base stations, have been ubiquitously connected to the Internet [1], [2]. An estimate from Cisco demonstrates that 50 billion such things shall be connected to the Internet by 2020 [3]. These ubiquitous IoTs will continuously generate all sorts of data from diverse spaces, including cyber space, physical space, and social space (i.e., cyber–physical–social system, CPSS) [4]. These human-centric intertwined data are forming cyber–physical–social big data and have some unique characteristics, e.g., multisource and heterogeneous data type, complicated data relationship, and real-time updated data streaming [5]–[7].

Due to the increasing improvement of hardware performance and 5G technologies, the extensively deployed IoTs are generally equipped with sufficient storage, communication, and computation capabilities [8]. They can perform some lightweight computational tasks, such as data prefiltering, data aggregation, and online processing. Therefore, edge/fog computing, an emerging computing paradigm, has gradually sprung up and extensively developed [9]. It allows computation tasks to be performed at the edge of the network and has some distinct advantages, e.g., location awareness, low latency, and support for mobility [2], [10]. By exploiting the advantages of edge/fog computing, it is conducive to satisfy the requirements of real-time information exchange and providing elastic services to end users [11]. However, how to represent, store, handle, and analyze the cyber–physical–social big data in an efficient manner under edge/fog computing environments remains a challenging problem and needs more exploration.

To represent and deal with the cyber–physical–social big data, tensor has been viewed as an efficient processing tool because of its prominent advantages in handling multidimensional data [12], [13]. Meanwhile, some tensor-based analysis approaches, such as tensor-based accurate recommendation [14], tensor-based multiclustering [15], tensor-based deep computation [16], [17], and tensor-based multimodal prediction [18], [19], play significant roles in practical applications and are applied to various fields, e.g., social network, online education, and intelligent transport [20].

However, during implementing the tensor-based big data analysis, the curse of dimensionality arising from high-order tensor is still the main bottleneck of widespread exploitation, especially under edge/fog computing environments. To address this, some typical tensor networks, such as hierarchical

Manuscript received February 13, 2019; revised November 1, 2019; accepted November 17, 2019. Date of publication July 30, 2020; date of current version August 6, 2020. This work was supported in part by the National Natural Science Foundation of China under Grant 61867002, Grant 71704160, Grant U1811264, Grant 61972425, and Grant 61802112. *(Corresponding author: Laurence T. Yang.)*

Huazhong Liu is with the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China, and also with the School of Information Science and Technology, Jiujiang University, Jiujiang 332005, China (e-mail: sharpshark_ding@163.com).

Laurence T. Yang is with the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China, and also with the Department of Computer Science, St. Francis Xavier University, Antigonish, NS B2G 2W5, Canada (e-mail: ltyang@gmail.com).

Jihong Ding is with the School of Education Science and Technology, Zhejiang University of Technology, Hangzhou 310014, China (e-mail: jhding@zjut.edu.cn).

Yimu Guo is with the Department of Social Network Application, Tencent Inc., Shenzhen 518052, China (e-mail: kknightgo@gmail.com).

Xia Xie is with the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: shelicy@hust.edu.cn).

Zhi-Jie Wang is with the College of Computer Science, Chongqing University, Chongqing 400044, China (e-mail: cszjwang@cqu.edu.cn).

Digital Object Identifier 10.1109/TCSS.2019.2957337

Tucker (HT), tensor train (TT), and quantized TT (QTT) decompositions, are presented successively [13]. Oseledets [21] proposes a TT decomposition approach, which decomposes a high-order tensor into a battery of low-order (typically second- or third-order) core tensors. TT has some prominent advantages, such as stable decomposition algorithm, fewer parameters, distributed storage of TT cores, and TT-based tensor operations. Thus, it has been adopted to solve many intractable high-dimensional problems, such as higher dimensions eigenvalues [22], high-dimensional operator equations [23], deep computation [24], and multimodal prediction [25], [26].

Although tensor-based data analysis methods are efficient, there remain many challenges in implementing tensor computations under edge/fog environments. First, the constructed tensor is hard to be directly stored because of the curse of dimensionality. It is necessary to select a low parameter-based tensor decomposition to alleviate the dilemma. Second, how to implement tensor computations and tensor analysis based on the decomposed results is also a challenge. It is hardly possible to perform tensor computations by reconstructing the original tensor based on decomposed results on edge/fog devices because of the curse of dimensionality. Thus, we may need to explore an approach to perform tensor operations directly based on decomposed results. Third, due to the relatively poor performance of edge/fog devices in computation, storage, bandwidth, and so on, it will be incapable to perform the whole tensor operation on a single edge/fog device. Thus, we should seek for a solution to break each tensor operation into some smaller ones to adapt to edge/fog devices. To the best of our knowledge, there is no holistic approach to tackle the above-mentioned challenges. Therefore, it is very meaningful to develop an efficient processing framework to represent, store, compute, and analyze the cyber–physical–social big data under edge/fog computing environments.

By taking the advantages of TT, this article focuses on presenting a set of tensor operations based on decomposed TT cores with their scalable implementation and then proposing a novel TT-based big data processing framework under edge/fog computing environments. The main idea is to equivalently convert the original high-order tensor operation to a battery of low-order (second- or third-order) TTcore-based operations, and these lightweight operations are then implemented on edge/fog devices in a distributed or parallel manner. Fig. 1 depicts the edge-fog-cloud computing architecture [2]. Traditionally, tensor-based data analysis methods are implemented on cloud [27]. Under edge/fog environments, various ubiquitous data are generated from edge plane and then uploaded to cloud via fog plane. However, if there are some relatively small-scale tensor-based computational tasks with quick feedback requirement, it might be incapable to upload them to cloud due to the long delay. Here, our proposed TT-based processing approach can be used to deal with these situations by fully utilizing the computation capability of edge/fog devices. Meanwhile, it can also provide a feasible solution if the computing resources on the cloud are not available or the total overheads on the cloud are very large, such as prices, delay, and networking. Through the proposed TT-based processing approach, it is possible to flexibly select more feasible computing solutions.

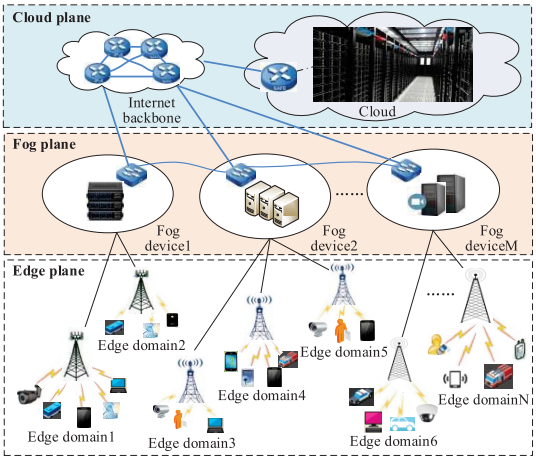


Fig. 1. Edge-fog-cloud computing architecture.

Concretely, we first summarize the existing TT-based tensor operations and present other TT-based tensor operations, including mode-n product and Tucker mode-n product. Then, we propose a two-layer scalable TT-based computation architecture (STT-CA), including inter-TTcore (containing vertical, horizontal, and hybrid schemes) and intra-TTcore scalable models. Afterward, according to various scalable models, we put forward a series of scalable TT-based tensor computations (STT-TCs) and give their detailed complexity analysis. Furthermore, a novel TT-based big data processing framework is proposed to adapt to the edge/fog computing environments.

To summarize, the major contributions of this article are listed as follows.

1) Summarize and present a set of TT-based tensor operations directly based on the decomposed TT cores and guarantee that the result remains TT format.
2) Put forward a two-layer STT-CA, including inter-TTcore and intra-TTcore scalable models.
3) Propose a series of STT-TCs according to various scalable models.
4) Present a TT-based big data processing framework to adapt to the edge/fog computing environments.

Besides, we conduct extensive experiments based on both random and real-world ubiquitous bus traffic data sets. The experimental results demonstrate that the proposed STT-TCs can significantly improve computation efficiency and reduce the running memory, which is conducive to tensor-based data analysis under edge/fog computing environments.

The rest of this article is organized as follows. Section II briefly recalls tensor preliminaries, as well as TT decomposition. Section III illustrates the TT-based tensor operations. In Section IV, an STT-CA is established. In Section V, some STT-TCs are presented in detail. A TT-based big data processing framework under edge/fog computing environments is proposed in Section VI. Section VII compares the experimental results, and Section VIII concludes this article.

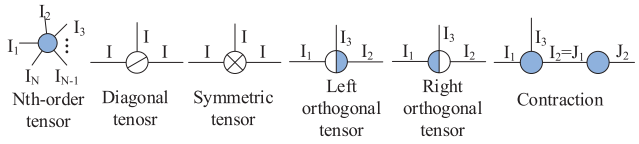


Fig. 2. Some examples of tensor symbols and operations using TND.

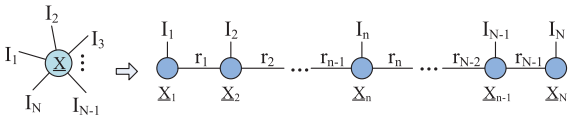


Fig. 3. TT format of an Nth-order tensor.

II. Preliminaries

In an Nth-order tensor $\underline{X} \in R^{I_1 \times I_2 \times \cdots \times I_N}$, N is called tensor's order and I_n $(1 \leq n \leq N)$ is the dimensionality of the nth order. To facilitate tensor-based data analysis, some tensor operations play significant roles in practical applications, such as mode-n product, single-mode product, and multiple-mode product. For more concrete descriptions about other tensor operations, please refer to [13] and [28]. To visualize the complex interactions and operations between different tensors, the tensor network diagram (TND) is gradually utilized. For more concrete descriptions about TND, please refer to [13] and [29]. Fig. 2 depicts the graphical representation of basic symbols through TND.

TT decomposition is to decompose a high-order tensor to a battery of low-order core tensors [21]. Formally, the definition of TT format is illustrated as follows.

Definition 1 (TT decomposition): An Nth-order tensor $\underline{X} \in R^{I_1 \times I_2 \times \cdots \times I_N}$ is defined to be the TT-format if it satisfies the following format:

$$\underline{X} = \underline{X_1} \bullet \underline{X_2} \bullet \cdots \bullet \underline{X_n} \bullet \cdots \bullet \underline{X_N} \tag{1}$$

where $\underline{X_n} \in R^{r_{n-1} \times r_n \times I_n}$ $(n = 1, \ldots, N; r_0 = r_N = 1)$ denotes core tensor (or core), $\{r_0, r_1, \ldots, r_N\}$ are called TT ranks, and $\bullet$ denotes the contraction operation (i.e., $\times_2^1$).

Fig. 3 illustrates the graphical representation of the TT format for an Nth-order tensor. For notational convenience, each core tensor is regarded as a third-order tensor, and the first order of $\underline{X_1}$ and the third order of $\underline{X_N}$ are equal to 1. Alternatively, if the tensor is assigned to a specified index $(i_1, i_2, \ldots, i_N)$, then the entrywise TT format can be represented as follows:

$$\underline{X}(i_1, i_2, \ldots, i_N) = \underline{X_1}(i_1)\underline{X_2}(i_2) \cdots \underline{X_N}(i_N) \tag{2}$$

where $\underline{X_n}(i_n) \in R^{r_{n-1} \times r_n}(\ n = 1, \ldots, N;\ r_0 = r_N = 1)$.

III. TT-Based Tensor Operations

This section presents a series of computation approaches to implement some tensor operations directly based on the decomposed TT cores without reconstructing their TT cores to the original tensor.

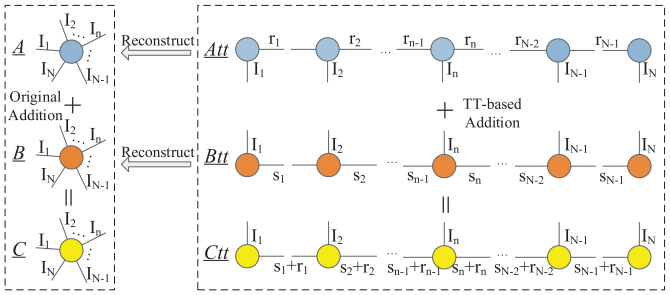


Fig. 4. Graphical representation of tensor addition in TT format.

A. Basic Tensor Operations in TT Format

In the previous work in [21], some basic tensor operations have been implemented in TT format, such as addition, scalar multiplication, minus, contraction, multilinear contraction, the Hadamard product, inner product, and norm. In these operations, if the operation is a binary operator, then the operands are with the same order and dimensionality. Given two tensors $\underline{A} \in R^{I_1 \times I_2 \times \cdots \times I_N}$ and $\underline{B} \in R^{I_1 \times I_2 \times \cdots \times I_N}$, we suppose their TT formats are known as follows:

$$\underline{A}(i_1, i_2, \ldots, i_N) = \underline{A_1}(i_1)\underline{A_2}(i_2) \cdots \underline{A_N}(i_N) \tag{3}$$

$$\underline{B}(i_1, i_2, \ldots, i_N) = \underline{B_1}(i_1)\underline{B_2}(i_2) \cdots \underline{B_N}(i_N) \tag{4}$$

where $\underline{A_n}(i_n) \in R^{r_{n-1} \times r_n}(\ n = 1, \ldots, N;\ r_0 = r_N = 1)$ and $\underline{B_n}(i_n) \in R^{s_{n-1} \times s_n}(\ n = 1, \ldots, N;\ s_0 = s_N = 1)$.

Based on these decomposed TT cores, we can implement some basic tensor operations directly in the TT format, and the result remains in the TT format. Next, we take a TT-based tensor addition as an example to illustrate the process, and the graphical representation is depicted in Fig. 4. Suppose there are two TT formats $\underline{Att}$ and $\underline{Btt}$, and $\underline{A}$ and $\underline{B}$ are their reconstructed tensors. On the one hand, if we add reconstructed tensors $\underline{A}$ and $\underline{B}$ according to the original tensor addition, the result is a new tensor $\underline{C}$. On the other hand, if we perform the TT-based tensor addition based on TT cores $\underline{Att}$ and $\underline{Btt}$ according to the TT-based computational rule in Table I, it will generate a new TT format $\underline{Ctt}$. At last, if we reconstruct TT cores $\underline{Ctt}$, then the reconstructed tensor will equal tensor $\underline{C}$. Finally, we summarize some TT-based computational rules and illustrate them in Table I. Its four columns represent the operation name, operands, tensor operations and results, and TT-based computational rules, respectively.

Especially, the tensor minus operation can be realized according to tensor addition and scalar multiplication operations, i.e., $\underline{A} - \underline{B} = \underline{A} + (-1)\underline{B}$. Furthermore, the Frobenius norm of the difference of any two tensors $\|\underline{A} - \underline{B}\|_F$ can be calculated by combining the tensor minus and Frobenius norm operations. Note that the TT ranks of the result after executing some TT-based operations (e.g., tensor addition) shall increase, and we can adopt the rounding algorithm proposed in [21] to compress the TT ranks according to practical applications.

TABLE I
COMPUTATION RULES OF SOME BASIC TENSOR OPERATIONS IN TT FORMAT

Name	Operand	Tensor Operation and Result	TT-based Computation Rule
Addition	$\underline{A} \in R^{I_1 \times I_2 \times \cdots \times I_N}$ and its TT cores in Eq. (3), $\underline{B} \in R^{I_1 \times I_2 \times \cdots \times I_N}$ and its TT cores in Eq. (4)	$\underline{C} = \underline{A} + \underline{B}, \underline{C} \in R^{I_1 \times I_2 \times \cdots \times I_N}$, $\underline{C}(i_1, i_2, \cdots, i_N) = \underline{C_1}(i_1)\underline{C_2}(i_2) \cdots \underline{C_N}(i_N)$	$\underline{C_k}(i_k) = \begin{cases} \left(\underline{A_1}(i_1)\ \underline{B_1}(i_1)\right), & k = 1 \\ \begin{pmatrix} \underline{A_k}(i_k) & 0 \\ 0 & \underline{B_k}(i_k) \end{pmatrix}, & k = 2, \cdots, N-1 \\ \begin{pmatrix} \underline{A_N}(i_N) \\ \underline{B_N}(i_N) \end{pmatrix}, & k = N \end{cases}$
Scalar multiplication	$\underline{A} \in R^{I_1 \times I_2 \times \cdots \times I_N}$ and its TT cores in Eq. (3), $\alpha \in R$	$\underline{B} = \alpha\underline{A}, \underline{B} \in R^{I_1 \times I_2 \times \cdots \times I_N}$, $\underline{B}(i_1, i_2, \cdots, i_N) = \underline{B_1}(i_1)\underline{B_2}(i_2) \cdots \underline{B_N}(i_N)$	$\underline{B_k}(i_k) = \begin{cases} \alpha\underline{A_p}(i_p), & k = p, p \in \{1, 2, \cdots, N\} \\ \underline{A_k}(i_k), & k = 1, 2, \cdots, N \text{ and } k \neq p \end{cases}$
Contraction (Tensor-by-vector)	$\underline{A} \in R^{I_1 \times I_2 \times \cdots \times I_N}$ and its TT cores in Eq. (3), $u_n \in R^{I_n}$	$\underline{B} = \underline{A} \times_n u_n$, $\underline{B} \in R^{I_1 \times I_2 \times \cdots I_{n-1} \times I_{n+1} \times \cdots \times I_N}$, $\underline{B}(i_1, \cdots, i_{n-1}, i_{n+1}, \cdots, i_N) = \underline{B_1}(i_1) \cdots \underline{B_{n-1}}(i_{n-1})\underline{B_{n+1}}(i_{n+1}) \cdots \underline{B_N}(i_N)$	$\underline{B_k}(i_k) = \begin{cases} \underline{A_{n-1}}(i_{n-1}) \left(\sum_{i_n=1}^{I_n} u_n(i_n)\underline{A_n}(i_n) \right), & k = n-1 \\ \underline{A_k}(i_k), & k = 1, 2, \cdots, N \text{ and } k \neq n-1, n \end{cases}$
Multilinear contraction	$\underline{A} \in R^{I_1 \times I_2 \times \cdots \times I_N}$ and its TT cores in Eq. (3), $u_k \in R^{I_k} (k = 1, 2, \cdots, N)$	$W = \underline{A} \times_1 u_1 \times_2 u_2 \cdots \times_N u_N$, $W \in R$	$T_k = \sum_{i_k=1}^{I_k} u_i(i_k)\underline{A_k}(i_k), k = 1, 2 \cdots, N$ $W = T_1 T_2 \cdots T_N$
Hadamard product	$\underline{A} \in R^{I_1 \times I_2 \times \cdots \times I_N}$, and its TT cores in Eq. (3), $\underline{B} \in R^{I_1 \times I_2 \times \cdots \times I_N}$ and its TT cores in Eq. (4)	$\underline{C} = \underline{A} \circledast \underline{B}, \underline{C} \in R^{I_1 \times I_2 \times \cdots \times I_N}$, $\underline{C}(i_1, i_2, \cdots, i_N) = \underline{C_1}(i_1)\underline{C_2}(i_2) \cdots \underline{C_N}(i_N)$	$\underline{C_k}(i_k) = \underline{A_k}(i_k) \otimes \underline{B_k}(i_k), k = 1, 2, \cdots, N$
Inner product	$\underline{A} \in R^{I_1 \times I_2 \times \cdots \times I_N}$ and its TT cores in Eq. (3), $\underline{B} \in R^{I_1 \times I_2 \times \cdots \times I_N}$ and its TT cores in Eq. (4)	$W = \underline{A} \cdot \underline{B}, W \in R$	$T_k = \sum_{i_k=1}^{I_k} \underline{A_k}(i_k) \otimes \underline{B_k}(i_k), k = 1, 2 \cdots, N$ $W = T_1 T_2 \cdots T_N$
Frobenius norm	$\underline{A} \in R^{I_1 \times I_2 \times \cdots \times I_N}$ and its TT cores in Eq. (3)	$W = \|\underline{A}\|_F, W \in R$	$T_k = \sum_{i_k=1}^{I_k} \underline{A_k}(i_k) \otimes \underline{A_k}(i_k), k = 1, 2 \cdots, N$ $W = \sqrt{T_1 T_2 \cdots T_N}$

B. Tensor-by-Matrix Operation in TT Format

In this section, we propose some other tensor-by-matrix operations based on the TT format, including mode-n product and Tucker mode-n product.

Theorem 1 (TT-Based Mode-n Product): Given the TT format of an Nth-order tensor $\underline{A} \in R^{I_1 \times I_2 \times \cdots \times I_N}$ in (3) and a matrix $U_n \in R^{I_n \times J_n}$, and suppose that the TT format of the mode-n product $\underline{B} = \underline{A} \times_n U_n$ is represented as $\underline{B}(i_1, i_2, \ldots, i_N) = \underline{B_1}(i_1)\underline{B_2}(i_2) \cdots \underline{B_N}(i_N)$, then their TT cores can be obtained according to the following generation rule:

$$\underline{B_k}(i_k) = \begin{cases} \sum_{i_n=1}^{I_n} U_n(i_n, j_n)\underline{A_n}(i_n), & k = n \\ \underline{A_k}(i_k), & k = 1, 2 \ldots, N \text{ and } k \neq n. \end{cases} \tag{5}$$

Proof: According to the definition of the mode-n product, we have

$$\begin{aligned} &\underline{B}(i_1, \ldots, i_{n-1}, j_n, i_{n+1}, \ldots, i_N) \\ &\quad = \sum_{i_n=1}^{I_n} \underline{A}(i_1, \ldots, i_{n-1}, i_n, i_{n+1}, \ldots, i_N) U_n(i_n, j_n) \\ &\quad = \sum_{i_n=1}^{I_n} \underline{A_1}(i_1) \cdots \underline{A_{n-1}}(i_{n-1})\underline{A_n}(i_n)\underline{A_{n+1}}(i_{n+1}) \\ &\qquad \cdots \underline{A_N}(i_N) U_n(i_n, j_n) \\ &\quad = \underline{A_1}(i_1) \cdots \underline{A_{n-1}}(i_{n-1}) \left(\sum_{i_n=1}^{I_n} U_n(i_n, j_n)\underline{A_n}(i_n) \right) \\ &\qquad \times \underline{A_{n+1}}(i_{n+1}) \cdots \underline{A_N}(i_N). \end{aligned}$$

It can be found that the result is equal to that of putting (5) to the TT format of tensor $\underline{B}$.

Theorem 2 (TT-Based Tucker Mode-n Product): Given the TT format of an Nth-order tensor $\underline{A} \in R^{I_1 \times I_2 \times \cdots \times I_N}$ in (3) and N matrices $U_k \in R^{I_k \times J_k}, k = 1, 2, \ldots, N$, and suppose that the TT format of Tucker mode-n product $\underline{B} = \underline{A} \times_1 U_1 \times_2 U_2 \cdots \times_N U_N$ is represented as $\underline{B}(j_1, j_2, \ldots, j_N) = \underline{B_1}(j_1)\underline{B_2}(j_2) \cdots \underline{B_N}(j_N)$, then the generation rule of their TT cores is defined as follows:

$$\underline{B_k}(j_k) = \sum_{i_k=1}^{I_k} U_k(i_k, j_k)\underline{A_k}(i_k), \quad k = 1, 2, \ldots, N. \tag{6}$$

Proof: According to the definition of the Tucker mode-n product, we have

$$\begin{aligned} &\underline{B}(j_1, \ldots, j_n, \ldots, j_N) \\ &\quad = \sum_{i_1 \cdots i_n \cdots i_N = 1}^{I_1 \cdots I_n \cdots I_N} \underline{A}(i_1, \ldots, i_n, \ldots, i_N) U_1(i_1, j_1) \\ &\qquad \cdots U_n(i_n, j_n) \cdots U_N(i_N, j_N) \\ &\quad = \sum_{i_1 \cdots i_n \cdots i_N = 1}^{I_1 \cdots I_n \cdots I_N} \underline{A_1}(i_1) \cdots \underline{A_n}(i_n) \cdots \underline{A_N}(i_N) U_1(i_1, j_1) \\ &\qquad \cdots U_n(i_n, j_n) \cdots U_N(i_N, j_N) \\ &\quad = \sum_{i_1 \cdots i_n \cdots i_N = 1}^{I_1 \cdots I_n \cdots I_N} U_1(i_1, j_1)\underline{A_1}(i_1) \cdots U_n(i_n, j_n)\underline{A_n}(i_n) \\ &\qquad \cdots U_N(i_N, j_N)\underline{A_N}(i_N) \\ &\quad = \left(\sum_{i_1=1}^{I_1} U_1(i_1, j_1)\underline{A_1}(i_1) \right) \cdots \left(\sum_{i_n=1}^{I_n} U_n(i_n, j_n)\underline{A_n}(i_n) \right) \\ &\qquad \cdots \left(\sum_{i_N=1}^{I_N} U_1(i_N, j_N)\underline{A_N}(i_N) \right). \end{aligned}$$

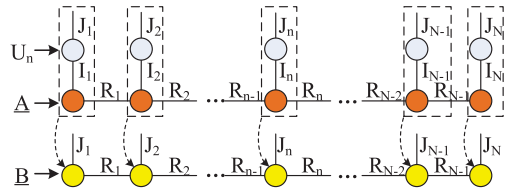


Fig. 5. Graphical representation of the Tucker mode-n product in TT format.

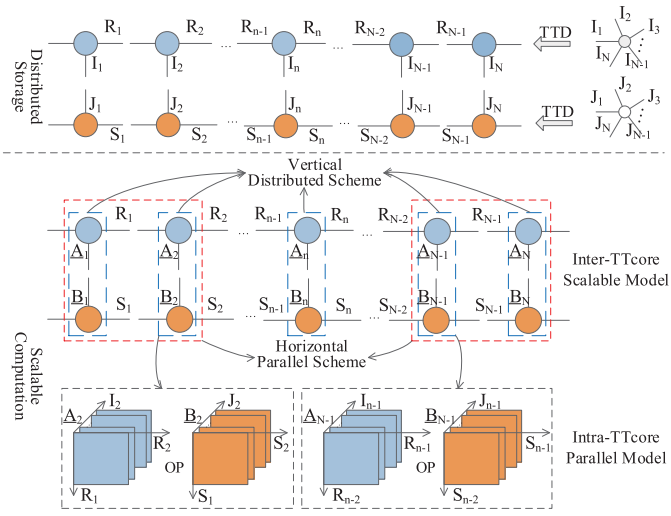


Fig. 6. STT-CA.

We can find that the result is exactly equal to that by putting (6) to the TT format of tensor $\underline{B}$.

The graphical representation of the Tucker mode-n product through TND is depicted in Fig. 5.

IV. STT-CA

This section presents an STT-CA. Generally, the TT cores after implementing TT decomposition can be stored in different nodes in a distributed way. According to the analysis in Section III, most of the tensor operations can be directly implemented in the TT format. Therefore, the original tensor operations can be transformed to the corresponding operations for their TT cores, and the result remains in the TT format. Because of their natural distributed storage and computational characteristics, these computations among TT cores are suitable to be executed in a distributed or parallel manner.

A. Overview of STT-CA

The overview of the STT-CA is illustrated in Fig. 6. Based on these distributed TT cores stored in different nodes, most of the TT-based tensor computations can be suitably realized in a scalable manner. According to the characteristics of all TT-based tensor operations, we design two-layer scalable computation models in STT-CA: inter-TTcore and intra-TTcore scalable models. The inter-TTcore scalable model mainly refers to the distributed or parallel computation among different pairs of TT cores. In the inter-TTcore scalable model, the original operations for high-order tensor are transformed to the current operations for low-order TT cores. Thus, its executable unit is TT core, and the degree of parallelism is tensor's order. According to the characteristics of TT-based tensor operations, there are three implementation schemes in the inter-TTcore scalable model, i.e., vertical distributed scheme, horizontal parallel scheme, and hybrid scalable scheme. For instance, $\underline{A_1}\ OP\ \underline{B_1} = \underline{C_1}$, $\underline{A_2}\ OP\ \underline{B_2} = \underline{C_2}$, ..., $\underline{A_N}\ OP\ \underline{B_N} = \underline{C_N}$ in Fig. 6 can be vertically executed in a distributed mode. Then, $\underline{C_1}\ OP\ \underline{C_2}$, ..., $\underline{C_{N-1}}\ OP\ \underline{C_N}$ can be horizontally executed in parallel. The intra-TTcore scalable model refers to the parallel computation of different slices in a pair of TT cores, especially when the dimensionality is huge, e.g., $\underline{A_2}\ OP\ \underline{B_2}$ or $\underline{A_{N-1}}\ OP\ \underline{B_{N-1}}$ in Fig. 6. In the intra-TTcore scalable model, the executable unit is data slice, the parallel granularity is further reduced from TT core to data slice, and the degree of parallelism is tensor's dimensionality.

B. Inter-TTcore Scalable Model

In the inter-TTcore scalable model, multiple operations among different pairs of TT cores can be concurrently executed. According to the characteristics of different tensor operations, there are three kinds of scalable schemes: the vertical distributed scheme, horizontal parallel scheme, and hybrid scalable scheme.

1) Vertical Distributed Scheme: The vertical distributed scheme refers to that the corresponding operations between the coupled TT cores of two different TT formats are concurrently executed. Generally, the vertical distributed scheme is applied to most binary tensor operators, such as tensor addition, minus, the Hadamard product, and the Tucker mode-n product. For instance, the implementation of the TT-based Tucker mode-n product in Fig. 5 is a typical vertical distributed example. According to the TT-based computational rule in Theorem 2, the original Tucker mode-n product operation can be transformed to the TTcore-based operations in (6), and these N TTcore-based operations can be executed in the vertical distributed mode.

2) Horizontal Parallel Scheme: The horizontal parallel scheme refers to that the corresponding operations among any two adjacent TT cores belonged to the same TT format are concurrently executed. The horizontal parallel scheme is ordinarily applied to the unary tensor operators, such as tensor extractions (including scalar, fiber, slice, and subtensor), in which tensor contractions are continually executed. Because tensor contraction satisfies the associative law but not the commutative law, adopting different associative orders will consume different execution times. Therefore, we design two modes in the horizontal parallel scheme: binary parallel and bidirectional parallel modes. The binary parallel mode refers to that the command operations among every two TT cores in each layer are concurrently executed. Given a TT format with N TT cores, an inverted binary tree with $\lceil log_2 N \rceil$ depth will be formed if the binary parallel mode is adopted. Fig. 7(a) is an example of the binary parallel mode for a TT format with six TT cores. The bidirectional parallel mode refers to that the command operations are concurrently executed from both ends to the middle of the TT format. Fig. 7(b) is a schematic of the bidirectional parallel mode. Facing different TTcore-based

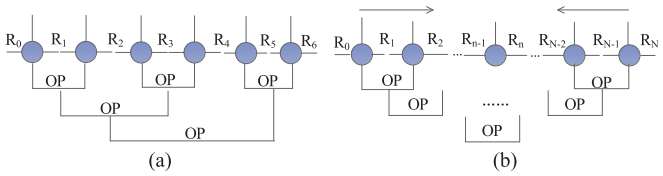


Fig. 7. Two horizontal parallel modes. (a) Binary parallel mode. (b) Bidirectional parallel mode.

horizontal parallel operations, we can select different parallel modes according to the time complexity of different tensor operations.

3) Hybrid Scalable Scheme: The hybrid scalable scheme refers to that the vertical distributed and horizontal parallel schemes are simultaneously implemented. In the hybrid scalable scheme, the execution sequence of the vertical and horizontal schemes need to satisfy the logical relationship of different tensor operations. For instance, in the TT-based tensor inner product operation, the vertical distributed scheme should be first executed, and then, the horizontal parallel scheme is exploited.

C. Intra-TTcore Parallel Model

In the intra-TTcore parallel model, the corresponding operations among different data slices in a specific pair of TT cores are concurrently executed. When the dimensionality is huge, the intra-TTcore parallel model can further improve the computation efficiency in finer-granted parallelism. For instance, the original tensor Hadamard product operation can be transformed to the TTcore-based Kronecker product, and it can be further transformed into the slice-based Kronecker product. Similarly, the finer-granted intra-TTcore parallelism can be applied to the tensor inner product, tensor Frobenius norm, and so on.

V. STT-TCs

In this section, a series of STT-TCs are illustrated. We select some typical tensor operations and illustrate their distributed or parallel implementation according to different scalable schemes in STT-CA and further analyze their complexity including computation, communication, and space complexity.

A. STT-TCs in Vertical Distributed Scheme

Vertical distributed scheme is applicable to the scalable TT-based implementation for most binary tensor operations, such as tensor addition, minus, the Hadamard product, inner product, the Frobenius norm, as well as some tensor contraction operations, such as multilinear contraction and the Tucker mode-n product. In the following, we take the STT Hadamard product as a typical example to illustrate the vertical distributed scheme.

1) STT Hadamard Product: Given two tensors $\underline{A}$ and $\underline{B}$ and their TT formats, illustrated in (3) and (4), according to the computational rule in Table I, we can find that the TT format $\underline{C_k}(i_k)$ of their Hadamard product $\underline{C} = \underline{A} \circledast \underline{B}$ can be calculated through $\underline{C_k}(i_k) = \underline{A_k}(i_k) \otimes \underline{B_k}(i_k), k = 1, 2, \ldots, N$. Obviously, these N pairs of Kronecker operations

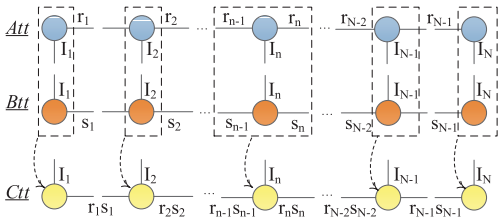


Fig. 8. Implementation of the TT-based Hadamard product in the vertical distributed scheme.

TABLE II
COMPLEXITY ANALYSIS OF THE HADAMARD PRODUCT

	Scalable TT-based	Serial TT-based	Original
Comp	$O(Ir^4)$	$O(NIr^4)$	$O(I^N)$
Comm	$O(Ir^2)$	$O(NIr^2))$	$O(I^N)$
Space	$O(Ir^4)$	$O(NIr^4)$	$O(2I^N)$

can be vertically executed in a distributed mode, and the result remains in the TT format. The graphical representation is depicted in Fig. 8.

2) Complexity Analysis for STT Hadamard Product: Without loss of generality, suppose that $I = max\{I_n\}$ and $r = max\{r_n, s_n\}$ $(n = 1, \ldots, N)$.

According to the computational rule in Table I and the representation in Fig. 8, we can obtain the computation complexity of the STT Hamamard product as $T_{STT}^{comp} = O(Ir^4)$. Besides, the communication complexity should also be considered in the distributed implementation. The communication time is mainly composed of establishing a new connection and transmitting data [30]. Therefore, the communication time is represented as

$$T^{comm} = T^{conn} + N^{data} t^{data} \tag{7}$$

where T^{conn} is the connection time, N^{data} denotes the volume of transmission data, and t^{data} represents the transmission time of every data word. The communication complexity of the STT Hadamard product is $T_{STT}^{comm} = T^{conn} + (Ir^2)t^{data} = O(Ir^2)$. Therefore, the total time complexity of STT Hadamard product can be calculated as follows:

$$T_{STT} = T_{STT}^{comp} + T_{STT}^{comm} = O(Ir^4 + Ir^2) = O(Ir^4). \tag{8}$$

Besides, the total time complexity of the serial TT-based Hadamard product is $T_{STT} = O(NIr^4)$. The total time complexity of the original Hadamard product is $T_{Ori} = O(I^N)$. Furthermore, the space complexities of scalable TT-based, serial TT-based, and the original Hadamard products can be represented as $O(Ir^4)$, $O(NIr^4)$, and $O(2I^N)$, respectively.

Finally, we summarize the complexity and illustrate them in Table II. It can be inferred that the STT approach can remarkably reduce both time and space complexity. Similarly, the complexity of other STT-TCs for vertical distributed scheme can also be analyzed according to their TT-based computational rules.

B. STT-TCs in Horizontal Parallel Scheme

The horizontal parallel scheme is suitable for some STT continuous contractions, such as exacting a scalar, fiber, slice,

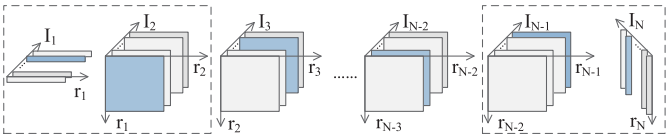


Fig. 9. Implementation of TT-based extracting scalar in the horizontal parallel scheme.

subtensor, and tensor reconstruction, as well as multilinear contraction, inner product, Frobenius norm, and so on. In the following, we take extracting an original tensor's scalar from its TT format as a typical example to illustrate the horizontal parallel scheme.

1) STT Extracting Scalar: Extracting scalar is to compute a specific value of the original tensor based on its TT format when all indexes are fixed. Similarly, we can calculate a fiber (or slice or subtensor) if we fix all but one index (or two or some indexes). Especially, the original tensor can be reconstructed if no index is fixed. These computations can be horizontally implemented in parallel by exploiting the binary or bidirectional parallel modes. Suppose the TT format of tensor $\underline{A}$ is described in (3), Fig. 9 depicts the graphical representation of STT extracting scalar.

2) Complexity Analysis for STT Extracting Scalar: Without loss of generality, suppose $I = max\{I_n\}, r = max\{r_n\}$ $(n = 1, \ldots, N)$.

If we implement the TT-based extracting scalar by exploiting binary parallel mode, then $\lceil log_2 N \rceil$ layers of contractions should be executed. The computation complexity in each layer is $O(r^3)$, but it reduces to $O(r^2)$ and $O(r)$ in the last two layers. Thus, the total computation complexity is $T_{Bin}^{Comp} = (\lceil log_2 N \rceil - 2)r^3 + r^2 + r = O(\lceil log_2 N \rceil r^3)$. Besides, according to (7) and Figs. 7 and 9, we can calculate the total communication complexity as $T_{Bin}^{comm} = T^{conn} + ((\lceil log_2 N \rceil - 2)r^2 + 2r)t^{data} = O(\lceil log_2 N \rceil r^2)$. Therefore, the total time complexity of TT-based extracting scalar when exploiting the binary parallel mode is

$$\begin{aligned} T_{Bin} &= T_{Bin}^{comp} + T_{Bin}^{comm} \\ &= O(\lceil log_2 N \rceil r^3 + \lceil log_2 N \rceil r^2) = O(\lceil log_2 N \rceil r^3). \end{aligned} \quad (9)$$

If we implement the TT-based extracting scalar by adopting bidirectional parallel mode, then $\lceil N/2 \rceil$ contractions should be executed from both ends. The computation complexity at each time is $O(r^2)$, but it reduces to $O(r)$ at the last time. Thus, the total computation complexity is $T_{Bid}^{comp} = (\lceil N/2 \rceil - 1)r^2 + r = O(\lceil N/2 \rceil r^2)$. Besides, the total communication complexity can be calculated as $T_{Bid}^{comm} = T^{conn} + (\lceil N/2 \rceil r)t^{data} = O(\lceil N/2 \rceil r)$. Therefore, the total time complexity of TT-based extracting scalar when adopting bidirectional parallel mode is

$$\begin{aligned} T_{Bid} &= T_{Bid}^{comp} + T_{Bid}^{comm} \\ &= O(\lceil N/2 \rceil r^2 + \lceil N/2 \rceil r) = O(\lceil N/2 \rceil r^2). \end{aligned} \quad (10)$$

If we implement the TT-based extracting scalar by adopting serial mode, then $N - 1$ contractions should be executed. The computation complexity at each time is $O(r^2)$, but it is reduced to $O(r)$ at the last time. Thus, the total computation complexity is $T_{Ser}^{comp} = (N - 2)r^2 + r = O(Nr^2)$. Besides, the total communication complexity is $T_{Ser}^{comm} = T^{conn} + ((N - 1)r)t^{data} = O(Nr)$. Therefore, the total time complexity of TT-based extracting scalar when adopting serial mode is

$$T_{Ser} = T_{Ser}^{comp} + T_{Ser}^{comm} = O(Nr^2 + Nr) = O(Nr^2). \quad (11)$$

Furthermore, the space complexities of the TT-based extracting scalar by exploiting the binary parallel, bidirectional parallel, and serial modes are $O(2r^2)$, $O(r^2 + r)$, and $O(r^2 + r)$, respectively. Finally, we summarize these complexities and illustrate them in Table III.

TABLE III
COMPLEXITY ANALYSIS OF TT-BASED EXTRACTING SCALAR

	Binary Parallel	Bidirectional Parallel	Serial
Comp	$O(\lceil log_2 N \rceil r^3)$	$O(\lceil N/2 \rceil r^2)$	$O(Nr^2)$
Comm	$O(\lceil log_2 N \rceil r^2)$	$O(\lceil N/2 \rceil r)$	$O(Nr)$
Space	$O(2r^2)$	$O(r^2 + r)$	$O(r^2 + r)$

According to the above-mentioned analysis, it is easy to infer that $T_{Bid} < T_{Ser}$. However, the relation between T_{Bin} and T_{Bid} needs to be further discussed. We can see from Table III that the communication time is significantly less than the computation time. Thus, we just need to compare their computation complexity. Let $T_{Bin}^{comp} < T_{Bid}^{comp}$, i.e., $(\lceil log_2 N \rceil - 2)r^3 + r^2 + r < (\lceil N/2 \rceil - 1)r^2 + r$, we can obtain $r < (\lceil N/2 \rceil - 2)/(\lceil log_2 N \rceil - 2)$. Therefore, it can be inferred the following.

1) $T_{Bin} \leq T_{Bid}$ if $r \leq (\lceil N/2 \rceil - 2)/(\lceil log_2 N \rceil - 2)$.
2) $T_{Bin} > T_{Bid}$ if $r > (\lceil N/2 \rceil - 2)/(\lceil log_2 N \rceil - 2)$.

Similarly, the complexity of other horizontal STT contractions (e.g., fiber, slice, subtensor, and tensor reconstruction) can be also analyzed in the same manner.

C. STT-TCs in Hybrid Scalable and Intra-TTcore Parallel Schemes

The hybrid scalable scheme is suitable for some complex STT tensor operations, such as the inner product and the Frobenius norm. In this section, we take STT inner product as a typical example to illustrate the hybrid scalable scheme. From the aforementioned discussion in Section IV-C, if the dimensionality is huge, the intra-TTcore parallel model can be exploited to further improve the computation efficiency. Therefore, intra-TTcore parallelism is analyzed together.

1) STT Inner Product: Given tensors $\underline{A}$ and $\underline{B}$ with their TT formats illustrated in (3) and (4), according to the TT-based computational rule for inner product $W = \underline{A} \cdot \underline{B}$ in Table I, $W = T_1 T_2 \cdots T_N$, where $T_k = \sum_{i_k=1}^{I_k} \underline{A_k}(i_k) \otimes \underline{B_k}(i_k)$, $k = 1, 2 \ldots, N$, we can find that the implementation of STT inner product belongs to the hybrid scalable scheme and includes three steps, which is depicted in Fig. 10(a). First, we perform N pairs of Kronecker products in the vertical distributed scheme, which is illustrated in detail in Section V-A. Second, we perform the summation for each TT core along the order located by I_n, which can also be executed in the vertical distributed scheme. Third, we perform the multilinear contraction and obtain the final result in the horizontal parallel scheme,

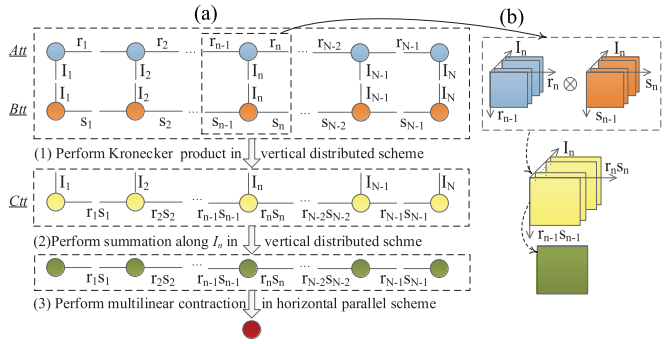


Fig. 10. Implementation of TT-based inner product in the hybrid scalable and intra-TTcore parallel schemes. (a) Inter-TTcore scalable model. (b) Intra-TTcore parallel model.

and the detailed implementation can be found in Section V-B. Furthermore, in the first and second steps, the intra-TTcore parallel model can be exploited to the computation for each pair of TT cores if the dimensionality is huge, which is depicted in Fig. 10(b).

2) Complexity Analysis for STT Inner Product: Without loss of generality, suppose that $I = max\{I_n\}$ and $r = max\{r_n, s_n\}$ $(n = 1, \ldots, N)$. From Fig. 10, we can see that there are three steps in the STT inner product. In the first and second steps, both the Kronecker product and summation for each pair of TT cores can be executed in the vertical distributed scheme. For each pair of TT cores, the computation time complexity of both the Kronecker product and summation are $O(Ir^4)$. According to the discussion in Section V-A2, the time complexity can be obtained based on (8), i.e., $T_{STT}^{(1,2)} = O(Ir^4)$. If each dimensionality (i.e., I_n) is divided to P slices, then the time complexity in intra-TTcore parallel is $T_{IntraSTT}^{(1,2)} = O((I/P)r^4)$. In the third step, the multilinear contraction can be executed in the horizontal parallel scheme. According to the analysis in Section V-B2, if we select the binary parallel mode, the time complexity can be obtained based on (9), i.e., $T_{Bin}^{(3)} = O(\lceil log_2 N \rceil r^6)$. Note that the dimensionality of each matrix in the third step is r^2. If the bidirectional parallel is exploited, we can obtain the time complexity based on (10), i.e., $T_{Bid}^{(3)} = O(\lceil N/2 \rceil r^4)$. Finally, we can obtain the total time complexity of STT inner product and illustrate them in Table IV. Besides, the communication complexity and space complexity can be similarly calculated according to the complexity analysis in Sections V-A2 and V-B2, which are illustrated in Table IV.

VI. TT-Based Big Data Processing Framework Under Edge/Fog Environments

Based on the proposed STT-TCs, we can convert the complex tensor operations to clusters of simple TTcore-based operations to adapt to edge/fog devices and perform them in a distributed or parallel manner. Therefore, we present a TT-based big data processing framework, which is depicted in Fig. 11, to realize data representation, decomposition, storage, computation, and analysis under

TABLE IV
Complexity Analysis of Inner Product

	Comp	Comm	Space
Inter STT (Binary)	$O(Ir^4+\lceil log_2 N \rceil r^6)$	$O(Ir^2+\lceil log_2 N \rceil r^4)$	Ir^4
Inter STT (Bidirectional)	$O(Ir^4+\lceil N/2 \rceil r^4)$	$O(Ir^2+\lceil N/2 \rceil r^2)$	Ir^4
Intra STT (Binary)	$O((I/P)r^4+\lceil log_2 N \rceil r^6)$	$O((I/P)r^2+\lceil log_2 N \rceil r^4)$	$(I/P)r^4$
Intra STT (Bidirectional)	$O((I/P)r^4+\lceil N/2 \rceil r^4)$	$O((I/P)r^2+\lceil N/2 \rceil r^2)$	$(I/P)r^4$
Serial TT-based	$O(NIr^4)$	$O(NIr^2)$	NIr^4
Original	$O(I^N)$	$O(I^N)$	$O(2I^N)$

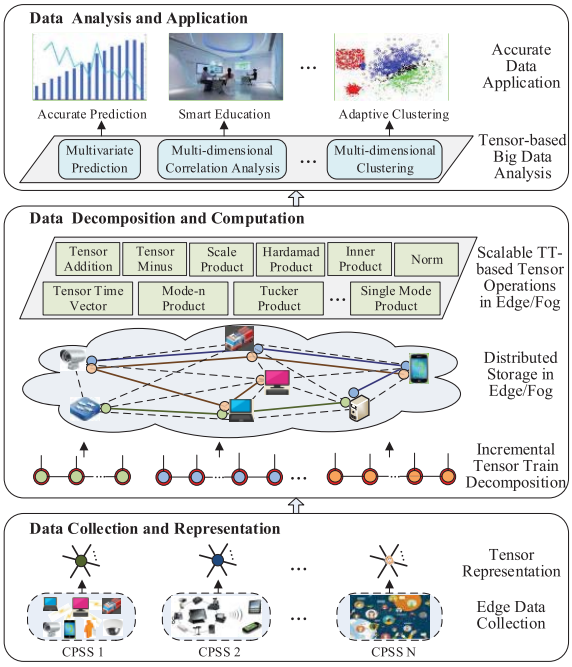


Fig. 11. TT-based big data processing framework under edge/fog computing environments.

edge/fog computing environments. The hierarchical framework is composed of three planes, i.e., data collection and representation, data decomposition and computation, and data analysis and application planes. The respective responsibility and function of each plane will be elaborated from a bottom–up view.

A. Data Collection and Representation

Data collection and representation plane is to collect the generated data and represent them in an efficient and compact way. Multisource and heterogeneous cyber–physical–social data are derived from ubiquitous IoTs or social mobile devices in various local CPSSs. To represent and analyze these data, the tensor is considered as an efficient tool because of its excellent characteristics in high-dimensional correlation analysis and low-rank approximation. According to the previous work of our team in [12], we can represent structured, semistructured, and unstructured data with different types (e.g., text, image, audio, and video) to different tensors.

B. Data Decomposition and Computation

These collected raw data generally contain a mass of noisy and redundant data; directly handling them shall take up tremendous storage space and consume tremendous computational resource and bandwidth. Therefore, we can first remove some dirty data and preserve core data containing prominent features (i.e., TT cores) by performing TT decomposition in the edge/fog plane. These decomposed results become smaller in size and higher in quality, which is conducive to efficient storage, computation, and communication. Moreover, cyber–physical–social big data are generally updated in a streaming way, and we can exploit the incremental decomposition method proposed in our previous work to reduce the decomposition time and further satisfy the edge/fog environments [29].

TT can decompose a high-order tensor into a sequence of low-order (typically second- or third-order) core tensors. These decomposed small TT cores can be stored in edge/fog devices or clouds in a distributed manner. Besides, based on the aforementioned STT-PCs in Section V, most tensor computations can be directly implemented based on TT cores in a scalable manner. These simple operations can be executed on lightweight edge/fog devices. Therefore, the proposed TT-based processing approach can be applied in many significant scenarios, for instance, when there are some relatively small-scale tensor-based computational tasks with quick feedback requirement in edge/fog planes, or when the computing resources on the cloud are not available, or when the total overheads (e.g., price, delay, and networking) on the cloud are very large.

C. Data Analysis and Application

After providing these TT-based tensor operations, some tensor-based big data analysis approaches can be implemented in the TT format, such as multivariate prediction, multidimensional correlation analysis, and multidimensional clustering algorithms. These tensor-based analysis approaches consider the integration and multidimensional correlation of heterogeneous data derived from diverse spaces, and the analysis results are commonly more accurate and targeted. They are beneficial for providing excellent services, such as personalized recommendation in smart education, proactive health-care services in smart monitor systems, and accurate traffic prediction in smart transport systems. These accurate services shall promote our daily lifestyle and improve our quality of life.

Therefore, through the proposed TT-based processing approach, we can fully make use of the performance of edge/fog devices and flexibly select more feasible computing solutions under edge-fog-cloud computing environments according to practical requirements. How to select the computing paradigms and allocate computational tasks to coordinately complete the data analysis can be further explored in our next work.

VII. EXPERIMENTS

To verify the efficiency of various STT-TCs in STT-CA, a series of experiments are conducted using different kinds of data sets. Furthermore, the comparisons of execution time and serial–parallel ratio for different computation approaches are performed.

TABLE V
TT RANKS UNDER VARIOUS TRAFFIC SLICE INTERVALS

# of Slices	r_0	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8
40	1	1	3	3	9	16	111	7	1
60	1	1	3	3	9	16	130	7	1
80	1	1	3	3	9	16	136	7	1
100	1	1	3	3	9	16	149	7	1
120	1	1	3	3	9	16	156	7	1

A. Experimental Design

We implement these experiments by exploiting the Java Toolkit AKKA and Python's NumPy package. All experiments are executed on an educational cloud platform, and we simulate one fog device configured an Intel's 16-core processor with 2.4 GHz and five edge devices configured an Intel's 8-core processor with 2 GHz. During the experiments, we adopt two kinds of data sets: random data sets and real-world CPSS data sets. The random data are generated according to different setting parameters, e.g., tensor's order, dimensionality, and TT ranks. The real-world CPSS data are derived from the public traffic system in Guangzhou, China. It contains eight million bus card records generated by two million users and four bus lines from August 1, 2014 to December 31, 2014[1]. For this data set, we first remove incomplete bus card records from the data set. Then, we integrate the remaining records (bus lines, card types, traffic, time, and so on) and other data (weather, temperature, and so on) to a fusion data. Afterward, we further analyze the characteristics of data, such as value range, mean, and variance.

Finally, we select the data in August 2014 as our experimental data set. After determining bus line, card type, month, weather, temperature, hour, passenger traffic, and week as eight attributes, we construct an eighth-order tensor with dimensions (2, 3, 12, 6, 10, 24, cnt_slice, 7), where cnt_slice represents the number of passenger traffic slices. In the data set, the maximum passenger traffic is 1200, and the passenger traffic is further divided into 40, 60, 80, 100, and 120 slices according to various traffic slice intervals, respectively. Since TT ranks also affect the computational efficiency of the TT-based tensor operations, we give the TT ranks after implementing TT decomposition (prescribed accuracy $eps = 0.01$ [29]) for different tensors under different traffic slice intervals, which are shown in Table V. Besides, we give the comparisons of entry number between the original tensor and decomposed TT cores, which is depicted in Table VI. We can see that the number of entries in decomposed TT cores is about 0.2% of that in the original tensor. The compression is helpful to efficient storage, computation, and communication.

In the experiments, the ratio of serial to scalable TT-based execution time is used to verify the computation efficiency, which is called the serial–parallel ratio. It differs slightly from the improvement factor in traditional parallel operations. The

[1]https://tianchi.aliyun.com/competition/information.htm?spm=5176.100067.5678.2.IsTw3H&raceId=231514

TABLE VI
COMPARISONS OF ENTRY NUMBER BETWEEN THE ORIGINAL TENSOR AND DECOMPOSED TT CORES

# of Slices	# of Original Tensor	# of TT Cores	Ratio
40	29030400	75474	0.00260
60	43545600	106290	0.00244
80	58060800	130154	0.00224
100	7257600	163286	0.00225
120	87091200	192714	0.00221

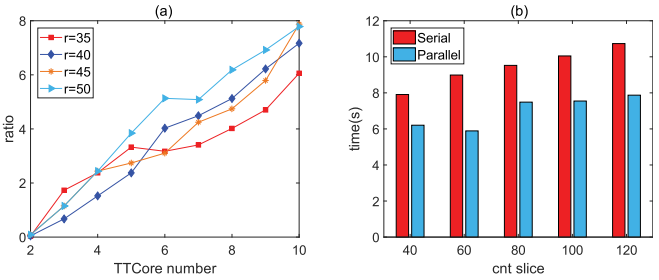


Fig. 12. (a) Serial–parallel ratio comparisons of the TT-based Hadamard product in vertical distributed scheme on random data sets. (b) Execution time comparisons of the TT-based Hadamard product between serial and vertical distributed schemes on real-world data set.

traditional parallel tasks are mainly focused on data blocks, while the parallel tasks proposed in this article put an emphasis on TT cores. Note that the execution time of original tensor operations cannot be compared in the experiments. This is because the execution time of original tensor operations is exponential with tensor's order. We can easily see from the complexity analysis in Section V that STT approaches have overwhelming superiority.

B. Evaluations

1) Comparisons of Execution Time for the TT-Based Hadamard Product in Vertical Distributed Scheme: To validate the efficiency of the TT-based Hadamard product operation in the vertical distributed scheme, we performed two sets of experiments. One is implemented using random data, and the other is conducted based on real-world data. In the random data experiments, we preset different orders of tensor and different TT ranks for comparisons. The range of tensor's order is set from the second order to the tenth order, and the dimensionality of each order is set to 200. Meanwhile, the TT ranks in the Hadamard product operation are set to 35, 40, 45, and 50, respectively. For the real-world data experiment, the number of orders and dimensions of the tensor can be referred to Section VII-A.

Fig. 12(a) depicts the serial–parallel ratio of the TT-based Hadamard product on two random data sets. It shows that the more TT cores participating in the TT-based Hadamard product, the greater the serial–parallel ratio. Thus, for higher order data, the advantages of vertical distributed computation shall be more prominent. Besides, we can see from Fig. 12(a) that the serial–parallel ratio of the TT-based Hadamard product in the vertical distributed scheme is almost linearly related to the growth of the TTcore number. In the real-world data experiment, Fig. 12(b) shows that the differences in the

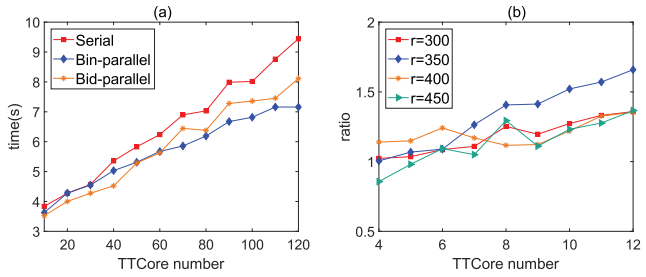


Fig. 13. (a) Comparisons of execution time among serial, binary parallel, and bidirectional parallel modes on the first group of random data sets. (b) Serial–parallel ratio comparisons of horizontal parallel scheme with various TT ranks on the second group of random data sets.

execution time between serial scheme and parallel scheme are not so obvious as that of the random data experiment. This is because in the tensor constructed by real-world data set, only three middle TT cores have large amounts of data. Therefore, the advantage of vertical distributed mode may not be used fully. However, it should be noticed that the distributed execution time is always less than the serial execution time. Moreover, both the serial execution time and distributed execution time increase as we increase the dimensionality of traffic order, but the increment of serial execution time is larger than the distributed execution time.

2) Comparisons of Execution Time for TT-Based Extracting Scalar in Horizontal Parallel Scheme: To verify the computational efficiency of implementing TT-based extracting scalar by binary parallel and bidirectional parallel modes, we conduct comparative experiments on serial mode and these two parallel modes. As before, we performed the experiments using random data sets and real-world CPSS data set.

When using random data, we conduct two groups of experiments. In the first group of experiments, the TT ranks are set to 10, and the order of TT is set to a range from 10 to 120 with 10 as the interval. As illustrated in Fig. 13(a), the execution time of two parallel modes are both less than the serial mode. Moreover, when the order is higher (i.e., 70 or more), the execution time of binary parallel mode is less than the bidirectional parallel mode. Otherwise, less execution time is needed for the bidirectional parallel mode. This observation is consistent with the complexity analysis in Section V-B2. In the second group of experiments, the number of TT cores varies from 4 to 12, and the TT ranks are ranging from 300 to 450. Then, we conduct experiments to witness the serial–parallel ratio of the horizontal parallel scheme under various conditions. According to the experimental results shown in Fig. 13(b), the serial–parallel ratio is ranging from 1 to 2 since there are less TT cores. However, the serial–parallel ratio will increase with the increment of TTcore number.

Besides, we also conduct TT-based extracting scalar experiments using real-world CPSS data set. Fig. 14 shows that the execution time of the bidirectional parallel mode is slightly less than the binary parallel mode, and the execution time of serial mode is the longest among the three modes. According to the complexity analysis in Table III, it can be inferred that the superiority of parallel modes is determined by the

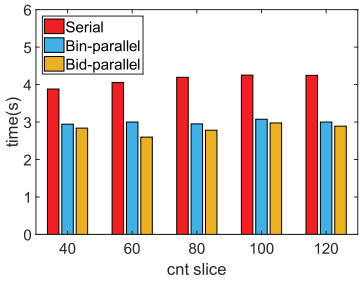


Fig. 14. Comparisons of execution time among serial, binary parallel, and bidirectional parallel modes on real-world data set.

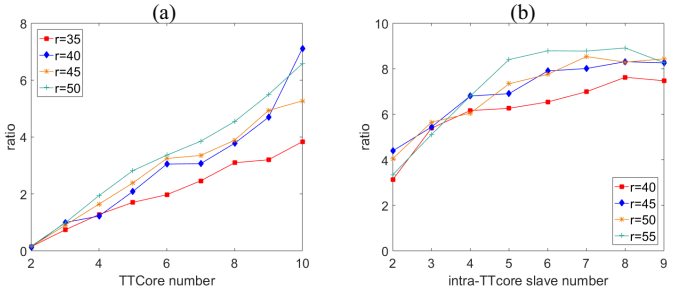


Fig. 15. (a) Serial–parallel ratio comparisons of hybrid scalable scheme with various TT ranks on the first group of random data sets. (b) Serial–parallel ratio comparisons of intra-TTcore parallel model with various TT ranks on the second group of random data sets.

tensor's order and TT ranks. Therefore, if the number of tensor's order is not too large, the parallel efficiency may not be fully utilized. However, if the tensor's dimensionality is large, we can exploit the intra-TTcore parallel scheme to further improve the computation efficiency.

3) Comparisons of Execution Time for TT-Based Inner Product in Hybrid Scalable and Intra-TTcore Parallel Schemes: To verify the efficiency of hybrid scalable and intra-TTcore parallel schemes, we performed TT-based inner product experiments with random data sets and real-world data set to obtain the serial–parallel ratio. In random data experiments, two groups of data were randomly generated. For the first group of experiments using the hybrid scalable scheme, we set the number of TT cores to a range from 2 to 10, and the TT ranks are set to 35, 40, 45, and 50, respectively. For the second group of experiments using the Intra-TTcore parallel model, the number of TT cores is fixed at five. A two-layer parallel architecture is adopted for TT cores whose dimensionality is greater than 200. Then, we can observe the influence of intra-TTcore parallel model on parallel efficiency by changing the intra-TTcore slave number. Besides, the data settings of real-world data set are shown in Section VII-A.

Fig. 15(a) illustrates the serial–parallel ratio of TT-based inner product operation in the hybrid scalable scheme. We can see that the serial–parallel ratio will increase as the number of TT cores increases. When the number of TT cores is 5, the serial–parallel ratio is about 1–3. For TT cores with large dimensionality (i.e., I_n), the TT cores are partitioned to multiple slices along the order located by I_n. Here, new slave nodes are employed for the intra-TTcore parallel model to further improve parallel efficiency. Fig. 15(b) depicts the

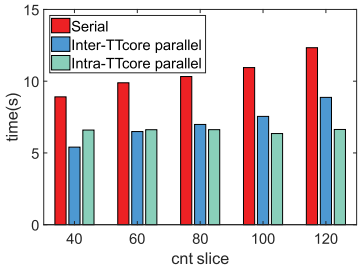


Fig. 16. Comparisons of execution time among serial, inter-TTcore parallel, and intra-TTcore parallel models on real-world data set.

experimental results in intra-TTcore parallel model. After joining the intra-TTcore parallel operations, the serial–parallel ratio will continue to increase as the number of slaves in the two-layer parallel architecture increases, which can reach to a maximum ratio of 9. However, the serial–parallel ratio will not always increase as the number of employed slaves continues to increase. Because more slaves in the intra-TTcore parallel model also bring additional communication overhead. Therefore, in the intra-TTcore parallel model, we should select appropriate number of slaves to perform intra-TTcore parallel operations.

In the eighth-order tensor constructed from real-world data set, the dimensionality of traffic order is relatively large. The parallel operations on the traffic order are performed in the intra-TTcore parallel experiment, and we employ two slaves to perform the intra-TTcore parallel operations. Fig. 16 shows that if the dimensionality of the traffic order is less than 80, there is no advantage when adopting the intra-TTcore parallel mode. However, when the dimensionality of the traffic order is greater than 80, the intra-TTcore parallel mode under two-layer parallel architecture can further improve the computational efficiency. The experimental result can be explained as follows. When the dimensionality of the traffic order is large, the intra-TTcore parallel mode can be used to reduce the computation burden of single TTcore, thereby improving the computational efficiency. However, when the dimensionality I_n of TTcore is small, the intra-TTcore parallel task can be completed fast enough by a single slave, so there is no need to apply the intra-TTcore parallel mode. Moreover, due to the communication overhead, the execution time of the intra-TTcore parallel mode becomes even longer.

From the extensive experimental results, we can see that the proposed STT-TCs approach can significantly improve the computation efficiency, which is conducive to tensor-based data analysis under edge/fog computing environments. In fact, the general STT-TCs approach can also be extended to the cloud to improve the efficiency of tensor-based data analysis.

VIII. Conclusion

To alleviate the dilemma caused by the curse of dimensionality in tensor-based analysis approaches for cyber–physical–social big data, this article focuses on proposing a set of TT-based tensor operations with their scalable computations and presenting a novel TT-based processing framework for cyber–physical–social big data under edge/fog computing environments. In this article, we first summarize a set of

TT-based computation approaches to implement most tensor operations directly based on decomposed TT cores and then propose an STT-CA, including inter-TTcore and intra-TTcore parallel models. Furthermore, we put forward a series of STT-TCs and propose a novel TT-based big data processing framework under edge/fog computing environments. Extensive experimental results based on both random and real-world data sets demonstrate that the proposed STT-TCs can significantly improve computation efficiency and reduce the storage space.

In the future, we shall further study the tensor-by-tensor operation in the TT format. Furthermore, we shall implement some tensor-based analysis approaches directly based on the TT format and analyze the superiority of TT-based approaches in computation time, storage, and analysis results.

REFERENCES

[1] Q. Zhang, L. T. Yang, Z. Chen, and P. Li, "High-order possibilistic c-means algorithms based on tensor decompositions for big data in IoT," *Inf. Fusion*, vol. 39, pp. 72–80, Jan. 2018.

[2] H. Liu, L. T. Yang, M. Lin, D. Yin, and Y. Guo, "A tensor-based holistic edge computing optimization framework for Internet of Things," *IEEE Netw.*, vol. 32, no. 1, pp. 88–95, Jan./Feb. 2018.

[3] D. Evans, "The Internet of Things: How the next evolution of the Internet is changing everything," CISCO, San Jose, CA, USA, White Paper, 2011, no. 1, pp. 1–11.

[4] R. Mitchell and I.-R. Chen, "A survey of intrusion detection techniques for cyber-physical systems," *ACM Comput. Surv.*, vol. 46, no. 4, p. 55, 2014.

[5] V. K. Naik, C. Liu, L. Yang, and J. Wagner, "Online resource matching for heterogeneous grid environments," in *Proc. IEEE Int. Symp. Cluster Comput. Grid (CCGrid)*, vol. 2, 2005, pp. 607–614.

[6] W. Yang, X. Liu, L. Zhang, and L. T. Yang, "Big data real-time processing based on storm," in *Proc. 12th IEEE Int. Conf. Trust, Secur. Privacy Comput. Commun.*, 2013, pp. 1784–1787.

[7] J. Zeng, L. T. Yang, and J. Ma, "A system-level modeling and design for cyber-physical-social systems," *ACM Trans. Embedded Comput. Syst.*, vol. 15, no. 2, 2016, Art. no. 35.

[8] J. Zhang *et al.*, "Power consumption analysis of video streaming in 4G LTE networks," *Wireless Netw.*, vol. 24, no. 8, pp. 3083–3098, 2018.

[9] X. Wang, L. T. Yang, X. Xie, J. Jin, and M. J. Deen, "A cloud-edge computing framework for cyber-physical-social services," *IEEE Commun. Mag.*, vol. 55, no. 11, pp. 80–85, Nov. 2017.

[10] H. Liu *et al.*, "A holistic optimization framework for mobile cloud task scheduling," *IEEE Trans. Sustain. Comput.*, vol. 4, no. 2, pp. 217–230, Apr./Jun. 2019.

[11] Y. Zhang, J. Ren, J. Liu, C. Xu, H. Guo, and Y. Liu, "A survey on emerging computing paradigms for big data," *Chin. J. Electron.*, vol. 26, no. 1, pp. 1–12, Jan. 2017.

[12] 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 Trans. Emerg. Topics Comput.*, vol. 2, no. 3, pp. 280–291, Sep. 2014.

[13] A. Cichocki, N. Lee, I. V. Oseledets, A.-H. Phan, Q. Zhao, and D. Mandic, "Low-rank tensor networks for dimensionality reduction and large-scale optimization problems: Perspectives and challenges PART 1," *Found Trends Mach. Learn.*, vol. 9, nos. 4–5, pp. 229–449, 2016.

[14] H. Liu, J. Ding, L. T. Yang, Y. Guo, X. Wang, and A. Deng, "Multi-dimensional correlative recommendation and adaptive clustering via incremental tensor decomposition for sustainable smart education," *IEEE Trans. Sustain. Comput.*, 2019, doi: 10.1109/TSUSC.2019.2954456.

[15] Y. Zhao, L. T. Yang, and R. Zhang, "A tensor-based multiple clustering approach with its applications in automation systems," *IEEE Trans. Ind. Informat.*, vol. 14, no. 1, pp. 283–291, Jan. 2018.

[16] P. Li, Z. Chen, L. T. Yang, Q. Zhang, and M. J. Deen, "Deep convolutional computation model for feature learning on big data in Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 14, no. 2, pp. 790–798, Feb. 2018.

[17] Q. Zhang, L. T. Yang, Z. Chen, P. Li, and M. J. Deen, "Privacy-preserving double-projection deep computation model with crowdsourcing on cloud for big data feature learning," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 2896–2903, Aug. 2018.

[18] H. Liu, L. T. Yang, J. Chen, M. Ye, J. Ding, and L. Kuang, "Multivariate multi-order Markov multi-modal prediction with its application in network traffic management," *IEEE Trans. Netw. Service Manage.*, vol. 16, no. 3, pp. 828–841, Sep. 2019.

[19] J. Ding, H. Liu, L. T. Yang, T. Yao, and W. Zuo, "Multi-user multivariate multi-order Markov based multi-modal user mobility pattern prediction," *IEEE Internet Things J.*, to be pulished, doi: 10.1109/JIOT.2019.2951134.

[20] 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 Trans. Big Data*, vol. 4, no. 3, pp. 325–340, Sep. 2018.

[21] I. V. Oseledets, "Tensor-train decomposition," *SIAM J. Sci. Comput.*, vol. 33, no. 5, pp. 2295–2317, 2011.

[22] S. V. Dolgov, B. N. Khoromskij, I. V. Oseledets, and D. V. Savostyanov, "Computation of extreme eigenvalues in higher dimensions using block tensor train format," *Comput. Phys. Commun.*, vol. 185, no. 4, pp. 1207–1216, 2014.

[23] D. Kressner and A. Uschmajew, "On low-rank approximability of solutions to high-dimensional operator equations and eigenvalue problems," *Linear Algebra Appl.*, vol. 493, pp. 556–572, Mar. 2016.

[24] Q. Zhang, L. T. Yang, Z. Chen, and P. Li, "A tensor-train deep computation model for industry informatics big data feature learning," *IEEE Trans. Ind. Informat.*, vol. 14, no. 7, pp. 3197–3204, Jul. 2018.

[25] H. Liu, L. T. Yang, J. Ding, Y. Guo, and S. S. Yau, "Tensor-train-based high-order dominant eigen decomposition for multimodal prediction services," *IEEE Trans. Eng. Manag.*, to be published, doi: 10.1109/TEM.2019.2912928.

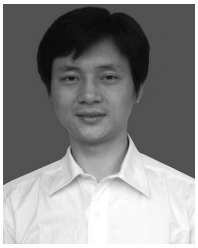
[26] H. Liu, L. T. Yang, T. Yao, J. Ding, Q. Wu, and A. Deng, "Tensor-train-based higher order dominant z-eigen decomposition for multi-modal prediction and its cloud/edge implementation," *IEEE Trans. Netw. Service Eng.*, 2019.

[27] H. Liu *et al.*, "Thermal-aware and DVFS-enabled big data task scheduling for data centers," *IEEE Trans. Big Data*, vol. 4, no. 2, pp. 177–190, Jun. 2018.

[28] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Review*, vol. 51, no. 3, pp. 455–500, 2009.

[29] H. Liu, L. T. Yang, Y. Guo, X. Xie, and J. Ma, "An incremental tensor-train decomposition for cyber-physical-social big data," *IEEE Trans. Big Data*, to be published, doi: 10.1109/TBDATA.2018.2867485.

[30] X. Wang, W. Wang, L. T. Yang, S. Liao, D. Yin, and M. J. Deen, "A distributed HOSVD method with its incremental computation for big data in cyber-physical-social systems," *IEEE Trans. Comput. Social Syst.*, vol. 5, no. 2, pp. 481–492, Jun. 2018.



Huazhong Liu received the B.S. degree in computer science from the School of Computer Science and Technology, Jiangxi Normal University, Nanchang, China, in 2004, and the M.S. degree in computer science from the College of Mathematics and Computer Science, Hunan Normal University, Changsha, China, in 2009. He is currently pursuing the Ph.D. degree with the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China. He is also a Lecturer with the School of Information Science and Technology, Jiujiang University, Jiujiang, China.

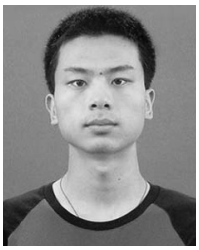
His research interests include big data, cloud computing, the Internet of Things, and scheduling optimization.

Laurence T. Yang (Fellow, IEEE) received the B.E. degree in computer science and technology from Tsinghua University, Beijing, China, in 1992, and the Ph.D. degree in computer science from the University of Victoria, Victoria, BC, Canada, in 2006.

He is currently a Professor with the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China, and the Department of Computer Science, St. Francis Xavier University, Antigonish, NS, Canada. His research interests include parallel and distributed computing, embedded and ubiquitous/pervasive computing, and big data. His research has been supported by the National Sciences and Engineering Research Council (NSERC), Canada, and the Canada Foundation for Innovation (CFI).

Jihong Ding received the B.S. and M.S. degrees in computer science from Hunan Normal University, Changsha, China, in 2004 and 2009, respectively, and the Ph.D. degree in educational technology from Central China Normal University, Wuhan, China, in 2016.

She is currently a Lecturer with the Zhejiang University of Technology, Hangzhou, China. Her research interests include educational resource recommendation and big data analysis.



Yimu Guo received the B.S. degree in computer science from the College of Information, Liaoning University, Shenyang, China, in 2015, and the M.S. degree in computer science from the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China, in 2017. He is currently a Software Engineer of the Department of Social Network Application, Tencent Inc., Shenzhen, China.

His research interests include big data and parallel computing.

Xia Xie received the B.E. degree in computer science and the Ph.D. degree in computer system architecture from the Huazhong University of Science and Technology, Wuhan, China, in 2002 and 2006, respectively.

She is currently an Associate Professor with Services Computing Technique and System Lab, Big Data Technology and System Lab, and Cluster and Grid Computing Lab, the School of Computer Science and Technology, the Huazhong University of Science and Technology. Her research interests include big data mining and performance evaluation.

Zhi-Jie Wang (Member, IEEE) received the Ph.D. degree in computer science from Shanghai Jiao Tong University, Shanghai, China, in 2015.

He was a Post-Doctoral Research Fellow with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong. He was a Research Associate Professor at the School of Data and Computer Science, Sun Yat-sen University (SYSU), Guangzhou, China. He is currently an Associate Professor with the College of Computer Science, Chongqing University (CQU), Chongqing, China. He has published some research articles in venues, such as the IEEE Transactions on Knowledge and Data Engineering, the IEEE Transactions on Multimedia, the IEEE Transactions on Audio, Speech and Language Processing, the IEEE Transactions on Parallel and Distributed Systems, the International Joint Conference on Artificial Intelligence (IJCAI), and the AAAI Conference on Artificial Intelligence (AAAI). His current research interests include distributed computing, artificial intelligence, databases, and data mining.

Dr. Wang is also a member of the China Computer Federation (CCF) and the Association for Computing Machinery (ACM).