IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received October 28, 2020, accepted December 3, 2020, date of publication December 10, 2020,
date of current version December 21, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3043825

Cascaded Coded Distributed Computing Schemes Based on Placement Delivery Arrays

JING JIANG AND LINGXIAO QU
Guangxi Key Laboratory of Multi-Source Information Mining and Security, Guangxi Normal University, Guilin 541004, China
Corresponding author: Jing Jiang (jjiang2008@hotmail.com)

The work of Jing Jiang was supported in part by the National Natural Science Foundation of China under Grant 11601096 and Grant 61672176; in part by Guangxi Natural Science Foundation under Grant 2016GXNSFCA380021, in part by the Guangxi Higher Institutions Program of Introducing 100 High-Level Overseas Talents; in part by the Guangxi Collaborative Innovation Center of Multi-source Information Integration and Intelligent Processing; in part by the Guangxi "Bagui Scholar" Teams for Innovation and Research Project; and in part by the Guangxi Talent Highland Project of Big Data Intelligence and Application.

ABSTRACT Li *et al*. introduced coded distributed computing (CDC) scheme to reduce the communication load in general distributed computing frameworks such as MapReduce. They also proposed cascaded CDC schemes where each output function is computed multiple times, and proved that such schemes achieved the fundamental trade-off between computation load and communication load. However, these schemes require exponentially large numbers of input files and output functions when the number of computing nodes gets large. In this paper, by using the structure of placement delivery arrays (PDAs), we construct several infinite classes of cascaded CDC schemes. We also show that the numbers of output functions in all the new schemes are only a factor of the number of computing nodes, and the number of input files in our new schemes is much smaller than that of input files in CDC schemes derived by Li *et al*.

INDEX TERMS Coded distributed computing, MapReduce, placement delivery array.

I. INTRODUCTION

With the amount of data being generated increasing rapidly, a computation task could have a huge amount of data to be processed. Hence it becomes more and more difficult to complete such a task by using a single server. Therefore, large scale distributed computing systems, where a computation task is distributed to many servers, are becoming very relevant. The MapReduce [9], Hadoop [2] are two of the popular distributed computing frameworks, and have wide application in many areas, for instance, [3]–[5], [10], [13], [14], [17], [18], [20], [21], [32]. In such frameworks, in order to compute the output functions, the computation consists of three phases: map phase, shuffle phase and reduce phase. In the map phase, distributed computing nodes process parts of the input data files locally, generating some intermediate values according to their designed map functions. In the shuffle phase, the nodes exchange the calculated intermediate values among each other, in order to obtain enough values to calculate the final output results by using their designed reduce functions. In the reduce phase, each node computes its designed reduce functions by using the intermediate values derived in the map phase and the shuffle phase.

The associate editor coordinating the review of this manuscript and approving it for publication was Muhammad Imran.

In these frameworks, one has to spend most of execution time on data shuffling. For example, 33% of the execution time is spent on data shuffling in a Facebook's Hadoop cluster [8]. 70% of the execution time is spent on data shuffling when running "SelfJoin" on the Amazon EC2 cluster [1]. Coded distributed computing (CDC) introduced in [16] is an efficient approach to reduce the communication load in shuffle phase. The authors in [16] characterized a fundamental tradeoff between "computation load" in the map phase and "communication load" in the shuffle phase on homogeneous computing networks, i.e., all the nodes have the same storage, computing and communication capabilities. All the schemes mentioned in this paper are based on homogeneous networks, unless otherwise specified. There are many studies on CDC. Some new CDC schemes are constructed by using combinatorial design in [12], [24]. In order to solve the straggler problem, [13], [15], [29] proposed CDC schemes by using error-correcting codes. The authors in [23], [25]–[27] investigated CDC schemes in heterogeneous networks, where nodes could have different storage, computation capabilities and communication constraints.

Observe that in [16], the author proposed CDC schemes with $N = \binom{K}{r}$ input files and $Q = \binom{K}{s}$ output functions, where K is the total number of computing nodes, r is the average number of nodes that map each file, and s is the

number of nodes that compute each reduce function. Obviously, the number of input files $N = \binom{K}{r}$ and the number of output functions $Q = \binom{K}{s}$ increase too quickly with K to be used in practice when K is large. That is, in order to achieve the gain of communication load, a large number of input files and a large number of output functions are needed. In practical scenarios, these requirements could significantly reduce the promised gains of the method [12]. It is desired to design CDC schemes with smaller numbers of input files and output functions. There are a few works paying attention to reducing the values of N and Q, for instance, [12], [24], [30]. However, the number s of nodes that compute each reduce function in most of the known schemes is a certain value, for example, $s = 1$ in the schemes in [12] and [30], $s = 1$ or $s = t$ where $t|K$, i.e., t is a factor of K, in the schemes in [24]. We list these known CDC schemes in Table 1.

However, in practice, the reduce functions are desired to be computed by multiple nodes, which allows for consecutive MapReduce procedures as the reduce function outputs can act as the input files for the next MapReduce procedures [33]. Such a kind of CDC schemes is always called *cascaded* CDC scheme.

In this paper, we focus on cascaded CDC schemes with smaller numbers of input files and output functions.

1) Based on placement delivery array (PDA), which was introduced to construct coded caching schemes in [28], we propose a construction of cascaded CDC schemes. That is, given a known PDA, one can obtain a class of cascaded CDC schemes. We list three classes of new schemes in Table 2.
2) The numbers of output functions in all the new schemes are only a factor of the number of computing nodes K, and the number of input files in our new schemes is exponentially smaller in K than that of the scheme in [16].
3) From the construction in this paper, we can obtain the schemes in [30], i.e., our new schemes include the schemes in [30] as a special case. In addition, our new schemes include some schemes in [12], [16] and [24] as a special case.

The rest of this paper is organized as follows. In Section II, we formulate a general distributed computing framework. In Section III, a construction of cascaded CDC schemes is proposed. In Section IV, we analyse the performance of our new schemes from the construction in Section III,. Finally conclusion is drawn in Section V.

II. PRELIMINARIES

In this section, we give a formulation of our problem. In this system, there are K distributed computing nodes $\mathcal{K} = \{0, 1, \ldots, K-1\}$, N files $\mathcal{W} = \{w_0, w_1, \ldots, w_{N-1}\}$, where $w_n \in \mathbb{F}_{2^D}$ for any $n \in \{0, 1, \ldots, N-1\}$, and Q output functions $\mathcal{Q} = \{\phi_0, \phi_1, \ldots, \phi_{Q-1}\}$, where $\phi_q : \mathbb{F}_{2^D}^N \to \mathbb{F}_{2^B}$ for any $q \in \{0, 1, \ldots, Q-1\}$, which maps all the files to a bit stream $u_q = \phi_q(w_0, w_1, \ldots, w_{N-1}) \in \mathbb{F}_{2^B}$. The goal of node k ($k \in \mathcal{K}$) is responsible for computing a subset of output functions, denoted by a set $\mathcal{Q}_k \subseteq \mathcal{Q}$.

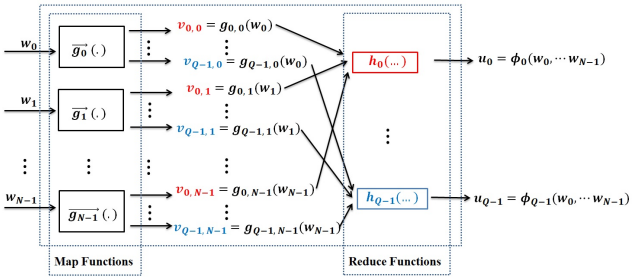

FIGURE 1. Illustration of a two-stage distributed computing framework.

As illustrated in Fig. 1 from [16], we assume that each output function ϕ_q, $q \in \{0, 1, \ldots, Q-1\}$, decomposes as: $\phi_q(w_0, w_1, \ldots, w_{N-1}) = h_q(g_{q,0}(w_0), g_{q,1}(w_1), \ldots, g_{q,N-1}(w_{N-1}))$, where

1) The "map" function $g_{q,n} : \mathbb{F}_{2^D} \to \mathbb{F}_{2^T}$, $n \in \{0, 1, \ldots, N-1\}$, maps the file w_n into the *intermediate value* (IVA) $v_{q,n} \triangleq g_{q,n}(w_n) \in \mathbb{F}_{2^T}$ for a given positive integer T.
2) The "reduce" function $h_q : \mathbb{F}_{2^T}^N \to \mathbb{F}_{2^B}$, maps the IVAs in $\{v_{q,n} \mid n \in \{0, 1, \ldots, N-1\}\}$ into the output value $u_q = h_q(v_{q,0}, v_{q,1}, \ldots, v_{q,N-1})$.

Following the above decomposition, the computation proceeds in the following three phases.

- Map Phase. For each $k \in \mathcal{K}$, node k computes $g_{q,n}$ for $q \in \{0, 1, \ldots, Q-1\}$ and $w_n \in \mathcal{W}_k$, where $\mathcal{W}_k \subseteq \mathcal{W}$ is the subset of files stored by k, i.e., node k locally computes a subset of IVAs
$$\mathcal{C}_k = \{v_{q,n} \mid \phi_q \in \mathcal{Q}, w_n \in \mathcal{W}_k\}.$$
- Shuffle Phase. Denote $\mathcal{Q}_k, k \in \mathcal{K}$, as the subset of output functions which will be computed by node k. In order to obtain the output value of ϕ_q where $\phi_q \in \mathcal{Q}_k$, node k needs to compute $h_q(v_{q,0}, v_{q,1}, \ldots, v_{q,N-1})$, i.e., it needs the IVAs that are not computed locally in the map phase. Hence, in this phase, the K nodes exchange some of their computed IVAs. Suppose that node k creates a message $X_k = \varphi_k(\mathcal{C}_k)$ of some length $l_k \in \mathbb{N}$, using a function $\varphi_k : \mathbb{F}_{2^T}^{|\mathcal{C}_k|} \to \mathbb{F}_{2^{l_k}}$. Then it multicasts this message to all other nodes, where each node receives it error-free.
- Reduce Phase. Using the shuffled messages X_0, X_1, $\ldots$, X_{K-1} and the IVAs in $\mathcal{C}_k$ it computed locally in the map phase, node k now could derive $(v_{q,0}, v_{q,1}, \ldots, v_{q,N-1}) = \psi_q(X_0, X_1, \ldots, X_{K-1}, \mathcal{C}_k)$ for some function $\psi_q : \mathbb{F}_{2^{l_0}} \times \mathbb{F}_{2^{l_1}} \times \ldots \times \mathbb{F}_{2^{l_{K-1}}} \times \mathbb{F}_{2^T}^{|\mathcal{C}_k|} \to \mathbb{F}_{2^T}^N$ where $\phi_q \in \mathcal{Q}_k$. More specifically, node k could derive the following IVAs
$$\{v_{q,n} \mid \phi_q \in \mathcal{Q}_k, n \in \{0, 1, \ldots, N-1\}\},$$
which is enough to compute output value $u_q = h_q(v_{q,0}, v_{q,1}, \ldots, v_{q,N-1})$.

TABLE 1. Some known CDC schemes.

Schemes and Parameters	Number of Nodes K	Computation Load r	Replication Factor s	Number of Files N	Number of Reduce Functions Q	Communication Load L
[16], $K, r, s \in \mathbb{N}^+$ with $1 \leq r, s \leq K$	K	r	s	$\binom{K}{r}$	$\binom{K}{s}$	$\sum_{=\max\{r+1,s\}}^{\min\{r+s,K\}} \frac{l\binom{K}{l}\binom{l-2}{r-1}\binom{r}{l-s}}{r\binom{K}{r}\binom{K}{s}}$
[12], $K, t \in \mathbb{N}^+$ with $t \geq 2$ and $t \mid K$	K	t	1	$(\frac{K}{r})^{r-1}$	K	$\frac{1}{r-1}(1-\frac{r}{K})$
[30], $K, t \in \mathbb{N}^+$ with $t \geq 2$ and $t \mid K$	K	$K-t$	1	$(\frac{K}{r}-1)(\frac{K}{r})^{r-1}$	K	$\frac{1}{r-1}(1-\frac{r}{K})$
[24], $K, t \in \mathbb{N}^+$ with $t \geq 2$ and $t \mid K$	K	t	t	$(\frac{K}{r})^{r-1}$	$(\frac{K}{r})^{r-1}$	$\frac{r^r(K-r)}{K^r(r-1)} + \sum_{=2}^{r}(\frac{r}{K})^r\binom{r}{}(\frac{K}{2})\frac{2}{2-1}$

TABLE 2. New CDC schemes.

Schemes and Parameters	Number of Nodes K	Computation Load r	Replication Factor s	Number of Files N	Number of Reduce Functions Q	Communication Load L
Scheme 1, $K, r, s \in \mathbb{N}^+$ with $1 \leq r, s \leq K$	K	r	s	$\binom{K}{r}$	$\frac{K}{\gcd(K,s)}$	$\frac{s}{r}(1-\frac{r}{K})$
Scheme 2, $K, t, s \in \mathbb{N}^+$ with $t \geq 2, t \mid K$ and $s \leq K$	K	t	s	$(\frac{K}{r})^{r-1}$	$\frac{K}{\gcd(K,s)}$	$\frac{s}{r-1}(1-\frac{r}{K})$
Scheme 3, $K, t, s \in \mathbb{N}^+$ with $t \geq 2, t \mid K$ and $s \leq K$	K	$K-t$	s	$(\frac{K}{r}-1)(\frac{K}{r})^{r-1}$	$\frac{K}{\gcd(K,s)}$	$\frac{s}{r-1}(1-\frac{r}{K})$

Define the *computation load* as $r = \frac{\sum_{k=0}^{K-1}|\mathcal{W}_k|}{N}$ and *communication load* as $L = \frac{\sum_{k=0}^{K-1} l_k}{QNT}$, i.e., r is the average number of nodes that map each file and L is the ratio of the total number of bits transmitted in shuffle phase to QNT. Li *et al.* [16] gave the following optimal computation-communication function.

$$L^*(r,s) = \sum_{l=\max\{r+1,s\}}^{\min\{r+s,K\}} \frac{\binom{K-r}{K-l}\binom{r}{l-s}}{\binom{K}{s}} \frac{l-r}{l-1}, \tag{1}$$

where K is the number of nodes, r is the computation load and s is the number of nodes that compute each reduce function. Moreover, the authors proposed some schemes achieving the above optimal computation-communication function.

Lemma 1 (See [16]): Suppose that K, r, and s are positive integers. Then there exists a CDC scheme with K nodes, $N = \binom{K}{r}$ files and $Q = \binom{K}{s}$ output functions, such that the communication load is

$$L_{Li}(r,s) = \sum_{l=\max\{r+1,s\}}^{\min\{r+s,K\}} \frac{\binom{K-r}{K-l}\binom{r}{l-s}}{\binom{K}{s}} \frac{l-r}{l-1},$$

where r is the computation load and s is the number of nodes that compute each reduce function.

For convenience, the above CDC schemes are called Li-CDC schemes in this paper. Obviously, in the Li-CDC schemes, the numbers of input files $N = \binom{K}{r}$ and output functions $Q = \binom{K}{s}$ are too large to be of practical use. In the next section, we will construct some classes of CDC schemes with smaller numbers of input files and output functions.

III. A NEW CONSTRUCTION OF CDC SCHEMES

In this section, we will give a construction of cascaded CDC schemes by using placement delivery array which was introduced by Yan *et al.* [28] to study coded caching schemes.

Definition 1 (See [28]): Suppose that K, N, Z and S are positive integers. $\mathbf{P} = (p_{n,k})$, $n \in \{0, 1, \ldots, N-1\}$, $k \in \{0, 1, \ldots, K-1\}$, is an $N \times K$ array composed of a specific symbol "$*$" and positive integers $0, 1, \cdots, S-1$. Then $\mathbf{P}$ is a (K, N, Z, S) placement delivery array (PDA for short) if

1) the symbol "$*$" occurs exactly Z times in each column;
2) each integer appears at least once in the array;
3) for any two distinct entries p_{n_1,k_1} and p_{n_2,k_2}, $p_{n_1,k_1} = p_{n_2,k_2} = u$ is an integer only if
   a. $n_1 \neq n_2$, $k_1 \neq k_2$, i.e., they lie in distinct rows and distinct columns; and
   b. $p_{n_1,k_2} = p_{n_2,k_1} = *$, i.e., the corresponding 2×2 subarray formed by rows n_1, n_2 and columns k_1, k_2 must be of the following form

$$\begin{pmatrix} u & * \\ * & u \end{pmatrix} \text{ or } \begin{pmatrix} * & u \\ u & * \end{pmatrix}.$$

A (K, N, Z, S) PDA $\mathbf{P}$ is *g-regular*, denoted as g-(K, N, Z, S) PDA, if each integer in $\{0, 1, \ldots, S-1\}$ occurs exactly g times in $\mathbf{P}$.

Example 1: We can directly check that the following array is a 3-(6, 4, 2, 4) PDA:

$$\mathbf{P}_{4\times 6} = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{pmatrix} * & * & 0 & * & 1 & 2 \\ * & 0 & * & 1 & * & 3 \\ 0 & * & * & 2 & 3 & * \\ 1 & 2 & 3 & * & * & * \end{pmatrix} \end{array}.$$

The concept of a PDA is a useful tool to construct coded caching schemes, for instance, [6], [7], [22], [28]. Recently, Yan *et al.* [30], [31] used PDAs to construct CDC schemes with $s = 1$, where s is the number of nodes that compute each reduce function.

Lemma 2 (See [30]): Suppose that there exists a $g-(K,N,Z,S)$ PDA with $g \geq 2$. Then there exists a CDC scheme satisfying the following properties:

1) it consists of K distributed computing nodes $\mathcal{K} = \{0, 1, \ldots, K-1\}$, N files and $Q = K$ output functions $\mathcal{Q} = \{\phi_0, \phi_1, \ldots, \phi_{Q-1}\}$;
2) node k, where $k \in \mathcal{K}$, is responsible for computing ϕ_k;
3) the computation load is $r = \frac{KZ}{N}$ and the number of IVAs multicasted by all the nodes is $\frac{gS}{g-1}$.

In the above scheme, the numbers of nodes and files are corresponding to the numbers of columns and rows of the PDA, respectively. Furthermore, node k, $k \in \{0, 1, \ldots, K-1\}$, is responsible for computing ϕ_k, i.e., distinct nodes are responsible for computing distinct output functions. So the number Q of output functions and the number of nodes are the same, i.e., $Q = K$. In order to obtain the main results of this section, the property that some nodes are responsible for computing the same output function is needed.

Example 2: Consider the 3-$(6, 4, 2, 4)$ PDA $\mathbf{P}_{4\times 6}$ in Example 1. By using $\mathbf{P}_{4\times 6}$, we will construct a CDC scheme with $K = 6$ nodes $\{0, 1, 2, 3, 4, 5\}$, $N = 4$ files $\{w_0, w_1, w_2, w_3\}$, $Q = 2$ output functions $\{\phi_0, \phi_1\}$ (note that $Q = K = 6$ in Lemma 2).

- Map phase. For each $k \in \{0, 1, 2, 3, 4, 5\}$, node k stores the files in

$$\mathcal{W}_k = \{w_n \mid n \in \{0, 1, 2, 3\}, p_{n,k} = *\}, \tag{2}$$

i.e.,

$$\mathcal{W}_0 = \{w_0, w_1\}, \mathcal{W}_1 = \{w_0, w_2\}, \mathcal{W}_2 = \{w_1, w_2\},$$
$$\mathcal{W}_3 = \{w_0, w_3\}, \mathcal{W}_4 = \{w_1, w_3\}, \mathcal{W}_5 = \{w_2, w_3\}.$$

- Shuffle phase. For each $k \in \{0, 1, 3, 4\}$, node k is responsible for computing ϕ_0, i.e., it needs to obtain output value $u_0 = h_0(v_{0,0}, v_{0,1}, v_{0,2}, v_{0,3})$. For each $k \in \{2, 5\}$, node k is responsible for computing ϕ_1. That is, node k is responsible for computing ϕ_{q_k}, where $(q_0, q_1, q_2, q_3, q_4, q_5) = (0, 0, 1, 0, 0, 1)$. We take node 0 as an example, i.e., in order to compute the output value u_0, node 0 should obtain all the IVAs in $\{v_{0,n} \mid n \in \{0, 1, 2, 3\}\}$. Since node 0 stores w_0 and w_1, it can locally compute the IVAs in

$$\{v_{0,n} \mid n \in \{0, 1\}\}. \tag{3}$$

Since $p_{2,0} = 0$, node 0 can derive $v_{0,2}$ from the following subarray $\mathbf{P}^0$ formed by the rows the rows n_1, n_2, n_3 and columns k_1,k_2,k_3, where $p_{n_1,k_1} = p_{n_2,k_2} = p_{n_3,k_3} = 0$.

$$\mathbf{P}^0 = \begin{array}{c} \\ 0 \\ 1 \\ 2 \end{array} \begin{array}{c} \begin{array}{ccc} 0 & 1 & 2 \end{array} \\ \begin{pmatrix} * & * & 0 \\ * & 0 & * \\ 0 & * & * \end{pmatrix} \end{array}$$

Observe $\mathbf{P}^0$. From (2), $p_{n,k} = *$ means that the node k can locally compute $v_{q_k,n}$. So, from $\mathbf{P}^0$, we know that node 0, 1 and 2 do not know $v_{q_0,2} = v_{0,2}$, $v_{q_1,1} = v_{0,1}$ and $v_{q_2,0} = v_{1,0}$, respectively. Divide $v_{0,2}$, $v_{0,1}$ and $v_{1,0}$ into 2 disjoint segments, respectively, i.e.,

$$v_{0,2} = (v_{0,2}^{(1)}, v_{0,2}^{(2)}), v_{0,1} = (v_{0,1}^{(0)}, v_{0,1}^{(2)}), v_{1,0} = (v_{1,0}^{(0)}, v_{1,0}^{(1)}).$$

For a segment $v_{q,n}^{(k)}$, the superscript k represents the node that can locally compute this IVA, which implies the segment $v_{q,n}^{(k)}$ will be transmitted by node k. That is

- node 0 multicasts the message $v_{0,1}^{(0)} \bigoplus v_{1,0}^{(0)}$ to nodes 1 and 2,
- node 1 multicasts the message $v_{0,2}^{(1)} \bigoplus v_{1,0}^{(1)}$ to nodes 0 and 2.
- node 2 multicasts the message $v_{0,2}^{(2)} \bigoplus v_{0,1}^{(2)}$ to nodes 0 and 1.

Since node 0 can compute $v_{1,0}$ and $v_{0,1}$ locally, it can derive $v_{0,2}^{(1)}$ and $v_{0,2}^{(2)}$ from the messages $v_{0,2}^{(1)} \bigoplus v_{1,0}^{(1)}$ multicasted by node 1 and $v_{0,2}^{(2)} \bigoplus v_{0,1}^{(2)}$ multicasted by node 2, respectively. This implies that node 0 can obtain $v_{0,2} = (v_{0,2}^{(1)}, v_{0,2}^{(2)})$. By using similar method, node 0 can obtain $v_{0,3}$. Together with the known IVAs in (3), node 0 can obtain all the IVAs in $\{v_{0,n} \mid n \in \{0, 1, 2, 3\}\}$. Similarly, node $k, k \in \{1, 3, 4\}$ can obtain all the IVAs in $\{v_{0,n} \mid n \in \{0, 1, 2, 3\}\}$ and node k, $k \in \{2, 5\}$ can obtain all the IVAs in $\{v_{1,n} \mid n \in \{0, 1, 2, 3\}\}$.

- Reduce phase. By using the IVAs derived in map phase and shuffle phase, for each $k_1 \in \{0, 1, 3, 4\}$ and $k_2 \in \{2, 5\}$, node k_1 and k_2 can compute output values u_0 and u_1, respectively. Since each node stores 2 files, the computation load is $r = \frac{2\times 6}{N} = \frac{2\times 6}{4} = 3$. For each integer $u \in \{0, 1, 2, 3\}$, the number of IVAs multicasted by the nodes from $\mathbf{P}^u$ is $\frac{1}{2} \times 3$. So the total number of IVAs multicasted by all the nodes is $4 \times \frac{1}{2} \times 3 = 6$.

Lemma 3: Suppose that there exists a g-(K, N, Z, S) PDA with $g \geq 2$. Then there exists a CDC scheme satisfying the following properties:

1) it contains K distributed computing nodes $\mathcal{K} = \{0, 1, \ldots, K-1\}$, N files and Q output functions $\mathcal{Q} = \{\phi_0, \phi_1, \ldots, \phi_{Q-1}\}$, where $Q \leq K$;
2) node k is responsible for computing ϕ_{q_k}, where $\phi_{q_k} \in \mathcal{Q}$;
3) the computation load is $r = \frac{KZ}{N}$ and the number of IVAs multicasted by all the nodes is $\frac{gS}{g-1}$.

We now pay our attention to the proof of Lemma 3. Given a g-(K, N, Z, S) PDA $\mathbf{P} = (p_{n,k})$, $n \in \{0, 1, \ldots, N-1\}$, $k \in \{0, 1, \ldots, K-1\}$, we can construct a CDC scheme with K nodes $\mathcal{K} = \{0, 1, \ldots, K-1\}$, N files $\mathcal{W} = \{w_0, w_1, \ldots, w_{N-1}\}$ and Q output functions $\mathcal{Q} = \{\phi_0, \phi_1, \ldots, \phi_{Q-1}\}$, where node $k \in \mathcal{K}$ is responsible for computing ϕ_{q_k} such that $\phi_{q_k} \in \mathcal{Q}$ and $\cup_{k\in\mathcal{K}}\phi_{q_k} = \mathcal{Q}$.

- Map phase. Node k, where $k \in \{0, 1, \ldots, K-1\}$, stores the files in

$$\mathcal{W}_k = \{w_n \mid n \in \{0, 1, \ldots, N-1\}, p_{n,k} = *\}. \tag{4}$$

Hence the node k can compute the IVAs in the set

$$\bigcup_{p_{n,k}=*} v_{q_k,n}. \tag{5}$$

We can also obtain the computation load

$$r = \frac{\sum_{k=0}^{K-1} |\mathcal{W}_k|}{N} = \frac{\sum_{k=0}^{K-1} Z}{N} = \frac{KZ}{N}.$$

- Shuffle phase. Note that node k, where $k \in \{0, 1, \ldots, K-1\}$, is responsible for computing the output function ϕ_{q_k}. According to the definition of g-regular, each integer $u \in \{0, 1, \ldots, S-1\}$ occurs g times in $\mathbf{P}$. Suppose that $p_{n_1,k_1} = p_{n_2,k_2} = \ldots = p_{n_g,k_g} = u$. From condition 3) of Definition 1, the subarray of $\mathbf{P}^u$ formed by the rows $n_1, n_2, \ldots, n_g$ and columns $k_1, k_2, \ldots, k_g$ is of the following form:

$$\mathbf{P}^u = \begin{array}{c} \\ n_1 \\ n_2 \\ \vdots \\ n_g \end{array} \begin{array}{c} \begin{array}{cccc} k_1 & k_2 & \cdots & k_g \end{array} \\ \begin{pmatrix} u & * & \cdots & * \\ * & u & \cdots & * \\ \vdots & \vdots & \vdots & \\ * & * & \cdots & u \end{pmatrix} \end{array}. \tag{6}$$

Suppose that node k_j, $j \in \{1, 2, \ldots, g\}$, is responsible for computing $\phi_{q_{k_j}}$. Divide $v_{q_{k_j},n_j}$ into $g-1$ segments, i.e.,

$$v_{q_{k_j},n_j} = (v^{(k_1)}_{q_{k_j},n_j}, \ldots, v^{(k_{j-1})}_{q_{k_j},n_j}, v^{(k_{j+1})}_{q_{k_j},n_j}, \ldots, v^{(k_g)}_{q_{k_j},n_j}). \tag{7}$$

The superscript k_l, $l \in \{1, 2, \ldots, g\}$, of a segment means that such a segment will be transmitted by k_l. For any $l \in \{1, 2, \ldots, g\}$, the node k_l multicasts the following message:

$$\bigoplus_{t \in \{1,2,\ldots,g\} \setminus \{l\}} v^{(k_l)}_{q_{k_t},n_t}. \tag{8}$$

Hence the number of IVAs multicasted by the nodes $k_1, k_2, \ldots, k_g$ is $\frac{g}{g-1}$ for the integer u. Since there are S integers in $\{0, 1, \ldots, S-1\}$, the total number of IVAs multicasted by all the nodes are $\frac{gS}{g-1}$.

In order to show the correctness of the scheme in the reduce phase, we now prove that for any $j \in \{1, 2, \ldots, g\}$, the node k_j will obtain the IVAs $v_{q_{k_j},n_j}$ from $\mathbf{P}^u$ in (6), where $p_{n_j,k_j} = u$.

1) Since $p_{n_1,k_1} = p_{n_2,k_2} = \ldots = p_{n_g,k_g} = u$, according to the definition of a PDA, for any $j \in \{1, 2, \ldots, g\}$, we have $p_{n_t,k_j} = *$ for any $t \in \{1, 2, \ldots, g\} \setminus \{j\}$. Then node k_j stores file w_{n_t} from (4), which implies that it can locally compute $v_{q_{k_l},n_t}$ for any $l \in \{1, 2, \ldots, g\}$. So k_j can compute $v_{q_{k_t},n_t}$ for any $t \in \{1, 2, \ldots, g\} \setminus \{j\}$.
2) We take k_1 as an example, i.e., node k_1 will obtain the IVA $v_{q_{k_1},n_1}$. According to (7), it need segments $v^{(k_2)}_{q_{k_1},n_1}$, $v^{(k_3)}_{q_{k_1},n_1}$, $\ldots$, $v^{(k_g)}_{q_{k_1},n_1}$. For the segment $v^{(k_2)}_{q_{k_1},n_1}$, from (8), node k_2 multicasts the message $\bigoplus_{t \in \{1,3,4,\ldots,g\}} v^{(k_2)}_{q_{k_t},n_t}$. From 1), k_1 can compute $v_{q_{k_t},n_t}$ for any $t \in \{2, 3, \ldots, g\}$, which implies it can locally compute $v^{(k_2)}_{q_{k_t},n_t}$ for any $t \in \{3, 4, \ldots, g\}$. So the node k_1 can obtain the segment $v^{(k_2)}_{q_{k_1},n_1}$ from the message $\bigoplus_{t \in \{1,3,4,\ldots,g\}} v^{(k_2)}_{q_{k_t},n_t}$ multicasted by k_2. Similarly, the node k_1 can obtain the segment $v^{(k_l)}_{q_{k_1},n_1}$ for any $l \in \{3, 4, \ldots, g\}$ from the message $\bigoplus_{t \in \{1,2,\ldots,g\} \setminus \{l\}} v^{(k_l)}_{q_{k_t},n_t}$ multicasted by k_l. Now the node k_1 recovers the IVA

$$v_{q_{k_1},n_1} = (v^{(k_2)}_{q_{k_1},n_1}, v^{(k_3)}_{q_{k_1},n_1}, \ldots, v^{(k_g)}_{q_{k_1},n_1}).$$

Similarly, for any $j \in \{2, 3, \ldots, g\}$, node k_j could recover $v_{q_{k_j},n_j}$ from $\mathbf{P}^u$ in (6).

- Reduce phase: Consider node k, where $k \in \{0, 1, \ldots, K-1\}$. Since the node k is responsible for computing ϕ_{q_k}, it needs to know the IVAs in the set

$$\{v_{q_k,n} \mid n \in \{0, 1, \ldots, N-1\}\} = (\bigcup_{p_{n,k}=*} v_{q_k,n}) \bigcup (\bigcup_{p_{n,k} \neq *} v_{q_k,n}).$$

From (5), the node k can locally compute $\bigcup_{p_{n,k}=*} v_{q_k,n}$. Hence it only needs to derive $\bigcup_{p_{n,k} \neq *} v_{q_k,n}$. For any $p_{n,k} \neq *$, there exists an integer $u \in \{0, 1, \ldots, S-1\}$ such that $p_{n,k} = u$. From the shuffle phase, the node k can get the IVA $v_{q_k,n}$ from $\mathbf{P}^u$ in (6). That is node k can derive all the IVAs in $\bigcup_{p_{n,k} \neq *} v_{q_k,n}$.

Next, we will use Lemma 3 to derive some cascaded CDC schemes where the parameter s of such schemes is a positive integer such that $s > 1$.

Theorem 1: Suppose that there exists a g-(K, N, Z, S) PDA with $g \geq 2$. Then for any positive integer s with $s \leq K$, there exists a cascaded CDC scheme consisting of K distributed computing nodes, N files and $Q = \frac{K}{\gcd(K,s)}$ output functions such that the computation load is $r = \frac{KZ}{N}$ and the communication load is $L = \frac{gsS}{(g-1)KN}$.

Remark 1: The number $Q = \frac{K}{\gcd(K,s)}$ of output functions in the above new scheme is only a factor of the number of nodes K, which is much smaller than the number $Q_{Li} = \binom{K}{s}$ of output functions in Li-CDC scheme.

Remark 2: Applying Theorem 1 with $s = 1$, the scheme is the same as the scheme in Lemma 2. That is, the schemes in Theorem 1 include the schemes in [30] as a special case.

The rest of the section is devoted to the proof of Theorem 1. Given a g-(K, N, Z, S) PDA $\mathbf{P} = (p_{n,k})$, $n \in \{0, 1, \ldots, N-1\}$, $k \in \{0, 1, \ldots, K-1\}$, we can construct a CDC scheme with K nodes $\mathcal{K} = \{0, 1, \ldots, K-1\}$, N files $\mathcal{W} = \{w_0, w_1, \ldots, w_{N-1}\}$ and $Q = \frac{K}{\gcd(K,s)}$ output functions $\mathcal{Q} = \{\phi_0, \phi_1, \ldots, \phi_{Q-1}\}$.

- Map phase. Node k, where $k \in \{0, 1, \ldots, K-1\}$, stores the files in

$$\mathcal{W}_k = \{w_n \mid n \in \{0, 1, \ldots, N-1\}, p_{n,k} = *\}, \tag{9}$$

Hence the node k can compute the IVAs in the set $\bigcup_{p_{n,k}=*} v_{q_k,n}$. We can also obtain the computation load

$$r = \frac{\sum_{k=0}^{K-1} |\mathcal{W}_k|}{N} = \frac{\sum_{k=0}^{K-1} Z}{N} = \frac{KZ}{N}.$$

- Shuffle phase. Node k, where $k \in \{0, 1, \ldots, K-1\}$, is responsible for computing a subset of output functions

$$\mathcal{Q}_k = \{\phi_{<ke>_Q}, \phi_{<ke+1>_Q}, \ldots, \phi_{<(k+1)e-1>_Q}\}, \quad (10)$$

where $e = \frac{sQ}{K}$ and $< a >_b$ is the least non-negative residue of a modulo b for any positive integers a and b. We can directly check that $|\mathcal{Q}_k| = e$ and each output function is computed exactly s times.
We divide the processes into e steps such that in the ith step, $1 \le i \le e$, node $k \in \mathcal{K}$ is responsible for computing $\phi_{<ke+i>_Q}$. Then in such a step, by using Lemma 3, the node k can derive enough numbers of IVAs for computing $\phi_{<ke+i>_Q}$ and the number of IVAs multicasted by all the nodes is $\frac{gS}{g-1}$. There are e steps, thus the total number of IVAs multicasted by all the nodes is $\frac{gSe}{g-1}$. So the communication load is

$$L = \frac{\frac{gSe}{g-1}}{QN} = \frac{\frac{gS\frac{sQ}{K}}{g-1}}{QN} = \frac{gsS}{(g-1)KN}.$$

Example 3: The following array is a 4-(10, 5, 3, 5) PDA.

$$\mathbf{P}_{5\times 10} = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{array} \\ \begin{pmatrix} * & * & * & * & * & * & 0 & 1 & 2 & 3 \\ * & * & * & 0 & 1 & 2 & * & * & * & 4 \\ * & 0 & 1 & * & * & 3 & * & * & 4 & * \\ 0 & * & 2 & * & 3 & * & * & 4 & * & * \\ 1 & 2 & * & 3 & * & * & 4 & * & * & * \end{pmatrix} \end{array}$$

From Theorem 1, for $s = 4$, we can construct a CDC scheme with $K = 10$ nodes $\mathcal{K} = \{0, 1, \ldots, 9\}$, $N = 5$ files $\mathcal{W} = \{w_0, w_1, w_2, w_3, w_4\}$, $Q = \frac{K}{\gcd(K,s)} = 5$ functions $\mathcal{Q} = \{\phi_0, \phi_1, \phi_2, \phi_3, \phi_4\}$. Then, according to (9) and (10),

$$\mathcal{W}_0 = \{w_0, w_1, w_2\}, \mathcal{W}_1 = \{w_0, w_1, w_3\},$$
$$\mathcal{W}_2 = \{w_0, w_1, w_4\}, \mathcal{W}_3 = \{w_0, w_2, w_3\},$$
$$\mathcal{W}_4 = \{w_0, w_2, w_4\}, \mathcal{W}_5 = \{w_0, w_3, w_4\},$$
$$\mathcal{W}_6 = \{w_1, w_2, w_3\}, \mathcal{W}_7 = \{w_1, w_2, w_4\},$$
$$\mathcal{W}_8 = \{w_1, w_3, w_4\}, \mathcal{W}_9 = \{w_2, w_3, w_4\},$$

$e = \frac{sQ}{K} = 2$, and

$$\mathcal{Q}_0 = \{\phi_0, \phi_1\}, \mathcal{Q}_1 = \{\phi_2, \phi_3\},$$
$$\mathcal{Q}_2 = \{\phi_4, \phi_0\}, \mathcal{Q}_3 = \{\phi_1, \phi_2\},$$
$$\mathcal{Q}_4 = \{\phi_3, \phi_4\}, \mathcal{Q}_5 = \{\phi_0, \phi_1\},$$
$$\mathcal{Q}_6 = \{\phi_2, \phi_3\}, \mathcal{Q}_7 = \{\phi_4, \phi_0\},$$
$$\mathcal{Q}_8 = \{\phi_1, \phi_2\}, \mathcal{Q}_9 = \{\phi_3, \phi_4\}.$$

We divide the processes into $e = 2$ steps.

- In the first step, node k, where $k \in \{0, 1, \ldots, 9\}$, is responsible for computing $\phi_{<ke>_Q} = \phi_{<2k>_5}$.

TABLE 3. The first step of shuffle phase.

Node	0	1	2	3	4	5	6	7	8	9
output function	ϕ_0	ϕ_2	ϕ_4	ϕ_1	ϕ_3	ϕ_0	ϕ_2	ϕ_4	ϕ_1	ϕ_3

We list them in Table 3. By using Lemma 3, node k, $k \in \{0, 1, \ldots, 9\}$ can derive enough IVAs for computing $\phi_{<2k>_5}$.

- In the second step, similar to the first step, node k, $k \in \{0, 1, \ldots, 9\}$ can derive enough IVAs for computing $\phi_{<ke+1>_Q} = \phi_{<2k+1>_5}$. We list them in Table 4.

TABLE 4. The second step of shuffle phase.

Node	0	1	2	3	4	5	6	7	8	9
output function	ϕ_1	ϕ_3	ϕ_0	ϕ_2	ϕ_4	ϕ_1	ϕ_3	ϕ_0	ϕ_2	ϕ_4

So node k, $k \in \{0, 1, \ldots, K-1\}$ can compute the output functions in $\mathcal{Q}_k$. The computation load is $r = \frac{10\times 3}{5} = 6$. The total number of IVAs multicasted by all the nodes is $\frac{1}{3} \times 4 \times 10 = \frac{40}{3}$.

IV. PERFORMANCE

From Theorem 1, we can directly obtain CDC schemes from known PDAs. We list some known results on g-(K, N, Z, S) PDAs in Table 5, where $t|K$ represents that t is a factor of K. The interested readers can be reffered to [6], [7], [22], [28] for more known results about PDAs.

A. THE FIRST NEW SCHEME

Let $\mathbf{P}$ be a $(t+1)$-$(K, \binom{K}{t}, \binom{K-1}{t-1}, \binom{K}{t+1})$ PDA from [19] (the PDA in the second row of Table 5). From Theorem 1, for any positive integer $s \le K$, one can obtain a CDC scheme, say Scheme 1, with K nodes, $N = \binom{K}{t}$ files and $Q = \frac{K}{\gcd(K,s)}$ output functions, where s is corresponding to the number of nodes that compute each output function. Furthermore, the computation load is

$$r = \frac{KZ}{N} = \frac{K\binom{K-1}{t-1}}{\binom{K}{t}} = t,$$

and the communication load is

$$L_1(r, s) = \frac{sgS}{(g-1)KN} = \frac{s(t+1)\binom{K}{t+1}}{((t+1)-1)K\binom{K}{t}}$$
$$= \frac{s}{t}(1 - \frac{t}{K}) = \frac{s}{r}(1 - \frac{r}{K}).$$

Note that if $s \ge \frac{rK}{K-r}$, we have $L_1(r, s) = \frac{s}{r}(1 - \frac{r}{K}) \ge 1$. In this case, the nodes do not need to transmit coded messages. Instead, each IVA can be multicasted by one node which can compute the IVA locally. By using this method, the communication load $L_1(r, s) = 1$. So the communication load is $L_1(r, s) = \min\{\frac{s}{r}(1 - \frac{r}{K}), 1\}$.

TABLE 5. Some known results on PDAs.

References and Parameters	g	K	N	Z	S
[19], $K, t \in \mathbb{N}^+$ with $1 \leq t \leq K-1$	$t+1$	K	$\binom{K}{t}$	$\binom{K-1}{t-1}$	$\binom{K}{t+1}$
[28], $K, t \in \mathbb{N}^+$ with $t \geq 2$ and $t \mid K$	t	K	$(\frac{K}{t})^{t-1}$	$(\frac{K}{t})^{t-2}$	$(\frac{K}{t})^{t} - (\frac{K}{t})^{t-1}$
[28], $K, t \in \mathbb{N}^+$ with $t \geq 2$ and $t \mid K$	$K-t$	K	$(\frac{K}{t}-1)(\frac{K}{t})^{t-1}$	$(\frac{K}{t}-1)^2(\frac{K}{t})^{t-2}$	$(\frac{K}{t})^{t-1}$

1) OPTIMALITY

When $s = 1$ and $1 \leq r < K-1$, (1) can be written as

$$\begin{aligned} L^*(r,1) &= \sum_{l=\max\{r+1,1\}}^{\min\{r+1,K\}} \frac{\binom{\text{K-r}}{K-l}\binom{r}{l-1}}{\binom{K}{1}} \frac{l-r}{l-1} \\ &= \sum_{l=\max\{r+1\}}^{\min\{r+1\}} \frac{\binom{\text{K-r}}{K-l}\binom{r}{l-1}}{\binom{K}{1}} \frac{l-r}{l-1} \\ &= \frac{1}{r}(1-\frac{r}{K}). \end{aligned}$$

Obviously, the communication load $L_1(r,1) = \frac{1}{r}(1-\frac{r}{K})$ of Scheme 1 achieves the optimal computation-communication trade-off.

Remark 3: In this case, one can directly check that our scheme is the same as the scheme with $s = 1$ proposed in [16].

When $1 \leq s \leq K$ and $r = K-1$, from (1), we have

$$\begin{aligned} L^*(K-1,s) &= \sum_{l=\max\{K-1+s,K\}}^{\min\{K-1+s,K\}} \frac{\binom{K-(K-1)}{K-l}\binom{K-1}{l-s}}{\binom{K}{s}} \frac{l-(K-1)}{l-1} \\ &= \sum_{l=\max\{K\}}^{\min\{K\}} \frac{\binom{K-(K-1)}{K-l}\binom{K-1}{l-s}}{\binom{K}{s}} \frac{l-(K-1)}{l-1} \\ &= \frac{s}{K-1}(1-\frac{K-1}{K}) \\ &= \frac{s}{r}(1-\frac{r}{K}). \end{aligned}$$

So in this case, the communication load of Scheme 1 achieves the optimal computation-communication trade-off.

Remark 4: One can show that the number of output functions in Scheme 1 is much smaller than that of output functions in Li-CDC scheme. The number of output functions in Li-CDC scheme is $Q_{Li} = \binom{K}{s}$, while the number of output functions in Scheme 1 is $Q_1 = \frac{K}{\gcd(K,s)}$. For example, if $s = K/w$ where w is a factor of K, then $Q_{Li} = \binom{K}{K/w}$ and $Q_1 = \frac{K}{\gcd(K,K/w)} = w$, respectively. We list the cases $w = 2$ and $w = 3$ when $2 \leq K \leq 20$ in Table 6 and in Table 7, respectively.

2) COMPARISON

For the other values of K, r and s, we conjecture that $H_1(r,s) = \frac{L_1(r,s)}{L^*(r,s)} \leq 2$. Unfortunately, the structure of the formula $L^*(r,s) = \sum_{l=\max\{r+1,s\}}^{\min\{r+s,K\}} \frac{\binom{\text{K-r}}{K-l}\binom{r}{l-s}}{\binom{K}{s}} \frac{l-r}{l-1}$ is too complex to prove this conjecture. However, we could find out some values of K, r and s satisfying that $H_1(r,s) = \frac{L_1(r,s)}{L^*(r,s)} \leq 2$.

TABLE 6. The numbers of output functions in Li-CDC scheme and Scheme 1 when K is even and $s = \frac{K}{2}$.

Number of Node K	$Q_{Li} = \binom{K}{s}$	$Q_1 = \frac{K}{\gcd(K,s)}$	K	Q_{Li}	Q_1
2	2	2	12	924	2
4	6	2	14	3432	2
6	20	2	16	12870	2
8	70	2	18	48620	2
10	252	2	20	184756	2

TABLE 7. The numbers of reduce functions in Li-CDC scheme and Scheme 1 when $3 \mid K$ and $s = \frac{K}{3}$.

Number of Node K	$Q_{Li} = \binom{K}{s}$	$Q_1 = \frac{K}{\gcd(K,s)}$	K	Q_{Li}	Q_1
3	3	3	12	495	3
6	15	3	15	3003	3
9	84	3	18	18564	3

Theorem 2: For any positive integers K, r and s with $r, s \leq K \leq 1000$, $H_1(r,s) = \frac{L_1(r,s)}{L^*(r,s)} \leq 2$.

Proof: With the aid of a computer, one can find out that $H_1(r,s) \leq 2$ holds for all the positive integers K, r and s with $r, s \leq K \leq 1000$. Here we only list some cases in Table 8. For the other cases, we omit it and the interested readers may contact the author for a copy.

TABLE 8. The ratio of $L_1(r,s)$ to $L^*(r,s)$ with $K = 16$.

Computation Load r	Replication Factor s	Communication Load $L^*(r,s)$	Communication Load $L_1(r,s)$	$H_1(r,s) = \frac{L_1(r,s)}{L^*(r,s)}$
3	1	0.2708	0.2708	1.0000
	2	0.4333	0.5417	1.2500
	3	0.5388	0.8125	1.5086
5	3	0.3293	0.4125	1.2528
	5	0.4540	0.6875	1.5143
	7	0.5406	0.9625	1.7804
8	5	0.2555	0.3125	1.2233
	8	0.3579	0.5000	1.3971
	10	0.4125	0.6250	1.5150

Remark 5: For the parameters K, r and s satisfying the conditions in Theorem 2, the communication loads of Scheme 1 are slightly larger than those of Li-CDC scheme. However, similar to Remark 4, one can show that the number of output functions in Scheme 1 is much smaller than that of output functions in Li-CDC scheme.

We also prove there exist some values of K, r and s such that $H_1(r,s) = \frac{L_1(r,s)}{L^*(r,s)} \leq 2$ by using theoretical analysis.

Lemma 4: For any positive integers K, r and s with $r \geq s$ and $K \geq \frac{3rs(7r-s+1)}{8(r-s+1)}$, $H_1(r,s) = \frac{L_1(r,s)}{L^*(r,s)} \leq 2$.

The proof of Lemma 4 could be found in Appendix A.

By using Lemma 4, we can provide a method to show $H_1(r,s) \leq 2$ for some positive integers s, r and K. More specifically, given positive integers r and s, one can find out an integer $K(r,s)$, such that $H_1(r,s) \leq 2$ if $K \geq K(r,s)$. For $K < K(r,s)$, with the aid of a computer, one may check that whether $H_1(r,s) \leq 2$ holds. We take the following result as an example.

Theorem 3: Suppose that K, r and s are positive integers. If $s \leq r \leq 8$ and $r \leq K$, then $H_1(r,s) = \frac{L_1(r,s)}{L^*(r,s)} \leq 2$.

Proof: We divide the proof into two parts.

1) $(r,s) \neq (8,8)$. We take $(r,s) = (2,2)$ as an example. According to Lemma 4, for any positive integer $K \geq \frac{3rs(7r-s+1)}{8(r-s+1)} = 19.5$, $H_1(r,s) \leq 2$ holds. For any positive integer $K < 19.5$, we can obtain $H_1(r,s) \leq 2$ from Theorem 2. Similarly, we can prove $H_1(r,s) \leq 2$ holds for any other pairs (r,s).
2) $(r,s) = (8,8)$. According to Lemma 4, for any positive integer $K \geq \frac{3rs(7r-s+1)}{8(r-s+1)} = 1176$, $H_1(r,s) \leq 2$ holds. For any positive integer $K \leq 1000$, we can obtain $H_1(r,s) \leq 2$ from Theorem 2. For any positive integer $1000 < K < 1176$, with the aid of a computer, one can show that $H_1(r,s) \leq 2$ holds.

This completes the proof.

Remark 6: It is easy to see that for all the parameters satisfying the conditions in Theorem 3, the communication loads of Scheme 1 are slightly larger than those of Li-CDC scheme, while the number of output functions in Scheme 1 is much smaller than that of output functions in Li-CDC scheme. Here we only list some cases in Table 9, where Q_{Li} and Q_1 are the numbers of output functions in Li-CDC scheme and Scheme 1, respectively.

TABLE 9. The numbers of output functions in Li-CDC scheme and Scheme 1.

Number of Node K	Computation Load r	Replication Factor s	$Q_{Li} = \binom{K}{s}$	$Q_1 = \frac{K}{\gcd(K,s)}$
16	3	2	120	8
	5	4	1820	4
	8	6	8008	8
20	3	2	190	10
	5	4	4845	5
	8	6	38760	10

B. THE SECOND NEW SCHEME

Let $\mathbf{P}$ be a t-$(K, (\frac{K}{t})^{t-1}, (\frac{K}{t})^{t-2}, (\frac{K}{t})^t - (\frac{K}{t})^{t-1})$ PDA from [28] (the PDA in the third row of Table 5). From Theorem 1, for any positive integer $s \leq K$, one can obtain a CDC scheme, say Scheme 2, with K nodes, $N = (\frac{K}{t})^{t-1}$ files and $Q = \frac{K}{\gcd(K,s)}$ output functions, where s is corresponding to the number of nodes that compute each output function. Furthermore, the computation load is

$$r = \frac{KZ}{N} = \frac{K(\frac{K}{t})^{t-2}}{(\frac{K}{t})^{t-1}} = t,$$

and the communication load is

$$L_2(r,s) = \frac{sgS}{(g-1)KN} = \frac{st((\frac{K}{t})^t - (\frac{K}{t})^{t-1})}{(t-1)K(\frac{K}{t})^{t-1}} = \frac{s(K-t)}{(t-1)K} = \frac{s}{r-1}(1-\frac{r}{K}).$$

Similar to the communication load of Scheme 1, it is possible that $L_2 = \frac{s}{r-1}(1-\frac{r}{K}) > 1$. By using similar method, we could have $L_2(r,s) = \min\{\frac{s}{r-1}(1-\frac{r}{K}), 1\}$. Obviously, the communication load in Scheme 2 is slightly larger than the communication load in Scheme 1, i.e., $L_2 = \frac{s}{r-1}(1-\frac{r}{K}) > \frac{s}{r}(1-\frac{r}{K}) = L_1$. Hence, similar to Scheme 1, we can also prove the following results.

Theorem 4: Suppose that $K \geq 5$, r and s are positive integers satisfying that $K \geq s$ and $r \geq 2$ is a factor of K. Then $H_2(r,s) = \frac{L_2(r,s)}{L^*(r,s)} \leq 2.1$ holds if one of the following conditions is satisfied:

1) $K \leq 1000$;
2) $r \geq s+2$ and $K \geq \frac{(111r-15s-111)rs}{44r-40s-44}$;
3) $s+2 \leq r \leq 10$.

The proof of Theorem 4 is included in Appendix B.

Furthermore, we can show that the number of files in Scheme 2 is much smaller than that of files in Li-CDC scheme. To do this, we need the following lemma.

Lemma 5: ([28]) For fixed rational number $a \in (0,1)$, let $K \in \mathbb{N}^+$ such that $aK \in \mathbb{N}^+$, when $K \to \infty$,

$$\binom{K}{aK} \sim \frac{e^{K(a\ln\frac{1}{a}+(1-a)\ln\frac{1}{1-a})}}{\sqrt{2\pi K(a-a^2)}}.$$

From Lemma 1, the number of files in Li-CDC scheme is $\binom{K}{aK}$, where $aK = r$. So according to Lemma 5, the number of files in Li-CDC scheme is

$$\binom{K}{aK} \sim \frac{e^{K(a\ln\frac{1}{a}+(1-a)\ln\frac{1}{1-a})}}{\sqrt{2\pi K(a-a^2)}}$$

when $K \to \infty$. On the other hand, the number of files in Scheme 2 is $(\frac{K}{r})^{r-1}$, which is exponentially smaller in K than the number of files in Li-CDC scheme.

Remark 7: From Theorem 4, the communication load of Scheme 2 is slightly larger than that of Li-CDC scheme. However, the number $\frac{K}{\gcd(K,s)}$ of output functions in Scheme 2 is much smaller than the number $\binom{K}{s}$ of output functions in Li-CDC scheme. Furthermore, according to the above discussions, the number of files in Scheme 2 is exponentially smaller in K than the number of files in Li-CDC scheme.

C. THE THIRD NEW SCHEME

Let $\mathbf{P}$ be a $(K-t)$-$(K, (\frac{K}{t}-1)(\frac{K}{t})^{t-1}, (\frac{K}{t}-1)^2(\frac{K}{t})^{t-2}, (\frac{K}{t})^{t-1})$ PDA from [28] (the PDA in the forth row of Table 5).

From Theorem 1, for any positive integer $s \leq K$, one can obtain a CDC scheme, say Scheme 3, with K nodes, $N = (\frac{K}{t}-1)(\frac{K}{t})^{t-1}$ files and $Q = \frac{K}{\gcd(K,s)}$ output functions, where s is corresponding to the number of nodes that compute each reduce function. Furthermore, the computation load is

$$r = \frac{KZ}{N} = \frac{K(\frac{K}{t}-1)^2(\frac{K}{t})^{t-2}}{(\frac{K}{t}-1)(\frac{K}{t})^{t-1}} = K - t,$$

and the communication load is

$$\begin{aligned} L_3(r,s) &= \frac{sgS}{(g-1)KN} = \frac{s(K-t)(\frac{K}{t})^{t-1}}{(K-t-1)K(\frac{K}{t}-1)(\frac{K}{t})^{t-1}} \\ &= \frac{st}{(K-t-1)K} = \frac{s(K-r)}{(r-1)K} = \frac{s}{r-1}(1-\frac{r}{K}). \end{aligned}$$

We note that the communication load of Scheme 3 is equal to that of Scheme 2, i.e., $L_3 = \frac{s}{r-1}(1-\frac{r}{K}) = L_2$, which implies that $H_3(r,s) = \frac{L_3(r,s)}{L^*(r,s)} = \frac{L_2(r,s)}{L^*(r,s)} = H_2(r,s)$. So we can prove the following results by using the same method as the proof of Theorem 4.

Theorem 5: Suppose that $K \geq 5$, r and s are positive integers satisfying that $s \leq K$, $r \leq K-2$ and $K-r$ is a factor of K. Then $H_3(r,s) = \frac{L_3(r,s)}{L^*(r,s)} \leq 2.1$ holds if one of the following conditions is satisfied:

1) $K \leq 1000$;
2) $r \geq s+2$ and $K \geq \frac{(111r-15s-111)rs}{44r-40s-44}$.

Since the proof of Theorem 5 is the same as the proof of Theorem 4-1) and 2), we omit it here. The difference between Theorem 4 and Theorem 5 is the range of the computation load r. In Theorem 4, r is a factor of K, while $K-r$ is a factor of K in Theorem 5.

Remark 8: Similar to the discussions of Scheme 2, the communication load of Scheme 3 is slightly larger than that of Li-CDC scheme. However, the number of output functions in Scheme 3 is much smaller than the number of output functions in Li-CDC scheme, and the number of files in Scheme 3 is exponentially smaller in K than the number of files in Li-CDC scheme.

V. CONCLUSION

In this paper, we paid our attention to cascaded CDC schemes on homogeneous computing networks. We showed that, in many cases, $\frac{L}{L_{Li}} \leq 2.1$, where L and L_{Li} are the communication loads of our new scheme and the scheme derived by Li *et al.*, respectively. Most importantly, the number of output functions in all the new schemes are only a factor of the number of computing nodes, and the number of input files in our new schemes is much smaller than that of input files in CDC schemes derived by Li *et al.*

APPENDIX A
PROOF OF LEMMA 4

Recall that the positive integers K, r and s satisfy $r \geq s$ and $K \geq \frac{3rs(7r-s+1)}{8(r-s+1)}$.

Firstly, consider the communication $L^*(r,s)$ of Li-CDC scheme. Since

$$\begin{aligned} K &\geq \frac{3rs(7r-s+1)}{8(r-s+1)} = \frac{3rs}{8}(7+\frac{6s-6}{r-s+1}) \\ &\geq \frac{21}{8}rs > r+s, \end{aligned} \tag{11}$$

we have

$$\begin{aligned} L^*(r,s) &= \sum_{l=\max\{r+1,s\}}^{\min\{r+s,K\}} \frac{\binom{K-r}{l-r}\binom{r}{l-s}}{\binom{K}{s}} \frac{l-r}{l-1} \\ &= \frac{1}{\binom{K}{s}} \sum_{l=\max\{r+1,s\}}^{r+s} \binom{K-r}{l-r}\binom{r}{l-s}\frac{l-r}{l-1} \\ &\geq \frac{1}{\binom{K}{s}} \binom{K-r}{s}\binom{r}{r}\frac{s}{r+s-1} \\ &= \frac{s(K-r)(K-r-1)\cdots(K-r-s+1)}{(r+s-1)K(K-1)\cdots(K-s+1)}. \end{aligned}$$

Then

$$\begin{aligned} &H_1(r,s) \\ &= \frac{L_1(r,s)}{L^*(r,s)} \\ &\leq \frac{s(K-r)}{Kr} \frac{(r+s-1)K(K-1)\cdots(K-s+1)}{s(K-r)(K-r-1)\cdots(K-r-s+1)} \\ &= \frac{r+s-1}{r} \frac{(K-1)\cdots(K-(s-1))}{(K-(r+1))\cdots(K-(r+s-1))}. \end{aligned} \tag{12}$$

In order to evaluate the above value of $H_1(r,s)$, the following lemma is needed.

Lemma 6: Suppose that K, a_1, a_2, ..., a_n are positive integers with $K > a_i, 1 \leq i \leq n$, and $(K-a_1)(K-a_2)\cdots(K-a_n) = K^n - b_1K^{n-1} + b_2K^{n-2} + \ldots + (-1)^{n-1}b_{n-1}K + (-1)^n b_n$. For any $1 \leq h \leq n-1$,

1) $b_{h+1} \leq \frac{\sum_{1\leq i\leq n} a_i}{h+1} b_h$;
2) $b_{h+1}K^{n-h-1} \leq b_hK^{n-h}$ holds if $K \geq \frac{\sum_{1\leq i\leq n} a_i}{h+1}$.

Proof: Firstly, we will prove the first result. It is not difficult to know that $b_h = \sum_{1\leq i_1<i_2<\cdots<i_h\leq n} a_{i_1}a_{i_2}\cdots a_{i_h}$ and $b_{h+1} = \sum_{1\leq i_1<i_2<\cdots<i_{h+1}\leq n} a_{i_1}a_{i_2}\cdots a_{i_{h+1}}$. Then

$$\begin{aligned} &b_{h+1} \\ &= \sum_{1\leq i_1<i_2<\cdots<i_{h+1}\leq n} a_{i_1}a_{i_2}\cdots a_{i_{h+1}} \\ &= \frac{1}{h+1} \sum_{1\leq i_1<i_2<\cdots<i_{h+1}\leq n} (h+1)a_{i_1}a_{i_2}\cdots a_{i_{h+1}} \\ &= \frac{1}{h+1} \sum_{1\leq i_1<i_2<\cdots<i_h\leq n} a_{i_1}a_{i_2}\cdots a_{i_h} \sum_{i_j\in\{1,\ldots,n\}\setminus\{i_1,i_2,\ldots,i_h\}} a_{i_j} \\ &= \frac{1}{h+1} \sum_{1\leq i_1<i_2<\cdots<i_h\leq n} a_{i_1}a_{i_2}\cdots a_{i_h} \\ &\quad \times\Big(\sum_{i_j\in\{1,\ldots,n\}} a_{i_j} - \sum_{i_j\in\{i_1,i_2,\ldots,i_h\}} a_{i_j}\Big) \\ &\leq \frac{\sum_{1\leq i\leq n} a_i}{h+1} b_h. \end{aligned}$$

That is $b_{h+1} \leq \frac{\sum_{1\leq i\leq n} a_i}{h+1} b_h$.

If $K \geq \frac{\sum_{1\leq i\leq n} a_i}{h+1}$, together with the above result, we have

$$\begin{aligned} b_{h+1}K^{n-h-1} &\leq \frac{\sum_{1\leq i\leq n} a_i}{h+1} b_h K^{n-h-1} \\ &= (b_h K^{n-h})\frac{\sum_{1\leq i\leq n} a_i}{K(h+1)} \\ &\leq b_h K^{n-h}. \end{aligned}$$

This completes the proof.

By using Lemma 6,

$$\begin{aligned} &(K-1)\cdots(K-(s-1)) \\ &= K^{s-1} - \sum_{1\leq i_1\leq s-1} i_1 K^{s-2} + \sum_{1\leq i_1<i_2\leq s-1} i_1 i_2 K^{s-3} \\ &\quad + \ldots + (-1)^{s-1} \sum_{1\leq i_1<i_2<\ldots<i_{s-1}\leq s-1} i_1 i_2 \ldots i_{s-1} \\ &\leq K^{s-1} - \sum_{1\leq i_1\leq s-1} i_1 K^{s-2} + \sum_{1\leq i_1<i_2\leq s-1} i_1 i_2 K^{s-3} \\ &\leq K^{s-1} - \sum_{1\leq i\leq s-1} iK^{s-2} + \frac{\sum_{1\leq i\leq s-1} i}{2} \sum_{1\leq j\leq s-1} jK^{s-3} \\ &= K^{s-1} - \frac{s(s-1)}{2}K^{s-2} + \frac{s^2(s-1)^2}{8}K^{s-3}, \end{aligned} \tag{13}$$

where the second and the third from last formula come from Lemma 6-1) and Lemma 6-2), respectively. Similarly, by using Lemma 6-2),

$$\begin{aligned} &(K-(r+1))\cdots(K-(r+s-1)) \\ &\geq K^{s-1} - \sum_{r+1\leq i_1\leq r+s-1} i_1 K^{s-2} \\ &= K^{s-1} - \frac{(2r+s)(s-1)}{2}K^{s-2}. \end{aligned} \tag{14}$$

So, according to (12), (13), (14),

$$\begin{aligned} &H_1(r,s) \\ &\leq \frac{r+s-1}{r} \frac{(K-1)\cdots(K-(s-1))}{(K-(r+1))\cdots(K-(r+s-1))} \\ &\leq \frac{r+s-1}{r} \frac{K^{s-1} - \frac{s(s-1)}{2}K^{s-2} + \frac{s^2(s-1)^2}{8}K^{s-3}}{K^{s-1} - \frac{(2r+s)(s-1)}{2}K^{s-2}} \\ &= \frac{r+s-1}{r} \frac{8K^2 - 4s(s-1)K + s^2(s-1)^2}{8K^2 - 4(2r+s)(s-1)K} \\ &= \frac{r+s-1}{r}\left(1 + \frac{8r(s-1)K + s^2(s-1)^2}{8K^2 - 4(2r+s)(s-1)K}\right). \end{aligned} \tag{15}$$

Since $r \geq s$, we have

$$\begin{aligned} 8r(s-1)K &< 8rsK, \\ -4(2r+s)(s-1)K &> -4(2r+r)sK = -12rsK. \end{aligned}$$

Since $rs < K$ from (11),

$$s^2(s-1)^2 \leq r^2 s^2 < rsK.$$

Hence

$$\begin{aligned} H_1(r,s) &< \frac{r+s-1}{r}\left(1 + \frac{8rsK + rsK}{8K^2 - 12rsK}\right) \\ &= \frac{r+s-1}{r}\left(1 + \frac{9rs}{8K - 12rs}\right) \\ &\leq \frac{r+s-1}{r}\left(1 + \frac{9rs}{8\frac{3rs(7r-s+1)}{8(r-s+1)} - 12rs}\right) \\ &= 2, \end{aligned}$$

where the second from last formula holds since $K \geq \frac{3rs(7r-s+1)}{8(r-s+1)}$.

This completes the proof.

APPENDIX B
PROOF OF THEOREM 4

With the aid of a computer, one can show that $H_2(r,s) \leq 2.1$ holds for all the positive integers satisfying condition 1) of Theorem 4.

Similar to the processes obtaining (15), one can obtain that

$$H_2(r,s) \leq \frac{r+s-1}{r-1}\left(1 + \frac{8r(s-1)K + s^2(s-1)^2}{8K^2 - 4(2r+s)(s-1)K}\right).$$

Since $r \geq s+2$ and $K \geq \frac{(111r-15s-111)rs}{44r-40s-44}$, we have $rs < K$. Then

$$s^2(s-1)^2 \leq r^2 s^2 < rsK.$$

We also obtain that

$$\begin{aligned} 8r(s-1)K &< 8rsK, \\ -4(2r+s)(s-1)K &> -4(2r+r)sK = -12rsK. \end{aligned}$$

Hence

$$\begin{aligned} H_1(r,s) &< \frac{r+s-1}{r-1}\left(1 + \frac{8rsK + rsK}{8K^2 - 12rsK}\right) \\ &= \frac{r+s-1}{r-1}\left(1 + \frac{9rs}{8K - 12rs}\right) \\ &< \frac{r+s-1}{r-1}\left(1 + \frac{9rs}{8\frac{(111r-15s-111)rs}{44r-40s-44} - 12rs}\right) \\ &= 2.1. \end{aligned}$$

So $H_2(r,s) \leq 2.1$ holds for all the positive integers satisfying condition 2) of Theorem 4.

We now consider the condition 3) of Theorem 4. We take $(r,s) = (4,2)$ as an example. According to Theorem 4-2) for any positive integer $K \geq \frac{(111r-15s-111)rs}{44r-40s-44} = 46.62$, $H_2(r,s) \leq 2.1$ holds. For any positive integer $K \leq 46$, we can obtain $H_2(r,s) \leq 2.1$ from Theorem 4-1). Similarly, we can prove $H_2(r,s) \leq 2.1$ holds for any other pairs (r,s) satisfying condition 3) of Theorem 4.

This completes the proof.

REFERENCES

[1] *Amazon Elastic Compute Cloud (Amazon EC2)*. Accessed: May 12, 2017. [Online]. Available: https://aws.amazon.com/ec2/
[2] *Apache Hadoop*. Accessed: Feb. 8, 2018. [Online]. Available: http://hadoop.apache.org/

[3] F. Chen, L. Hu, P. Liu, and M. Feng, "A robust diffusion estimation algorithm for asynchronous networks in IoT," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 9103–9115, Sep. 2020.
[4] F. Chen and X. Shao, "Broken-motifs diffusion LMS algorithm for reducing communication load," *Signal Process.*, vol. 133, pp. 213–218, Apr. 2017.
[5] F. Chen, T. Shi, S. Duan, L. Wang, and J. Wu, "Diffusion least logarithmic absolute difference algorithm for distributed estimation," *Signal Process.*, vol. 142, pp. 423–430, Jan. 2018.
[6] M. Cheng, J. Jiang, Q. Wang, and Y. Yao, "A generalized grouping scheme in coded caching," *IEEE Trans. Commun.*, vol. 67, no. 5, pp. 3422–3430, May 2019.
[7] M. Cheng, J. Jiang, Q. Yan, and X. Tang, "Coded caching schemes for flexible memory sizes," *IEEE Trans. Commun.*, vol. 67, no. 6, pp. 4166–4176, Jun. 2019.
[8] M. Chowdhury, M. Zaharia, J. Ma, M. I. Jordan, and I. Stoica, "Managing data transfers in computer clusters with orchestra," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, pp. 98–109, Oct. 2011.
[9] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, Jan. 2008.
[10] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 770–778.
[11] M. Kiamari, C. Wang, and A. S. Avestimehr, "On heterogeneous coded distributed computing," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2017, pp. 1–7.
[12] K. Konstantinidis and A. Ramamoorthy, "Resolvable designs for speeding up distributed computing," 2019, *arXiv:1908.05666*. [Online]. Available: http://arxiv.org/abs/1908.05666
[13] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Speeding up distributed machine learning using codes," *IEEE Trans. Inf. Theory*, vol. 64, no. 3, pp. 1514–1529, Mar. 2018.
[14] K. Lee, C. Suh, and K. Ramchandran, "High-dimensional coded matrix multiplication," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2017, pp. 2418–2422.
[15] S. Li, M. A. Maddah-Ali, and A. S. Avestimehr, "A unified coding framework for distributed computing with straggling servers," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Dec. 2016, pp. 1–6.
[16] S. Li, M. A. Maddah-Ali, Q. Yu, and A. S. Avestimehr, "A fundamental tradeoff between computation and communication in distributed computing," *IEEE Trans. Inf. Theory*, vol. 64, no. 1, pp. 109–128, Jan. 2018.
[17] S. Li, Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "A scalable framework for wireless distributed computing," *IEEE/ACM Trans. Netw.*, vol. 25, no. 5, pp. 2643–2654, Oct. 2017.
[18] S. Li, Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Edge-facilitated wireless distributed computing," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Washington, DC, USA, Dec. 2016, pp. 1–7.
[19] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *IEEE Trans. Inf. Theory*, vol. 60, no. 5, pp. 2856–2867, May 2014.
[20] H. Park, K. Lee, J.-Y. Sohn, C. Suh, and J. Moon, "Hierarchical coding for distributed computing," 2018, *arXiv:1801.04686*. [Online]. Available: http://arxiv.org/abs/1801.04686
[21] S. Prakash, A. Reisizadeh, R. Pedarsani, and S. Avestimehr, "Coded computing for distributed graph analytics," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2018, pp. 1–5.
[22] C. Shangguan, Y. Zhang, and G. Ge, "Centralized coded caching schemes: A hypergraph theoretical approach," *IEEE Trans. Inf. Theory*, vol. 64, no. 8, pp. 5755–5766, Aug. 2018.
[23] N. Shakya, F. Li, and J. Chen, "On distributed computing with heterogeneous communication constraints," in *Proc. 52nd Asilomar Conf. Signals, Syst., Comput.*, Oct. 2018, pp. 1795–1799.
[24] N. Woolsey, R.-R. Chen, and M. Ji, "A new combinatorial design of coded distributed computing," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2018, pp. 726–730.
[25] N. Woolsey, R.-R. Chen, and M. Ji, "Cascaded coded distributed computing on heterogeneous networks," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2019, pp. 2644–2648.
[26] N. Woolsey, R.-R. Chen, and M. Ji, "Coded distributed computing with heterogeneous function assignments," 2019, *arXiv:1902.10738*. [Online]. Available: http://arxiv.org/abs/1902.10738
[27] F. Xu and M. Tao, "Heterogeneous coded distributed computing: Joint design of file allocation and function assignment," 2019, *arXiv:1908.06715*. [Online]. Available: http://arxiv.org/abs/1908.06715
[28] Q. Yan, M. Cheng, X. Tang, and Q. Chen, "On the placement delivery array design for centralized coded caching scheme," *IEEE Trans. Inf. Theory*, vol. 63, no. 9, pp. 5821–5833, Sep. 2017.
[29] Q. Yan, M. Wigger, S. Yang, and X. Tang, "A fundamental storage-communication tradeoff in distributed computing with straggling nodes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2019, pp. 2803–2807.
[30] Q. Yan, S. Yang, and M. Wigger, "Storage, computation, and communication: A fundamental tradeoff in distributed computing," 2018, *arXiv:1806.07565*. [Online]. Available: http://arxiv.org/abs/1806.07565
[31] Q. Yan, S. Yang, and M. Wigger, "Storage, computation, and communication: A fundamental tradeoff in distributed computing," in *Proc. IEEE Inf. Theory Workshop (ITW)*, Guangzhou, China, Nov. 2018, pp. 1–5.
[32] Q. Yu, M. Maddah-Ali, and S. Avestimehr, "Polynomial codes: An optimal design for high-dimensional coded matrix multiplication," in *Proc. 31st Annu. Conf. Neural Inf. Process. Syst. (NIPS)*, Long Beach, CA, USA, May 2017, pp. 4403–4413.
[33] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," *HotCloud*, vol. 10, no. 10, p. 95, Jun. 2010.

JING JIANG received the Ph.D. degree from the Department of Social Systems and Management, Graduate School of Systems and Information Engineering, University of Tsukuba, Tsukuba, Japan, in 2015. He is currently an Associate Professor with the School of Computer Science and Information Technology, Guangxi Normal University, Guilin, China. His research interests include combinatorial design theory, coding theory, and their interactions.

LINGXIAO QU received the B.S. degree from the Department of Computer Science, School of Control and Computer Engineering, North China Electric Power University, Baoding, China, in 2017. She is currently pursuing the M.S. degree with the School of Computer Science and Information Technology, Guangxi Normal University, Guilin, China. Her research interests include distributed computing and combinatorial design theory.