Asymptotically Optimal Cascaded Coded Distributed Computing via Combinatorial Designs

Minquan Cheng, Youlong Wu, and Xianxian Li

Abstract

Coded distributed computing (CDC) introduced by Li *et al.* can greatly reduce the communication load for MapReduce computing systems. In the general cascaded CDC with K workers, N input files and Q Reduce functions, each input file will be mapped by r workers and each Reduce function will be computed by s workers such that coding techniques can be applied to achieve the maximum multicast gain. The main drawback of most existing CDC schemes is that they require the original data to be split into a large number of input files that grows exponentially with K, which can significantly increase the coding complexity and degrade system performance. In this paper, we first use a classic combinatorial structure t-design, for any integer $t \geq 2$, to develop a low-complexity and asymptotically optimal CDC with $r = s$. The main advantages of our scheme via t-design are two-fold: 1) having much smaller N and Q than the existing schemes under the same parameters K, r and s; and 2) achieving smaller communication loads compared with the state-of-the-art schemes. Remarkably, unlike the previous schemes that realize on large operation fields, our scheme operates on the minimum binary field $\mathbb{F}_2$. Furthermore, we show that our construction method can incorporate the other combinatorial structures that have a similar property to t-design. For instance, we use t-GDD to obtain another asymptotically optimal CDC scheme over $\mathbb{F}_2$ that has different parameters from t-design. Finally, we show that our construction method can also be used to construct CDC schemes with $r \neq s$ that have small file number and Reduce function number.

Index Terms

I. Introduction

Distributed computing systems have been widely applied to execute large-scale computing tasks, since they can greatly speed up task execution by letting distributed computing nodes execute computation jobs in parallel and exploiting distributed computing and storage resources. However, when exchanging a massive amount of data among the computing nodes, distributed computing systems would suffer a severe communication bottleneck due to the limited communication resource and high transmitted traffic load. For instance, in the TeraSort [1] and SelfJoin [2] applications running in Amazon EC 2 cluster, the time cost of data exchange occupies $65\% \sim 70\%$ of the overall job execution time [3].

Coded distributed computing (CDC), proposed by Li *et al.* in [4], is considered as a prominent approach to reduce the communication load for distributed computing systems such as MapReduce [5] and Spark [6], by introducing repetitive computations on the input data to create coding multicast opportunities. Consider a general (K, r, s, N, Q) MapReduce system with K computing workers, N input data files with equal size and Q output values each of which is the function of the N input data files. It consists of "Map", "Shuffle" and "Reduce" phases. In the map phase, every input file will be exclusively mapped by a distinct r-subset of workers to Q intermediate values (IVs). In the Shuffle phase, the workers generate coded symbols from the local IVs, and multicast them to other workers such that all desired IVs can be recovered by the desired nodes. In the Reduce phase, each output value will be exclusively reduced by a distinct s-subset of workers, based on the locally computed and the recovered IVs. When $s = 1$, i.e., each output value is computed exactly once, the CDC scheme is similar to the coded caching scheme for the D2D network [4], [7]. When $s \geq 1$, it is called cascaded CDC as it can support multi-round computation where the output values of the previous round serve as the input of the next round.

The CDC approach has attracted enormous attention, and many works have been conducted focusing on the scalability and optimality of CDC in various settings. For instance, the linear dependency of IVs and the properties of map functions were exploited to improve the computation-communication trade-off [8]. Saurav *et al.* described the MapReduce computations on graphs and and leveraged the graph structure to create coded multicast opprotunies [9]. The resource allocation problem of CDC has been investigated in [10], [11], in which optimal CDC schemes to minimize the total execution time were proposed. The heterogeneous CDC with different storages and computation loads among nodes were considered in [12]–[17]. For more works about CDC, please see a survey in [18].

M. Cheng and X. Li are with Guangxi Key Lab of Multi-source Information Mining & Security, Guangxi Normal University, Guilin 541004, China (e-mail: chengqinshi@hotmail.com, lixx@gxnu.edu.cn).

Y. Wu is with the School of Information Science and Technology, ShanghaiTech University, Shanghai 201210, China (e-mail: wuyl1@shanghaitech.edu.cn).

A. *Research Motivation*

For the cascaded case where each Reduce function is computed multiple times, the main idea of most existing CDC schemes is to use some combinatorial structures to design the data placement and Reduce function assignment such that the communication load is as small as possible by means of the Maximum separable (MDS) codes or randomly linear combination method. For instance, the first and classic cascaded CDC scheme [4] uses all the r-subset and s-subset of $[K]$ to design the data placement and Reduce function assignment respectively where $1 \leq r, s \leq K$. Then in the shuffle phase, for any subset $\mathcal{S} \subseteq [K]$ with a cardinality $|\mathcal{S}| \in \{\max\{r+1, s\}, \ldots, \min\{r+s, K\}\}$, each node $k \in \mathcal{S}$ will use MDS codes to generate and deliver linear combinations of IV segments based on its locally computed IVs. Each linear combination is decodable by the exclusively desired nodes in $\mathcal{S}\backslash\{k\}$, i.e., achieving a multicast gain $|\mathcal{S}| - 1$. The authors in [4] showed that their scheme achieves the minimum communication load, but at the cost of requiring exponentially large numbers of both input files and Reduce functions in terms of $\binom{K}{r}$ and $\binom{K}{s}$, respectively. This leads to unexpected performance loss in practical implementations when K is relatively large. For example, in the CodedTeraSort experiment sorting 12 GB data with $K = 16$ workers, $r = 6$, and 100 Mbps network speed, each worker needs to generate all file indices and initialize $\binom{K}{r+1}$ multicast groups to transfer all IVs to the intended workers. The corresponding time cost will dominate the overall execution time [4].

To reduce the required numbers of both input files and Reduce functions, a hypercube scheme, where the data placement and Reduce function assignment are generated by the hypercube structure was proposed in [19] for the case $r = s$. In their scheme, the communications take place in multiple rounds with multicast gains equal to $r, \ldots, \min\{2r-1, K-1\}$, in each of which one worker generates linear combinations based on the local IVs and broadcasts them to the other workers. The authors also showed that the communication load is asymptotically optimal. However, their scheme still requires exponentially large numbers of both input files and Reduce functions in terms of $(\frac{K}{r})^{r-1}$. The authors in [20] used a symmetric balanced incomplete block design (SBIBD) to generate the data placement and Reduce function assignment to obtain an asymptotically optimal scheme with $K = N = Q$. In their scheme, each coded symbol carries contents desired by $r-1$ workers, i.e., the multicast gain is $r-1$.

The authors [21] used the placement delivery array (PDA) to generate cascaded CDC schemes. These schemes have an operation field $\mathbb{F}_2$ and are one-shot delivery. It is worth noting that the PDA was originally proposed to reduce the subpacketization of coded caching problem when each worker requests distinct contents, not able to characterize the worker demands if some workers request the same content [22]–[34]. This leads to the PDA-based CDC schemes [21] failing to exploit the common IVs desired by multiple workers, and thus incurring redundant communication load. We list all the above results in Table I.

TABLE I: Existing cascaded CDC schemes

Parameters	Number of Nodes K	Computation Load r	Replication Factor s	Number of Files N	Number of Reduce Functions Q	Communication Load $L_{\text{comm.}}$	Operation Field $\mathbb{F}_2$
[4]: $K, r, s \in \mathbb{N}^+$, $1 \leq r, s \leq K$	K	r	s	$\binom{K}{r}$	$\binom{K}{s}$	$\sum\limits_{l=\max\{r+1,s\}}^{\min\{r+s,K\}} \left(\frac{l-r}{l-1} \cdot \frac{\binom{K-r}{K-l}\binom{r}{l-s}}{\binom{K}{s}} \right)$	No
[19]: $K, r \in \mathbb{N}^+$, K is divisible by r	K	r	r	$(\frac{K}{r})^{r-1}$	$(\frac{K}{r})^{r-1}$	$\frac{1}{2} - \frac{1}{2} \cdot \left(\frac{r}{K}\right)^r + \left(1 - \frac{r}{K}\right)^r \cdot \frac{1}{4r-2}$	No
[20]: (K, r, λ) SBIBD, $K, r, \lambda \in \mathbb{N}^+$, $2 \leq \lambda \leq r$	K	r	r	K	K	$\frac{r}{r-1} \cdot \frac{K-r}{K}$	No
[21]: $K, r, s \in \mathbb{N}^+$, $1 \leq r, s \leq K$	K	r	s	$(\frac{K}{r})^{r-1}$	$\frac{K}{\text{gcd}(K,s)}$	$\frac{s}{r-1}(1 - \frac{r}{K})$	Yes

B. *Contribution and Organization*

In this paper, we first show that for the cascaded case with $s = r$, by delicately designing the data placement and Reduce function assignment using a combinatorial structure t-design [35], for any positive integer $t \geq 2$, we can construct a new class of asymptotically optimal schemes, i.e., the scheme for Theorem 1 in Table II. For the special 2-design, our scheme achieves a multicast gain $r + s - 2 = 2r - 2$ that is just one less than the maximum multicast gain of the scheme proposed in [4] (See Remark 5). Comparing Table I and Table II, our scheme has the following advantages.

- Compared with the scheme in [4], [19], our scheme has much smaller file number N and Reduce function number Q. More surprisingly, in addition to smaller N, Q, our scheme achieves a less communication load than that of [19] when $\frac{K}{r} > 1 + (2r+1)^{\frac{1}{r}}$, and than that of [4] for some case (see Example 3).
- Compared with the scheme in [20], our scheme has a smaller communication load under the same parameters K, $r = s$, file number N and Reduce function number Q when $r \leq K/2$. Furthermore, SBIBD is the special structure of the 2-design which will be introduced in Section II-B. This implies that we can obtain new CDC schemes with more flexible parameters compared with the scheme in [20].

- Compared with the scheme in [21], our scheme has much smaller file number N and smaller communication load under the same parameters K, $r = s$ when $r \leq K/2$.

In fact, by using other combinatorial structures having a similar property to the second property of t-design (see Definition 2), we can also construct other low-complexity and communication-efficient CDC schemes. For instance, we apply the combinatorial structure t-GDD to design another CDC scheme that is asymptotically optimal over $\mathbb{F}_2$and has different parameters from the t-design scheme for Theorem 1 (see Remark 1 in Subsection II-B). The results of t-GDD scheme for Theorem 2 are listed in Table II.

Finally, we show that our constructing method can also be extended to the case $r \neq s$, which is the scheme for Theorem 3 in Table II. By comparison with the scheme in [21], our third scheme has a significant advantage on the file number.

TABLE II: New schemes where t, N, M, $\lambda \in \mathbb{N}^+$, $1 \leq t \leq M \leq N$ and $M \leq M$

Parameters	Number of Nodes K	Computation Load r	Replication Factor s	Number of Files N	Number of Reduce Functions Q	Communication Load $L_{\text{comm.}}$	Operation Field $\mathbb{F}_2$
Theorem 1	$\frac{\lambda\binom{N}{t}}{\binom{M}{t}}$	$\frac{KM}{N}$	$\frac{KM}{N}$	N	N	$\frac{N-1}{2N} < \frac{1}{2}$	Yes
Theorem 2	$\frac{\lambda\binom{m}{t}q^t}{\binom{M}{t}}$	$\frac{KM}{mq}$	$\frac{\lambda\binom{m-1}{t-1}q^{t-1}}{\binom{M-1}{t-1}}$	mq	mq	$\frac{1}{2} + \frac{q-2}{mq}$	Yes
Theorem 3	$\frac{\lambda\binom{N}{t}}{\binom{M}{t}}$	$\frac{KM}{N}$	$\frac{\lambda(N-t+1)}{M-t+1}$	N	$\binom{N}{t-1}$	$\frac{N-t+1}{Nt} < \frac{1}{t}$	Yes

The rest of this paper is organized as follows. Section II describes the system model and some useful combinatorial structures. Section III introduces our main results including two new schemes and their performance analyses. Section IV provides the detailed construction of the two proposed cascaded CDC schemes. Section V provides an extension scheme for the case $r \neq s$. Finally, we conclude the paper in Section VI.

C. Notations

In this paper, we use the following notations unless otherwise stated.

- Bold capital letters, bold lowercase letters, and calligraphic fonts will be used to denote arrays, vectors, and sets, respectively.
- We assume that all the sets are in increasing order; for a set $\mathcal{V}$, we let $\mathcal{V}(j)$ represent the j-th smallest element of $\mathcal{V}$ and let $\mathcal{V}(\mathcal{J}) = \{\mathcal{V}(j) | j \in \mathcal{J}\}$.
- $|\cdot|$ is used to represent the cardinality of a set or the length of a vector.
- For any positive integers a, b, t with $a < b$ and $t \leq b$, and any nonnegative set $\mathcal{V}$, let $[a, b] = \{a, a+1, \ldots, b\}$, especially $[1, b]$ be shorten by $[b]$, and $\binom{[b]}{t} = \{\mathcal{V} \mid \mathcal{V} \subseteq [b], |\mathcal{V}| = t\}$, i.e., $\binom{[b]}{t}$ is the collection of all t-sized subsets of $[b]$. We use $a|b$ to denote that b is divisible by a.

II. Premilaries

In this section, we formulate the cascaded coded distributed computing problem and introduce some combinatorial structures that will be useful to our scheme design.

A. Cascaded Coded Distributed Computing System

In a coded distributed computing system, there are K distributed computing workers which will compute Q Reduce functions by taking advantage of N input files each of equal size. Denote the worker set, N files, and Q functions by $\mathcal{K} = [K]$, $\mathcal{W} = \{w_1, w_2, \ldots, w_N\}$, and $\mathcal{Q} = \{\phi_1, \phi_2, \ldots, \phi_Q\}$, respectively. For each function ϕ_q where $q \in [Q]$, let $\mathcal{A}_q$ represent the worker set each of which is arranged to compute the Reduce function

$$\begin{aligned} u_q &\triangleq \phi_q(w_1, w_2, \ldots, w_N) \\ &\triangleq h_q(g_{q,1}(w_1), g_{q,2}(w_2), \ldots, g_{q,N}(w_N)) \in \mathbb{F}_{2^E}, \end{aligned}$$

for some integer E where for any $q \in [Q]$ and $n \in [N]$, $g_{q,n}(\cdot)$ is called Map function and $h_q(\cdot)$ is called Reduce function. The parameter $v_{q,n} \triangleq g_{q,n}(w_n)$ where $q \in [Q]$ and $n \in [N]$ is called intermediate value (IV). We assume that each IV has T bits, for some positive T. In order to support multiple-round computing where the reduced results of the previous round are the inputs of the next round Map operation, each Reduce function is assumed to be computed by $s \in [K]$ workers. As illustrated in Fig. 1, a cascaded coded distributed computing consists of the following three phases.

- **Map Phase.** Each worker $k \in \mathcal{K}$ first stores M files, denoted by $\mathcal{Z}_k$. For each file w_n, let $\mathcal{D}_n$ represent the worker set each of which stores file w_n. Then the files stored by worker k can be written as follows.

$$\mathcal{Z}_k = \{w_n \mid n \in [N], k \in \mathcal{D}_n\}. \tag{1}$$

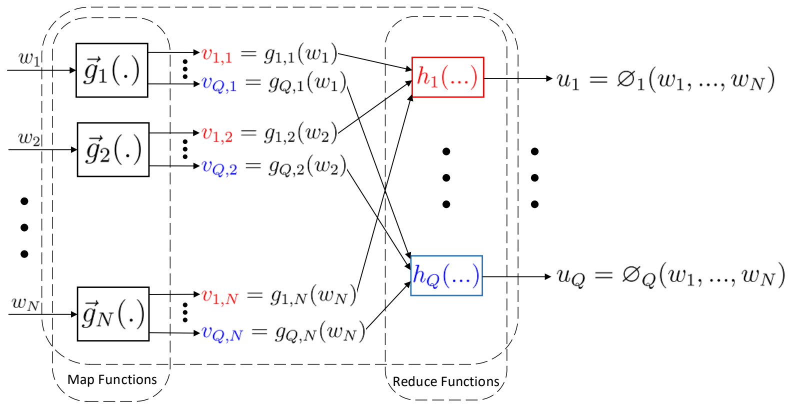

Fig. 1: Illustration of a two-stage distributed computing framework. The overall computation is decomposed into computing a set of Map and Reduce functions.

Using the stored files in (1) and Map functions $\{g_{q,n}(\cdot)\}$, worker k could compute the following IVs

$$\mathcal{I}_k = \{v_{q,n} = g_{q,n}(w_n) \mid q \in [Q], n \in [N], k \in \mathcal{D}_n\}.$$

- **Shuffle Phase.** According to the arranged computing tasks, i.e., worker $k \in \mathcal{K}$ is arranged to compute the following Reduce functions

$$\mathcal{Q}_k = \{\phi_q(w_1, w_2, \dots, w_N) \mid q \in [Q], k \in \mathcal{A}_q\}. \tag{2}$$

Since each worker can not store all the files, all the workers should exchange their locally computed IVs to ensure that each worker could derive the IVs which are not locally computed by itself. Assume that worker $k \in \mathcal{K}$ sends a coded message X_k with length l_k generated by its locally computed IVs to the other workers.
- **Reduce Phase.** By receiving the coded signals $\mathcal{X} = \{X_1, X_2, \dots, X_K\}$ and its locally computed IVs in $\mathcal{I}_k$, worker $k \in \mathcal{K}$ can compute each Reduce function in $\mathcal{Q}_k$.

Using the same definitions in [4], we also define the computation load as

$$r \triangleq \frac{\sum_{k=1}^{K} |\mathcal{Z}_k|}{N}, \tag{3}$$

and the communication load as

$$L_{\text{comm.}} \triangleq \frac{\sum_{k=1}^{K} |X_k|}{QNT} = \frac{\sum_{k=1}^{K} l_k}{QNT}. \tag{4}$$

That is, r is the average number of workers that map each file and $L_{\text{comm.}}$ is the ratio of the amount of transmitted data to QNT. When $N/\binom{K}{r} \in \mathbb{N}, Q/\binom{K}{s} \in \mathbb{N}$ (or $N \to \infty, Q \to \infty$), Li *et al.* established the optimal trade-off between the computation load and the communication load [4], which is given as follows.

Lemma 1 ([4]). For any positive integers K, r and s with r, $s \le K$, there exists a cascaded CDC scheme achieving the optimal communication load

$$L^*_{\text{comm.}}(r, s) = \sum_{l=\max\{r+1,s\}}^{\min\{r+s,K\}} \left(\frac{\binom{K-r}{K-l}\binom{r}{l-s}}{\binom{K}{s}} \cdot \frac{l-r}{l-1} \right), \tag{5}$$

where r is the computation load and s is the number of workers that calculate each function. □

Despite the optimality of (5), the requirement $N/\binom{K}{r} \in \mathbb{N}$ could lead to unexpected performance loss when N is sufficiently large. For example, in the CodedTeraSort experiment sorting 12 GB data with $K = 16$ workers and 100 Mbps network speed, each worker needs to generate all file indices and initialize $\binom{K}{r+1}$ multicast groups to transfer all IVs to the intended workers. The corresponding code generation time, namely code generation time, will increase exponentially with computation load r as $\binom{K}{r+1}$, and dominate the overall execution time when $r \ge 6$. Besides, the condition $Q/\binom{K}{s} \in \mathbb{N}$ requires the number of Reduce functions Q exponentially increasing with s, which may not hold for some computing tasks with small Q. To address the issues above, we aim to design low-complexity and communication-efficient schemes that reduce the communication load $L_{\text{comm.}}$ for any given r, while keeping small values of N and Q.

B. Combinatorial Design Structures

Definition 1 ([35], Design)**.** A design is a pair $(\mathcal{X},\mathfrak{B})$ such that the following properties are satisfied:

- $\mathcal{X}$ is a set of elements called points, and
- $\mathfrak{B}$ is a collection of nonempty subsets of $\mathcal{X}$ called blocks.

$\square$

A design is called r-regular if each point occurs in exactly r blocks. A r-regular design containing N points and K blocks each of which has size M is denoted by r- regular (N, M, K) design. Clearly, we have $KM = rN$. In addition, a design is called η-cross if the intersection of any two different blocks has exactly η points.

Definition 2. ([35], t-design) Let N, K, M and t and λ be five positive integers. A t-(N, M, K, r, λ) design is a design $(\mathcal{X},\mathfrak{B})$ where $\mathcal{X}$ has N points and $\mathfrak{B}$ has K blocks that satisfy

- $|\mathcal{B}| = M$ for any $\mathcal{B} \in \mathfrak{B}$;
- every t-subset of $\mathcal{X}$ is contained in exactly λ blocks;
- every point occurs exactly in r blocks.

$\square$

From Definition 2, we can obtain that the number of blocks and the occurrence number of each point in blocks of $\mathfrak{B}$ are

$$K = \frac{\lambda\binom{N}{t}}{\binom{M}{t}}, \qquad r = \frac{KM}{N} = \frac{\lambda\binom{N-1}{t-1}}{\binom{M-1}{t-1}}, \tag{6}$$

respectively. So any t-design is always regular and a t-(N, M, K, r, λ) design is also shorten as t-(N, M, λ) design in this paper. A t-(N, M, λ)-design is also a t'-$(N, M, \lambda_{t'})$ where $t' \leq t$ and

$$\lambda_{t'} = \frac{\lambda\binom{N-t'}{t-t'}}{\binom{M-t'}{t-t'}}. \tag{7}$$

Definition 3 (Dual design)**.** For any design $(\mathcal{X},\mathfrak{B})$, we regard the blocks $\mathfrak{B}$ as points and $\mathcal{X}$ as block set where $\mathcal{B} \in \mathfrak{B}$ is contained by $x \in \mathcal{X}$ if and only if $x \in \mathcal{B}$, then the obtained design $(\mathfrak{B},\mathcal{X})$ is called the dual design of $(\mathcal{X},\mathfrak{B})$. $\square$

Clearly, a design is a dual design of its dual design. In addition, it is not difficult to obtain the following result.

Lemma 2 (Regular and cross design via t-design)**.** The dual design of a t-(N, M, K, r, λ) design is a design where

- there are exactly K points each of which occurs in exactly M blocks;
- there are N blocks each of which has size $r = \lambda\binom{N-1}{t-1}/\binom{M-1}{t-1}$;
- any two distinct blocks intersect in exactly $\lambda_2 = \lambda\binom{N-2}{t-2}/\binom{M-2}{t-2}$ points.

$\square$

So the dual design of any t-(N, M, λ) design is M-regular and λ_2-cross. The 2-(N, M, λ) design is always called (N, M, λ) balanced incomplete block design (in short BIBD). From (6) we can obtain that the number of blocks and the occurrence number of each point are

$$K = \frac{\lambda N(N-1)}{M(M-1)}, \qquad r = \frac{KM}{N} = \frac{\lambda N(N-1)}{M(M-1)} \cdot M \cdot \frac{1}{N} = \frac{\lambda(N-1)}{M-1} \tag{8}$$

respectively. A BIBD in which $K = N$ (or, equivalently, $r = M$ or $\lambda(N-1) = M(M-1)$) is called a symmetric BIBD (SBIBD). It is worth noting that any SBIBD is regular and cross by the following well-known result.

Lemma 3 ([36])**.** Suppose that $(\mathcal{X},\mathfrak{B})$ is a (N, M, λ) SBIBD. Then $|\mathcal{B} \cap \mathcal{B}'| = \lambda$ holds for all distinct $\mathcal{B}$, $\mathcal{B}' \in \mathfrak{B}$. $\square$

Example 1. When $N = 7$ and $M = 3$, let us see the design $(\mathcal{X},\mathfrak{B})$ where $\mathcal{X} = \{1, 2, 3, 4, 5, 6, 7\}$ and

$$\mathfrak{B} = \{\mathcal{B}_1 = \{1, 2, 4\}, \mathcal{B}_2 = \{2, 3, 5\}, \mathcal{B}_3 = \{3, 4, 6\}, \mathcal{B}_4 = \{4, 5, 7\}, \mathcal{B}_5 = \{5, 6, 1\}, \mathcal{B}_6 = \{6, 7, 2\}, \mathcal{B}_7 = \{7, 1, 3\}\}. \tag{9}$$

By Definition 3 we have its dual design $(\mathfrak{B},\mathcal{X})$ where

$$\begin{aligned} \mathcal{X} = \{&1 = \{\mathcal{B}_1, \mathcal{B}_5, \mathcal{B}_7\},\ 2 = \{\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_6\},\ 3 = \{\mathcal{B}_2, \mathcal{B}_3, \mathcal{B}_7\},\ 4 = \{\mathcal{B}_1, \mathcal{B}_3, \mathcal{B}_4\},\ 5 = \{\mathcal{B}_2, \mathcal{B}_4, \mathcal{B}_5\},\\ &6 = \{\mathcal{B}_3, \mathcal{B}_5, \mathcal{B}_6\},\ 7 = \{\mathcal{B}_4, \mathcal{B}_6, \mathcal{B}_7\}\}. \end{aligned} \tag{10}$$

We can easily to verify that the designs $(\mathcal{X},\mathfrak{B})$ and $(\mathfrak{B},\mathcal{X})$ are $(7, 3, 1)$ SBIBDs. So the cardinality of any two blocks in $\mathfrak{B}$ of the design $(\mathcal{X},\mathfrak{B})$ (and in $\mathcal{X}$ of the dual design $(\mathfrak{B},\mathcal{X})$) is 1. $\square$

Recently, P. Keevash in [37] and S. Glock et al., in [38] respectively proved the existence conjecture for t-design, i.e., the following result.

Lemma 4 (The existence conjecture for t-design [37], [38])**.** Given t, M and λ, there exists an integer $N_0(t,M,\lambda)$ such that for any $N > N_0(t,M,\lambda)$, a t-(N,M,λ) design exists if and only if for any $0 \leq i \leq t-1$, the following condition holds

$$\lambda\binom{N-i}{t-i} \equiv 0 \pmod{\binom{M-i}{t-i}}.$$

□

Next, we introduce another special design, i.e., group divisible design (GDD), which is also useful in our proposed scheme.

Definition 4. ([35], t-GDD) Let M, t, q and m be positive integers with $t \leq M \leq m$. A (m,q,M,λ) *group divisible t-design* (t-(m,q,M,λ) GDD) is a triple $(\mathcal{X},\mathfrak{G},\mathfrak{B})$ where

- $\mathcal{X}$ is a set of mq points,
- $\mathfrak{G} = \{\mathcal{G}_1,\mathcal{G}_2,\ldots,\mathcal{G}_m\}$ is a partition of $\mathcal{X}$ into m subsets each of which has size q (called groups),
- $\mathfrak{B}$ is a family of M-blocks of $\mathcal{X}$ such that every block intersects every group at most one point, and every t-subset of points from t distinct groups belongs to exactly λ blocks.

□

We can obtain the number of blocks and the occurrence number of each point in $\mathfrak{B}$ by Definition 4 as

$$K = \frac{\lambda\binom{m}{t}q^t}{\binom{M}{t}}, \qquad r = \frac{KM}{N} = \frac{KM}{mq} = \frac{\lambda\binom{m-1}{t-1}q^{t-1}}{\binom{M-1}{t-1}}, \tag{11}$$

respectively. By the above introductions of t-design and t-GDD, we can obtain the following observations.

Remark 1.
- By Definition 2 and Definition 4 we can see that the third property of t-GDD is similar to the second property of t-design, i.e., every t-subset of points (for a special condition in t-GDD) occurs exactly in λ blocks.
- From (6) and (11), we can not obtain the same parameters K and r for the same point set, block size M, and the parameters t and λ.

Similar to Lemma 2, the following result can be obtained.

Lemma 5 (Regular and almost cross design via t-GDD)**.** The dual design of a t-(m,q,M,λ) GDD is a design where

- there are exactly K points each of which occurs in exactly M blocks;
- there are mq blocks each of which has size $r = \lambda\binom{m-1}{t-1}q^{t-1}/\binom{M-1}{t-1}$;
- there are $m\binom{q}{2}$ pairs of distinct blocks whose intersection is an empty set, i.e., the two different points in the same group of the GDD;
- there are $\binom{mq}{2} - m\binom{q}{2}$ pairs of distinct blocks whose intersection has exactly $\lambda_2 = \lambda\binom{m-2}{t-2}q^{t-2}/\binom{M-2}{t-2}$ points, i.e., the number of pairs containing the points included in the block of the GDD.

□

Example 2. When $m = M = 3$ and $q = 2$, let $\mathcal{X} = \{1,2,3,4,5,6\}$, groups

$$\mathfrak{G} = \{\mathcal{G}_1 = \{1,2\},\ \mathcal{G}_2 = \{3,4\},\ \mathcal{G}_3 = \{5,6\}\}$$

and blocks

$$\mathfrak{B} = \{\mathcal{B}_1 = \{1,3,5\}, \mathcal{B}_2 = \{1,4,6\}, \mathcal{B}_3 = \{2,4,5\}, \mathcal{B}_4 = \{2,3,6\}\}.$$

It is easy to check that the above design $(\mathcal{X},\mathfrak{G},\mathfrak{B})$ is a 2-$(3,2,3,1)$ GDD by Definition 4. By Definition 3 we have its dual design $(\mathfrak{B},\mathcal{X})$ where

$$\mathcal{X} = \{1 = \{\mathcal{B}_1,\mathcal{B}_2\},\ 2 = \{\mathcal{B}_3,\mathcal{B}_4\},\ 3 = \{\mathcal{B}_1,\mathcal{B}_4\},\ 4 = \{\mathcal{B}_2,\mathcal{B}_3\},\ 5 = \{\mathcal{B}_1,\mathcal{B}_3\},\ 6 = \{\mathcal{B}_2,\mathcal{B}_4\}\}. \tag{12}$$

We can verify that the intersection of any two blocks from block sets $\mathcal{X}_1 = \{1,2\}$, $\mathcal{X}_2 = \{3,4\}$ and $\mathcal{X}_3 = \{5,6\}$ is an empty set, and the intersection of any two blocks from two of $\mathcal{X}_1$, $\mathcal{X}_2$ and $\mathcal{X}_3$ contains exactly one point. For instance the intersection of blocks 1 and 2 is an empty set, and the intersection of blocks 1 and 4 is $\{\mathcal{B}_2\}$. □

There are many constructions and existences of the t-designs and t-GDDs, especially on the results of BIBDs. For the detailed constructions, please see [35, Section II and IV].

III. Main Results

In this section, we will first present the results of our two schemes based on t-design and t-GDD, respectively for the case $r = s$, and then show that our new schemes are asymptotically optimal. Finally, we compare our schemes with that of the state-of-art schemes, and show that ours have much smaller file number and Reduce function number than that of the schemes in [4], [19], [21] and have smaller communication loads than that of the schemes in [20], [21].

Given a t-design $(\mathcal{X}, \mathfrak{B})$, by taking the block set $\mathfrak{B}$ as worker set and the point set $\mathcal{X}$ to generate the data placement and Reduce function assignment, and using the second property of t-design to generate the delivery strategy for the IVs, the following Theorem can be obtained, whose detailed proof is given in Section IV-A.

Theorem 1 (Scheme via t-design)**.** If there exists a t-(N, M, K, r, λ) design, we can obtain a cascaded CDC scheme with $K = \lambda\binom{N}{t}/\binom{M}{t}$ distributed computing workers, N files, N Reduce functions such that each Reduce function is computed by $s = \lambda\binom{N-1}{t-1}/\binom{M-1}{t-1}$ workers, the computation load $r = \lambda\binom{N-1}{t-1}/\binom{M-1}{t-1}$, and the communication load $L_{t\text{-Design}} = \frac{N-1}{2N}$. $\square$

According to the proof of Theorem 1 in Subsection IV-A, we can see that the key point of designing the delivery strategy is the second property of t-design. This implies that when a design has a similar property to the second property of t-design, we can also obtain a new scheme based on such design. For instance, given a t-GDD $(\mathcal{X}, \mathfrak{G}, \mathfrak{B})$, by taking the block set $\mathfrak{B}$ as worker set and the point set $\mathcal{X}$ to generate the data placement and Reduce function assignment, and using the third property of t-GDD to generate the delivery strategy for the IVs, the following Theorem can be obtained, whose proof is given in IV-B.

Theorem 2 (Scheme via t-GDD)**.** If there exists a t-(m, q, M, λ) GDD, we can obtain a cascaded CDC scheme with $K = \lambda\binom{m}{t}q^t/\binom{M}{t}$ distributed computing workers, mq files, mq Reduce functions such that each Reduce function is computed by $s = \lambda\binom{m-1}{t-1}q^{t-1}/\binom{M-1}{t-1}$ workers, the computation load $r = \lambda\binom{m-1}{t-1}q^{t-1}/\binom{M-1}{t-1}$, and the communication load $L_{t\text{-GDD}} = \frac{1}{2} + \frac{q-2}{mq}$. $\square$

Remarkably, the communication loads in Theorem 1 and Theorem 2 are asymptotically optimal as stated in the following Proposition, whose proof is included in Appendix A.

Proposition 1 (Asymptotic Optimality)**.** Consider the cascaded CDC system described in Section II-A with computation load $r = s$ and $N = Q$.

- If $K \gg r$, $K \to \infty$, and $N \to \infty$, then the t-design scheme in Theorem 1 is optimal, i.e., $L_{t\text{-Design}} = L^*_{\text{comm.}} = \frac{1}{2}$.
- Besides, if $m \to \infty$ and $m \gg q$, then the t-GDD scheme in Theorem 2 is optimal, i.e., $L_{t\text{-GDD}} = L^*_{\text{comm.}} = \frac{1}{2}$.

$\square$

Now we compare the communication loads in Theorem 1 and Theorem 2 with that of the state-of-art works in [19]–[21].

Firstly, by Lemma 4, we know that we can obtain arbitrary t-(N, M, λ) designs for any parameters t, N, M and λ when N is large. Then by Theorem 1, we can obtain the cascaded CDC scheme with N files, $Q = N$ Reduce functions, the computation load $r = \lambda\binom{N-1}{t-1}/\binom{M-1}{t-1}$ and the communication load $L_{\text{comm.}} = \frac{N-1}{2N}$ for any parameters t, N, M and λ when N is large.

Secondly, by Definition 2 we know that SBIBD is a special 2-design with the point number equal to the block number, i.e., $N = K$. This implies that we can obtain the scheme with more flexible parameters compared with the scheme in [20].

Thirdly, it is well known that there are many existing results of 2-designs (i.e., BIBD) and 2-GDDs which are listed in [35, Section II and IV]. Then by Theorem 1 and Theorem 2, the schemes listed in Table III can be obtained.

TABLE III: New schemes via (N, M, λ) BIBD and 2-(m, q, M, λ) GDD

Schemes	Number of Nodes K	Computation Load r	Replication Factor s	Number of Files N	Number of Reduce Functions Q	Communication Load $L_{\text{comm.}}$	Operation Field $\mathbb{F}_2$
In Theorem 1	$\frac{\lambda N(N-1)}{M(M-1)}$	$\frac{\lambda(N-1)}{M-1}$	$\frac{\lambda(N-1)}{M-1}$	N	N	$\frac{N-1}{2N} < \frac{1}{2}$	Yes
In Theorem 2	$\frac{\lambda m(m-1)q^2}{M(M-1)}$	$\frac{\lambda(m-1)q}{M-1}$	$\frac{\lambda(m-1)q}{M-1}$	mq	mq	$\frac{1}{2} + \frac{q-2}{mq}$	Yes

For the $t = 2$-design scheme in Theorem 1, denote the corresponding number of files and the number of cached files as N_1 and M_1, respectively. Then, we compare our scheme in Theorem 1 with the schemes in [19]–[21] when the parameters $K = \frac{\lambda N_1(N_1-1)}{M_1(M_1-1)}$, $r = s = \frac{N_1-1}{M_1-1}$. Then the schemes in [19]–[21] can be obtained in Table IV. For any positive integers M_1, N_1 and λ, it is not difficult to check that the following inequality always holds

$$\left(1 - \frac{M_1}{N_1}\right)^{\frac{\lambda(N_1-1)}{M_1-1}} \cdot \frac{1}{\frac{2\lambda(N_1-1)}{M-1} - 1} - \left(\frac{M_1}{N_1}\right)^{\frac{\lambda(N_1-1)}{M_1-1}}$$

when

$$\frac{N_1}{M_1} > 1 + \left(\frac{2\lambda(N_1-1)}{M_1-1} + 1\right)^{\frac{M_1-1}{\lambda(N_1-1)}}, \quad \text{i.e.,} \quad \frac{K}{r} > 1 + (2r+1)^{\frac{1}{r}}$$

TABLE IV: Existing cascaded CDC schemes where the positive integers M_1, N_1 and λ satisfying $2 < M_1 < N_1$, $M_1(M_1-1)|\lambda N_1(N_1-1)$, $(M_1-1)|\lambda(N_1-1)$ and $M_1|N_1$.

Schemes	Number of Nodes K	Computation Load r	Replication Factor s	Number of Files N	Number of Reduce Functions Q	Communication Load $L_{\text{comm.}}$	Operation Field $\mathbb{F}_2$
[19]	$\frac{\lambda N_1(N_1-1)}{M_1(M_1-1)}$	$\frac{\lambda(N_1-1)}{M_1-1}$	$\frac{\lambda(N_1-1)}{M_1-1}$	$(\frac{N_1}{M_1})^{\frac{\lambda(N_1-1)}{M_1-1}-1}$	$(\frac{N_1}{M_1})^{\frac{\lambda(N_1-1)}{M_1-1}-1}$	$\frac{1}{2}-\frac{1}{2}\cdot\left(\left(\frac{M_1}{N_1}\right)^{\frac{\lambda(N_1-1)}{M_1-1}}+\left(1-\frac{M_1}{N_1}\right)^{\frac{\lambda(N_1-1)}{M_1-1}}\cdot\frac{1}{\frac{2\lambda(N_1-1)}{M_1-1}-1}\right)$	No
[20]				$\frac{\lambda N_1(N_1-1)}{M_1(M_1-1)}$	$\frac{\lambda N_1(N_1-1)}{M_1(M_1-1)}$	$\frac{\frac{\lambda(N_1-1)}{M_1-1}}{\frac{\lambda(N_1-1)}{M_1-1}-1}\cdot\left(1-\frac{M_1}{N_1}\right)>1-\frac{M_1}{N_1}$	No
[21]				$(\frac{N_1}{M_1})^{\frac{\lambda(N_1-1)}{M_1-1}-1}$	$\frac{N_1}{M_1}$		Yes

where $K=\frac{\lambda N_1(N_1-1)}{M_1(M_1-1)}$ and $r=\frac{\lambda(N_1-1)}{M_1-1}$. Clearly, when N_1 is much larger than M_1, the above inequality always holds. So we have the following statements.

Remark 2. By Table III and Table IV, for the same parameters K, $r=s$ our scheme in Theorem 1 has

- much smaller file number N and Reduce function number Q, and less communication load than that of the scheme in [19];
- smaller file number N and Reduce function number Q, and less communication load than that of the scheme in [20];
- smaller file number N, some larger Reduce function number Q, and less communication load than that of the scheme in [21];

□

Now let us consider comparisons between our scheme in Theorem 2 and schemes in [19]–[21] respectively when the parameters $K=\frac{\lambda m(m-1)q^2}{M(M-1)}$ and $r=s=\frac{\lambda(m-1)q}{M-1}$. Then the schemes in [19]–[21] can be obtained in Table V. Similar to the above comparisons the following statements can be obtained.

TABLE V: Existing cascaded CDC schemes

Schemes	Number of Nodes K	Computation Load r	Replication Factor s	Number of Files N	Number of Reduce Functions Q	Communication Load $L_{\text{comm.}}$	Operation Field $\mathbb{F}_2$
[19]	$\frac{\lambda m(m-1)q^2}{M(M-1)}$	$\frac{\lambda(m-1)q}{M-1}$	$\frac{\lambda(m-1)q}{M-1}$	$(\frac{mq}{M})^{\frac{\lambda(m-1)q}{M-1}-1}$	$(\frac{mq}{M})^{\frac{\lambda(m-1)q}{M-1}-1}$	$\frac{1}{2}-\frac{1}{2}\cdot\left(\left(\frac{M}{mq}\right)^{\frac{\lambda(m-1)q}{M-1}}+\left(1-\frac{M}{mq}\right)^{\frac{\lambda(m-1)q}{M-1}}\cdot\frac{1}{\frac{2\lambda(m-1)q}{M-1}-1}\right)$	No
[20]				$\frac{\lambda(m-1)q}{M-1}$	$\frac{\lambda(m-1)q}{M-1}$	$\frac{\frac{\lambda(m-1)q}{M-1}}{\frac{\lambda(m-1)q}{M-1}-1}\cdot\left(1-\frac{M}{mq}\right)>1-\frac{M}{mq}$	No
[21]				$(\frac{mq}{M})^{\frac{\lambda(m-1)q}{M-1}-1}$	$\frac{mq}{M}$		Yes

Remark 3. By Table III and Table V, for the same parameters K, $r=s$ our scheme in Theorem 2 has

- much smaller file number N and Reduce function number Q than that of the scheme in [19] with the approximately same communication load;
- has smaller file number N and Reduce function number, Q, and less communication load that that of the scheme in [20];
- smaller file number N and some larger Reduce function number Q and less communication load that that of the scheme in [21];

□

Finally, let us take the well known existing $\left(\frac{p^{m+1}-1}{p-1},\frac{p^m-1}{p-1},\frac{p^{m-1}-1}{p-1}\right)$ SBIBD and 2-$(p,p,p,1)$ GDD for any prime power p and positive integer $m\geq 2$ [35]. So the following schemes listed in Table VI can be obtained.

Remark 4. By Table VI the following statements hold.

- Our scheme for Theorem 2 (i.e., the fourth row of Table VI) has the communication load approximating the communication load of the scheme in [19] while having much smaller file number and Reduce function number than that of the scheme in [19], [21] when p is large, and has much smaller file number and less communication load than that of the scheme in [21].
- For any positive integer m and any prime power $p\geq 3$, i.e., with the same order of parameters K, N, Q, $r=s$, the communication load of our scheme for Theorem 1 (i.e., the seventh row of Table VI) is much less than that of the scheme in [20].

□

TABLE VI: Cascaded CDC schemes via $\left(\frac{p^{m+1}-1}{p-1}, \frac{p^m-1}{p-1}, \frac{p^{m-1}-1}{p-1}\right)$ SBIBD and 2-$(p,p,p,1)$ GDD with any prime power p.

Schemes	Number of Nodes K	Computation Load r	Replication Factor s	Number of Files N	Number of Reduce Functions Q	Communication Load $L_{\text{comm.}}$	Operation Field $\mathbb{F}_2$
[19]	p^2	p	p	p^{p-1}	p^{p-1}	$\frac{1}{2}-\frac{1}{2}\cdot\left(\frac{1}{p}\right)^p + \left(1-\frac{1}{p}\right)^p\cdot\frac{1}{4p-2}$	No
[21]					p	1	Yes
Theorem 2				p^2	p^2	$\frac{1}{2}+\frac{p-2}{p^2}$	Yes
Theorem 1 with $m=p$	p^2+p+1	$p+1$	$p+1$	p^2+p+1	p^2+p+1	$\frac{1}{2}-\frac{1}{2(p^2+p+1)}$	Yes
[20]	$\frac{p^{m+1}-1}{p-1}$	$\frac{p^m-1}{p-1}$	$\frac{p^{m-1}-1}{p-1}$	$\frac{p^{m+1}-1}{p-1}$	$\frac{p^{m+1}-1}{p-1}$	$>\frac{p-1}{p}$	No
Theorem 1						$<\frac{1}{2}$	Yes

IV. Novel Schemes for Theorem 1 and Theorem 2

In this section, we introduce how to use combinatorial design structures to construct CDC schemes to improve communication efficiency with small numbers of Reduce functions and input files.

Before describing our schemes, we first recall (1) and (2) which mean that the storing files strategy and arranging Reduce function strategy are determined by the subsets $\mathcal{D}_n$ (i.e., the worker set each of which stores file w_n) and $\mathcal{A}_q$ (i.e., the worker set each of which is arranged to compute the Reduce function $\phi_q(w_1, w_2, \ldots, w_N)$). This implies that in order to design a desired scheme, we only need to study

$$\text{the Reduce function arranged set } \mathfrak{A} = \{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_Q\} \quad \text{and} \tag{13a}$$

$$\text{the file stored set } \qquad \mathfrak{D} = \{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_N\}. \tag{13b}$$

Thus, we can study the Reduce function arranged set $\mathfrak{A}$ and the file stored set $\mathfrak{D}$ defined in (13) to design cascaded CDC schemes with the computation load and the communication load as small as possible. When we let the Reduce function arranged set $\mathfrak{A}$ equal to the file stored set $\mathfrak{B}$ defined in (13), i.e., $\mathfrak{A} = \mathfrak{D}$, by using the classic combinatorial structure, we can obtain two new cascaded coded distributed computing schemes based on t-design (see Definition 2) and t-GDD (see Definition 4), respectively.

A. New CDC Scheme in Theorem 1 via t-design

Suppose that $(\mathcal{X}, \mathfrak{B})$ is a t-(N, M, K, r, λ) design where $\mathcal{X} = \{x_1, x_2, \ldots, x_N\}$ and $\mathfrak{B} = \{\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_K\}$. By Lemma 2, its dual design $(\mathfrak{B}, \mathcal{X})$ is M-regular and λ_2-cross where there there are exactly $K = \lambda\binom{N}{t}/\binom{M}{t}$ points each of which occurs in exactly M blocks, there are N blocks each of which has size $r = \lambda\binom{N-1}{t-1}/\binom{M-1}{t-1}$, and any two distinct blocks intersect in exactly $\lambda_2 = \lambda\binom{N-2}{t-2}/\binom{M-2}{t-2}$ points. Let $\mathcal{X}$ be the Reduce function arranged set $\mathfrak{A}$ and the file stored set $\mathfrak{D}$ defined in (13). We will construct a cascaded CDC scheme with K workers $\mathcal{K} = \mathfrak{B}$, N files $\mathcal{W} = \{w_{x_1}, w_{x_2}, \ldots, w_{x_N}\}$ and N functions $\mathcal{Q} = \{\phi_{x_1}, \phi_{x_2}, \ldots, \phi_{x_N}\}$ where each worker stores M files and each Reduce function is computed by $s = r$ workers.

In the map phase, let $\mathcal{X}$ be the Reduce function arranged set $\mathfrak{A}$. From (1) each worker $\mathcal{B} \in \mathfrak{B}$ stores the files $\mathcal{Z}_{\mathcal{B}} = \{w_x \mid x \in \mathcal{B}, x \in \mathcal{X}\}$. Recall the dual design $(\mathfrak{B}, \mathcal{X})$ is M-regular, then we have $|\mathcal{Z}_k| = M$. So the computation load is

$$r = \frac{\sum_{i=1}^{K} |\mathcal{Z}_i|}{N} = \frac{KM}{N} = \frac{\lambda M\binom{N}{t}}{N\binom{M}{t}}.$$

In the shuffle phase, from (2), each worker $\mathcal{B} \in \mathfrak{B}$ is arranged to compute the Reduce functions

$$\mathcal{Q}_{\mathcal{B}} = \{u_y = \phi_y(w_{x_1}, w_{x_2}, \ldots, w_{x_N}) \mid y \in \mathcal{X}, y \in \mathcal{B}\}.$$

Using the stored files and functions $\mathcal{Q}$, worker $\mathcal{B}$ could compute the IVs $\mathcal{I}_{\mathcal{B}} = \{v_{y,x} = g_{y,x}(w_x) \mid x, y \in \mathcal{X}, x \in \mathcal{B}\}$. Then for any $x, y \in \mathcal{X}$ and any block $\mathcal{B} \in \mathfrak{B}$, the IV $v_{y,x}$ is required and is not locally computed by worker $\mathcal{B}$ if and only if $y \in \mathcal{B}$ and $x \notin \mathcal{B}$, and the IV $v_{y,x}$ is locally computed by worker $\mathcal{B}$ if and only if $x \in \mathcal{B}$. According to the above investigations, we will design a delivery strategy for exchanging the IVs among the workers by means of the cross property of the dual design $(\mathfrak{B}, \mathcal{X})$. Since $(\mathfrak{B}, \mathcal{X})$ is λ_2 cross [1], let

$$\{\mathcal{B}_{1,1}, \mathcal{B}_{1,2}, \ldots, \mathcal{B}_{1,\lambda_2}\} = x \cap y,\ \{\mathcal{B}_{2,1}, \mathcal{B}_{2,2}, \ldots, \mathcal{B}_{2,s-\lambda_2}\} = x \setminus y,\ \{\mathcal{B}_{3,1}, \mathcal{B}_{3,2}, \ldots, \mathcal{B}_{3,s-\lambda_2}\} = y \setminus x.$$

Then each worker $\mathcal{B}_{1,i_1}$ where $i_1 \in [\lambda_2]$ can locally compute the IVs $v_{y,x}$ and $v_{x,y}$ since it stores the files w_x and w_y; each worker $\mathcal{B}_{2,i_2}$ where $i_2 \in [s-\lambda_2]$ can locally compute the IV $v_{y,x}$ since it stores file w_x and requires the IV $v_{x,y}$ from the

[1]This is derived by (7) which is determined by the second property of t-design.

other workers since it does not store the files w_y but is arranged to compute the Reduce function u_y; and each worker $\mathcal{B}_{3,i_3}$ where $i_3 \in [s-\lambda_2]$ can locally compute the IV $v_{x,y}$ since it stores file w_y and requires the IV $v_{y,x}$ from the other workers since it does not store the files w_x but is arranged to compute the Reduce function u_x.

Based on the above investigations, we divide the IVs $v_{y,x}$ and $v_{x,y}$ into λ_2 sub-IVs

$$v_{y,x} = (v_{y,x}^{(1)}, v_{y,x}^{(2)}, \ldots, v_{y,x}^{(\lambda_2)}), \qquad v_{x,y} = (v_{x,y}^{(1)}, v_{x,y}^{(2)}, \ldots, v_{x,y}^{(\lambda_2)}).$$

Each worker $\mathcal{B}_{1,i_1}$, for $i_1 \in [\lambda_2]$, broadcasts the coded signal $X_{y,x}^{(i_1)} = v_{y,x}^{(i_1)} \oplus v_{x,y}^{(i_1)}$ to the workers $\mathcal{B}_{2,i_2}$, $\mathcal{B}_{3,i_3}$ for all $i_2, i_3 \in [s-\lambda_2]$. Clearly, each worker $\mathcal{B}_{2,i_2}$ and each $\mathcal{B}_{3,i_3}$ can decode their required sub-IVs $v_{x,y}^{(i_1)}$ and $v_{y,x}^{(i_1)}$ respectively. So there are exactly $\lambda_2 \cdot \binom{N}{2}$ sub-IVs transmitted. Recall that each IV has T bits. Then the communication load is

$$L_{\text{comm.}} = \frac{\lambda_2 \cdot \binom{N}{2} \cdot \frac{T}{\lambda_2}}{NQT} = \frac{N(N-1)}{2N^2} = \frac{N-1}{2N}.$$

Thus, using t-design, we can obtain Theorem 1.

Remark 5 (Multicast gain)**.** From the above introduction, each of the sub-IVs $v_{y,x}^{(i_1)}$ and $v_{x,y}^{(i_1)}$ where $i_1 \in [\lambda_2]$ is required by $s-\lambda_2 = r-\lambda_2$ workers who can not locally compute it themselves. So the coded signal $v_{y,x}^{(i_1)} \oplus v_{x,y}^{(i_1)}$ is useful for $2(r-\lambda_2)$ workers, i.e., the multicast gain is $2(r-\lambda_2)$. So our scheme generated by a $(N, M, K, r, 1)$ BIBD achieves a multicast gain $2(r-\lambda_2) = 2r-2$ that is just one less than the maximum multicast gain of the scheme proposed in [4].

Finally, we use the $(7,3,1)$ SBIBD in Example 1 to further explain our construction idea.

Example 3. When $N = Q = K = 7$, we have 7 files $\mathcal{W} = \{w_1, w_2, \ldots, w_7\}$ and 7 functions $\mathcal{Q} = \{\phi_1, \phi_2, \ldots, \phi_7\}$. We can use the dual design of the $(7,3,1)$ SBIBD $(\mathcal{X}, \mathfrak{B})$ in Example 1 to generate the following scheme. In the map phase, all the workers store the following files respectively.

$$\mathcal{Z}_{\mathcal{B}_1} = \{w_1, w_2, w_4\},\ \mathcal{Z}_{\mathcal{B}_2} = \{w_2, w_3, w_5\},\ \mathcal{Z}_{\mathcal{B}_3} = \{w_3, w_4, w_6\},\ \mathcal{Z}_{\mathcal{B}_4} = \{w_4, w_5, w_7\},$$
$$\mathcal{Z}_{\mathcal{B}_5} = \{w_1, w_5, w_6\},\ \mathcal{Z}_{\mathcal{B}_6} = \{w_2, w_6, w_7\},\ \mathcal{Z}_{\mathcal{B}_7} = \{w_1, w_3, w_7\}.$$

So the computation load is $r = \frac{N-1}{M-1} = \frac{7-1}{3-1} = 3$.

Assume that the Reduce functions are arranged for workers as follows.

$$\mathcal{Q}_{\mathcal{B}_1} = \{\phi_1, \phi_2, \phi_4\},\ \mathcal{Q}_{\mathcal{B}_2} = \{\phi_2, \phi_3, \phi_5\},\ \mathcal{Q}_{\mathcal{B}_3} = \{\phi_3, \phi_4, \phi_6\},\ \mathcal{Q}_{\mathcal{B}_4} = \{\phi_4, \phi_5, \phi_7\},$$
$$\mathcal{Q}_{\mathcal{B}_5} = \{\phi_1, \phi_5, \phi_6\},\ \mathcal{Q}_{\mathcal{B}_6} = \{\phi_2, \phi_6, \phi_7\},\ \mathcal{Q}_{\mathcal{B}_7} = \{\phi_1, \phi_3, \phi_7\}.$$

Here each Reduce function is computed by $s = 3$ workers.

Recall that all the workers can locally compute the following IVs.

$$\begin{aligned}
\mathcal{I}_{\mathcal{B}_1} &= \{v_{q,n} \mid q \in [7], n \in \{1,2,4\}\}, & \mathcal{I}_{\mathcal{B}_2} &= \{v_{q,n} \mid q \in [7], n \in \{2,3,5\}\},\\
\mathcal{I}_{\mathcal{B}_3} &= \{v_{q,n} \mid q \in [7], n \in \{3,4,6\}\}, & \mathcal{I}_{\mathcal{B}_4} &= \{v_{q,n} \mid q \in [7], n \in \{4,5,7\}\},\\
\mathcal{I}_{\mathcal{B}_5} &= \{v_{q,n} \mid q \in [7], n \in \{1,5,6\}\}, & \mathcal{I}_{\mathcal{B}_6} &= \{v_{q,n} \mid q \in [7], n \in \{2,6,7\}\},\\
\mathcal{I}_{\mathcal{B}_7} &= \{v_{q,n} \mid q \in [7], n \in \{1,3,7\}\}.
\end{aligned}$$

Then the IVs required by the workers are listed in Table VII. In this case, all the workers can send the coded signals listed in Table VIII. For instance, worker $\mathcal{B}_1$ sends the coded signal $v_{1,2} \oplus v_{2,1}$. After receiving $v_{1,2} \oplus v_{2,1}$, worker $\mathcal{B}_2$ and worker $\mathcal{B}_6$ can individually decode the requiring IV $v_{2,1}$ with the locally computed IV $v_{1,2}$; worker $\mathcal{B}_5$ and worker $\mathcal{B}_7$ can individually decode the requiring IV $v_{1,2}$ with its locally computed IV $v_{2,1}$. Similarly, all the other workers can obtain their required IVs respectively. Then the communication load is $L_{\text{comm.}} = \frac{3\times 7}{7^2} = \frac{3}{7} = \frac{N-1}{2N}$. When $K = 7$, $r = s = 3$, we can obtain a cascaded

TABLE VII: Intermediate values $\{v_{q,n}\}$ required by workers in $\mathfrak{B}$.

Parameters	worker set $\mathfrak{B}$						
q, n	$\mathcal{B}_1$	$\mathcal{B}_2$	$\mathcal{B}_3$	$\mathcal{B}_4$	$\mathcal{B}_5$	$\mathcal{B}_6$	$\mathcal{B}_7$
q	$1,2,4$	$2,3,5$	$3,4,6$	$4,5,6$	$1,5,6$	$2,6,7$	$1,3,7$
n	$3,5,6,7$	$1,4,6,7$	$1,2,5,7$	$1,2,3,6$	$2,3,4,7$	$1,3,4,5$	$2,4,5,6$

TABLE VIII: Coded signals sent by workers in $\mathfrak{B}$.

worker set $\mathfrak{B}$						
$\mathcal{B}_1$	$\mathcal{B}_2$	$\mathcal{B}_3$	$\mathcal{B}_4$	$\mathcal{B}_5$	$\mathcal{B}_6$	$\mathcal{B}_7$
$v_{1,2} \oplus v_{2,1}$	$v_{2,3} \oplus v_{3,2}$	$v_{3,4} \oplus v_{4,3}$	$v_{4,5} \oplus v_{5,4}$	$v_{1,5} \oplus v_{5,1}$	$v_{2,6} \oplus v_{6,2}$	$v_{1,3} \oplus v_{3,1}$
$v_{1,4} \oplus v_{4,1}$	$v_{2,5} \oplus v_{5,2}$	$v_{3,6} \oplus v_{6,3}$	$v_{4,7} \oplus v_{7,4}$	$v_{1,6} \oplus v_{6,1}$	$v_{2,7} \oplus v_{7,2}$	$v_{1,7} \oplus v_{7,1}$
$v_{2,4} \oplus v_{4,2}$	$v_{3,5} \oplus v_{5,3}$	$v_{4,6} \oplus v_{6,4}$	$v_{5,7} \oplus v_{7,5}$	$v_{5,6} \oplus v_{6,5}$	$v_{6,7} \oplus v_{7,6}$	$v_{3,7} \oplus v_{7,3}$

CDC scheme in [4] with $N = Q = 35$, $K = 7$, $r = s = 3$ and communication load $L_{\text{comm.}} = \frac{11}{25}$ which is larger than the communication load of ours[2] and a scheme in [20] with $N = Q = 7$, $K = 7$, $r = s = 3$ and communication load $L_{\text{comm.}} = \frac{6}{7}$. Clearly, the scheme in [20] has the same communication load as our scheme. However when $r \leq K/2$, the scheme in [20] has a larger communication load than that of ours when the parameters K, N, Q and $r = s$ are the same as ours. This statement can be easily obtained since when $t \leq K/2$ the communication load of the scheme in [20] is $\frac{t}{t-1} \cdot \frac{K-t}{K} > 1/2$ which is larger than the communication load of ours. □

B. New CDC Scheme in Theorem 2 via t-GDD

Remark 6. By Footnote 1, we can see that the key point to design the delivery strategy is the second property of t-design. This implies that if a combinatorial structure also has a similar property to the second property of t-design, we can also use it to generate a cascaded CDC scheme by our constructing method, i.e., taking the blocks as workers, the point set as the Reduce function arranged set $\mathfrak{A}$ and the file stored set $\mathfrak{D}$ respectively, and then according to the property which is similar to the second property of t-design to design the delivery strategy. By the first statement of Remark 1, the third property of t-GDD is similar to the second property of t-design, so in the following, we will take t-GDD to show our claim.

Suppose a triple $(\mathcal{X}, \mathfrak{G}, \mathfrak{B})$ is a t-(m, q, M, λ) GDD where $\mathcal{X} = \{x_1, x_2, \ldots, x_{mq}\}$ and $\mathfrak{B} = \{\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_K\}$. Now let us consider it dual design $(\mathfrak{B}, \mathcal{X})$ of $(\mathcal{X}, \mathfrak{G}, \mathfrak{B})$. By Lemma 5, its dual design $(\mathfrak{B}, \mathcal{X})$ is M-regular design where there are exactly K points each of which occurs in exactly M blocks; there are mq blocks each of which has size $r = \lambda\binom{m-1}{t-1}q^{t-1}/\binom{M-1}{t-1}$; there are $m\binom{q}{2}$ pairs of distinct blocks whose intersection is an empty set, i.e., the two different points in the same group of the GDD; and there are $\binom{mq}{2} - m\binom{q}{2}$ pairs of distinct blocks whose intersection has exactly $\lambda_2 = \lambda\binom{m-2}{t-2}q^{t-2}/\binom{M-2}{t-2}$ points. Here each point of $\mathcal{X}$ is regarded as a block of the dual design. Now using the above dual design $(\mathfrak{B}, \mathcal{X})$, in the following we will construct a cascaded CDC scheme with K workers $\mathcal{K} = \mathfrak{B}$, mq files $\mathcal{W} = \{w_{x_1}, w_{x_2}, \ldots, w_{x_{mq}}\}$ and mq functions $\mathcal{Q} = \{\phi_{x_1}, \phi_{x_2}, \ldots, \phi_{x_{mq}}\}$ where each worker stores M files and each Reduce function is arranged to $s = r$ workers to compute.

In the map phase, from (1) each worker $\mathcal{B} \in \mathfrak{B}$ stores the files $\mathcal{W}_{\mathcal{B}} = \{w_x \mid x \in \mathcal{B}, x \in \mathcal{X}\}$. Recall the dual design $(\mathfrak{B}, \mathcal{X})$ is M-regular, then we have $|\mathcal{Z}_k| = M$. So the computation load is

$$r = \frac{\sum_{k=1}^{K} |\mathcal{Z}_k|}{N} = \frac{KM}{N} = \frac{\lambda\binom{m-1}{t-1}q^{t-1}}{\binom{M-1}{t-1}}.$$

In the shuffle phase, let $\mathcal{X}$ be the Reduce function arranged set $\mathfrak{A}$. From (2), each worker $\mathcal{B} \in \mathfrak{B}$ is arranged to compute the Reduce functions $\mathcal{Q}_{\mathcal{B}} = \{u_y = \phi_y(w_{x_1}, w_{x_2}, \ldots, w_{x_N}) \mid y \in \mathcal{X}, y \in \mathcal{B}\}$. Using the stored files and functions $\mathcal{Q}$, worker $\mathcal{B}$ could compute the IVs $\mathcal{I}_{\mathcal{B}} = \{v_{y,x} = g_{y,x}(w_x) \mid x, y \in \mathcal{X}, x \in \mathcal{B}\}$. Then the delivery strategy is divided into two classes. The first for the $\binom{mq}{2} - m\binom{q}{2}$ pairs of distinct blocks whose intersection has exactly $\lambda_2 = \lambda\binom{m-2}{t-2}q^{t-2}/\binom{M-2}{t-2}$ points. Then we also apply the delivery strategy in the proof of Theorem 1. Similarly, we can also obtain that there are exactly $\lambda_2 \cdot \left(\binom{mq}{2} - m\binom{q}{2}\right)$ coded sub-IVs to be transmitted each of size $\frac{T}{\lambda_2}$, and the amount of the transmitted signal is

$$\lambda_2 \cdot \left(\binom{mq}{2} - m\binom{q}{2}\right) \cdot \frac{T}{\lambda_2} = \left(\binom{mq}{2} - m\binom{q}{2}\right) \cdot T.$$

The second class is for the left mq^2 IVs where there are mq IVs that can be computed by the workers themselves according to their stored files, and there are $mq(q-1)$ IVs that should be transmitted to the workers who require and do not compute by themselves since they do not store the related files. Recall that each point occurs in exactly $r = \lambda\binom{m-1}{t-1}q^{t-1}/\binom{M-1}{t-1}$ blocks of the GDD and any two points in the same group never occur in the same block. Let $\mathcal{G}_i = \{x_{i,1}, x_{i,2}, \ldots, x_{i,q}\}$ for each $i \in [m]$ where $\mathcal{X} = \bigcup_{i=1}^{m} \mathcal{G}_i$. Then for each IV $v_{x,x'}$ where $x, x' \in \mathcal{G}_i$ with $i \in [m]$, it is stored by workers each of which is represented by the block containing x', and required by the workers each of which is represented by the block containing x. Denote all the block sets each of which contains x and x' by

$$\mathfrak{B}_x = \{\mathcal{B}_1^x,\ \mathcal{B}_2^x,\ \ldots,\ \mathcal{B}_r^x\} \qquad \text{and} \qquad \mathfrak{B}_{x'} = \{\mathcal{B}_1^{x'},\ \mathcal{B}_2^{x'},\ \ldots,\ \mathcal{B}_r^{x'}\},$$

respectively. Then we divide $v_{x,x'}$ into r sub-IVs $v_{x,x'}^{(1)}$, $v_{x,x'}^{(2)}$, …, $v_{x,x'}^{(r)}$. Then the worker $\mathcal{B}_h^{x'}$ sends the sub-IV $v_{x,x'}^{(h)}$ to the workers in $\mathfrak{B}_x$, for $h \in [r]$. Clearly, each worker in $\mathfrak{B}_x$ can obtain the $v_{x,x'}$ by receiving all the r sub-IVs from the workers in $\mathfrak{B}_{x'}$. So there are $mq(q-1)r$ sub-IVs transmitted in all. Recall that each IV has T bits. Then the amount of the transmitted signal is $mq(q-1)r \cdot \frac{1}{r} \cdot T = mq(q-1)T$. From the above discussion, we achieve the communication load in Theorem 2.

Finally, let us use the 2-$(3, 3, 3, 1)$ GDD in Example 2 to further explain our construction idea.

[2]It is well known that the communication load in [4] is minimum. Then phenomenon that our communication load is smaller than that of the scheme in [4] is due to the fact that the file number and Reduce function numbers are larger than that of our scheme.

Example 4. When $N = Q = 6$, $K = 4$, we have 6 files $\mathcal{W} = \{w_1, w_2, \ldots, w_6\}$ and 6 functions $\mathcal{Q} = \{\phi_1, \phi_2, \ldots, \phi_6\}$. We can use the dual design of the 2-(3, 2, 3, 1) GDD $(\mathcal{X}, \mathfrak{G}, \mathfrak{B})$ in Example 2 to generate the following scheme. In the map phase, all the workers store the following files respectively.

$$\mathcal{Z}_{\mathcal{B}_1} = \{w_1, w_3, w_5\},\ \mathcal{Z}_{\mathcal{B}_2} = \{w_1, w_4, w_6\},\ \mathcal{Z}_{\mathcal{B}_3} = \{w_2, w_4, w_5\},\ \mathcal{Z}_{\mathcal{B}_4} = \{w_2, w_3, w_6\}.$$

So the computation load is $r = \frac{bM}{N} = \frac{4\times 3}{6} = 2$. Assume that u_1 is arranged to the workers $\mathcal{B}_1$ and $\mathcal{B}_2$; u_2 is arranged to the workers $\mathcal{B}_3$ and $\mathcal{B}_4$; u_3 is arranged to the workers $\mathcal{B}_1$ and $\mathcal{B}_4$; u_4 is arranged to the workers $\mathcal{B}_2$ and $\mathcal{B}_3$; u_5 is arranged to the workers $\mathcal{B}_1$ and $\mathcal{B}_3$; u_6 is arranged to the workers $\mathcal{B}_2$ and $\mathcal{B}_4$. Since all the workers can locally compute the following IVs.

$$\begin{aligned}
\mathcal{I}_{\mathcal{B}_1} &= \{v_{q,n} \mid q \in [6], n \in \{1,3,5\}\}, \quad & \mathcal{I}_{\mathcal{B}_2} &= \{v_{q,n} \mid q \in [6], n \in \{1,4,6\}\},\\
\mathcal{I}_{\mathcal{B}_3} &= \{v_{q,n} \mid q \in [6], n \in \{2,4,5\}\}, \quad & \mathcal{I}_{\mathcal{B}_4} &= \{v_{q,n} \mid q \in [6], n \in \{2,3,6\}\},
\end{aligned}$$

then worker $\mathcal{B}_1$ needs the IVs $\{v_{q,n} \mid q \in \{1,3,5\}, n \in \{2,4,6\}\}$; worker $\mathcal{B}_2$ needs the IVs $\{v_{q,n} \mid q \in \{1,4,6\}, n \in \{2,3,5\}\}$; worker $\mathcal{B}_3$ needs the IVs $\{v_{q,n} \mid q \in \{2,4,5\}, n \in \{1,3,6\}\}$; and worker $\mathcal{B}_4$ needs the IVs $\{v_{q,n} \mid q \in \{2,3,6\}, n \in \{1,4,5\}\}$. All the transmitted coded signals are listed in Table IX. Here

TABLE IX: Coded signals sent by workers in $\mathfrak{B}$.

worker set $\mathfrak{B}$			
$\mathcal{B}_1$	$\mathcal{B}_2$	$\mathcal{B}_3$	$\mathcal{B}_4$
$v_{1,3}+v_{3,1}$	$v_{1,4}+v_{4,1}$	$v_{2,4}+v_{4,2}$	$v_{2,3}+v_{3,2}$
$v_{1,5}+v_{5,1}$	$v_{1,6}+v_{6,1}$	$v_{2,5}+v_{5,2}$	$v_{2,6}+v_{6,2}$
$v_{3,5}+v_{5,3}$	$v_{4,6}+v_{6,4}$	$v_{4,5}+v_{5,4}$	$v_{3,6}+v_{6,3}$
$v^{(1)}_{2,1}$	$v^{(2)}_{2,1}$	$v^{(1)}_{1,2}$	$v^{(2)}_{1,2}$
$v^{(1)}_{4,3}$	$v^{(1)}_{3,4}$	$v^{(2)}_{3,4}$	$v^{(2)}_{4,3}$
$v^{(1)}_{6,5}$	$v^{(1)}_{5,6}$	$v^{(2)}_{6,5}$	$v^{(2)}_{5,6}$

$$v_{1,2} = (v^{(1)}_{1,2}, v^{(2)}_{1,2}),\ v_{2,1} = (v^{(1)}_{2,1}, v^{(2)}_{2,1}),\ v_{3,4} = (v^{(1)}_{3,4}, v^{(2)}_{3,4}),\ v_{4,3} = (v^{(1)}_{4,3}, v^{(2)}_{4,3}),\ v_{5,6} = (v^{(1)}_{5,6}, v^{(2)}_{5,6}),\ v_{6,5} = (v^{(1)}_{6,5}, v^{(2)}_{6,5}).$$

We can check that all the workers can decode their required IVs. Then the communication load is $L_{\text{comm.}} = \frac{3\times 4 + 3\times 4\times \frac{1}{2}}{6^2} = \frac{1}{2}$. $\square$

V. Extentions: Case $r \neq s$

In this section we will show that our constructing method can be extended to the case $r \neq s$. Suppose that $(\mathcal{X}, \mathfrak{B})$ is a t-(N, M, K, r, λ) design where $\mathcal{X} = \{x_1, x_2, \ldots, x_N\}$ and $\mathfrak{B} = \{\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_K\}$. By Lemma 2, its dual design $(\mathfrak{B}, \mathcal{X})$ is M-regular design where there are exactly K points each of which occurs in exactly M blocks, there are N blocks each of which has size $r = \lambda\binom{N-1}{t-1}/\binom{M-1}{t-1}$. Here each point of $\mathcal{X}$ is regarded as a block of the dual design and $\mathcal{X}$ is regarded as the file stored set. For instance and without loss of generality, let $x_1 = \{\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_r\}$. Then this implies that the file w_{x_1} is stored by workers in x_1. Let $\mathfrak{A} = \binom{\mathcal{X}}{t-1} = \{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_{\binom{N}{t-1}}\}$ denote the Reduce function arranged set.

Now using the $(\mathcal{X}, \mathfrak{B})$ and its above dual design $(\mathfrak{B}, \mathcal{X})$, in the following we will construct a cascaded CDC scheme with K workers $\mathcal{K} = \mathfrak{B}$, N files $\mathcal{W} = \{w_{x_1}, w_{x_2}, \ldots, w_{x_N}\}$ and $Q = \binom{N}{t-1}$ functions $\mathcal{Q} = \{\phi_{\mathcal{A}_1}, \phi_{\mathcal{A}_2}, \ldots, \phi_{\mathcal{A}_Q}\}$ where each worker stores M files and each Reduce function is assigned to $s = \lambda_{t-1} = \frac{\lambda(N-t+1)}{M-t+1}$ workers to compute.

In the map phase, from (1) each worker $\mathcal{B} \in \mathfrak{B}$ stores the files $\mathcal{Z}_{\mathcal{B}} = \{w_x \mid x \in \mathcal{B}, x \in \mathcal{X}\}$. Recall the dual design $(\mathfrak{B}, \mathcal{X})$ is M-regular, then we have $|\mathcal{Z}_k| = M$. So the computation load is

$$r = \frac{\sum_{i=1}^{K} |\mathcal{Z}_i|}{N} = \frac{KM}{N} = \frac{\lambda\binom{N-1}{t-1}}{\binom{M-1}{t-1}}.$$

Using the stored files and functions $\mathcal{Q}$, worker $\mathcal{B}$ could compute the IVs

$$\mathcal{I}_{\mathcal{B}} = \left\{ v_{\mathcal{A},x} = g_{\mathcal{A},x}(w_x) \;\middle|\; x \in \mathcal{X}, x \in \mathcal{B}, \mathcal{A} \in \binom{\mathcal{X}}{t-1} \right\}.$$

In the shuffle phase, from (2), each worker $\mathcal{B} \in \mathfrak{B}$ is arranged to compute the Reduce functions

$$\mathcal{Q}_{\mathcal{B}} = \left\{ u_{\mathcal{A}} = \phi_{\mathcal{A}}(w_{x_1}, w_{x_2}, \ldots, w_{x_N}) \;\middle|\; \mathcal{A} \in \mathfrak{A}, \mathcal{A} \subseteq \mathcal{B} \right\}.$$

Then for any $x \in \mathcal{X}$, $\mathcal{A} \in \binom{\mathcal{X}}{t-1}$ and any block $\mathcal{B} \in \mathfrak{B}$, the IV $v_{\mathcal{A},x}$ is required and is not locally computed by worker $\mathcal{B}$ if and only if $\mathcal{A} \subseteq \mathcal{B}$ and $x \notin \mathcal{B}$, and the IV $v_{\mathcal{A},x}$ is locally computed by worker $\mathcal{B}$ if and only if $x \in \mathcal{B}$. According to the above investigations, we will design a delivery strategy for exchanging the IVs among the workers by means of the design $(\mathcal{X}, \mathfrak{B})$. Recall that each t-subset $\mathcal{C} = \{x_1, x_2, \ldots, x_t\}$ occurs exactly λ blocks $\mathcal{B}_1$, $\mathcal{B}_2$, …, $\mathcal{B}_\lambda$, i.e., $\mathcal{C} \subseteq \bigcap_{j=1}^{\lambda} \mathcal{B}_j$. Define

$\mathcal{A}_i = \mathcal{C} \setminus \{x_i\}$ for each $i \in [t]$. We divide each IV $v_{\mathcal{A}_i,x}$ where $i \in [t]$ into λ sub-IVs $v_{\mathcal{A}_i,x} = (v^{(1)}_{\mathcal{A}_i,x}, v^{(2)}_{\mathcal{A}_i,x}, \ldots, v^{(\lambda)}_{\mathcal{A}_i,x})$. Each worker $\mathcal{B}_j$, for $j \in [\lambda]$, broadcasts the coded signal $X^{(j)} = \oplus_{i\in[t]} v^{(j)}_{\mathcal{A}_i,x_i}$ to the workers $\mathcal{B}$ satisfying $x_i \notin \mathcal{B}$ and $\mathcal{A}_i \subseteq \mathcal{B}$ for all $i \in [t]$. From the above introduction, we can see that each worker $\mathcal{B}$ satisfying $x_i \notin \mathcal{B}$ and $\mathcal{A}_i \subseteq \mathcal{B}$ can decode their required sub-IVs $v^{(j)}_{\mathcal{A}_i,x_i}$ since $x_{i'} \in \mathcal{B}$ by the assumption that $x_{i'} \in \mathcal{A}_i$ for all $i' \in [t] \setminus \{i\}$. So there are exactly $\lambda \cdot \binom{N}{t}$ coded sub-IVs transmitted. Recall that each IV has T bits. Thus, we achieve the communication load

$$L_{\text{comm.}} = \frac{\lambda \cdot \binom{N}{t} T/\lambda}{N\binom{N}{t-1}T} = \frac{N-t+1}{Nt}.$$

We formally state this result in the following Theorem.

Theorem 3. If there exists a t-(N, M, K, r, λ) design, we can obtain a cascaded CDC scheme with $K = \lambda\binom{N}{t}/\binom{M}{t}$ distributed computing workers, N files, $\binom{N}{t-1}$ Reduce functions such that each Reduce function is computed by $s = \frac{\lambda(N-t+1)}{M-t+1}$ workers, the computation load $r = \lambda\binom{N-1}{t-1}/(\binom{M-1}{t-1})$, and the communication load $L_{\text{comm.}} = \frac{N-t+1}{Nt}$. □

Finally, let us take 3-$(N_1, M_1, 1)$ design to introduce the performance of the scheme in Theorem 3. That is, by Theorem 3 the scheme in the second row of Table X is obtained. Since [19] only studied the schemes with $r = s$ and recall that [20] also mainly focused on the case with $r = s$, we only need to compare with the scheme in [21]. By Table I, we can obtain a scheme in [21] with $K = \frac{N_1(N_1-1)(N_1-2)}{M_1(M_1-1)(M_1-2)}$ distributed computing workers, $N = (\frac{N_1}{M_1})^{\frac{(N_1-1)(N_1-2)}{(M_1-1)(M_1-2)}-1}$ files, $Q = \frac{N_1(N_1-1)}{M_1(M_1-1)}$ Reduce functions such that each Reduce function is computed by $s = \frac{N_1-2}{M_1-2}$ workers, the computation load $r = \frac{(N_1-1)(N_1-2)}{(M_1-1)(M_1-2)}$, and the communication load

$$\begin{aligned} L_{\text{comm.}} =& \frac{s}{r-1}\cdot\left(1-\frac{r}{K}\right) = \frac{\frac{N_1-2}{M_1-2}}{\frac{(N_1-1)(N_1-2)}{(M_1-1)(M_1-2)}-1}\cdot\left(1-\frac{\frac{(N_1-1)(N_1-2)}{(M_1-1)(M_1-2)}}{\frac{N_1(N_1-1)(N_1-2)}{M_1(M_1-1)(M_1-2)}}\right) = \frac{1}{\frac{N_1-1}{M_1-1}-\frac{M_1-2}{N_1-2}}\cdot\left(1-\frac{M_1}{N_1}\right) \\ <& \frac{1}{\frac{N_1-1}{M_1-1}-\frac{M_1-1}{N_1-1}}\cdot\left(1-\frac{M_1}{N_1}\right) = \frac{N_1-1}{N_1}\cdot\frac{M_1-1}{N_1+M_1}. \end{aligned}$$

From Table X, we can see that for the same parameters K, r and s, the scheme in Theorem 3 has a much smaller file number N while increasing some communication load compared with the scheme in [21].

TABLE X: The schemes in [4], [19]–[21] and the scheme via 3-$(N_1, M_1, 1)$ design in Theorem 3

Parameters	Number of Nodes K	Computation Load r	Replication Factor s	Number of Files N	Number of Reduce Functions Q	Communication Load $L_{\text{comm.}}$	Operation Field $\mathbb{F}_2$
Theorem 3	$\frac{N_1(N_1-1)(N_1-2)}{M_1(M_1-1)(M_1-2)}$	$\frac{(N_1-1)(N_1-2)}{(M_1-1)(M_1-2)}$	$\frac{N_1-2}{M_1-2}$	N_1	$\frac{N_1(N_1-1)}{2}$	$\frac{N_1-2}{3N_1}$	Yes
[21]				$\left(\frac{N_1}{M_1}\right)^{\frac{(N_1-1)(N_1-2)}{(M_1-1)(M_1-2)}-1}$	$\frac{N_1(N_1-1)}{M_1(M_1-1)}$	$< \frac{N_1-1}{N_1}\cdot\frac{M_1-1}{N_1+M_1}$	Yes

Remark 7 (Other schemes by using the construction of Theorem 3)**.** Similar to the proof of Theorem 3, we can also obtain a scheme by regarding $\mathcal{X}$ as the stored file set and denoting $\mathfrak{A} = \binom{\mathcal{X}}{t-1}$ the Reduce function arranged set based on a t-(m, q, M, λ) GDD. In addition, we can also obtain the other two schemes by regarding the $\mathfrak{A} = \binom{\mathcal{X}}{t-1}$ as the stored file set and denoting $\mathcal{X}$ the Reduce function arranged set based on a t-(N, M, λ) design $(\mathcal{X}, \mathfrak{B})$ and t-(m, q, M, λ) GDD $(\mathcal{X}, \mathfrak{G}, \mathfrak{B})$, respectively. So this implies that our constructing method can also be extended to other combinatorial structures that have a similar property as the second property of t-design. □

VI. Conclusion

In this paper, different from the existing constructing methods based on MDS, linear combination, or PDA methods, we constructed two classes of the cascaded CDC schemes with $r = s$ via t-design and t-GDD. Remarkably, unlike the existing construction methods which require a large operation field and an exponentially large number of input files, our schemes can notably relax the requirement both on the number of input files and the number of Reduce functions, and simply operate on the binary field while maintaining the asymptotic optimality. Moreover, for some values of K, r, and s, our schemes achieve less communication load than the state-of-art schemes. Finally, our construction method can be used to construct new schemes for the case $r \neq s$ with small file number and Reduce function number.

Appendix A
Proof of Asymptotic Optimality

From (5), the optimal load $L^*_{\text{comm.}}(r,s)$ can be written as

$$\begin{aligned}
L^*_{\text{comm.}}(r,s) &= \sum_{l=\max\{r+1,s\}}^{\min\{r+s,K\}} \left(\frac{l-r}{l-1} \cdot \frac{\binom{K-r}{K-l}\binom{r}{l-s}}{\binom{K}{s}} \right) \\
&\geq \frac{s}{r+s-1} \cdot \frac{\binom{K-r}{K-l}\binom{r}{r}}{\binom{K}{s}} \qquad (14) \\
&= \frac{s}{r+s-1} \cdot \frac{K-r}{K} \cdot \frac{K-r-1}{K-1} \cdot \frac{K-r-s+1}{K-s+1} \\
&\geq \frac{s}{r+s-1} \cdot \left(1 - \frac{r}{K-s+1}\right)^s \\
&\approx \frac{s}{r+s-1} \cdot (1+o(1)), \quad \text{when} \quad K \gg r, s \text{ and } K \to \infty. \qquad (15)
\end{aligned}$$

When $r = s$, (15) can be written as follows.

$$L^*_{\text{comm.}}(r,s) > \frac{r}{2r-1} \cdot (1+o(1)) \quad \text{where} \quad K \gg r \text{ and } K \to \infty.$$

Substituting $r = \lambda\binom{N-1}{t-1}/\binom{M-1}{t-1}$ into the above formula, for any positive integers N, m and t with $t \leq M < N$, we have

$$\begin{aligned}
L^*_{\text{comm.}}(r,s) &> \frac{\lambda\binom{N-1}{t-1}/\binom{M-1}{t-1}}{2\lambda\binom{N-1}{t-1}/\binom{M-1}{t-1} - 1} \cdot (1+o(1)) \qquad (16) \\
&\approx \frac{1}{2}, \quad \text{when} \quad N \gg M, t \text{ or } N \to \infty. \qquad (17)
\end{aligned}$$

Now we consider the achievable communication loads in Theorem 1 and Theorem 2, respectively. By Theorem 1, we have a cascaded CDC scheme with $K = \lambda\binom{N}{t}/\binom{M}{t}$ distributed computing workers, N files, N Reduce functions such that each Reduce function is computed by $s = \lambda\binom{N-1}{t-1}/\binom{M-1}{t-1}$ workers, the computation load $r = \lambda\binom{N-1}{t-1}/\binom{M-1}{t-1}$, and the communication load $L_{\text{comm.}} = \frac{N-1}{2N}$. Clearly, when N is larger, our communication load $\frac{N-1}{2N}$ approximates $\frac{1}{2}$. Similarly, we can also show that the cascaded CDC scheme in Theorem 2 has our communication load approximating $\frac{1}{2}$ when m is large. Together with (17), we obtain that our schemes for Theorem 1 and Theorem 2 are asymptotically optimal.

References

[1] Y. Guo, J. Rao, and X. Zhou, "ishuffle: Improving hadoop performance with shuffle-on-write," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, pp. 1649–1662, 2017.

[2] F. Ahmad, S. T. Chakradhar, A. Raghunathan, and T. N. Vijaykumar, "Tarazu: optimizing mapreduce on heterogeneous clusters," in *ASPLOS XVII*, 2012.

[3] M. Chowdhury, M. A. Zaharia, J. Ma, M. I. Jordan, and I. Stoica, "Managing data transfers in computer clusters with orchestra," *Proceedings of the ACM SIGCOMM 2011 conference*, 2011.

[4] Q. Y. S. Li, M. A. Maddah-Ali and A. S. Avestimehr, "A fundamental tradeoff between computation and communication in distributed computing," *IEEE Trans. Inf. Theory*, vol. 64, no. 1, p. 109–128, 2018.

[5] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, p. 107–113, 2008.

[6] M. J. F. S. S. M. Zaharia, M. Chowdhury and I. Stoica, "Spark: Cluster computing with working sets," *HotCloud*, vol. 10, no. 10-10, p. 95, 2010.

[7] M. Ji, G. Caire, and A. F. Molisch, "Fundamental limits of caching in wireless d2d networks," *IEEE Transactions on Information Theory*, vol. 62, pp. 849–869, 2014.

[8] S. Horii, "Improved computation-communication trade-off for coded distributed computing using linear dependence of intermediate values," in *2020 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2020, pp. 179–184.

[9] S. Prakash, A. Reisizadeh, R. Pedarsani, and A. S. Avestimehr, "Coded computing for distributed graph analytics," *IEEE Transactions on Information Theory*, 2020.

[10] M. A. M.-A. Q. Yu, S. Li and A. S. Avestimehr, "How to optimally allocate resources for coded distributed computing?" *in Proc. IEEE Int. Conf. Commun. (ICC)*, pp. 1–7, May 2017.

[11] H. Chen and Y. Wu, "Coded computing for master-aided distributed computing systems," in *2020 IEEE Information Theory Workshop (ITW)*. IEEE, 2021, pp. 1–5.

[12] N. Woolsey, R.-R. Chen, and M. Ji, "A new combinatorial coded design for heterogeneous distributed computing," *IEEE Transactions on Communications*, 2021.

[13] F. Xu, S. Shao, and M. Tao, "New results on the computation-communication tradeoff for heterogeneous coded distributed computing," *IEEE Transactions on Communications*, vol. 69, no. 4, pp. 2254–2270, 2021.

[14] B. Wang, J. Xie, K. Lu, Y. Wan, and S. Fu, "On batch-processing based coded computing for heterogeneous distributed computing systems," *IEEE Transactions on Network Science and Engineering*, 2021.

[15] A. Reisizadeh, S. Prakash, R. Pedarsani, and A. S. Avestimehr, "Coded computation over heterogeneous clusters," *IEEE Transactions on Information Theory*, vol. 65, no. 7, pp. 4227–4242, 2019.

[16] D. Kim, H. Park, and J. Choi, "Optimal load allocation for coded distributed computation in heterogeneous clusters," *arXiv preprint arXiv:1904.09496*, 2019.

[17] Y. Wang and Y. Wu, "Coded mapreduce with pre-set data and reduce function assignments," in *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, 2022, pp. 1924–1929.

[18] J. S. Ng, W. Y. B. Lim, N. C. Luong, Z. Xiong, A. Asheralieva, D. Niyato, C. Leung, and C. Miao, "A comprehensive survey on coded distributed computing: Fundamentals, challenges, and networking applications," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 3, pp. 1800–1837, 2021.

[19] R. C. N. Woolsey and M. Ji, "A combinatorial design for cascaded coded distributed computing on general networks," in *IEEE Trans. Commun.,*, vol. 69, no. 9, 2021, pp. 5686–5700.

[20] W. W. J. Jiang and L. Zhou, "Cascaded coded distributed computing schemes based on symmetric designs," in *IEEE Trans. Commun.,*, vol. 70, no. 11, 2022, pp. 7179–7190.

[21] J. Jiang and L. Qu, "Cascaded coded distributed computing schemes based on placement delivery arrays," *IEEE Access*, vol. 8, pp. 221 385–221 395, 2020.

[22] Q. Yan, M. Cheng, X. Tang, and Q. Chen, "On the placement delivery array design for centralized coded caching scheme," *IEEE Transactions on Information Theory*, vol. 63, no. 9, pp. 5821–5833, 2017.

[23] Q. Yan, X. Tang, Q. Chen, and M. Cheng, "Placement delivery array design through strong edge coloring of bipartite graphs," *IEEE Communications Letters*, vol. 22, no. 2, pp. 236–239, 2018.

[24] C. Shangguan, Y. Zhang, and G. Ge, "Centralized coded caching schemes: A hypergraph theoretical approach," *IEEE Transactions on Information Theory*, vol. 64, no. 8, pp. 5755–5766, 2018.

[25] M. Cheng, J. Jiang, Q. Yan, and X. Tang, "Constructions of coded caching schemes with flexible memory size," *IEEE Transactions on Communications*, vol. 67, no. 6, pp. 4166–4176, 2019.

[26] M. Cheng, J. Jiang, X. Tang, and Q. Yan, "Some variant of known coded caching schemes with good performance," *IEEE Transactions on Communications*, vol. 68, no. 3, pp. 1370–1377, 2020.

[27] M. Cheng, J. Jiang, Q. Wang, and Y. Yao, "A generalized grouping scheme in coded caching," *IEEE Transactions on Communications*, vol. 67, no. 5, pp. 3422–3430, 2019.

[28] M. Cheng, J. Wang, X. Zhong, and Q. Wang, "A framework of constructing placement delivery arrays for centralized coded caching," *IEEE Transactions on Information Theory*, vol. 67, no. 11, pp. 7121–7131, 2021.

[29] M. Cheng, Y. Li, X. Zhong, and R. Wei, "Improved constructions of coded caching schemes for combination networks," *IEEE Trans. Commun.*, vol. 68, no. 10, pp. 5965–5975, Oct. 2020.

[30] J. Wang, M. Cheng, K. Wan, and G. Caire, "Novel frameworks for coded caching via cartesian product with reduced subpacketization," *arXiv preprint arXiv:2108.08486*, 2021.

[31] X. Zhong, M. Cheng, and R. Wei, "Coded caching schemes with linear subpacketizations," *IEEE Trans. Commun.*, vol. 69, no. 6, pp. 3628–3637, Jun. 2021.

[32] J. Michel and Q. Wang, "Placement delivery arrays from combinations of strong edge colorings," *IEEE Transactions on Communications*, vol. 68, no. 10, pp. 5953–5964, 2020.

[33] X. Zhong, M. Cheng, and J. Jiang, "Placement delivery array based on concatenating construction," *IEEE Communications Letters*, vol. 24, no. 6, pp. 1216–1220, 2020.

[34] M. Zhang, K. Wan, M. Cheng, and G. Caire, "Coded caching for two-dimensional multi-access networks," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2022, pp. 1707–1712.

[35] C. Colbourn and J. Dinitz, "Handbook of combinatorial designs," in *Chapman*, Hall/CRC, 2006, vol. 42.

[36] D. R. Stinson, *Combinatorial designs: constructions and analysis*. Springer,NY, 2004.

[37] P. Keevash, "The existence of designs," *arXiv: 1401.3665*, Jan. 2014.

[38] A. L. Stefan Glock, Daniela Kuhn and D. Osthus, "The existence of designs via iterative absorption," *arXiv: 1611.06827*, Feb. 2016.