A Storage-Computation-Communication Tradeoff for Distributed Computing

Qifa Yan
LTCI, Télécom ParisTech
75013 Paris, France
Email: qifa.yan@telecom-paristech.fr

Sheng Yang
L2S, CentraleSupélec
91190 Gif-sur-Yvette, France
Email: sheng.yang@centralesupelec.fr

Michèle Wigger
LTCI, Télécom ParisTech
75013 Paris, France
Email: michele.wigger@telecom-paristech.fr

***Abstract*—This paper investigates distributed computing systems where computations are split into "Map" and "Reduce" functions. A new coded scheme, called *distributed computing and coded communication (D3C)*, is proposed, and its communication load is analyzed as a function of the available storage space and the number of intermediate values (IVA) to be computed. D3C achieves the smallest possible communication load for a given storage space, while a smaller number of IVAs need to be computed compared to Li *et al.*'s coded distributed computing (CDC) scheme. More generally, our scheme can flexibly trade between storage space and the number of IVAs to be computed. Communication load is then analyzed for any given tradeoff.**

I. Introduction

Distributed computing has emerged as one of the most important approaches to big data analysis, where data-parallel computations are executed accross clusters consisting of many machines. Platforms like MapReduce [1] or Dryad [2], for example, can perform computation tasks that involve data sets with tens of terabytes and more. In these systems, computations are distributed in the following way. Each cluster (referred to as *node* in the sequel) $k \in \{1, \ldots, K\}$ is assigned the computation of a specific output function

$$\phi_k(w_1, \cdots, w_N) = h_k(g_{k,1}(w_1), \cdots, g_{k,N}(w_N)), \quad (1)$$

which can depend on all data elements $w_1, \ldots, w_N$ but decomposes into smaller *map functions* $g_{k,1}, \ldots, g_{k,N}$ each depending only on a single data element. The computation of the output functions $\phi_1, \ldots, \phi_K$ is carried out in three phases. During the first *map phase*, each node k locally stores a subset of the input data $\mathcal{M}_k \subseteq \{w_1, \ldots, w_N\}$ and calculates all intermediate values (IVAs) that depend on the stored data:

$$\{g_{\ell,i}(w_i)\colon\ \ell \in \{1, \ldots, K\},\ i \in \mathcal{M}_k\}.$$

In the subsequent *shuffling phase*, the nodes exchange the IVAs computed during the map phase. The goal is that at the end of this second phase, each node k is aware of all IVAs $g_{k,1}(w_1), \ldots, g_{k,N}(w_N)$ required to calculate its output function ϕ_k. During the final *reduce phase*, each node k then calculates the output function ϕ_k as in (1) based on the previously collected IVAs and the *reduce function* ϕ_k.

Li *et al.* [3] showed that the communication cost of such a distributed computation system can significantly be reduced if the nodes exploit multicast coding opportunities during the shuffling phase. More specifically, they designed a coding scheme for the map and shuffling phases, termed *coded distributed computation (CDC)*, and proved that it has smallest communication load among all distributed computation schemes with same storage requirements. Subsequently, various works extended the results in [3]. For example, the works in [4], [5] account for straggling nodes; [6] studies optimal allocation of computation resources; and [7] extends the works to wireless networks.

In this work, we propose to refine the performance measure and the coding scheme by Li *et al.* [3]. Specifically, we account also for the number of intermediate values (IVAs) that each of the nodes has to calculate. In this sense, we extend the *storage-communication tradeoff* studied in [3] to a *storage-computation-communication tradeoff* where we wish to characterize the smallest communication load required in the shuffling phase for a given storage space and a given number of IVAs totally calculated during the map phase. Notice that Li *et al.* [3] termed their tradeoff *computation-communication tradeoff* because they assume that each node calculates all the IVAs that can be obtained from the data stored at that node, irrespective of whether these IVAs are used in the sequel or not. In this sense, the total number of calculated IVAs is really a measure of the total storage space available at the nodes, this is why we refer to it as storage-communication tradeoff. The main result of this present paper is a new coding scheme, that we term *distributed computing and coded communication (D3C)*. Our scheme in particular achieves the storage-computation tradeoff in [3] but with a reduced number of calculated IVAs compared to [3].

Notations: Let $\mathbb{N}^+$ be the set of positive integers. For $m, n \in \mathbb{N}^+$, denote the n dimensional vector space over the finite field with cardinality 2^m by $\mathbb{F}_{2^m}^n$, and the integer set $\{1, 2, \cdots, n\}$ by $[n]$. Scalars are denoted by upper or lower case letters. Sets or subsets are denoted by calligraphic font, and a collection (set of set) will be denoted by bold font. The cardinality of a set $\mathcal{S}$ is denoted by $|\mathcal{S}|$. The bitwise Exclusive OR (XOR) operation is denoted by $\oplus$.

II. System Model

We consider a system consisting of K distributed computing nodes and N input files each of F bits, where $K, N, F \in \mathbb{N}^+$.

Specifically, given any N files

$$\mathcal{W} := \{w_1, \cdots, w_N\}, \quad w_n \in \mathbb{F}_{2^F}, \forall\ n \in [N],$$

the goal of Node k ($k \in [K]$) is to compute an output function $\phi_k : \mathbb{F}_{2^F}^N \to \mathbb{F}_{2^B}$, which maps all the files to a bit stream $u_k = \phi_k(w_1, \cdots, w_N) \in \mathbb{F}_{2^B}$ of length B for some $B \in \mathbb{N}^+$.

Following the conventions of [3], [7], we assume that the computation of the output functions ϕ_k can be decomposed as:

$$\phi_k(w_1, \cdots, w_N) = h_k(g_{k,1}(w_1), \cdots, g_{k,N}(w_N)),$$

where we define the functions g and h as follows.

- The "map" function

  $$g_{k,n} : \mathbb{F}_{2^F} \to \mathbb{F}_{2^T},\ k \in [K],\ n \in [N],$$

  maps the file w_n into a binary intermediate value (IVA) of T bits, i.e., $v_{k,n} \triangleq g_{k,n}(w_n) \in \mathbb{F}_{2^T}$ for some $T \in \mathbb{N}^+$.
- The "reduce" function

  $$h_k : \mathbb{F}_{2^T}^N \to \mathbb{F}_{2^B},\ k \in [K],$$

  maps the intermediate values

  $$\mathcal{V}_k \triangleq \{v_{k,n} : n \in [N]\}$$

  of the output function ϕ_k into the output stream $u_k = h_k(v_{k,1}, \cdots, v_{k,N})$.

Note that, such a decomposition always exists. For example, one trivial decomposition is setting the map functions be identity functions and the reduce functions be the output functions, i.e., $g_{k,n}(w_n) = w_n$, and $h_k = \phi_k$, $\forall\ n \in [N], k \in [K]$.

*1) **Map Phase**:* A central controller assigns a subset of files $\mathcal{M}_k \subset \mathcal{W}$ to Node k, for all $k \in [K]$. Having access to the files in $\mathcal{M}_k$, Node k computes a subset of IVAs $\mathcal{C}_{k,n} = \{v_{q,n} : q \in \Lambda_{k,n}\}$, where $\Lambda_{k,n} \subset [K]$ for each file $w_n \in \mathcal{M}_k$. Denote the set of IVAs computed at Node k by $\mathcal{C}_k$, i.e.,

$$\mathcal{C}_k \triangleq \cup_{n:w_n \in \mathcal{M}_k} \mathcal{C}_{k,n}. \tag{2}$$

To measure the storage and computation cost of the system, we introduce the following two definitions.

Definition 1 (Storage Space)**.** *We define the* storage space*, denoted by r, as the total number of files stored across the K nodes, normalized by the total number of files N, i.e.,*

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

Definition 2 (Computation Load)**.** *We define the computation load, denoted by c, as the total number of map functions computed across the K nodes, normalized by the total number of map functions NK, i.e.,*

$$c = \frac{\sum_{k=1}^K |\mathcal{C}_k|}{NK}. \tag{4}$$

*2) **Shuffle Phase**:* To compute the output function ϕ_k, Node k needs to collect the IVAs required by ϕ_k but not computed locally in the map phase, i.e., $\mathcal{V}_k \backslash \mathcal{C}_k$. After the map phase, the K nodes proceed to exchange the required IVAs. To be precise, each node k creates a signal $X_k \in \mathbb{F}_{2^{l_k}}$ for some $l_k \in \mathbb{N}$, as a function of the IVAs computed in the map phase, i.e., for some encoding function

$$\varphi_k : \mathbb{F}_{2^T}^{|\mathcal{C}_k|} \to \mathbb{F}_{2^{l_k}}.$$

Node k multicasts the signal

$$X_k = \varphi_k\left(\mathcal{C}_k\right),$$

to all the other nodes. Then we assume that each node k receives the signals $\{X_i\}_{i \in [K] \backslash \{k\}}$ without error.

Definition 3 (Communication Load)**.** *We define the communication load L, as the total length of the bits transmitted by the K nodes during the shuffle phase normalized by the total length of all intermediate values NKT, i.e.,*

$$L \triangleq \frac{\sum_{k=1}^K l_k}{NKT}.$$

*3) **Reduce Phase**:* With the signals $\{X_i\}_{i \in [K]}$, exchanged in the shuffle phase and the IVAs $\mathcal{C}_k$ computed locally, Node k constructs all the IVAs $\mathcal{V}_k$ using some decoding function

$$\psi_k : \mathbb{F}_{2^{l_1}} \times \mathbb{F}_{2^{l_2}} \times \cdots \mathbb{F}_{2^{l_K}} \times \mathbb{F}_{2^T}^{|\mathcal{C}_k|} \to \mathbb{F}_{2^T}^N,$$

So Node k computes

$$(v_{k,1}, \cdots, v_{k,N}) = \psi_k\left(X_1, \cdots, X_K, \mathcal{C}_k\right),$$

followed by the reduce function

$$u_k = h_k(v_{k,1}, \cdots, v_{k,N}). \tag{5}$$

Definition 4. *A distributed computing system achieves a storage-computation-communication (SCC) triple (r, c, L), if for any $\epsilon > 0$, when N is sufficiently large, there exists a map-shuffle-reduce procedure such that the storage space, computation and communication loads do not exceed $r + \epsilon$, $c + \epsilon$, and $L + \epsilon$, respectively. In particular, define the optimal communication load as*

$$L^*(r, c) \triangleq \inf\big\{L : (r, c, L) \textit{ is achievable}\big\}. \tag{6}$$

Note that it is without loss of generality to consider the case

$$1 \le c \le r < K.$$

Indeed, as $|\mathcal{C}_k| \le |\mathcal{M}_k| K$ by (2), thus $c \le r$ from (3) and (4). Further, since each IVA needs to be computed at least once somewhere in the system, we have $c \ge 1$. Finally, if $r \ge K$, each node stores all files and can locally compute all the IVAs required for its output function.

Remark 1. If each Node k computes the IVAs required by all output functions from all the files that it stores, i.e., $\mathcal{C}_k = \{v_{q,n} : q \in [K], w_n \in \mathcal{M}_k\}$, then $|\mathcal{C}_k| = |\mathcal{M}_k| \cdot K$, and

$$c = \frac{\sum_{k=1}^K |\mathcal{C}_k|}{NK} = \frac{\sum_{k=1}^K |\mathcal{M}_k| \cdot K}{NK} = r.$$

This is exactly the case investigated in [3], in which the authors established the fundamental storage-communication tradeoff. The *optimal* tradeoff curve is given by the lower

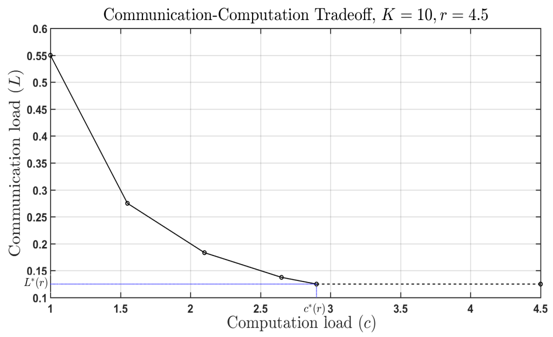

Fig. 1: The communication-computation curve $L(r,c)$ for $K = 10, r = 4.5$. The corner points correspond to different values of g as given in (8) and (10).

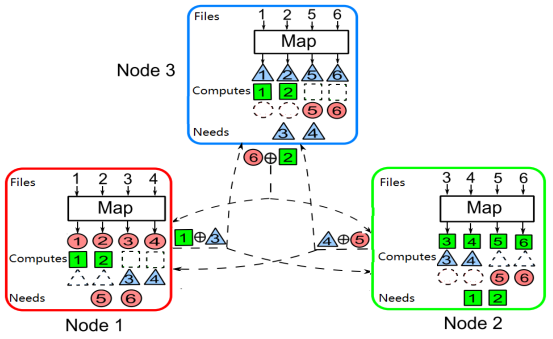

Fig. 2: D3C scheme achieving $L^*(r)$, for a system with $K = 3, r = 2$.

convex envelope of $\{(r, L^*(r)) : r = 1, \cdots, K\}$, where

$$L^*(r) \triangleq \frac{1}{r}\left(1 - \frac{r}{K}\right). \tag{7}$$

In other words, the optimal computation load in (6) is known for the case $c = r$ where $L^*(r,r) = L^*(r)$. Intuitively, however, computing the IVAs for all other nodes may be highly redundant across the whole system. This has motivated our investigation in the more general case where the computation load c can be strictly smaller than the storage space r.

III. Main Result

Theorem 1. *In a distributed computing system with K nodes, for any storage space $r \in [1, K)$, and*

$$c \in \left\{\frac{r}{K} + \left(1 - \frac{r}{K}\right) g \;:\; g = 1, 2, \cdots, \lfloor r \rfloor\right\}, \tag{8}$$

the following communication load $L(r,c)$ is achievable:

$$L(r,c) = \frac{1}{c - r/K} \cdot \left(1 - \frac{r}{K}\right)^2. \tag{9}$$

Define

$$c^*(r) \triangleq \frac{r}{K} + \left(1 - \frac{r}{K}\right) \cdot g_r,$$

where

$$g_r \triangleq \lfloor r \rfloor + \frac{(r - \lfloor r \rfloor)(K - \lceil r \rceil)}{K - r}. \tag{10}$$

For $c^*(r) \le c \le r$, $L^*(r)$ in (7) is achievable.

For general $c \in [1, c^*(r)]$, the lower convex envelope of (8)–(9) and the point $\big(c^*(r), L^*(r)\big)$ is achievable, where $L^*(r)$ is given by (7).

Notice that for any fixed $r \in [1, K)$, the achievable communication load $L(r,c)$ is a piecewise linear function over $c \in [0, r]$. Fig. 1 illustrates the case $K = 10$, $r = 4.5$. When c increases from 1 to $c^*(r)$, the communication load decreases from $1 - \frac{r}{K}$ to $L^*(r)$. The communication load becomes constant when $c \ge c^*(r)$. When $c \in [c^*(r), r]$, from Theorem 1, we have $L^*(r,c) \le L^*(r) = L^*(r,r)$, and thus

$$L^*(r,r) \ge L^*(r,c) \ge L^*(r,r),$$

where the last inequality holds because $L^*(r,c)$ is (non-strictly) decreasing respect to c. Hence, we establish the following optimality.

Corollary 1. *In a distributed computing system with K nodes and storage space $r \in [1, K)$, for $c \in [c^*(r), r]$,*

$$L^*(r,c) = L^*(r).$$

Thus, for fixed $r \in [1, K)$, to achieve the optimal communication load $L^*(r)$, computation load $c^*(r)$ suffices. This computation load is significantly lower than when each node computes IVAs for all output functions from all its stored files (Remark 1). The following example illustrates this saving.

Example 1. Let $K = 2$, $r = 2$, and $N = 6$. Consider Fig. 2. The top-most line in each of the three boxes indicates the files stored at the node. Below this line, the computed IVAs are depicted, where red circles indicate IVAs $\{v_{1,1}, \ldots, v_{1,6}\}$, green squares IVAs $\{v_{2,1}, \ldots, v_{2,6}\}$, and blue triangles IVAs $\{v_{3,1}, \ldots, v_{3,6}\}$. Nodes 1, 2 and 3 need to collect the IVAs denoted by the circles, square and triangles respectively. The last line of each box indicates the IVAs that the node needs to learn during the shuffling phase.

The dashed circles/squares/triangles stand for the IVAs that would be computed in addition in the CDC scheme, see [3, Example 1]. In other words, they represent the saving in computation load. In fact for this example, D3C and CDC have computation loads $\frac{4}{3}$ and 2, respectively.

To visualize savings in computation load, we plot $c^*(r)$ as a function of the storage space r as well as the line $c = r$ in Fig. 3. It implies that when the storage space exceeds a certain threshold, the necessary computation load is decreasing in the storage space. This is because with larger storage space, more IVAs can be locally computed by the nodes requesting them, thus each node computes less IVAs for coded communication, which consumes more computation.

We will prove Theorem 1 by proposing a distributed computing scheme, termed *ditributed computing and coded communication* (D3C) scheme. For $c = c^*(r)$, the shuffle phase of D3C scheme specializes to the shuffle phase of the CDC scheme. But the map phase of our D3C scheme only computes those IVAs that are necessary for communication and decoding. This special case has been investigated in [8], and similar conclusions are obtained there.

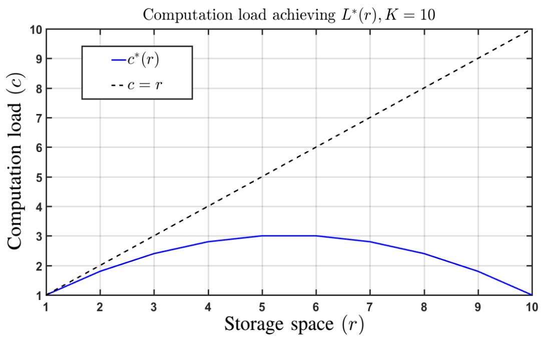

Fig. 3: The computation load achieving $L^*(r)$.

IV. Distributed Computing and Coded Communication Scheme

In this section, we describe the proposed scheme that achieves the result in Theorem 1. Define

$$g = \frac{c - r/K}{1 - r/K}.$$

We will first present the scheme for the case $r \in \{1, \cdots, K\}$, and $g \in \{1, \cdots, r\}$, which will be referred to as the basic D3C scheme hereafter. Then we will extend the basic D3C scheme to obtain the achievability for other cases.

For the case $r \in \{1, \cdots, K\}, g \in \{1, \cdots, r\}$, the N input files are partitioned into $\binom{K}{r}\binom{r}{g}$ batches, each containing

$$\eta_g := \frac{N}{\binom{K}{r}\binom{r}{g}} \tag{11}$$

files. Associate each batch with an element $(\mathcal{S}, \mathcal{T})$ of the set

$$\mathbf{\Omega} := \big\{(\mathcal{S}, \mathcal{T}) \colon \ \mathcal{T} \subset \mathcal{S} \subset [K], \ \ |\mathcal{S}| = r, \ \ |\mathcal{T}| = g\big\},$$

and let $\mathcal{W}_{\mathcal{S},\mathcal{T}}$ denote the batch of η_g files associated to a pair $(\mathcal{S}, \mathcal{T})$. Notice that

$$\mathcal{W} := \{w_1, \cdots, w_N\} = \bigcup_{(\mathcal{S},\mathcal{T}) \in \mathbf{\Omega}} \mathcal{W}_{\mathcal{S},\mathcal{T}}.$$

Let further $\mathcal{U}_{k,\mathcal{S},\mathcal{T}}$ be the set of IVAs for output function ϕ_k that can be computed from the files in $\mathcal{W}_{\mathcal{S},\mathcal{T}}$:

$$\mathcal{U}_{k,\mathcal{S},\mathcal{T}} \triangleq \{v_{k,n} \colon n \in [N] \text{ so that } w_n \in \mathcal{W}_{\mathcal{S},\mathcal{T}}\}.$$

We now describe the three distributed-computing phases.

1) *Map phase:* Node k stores the files in $\mathcal{W}_{\mathcal{S},\mathcal{T}}$ if and only if $k \in \mathcal{S}$. Thus:

$$\mathcal{M}_k = \bigcup_{\substack{(\mathcal{S},\mathcal{T}) \in \mathbf{\Omega}\colon \\ k \in \mathcal{S}}} \mathcal{W}_{\mathcal{S},\mathcal{T}}.$$

Node k further computes the IVAs in

$$\mathcal{C}_k = \mathcal{C}_k^1 \cup \mathcal{C}_k^2, \tag{12}$$

where

$$\mathcal{C}_k^1 := \bigcup_{\substack{(\mathcal{S},\mathcal{T}) \in \mathbf{\Omega}\colon \\ k \in \mathcal{S}}} \mathcal{U}_{k,\mathcal{S},\mathcal{T}}, \tag{13}$$

$$\mathcal{C}_k^2 := \bigcup_{\substack{(\mathcal{S},\mathcal{T}) \in \mathbf{\Omega}\colon \\ k \in \mathcal{T}}} \ \bigcup_{q \in [K] \backslash \mathcal{S}} \mathcal{U}_{q,\mathcal{S},\mathcal{T}}. \tag{14}$$

It's easy to verify that all the IVAs in $\mathcal{C}_k$ can be computed from $\mathcal{M}_k$, i.e., the files stored at Node k.

2) *Shuffling Phase:* For each element $(\mathcal{S}, \mathcal{T}) \in \mathbf{\Omega}$ and each index $j \in [K] \backslash \mathcal{S}$, we partition the set $\mathcal{U}_{j,\mathcal{S},\mathcal{T}}$ into g equal smaller subsets

$$\mathcal{U}_{j,\mathcal{S},\mathcal{T}} = \big\{\mathcal{U}^k_{j,\mathcal{S},\mathcal{T}} \colon \ k \in \mathcal{T}\big\}.$$

Define now the set

$$\mathbf{\Pi} := \big\{(\mathcal{I}, \mathcal{J}) \colon \ \mathcal{J} \subset \mathcal{I} \subset [K], \ \ |\mathcal{I}| = r+1, \ \ |\mathcal{J}| = g+1\big\}.$$

For each $(\mathcal{I}, \mathcal{J}) \in \mathbf{\Pi}$ and $k \in \mathcal{J}$, by (14), Node k can compute the signal

$$X^k_{\mathcal{I},\mathcal{J}} := \bigoplus_{i \in \mathcal{J} \backslash \{k\}} \mathcal{U}^k_{i,\mathcal{I}\backslash\{i\},\mathcal{J}\backslash\{i\}}$$

from the IVAs calculated during the map phase. A given Node k thus sends (shuffles) the multicast signal

$$X_k = \big\{X^k_{\mathcal{I},\mathcal{J}} \colon \ (\mathcal{I}, \mathcal{J}) \in \mathbf{\Pi} \text{ such that } k \in \mathcal{J}\big\}.$$

3) *Reduce Phase:* Notice that $\mathcal{C}_k^2$ only contains IVAs $v_{q,n}$ where $q \neq k$. Thus, by (12)–(13), during the shuffling phase each node k needs to learn all IVAs in

$$\bigcup_{\substack{(\mathcal{S},\mathcal{T}) \in \mathbf{\Omega}\colon \\ k \notin \mathcal{S}}} \mathcal{U}_{k,\mathcal{S},\mathcal{T}}.$$

Fix an arbitrary pair $(\mathcal{S}, \mathcal{T}) \in \mathbf{\Omega}$ such that $k \notin \mathcal{S}$ and an element $j \in \mathcal{T}$. From the received multicast message $X^j_{\mathcal{S}\cup\{k\},\mathcal{T}\cup\{k\}}$ during the shuffling phase and its locally calculated IVAs during the map phase

$$\Big\{\mathcal{U}^j_{i,\mathcal{S}\cup\{k\}\backslash\{i\},\mathcal{T}\cup\{k\}\backslash\{i\}} \colon \ \ i \in \mathcal{T}\backslash\{j\}\Big\},$$

Node k can recover the missing IVAs $\mathcal{U}^j_{k,\mathcal{S},\mathcal{T}}$ through a simple XOR operation. After collecting all missing IVAs, Node k can proceed to compute the reduce function (5).

4) *Analysis:* We analyze the performance of the scheme.

1) *Storage Space:* The number of batches in $\mathcal{M}_k$ is $\binom{K-1}{r-1}\binom{r}{g}$, each consisting of η_g files. Thus, the storage space is

$$\frac{1}{N} \cdot K \cdot \binom{K-1}{r-1}\binom{r}{g} \cdot \eta_g = r.$$

2) *Computation Load:* Since $\mathcal{C}_k^1 \cap \mathcal{C}_k^2 = \emptyset$:

$$|\mathcal{C}_k| = |\mathcal{C}_k^1| + |\mathcal{C}_k^2|. \tag{15}$$

By (11), (13), (14), we have

$$|\mathcal{C}_k^1| = \binom{K-1}{r-1}\binom{r}{g} \cdot \eta_g = \frac{rN}{K}, \tag{16}$$

$$\begin{aligned} |\mathcal{C}_k^2| &= \binom{K-1}{r-1}\binom{r-1}{g-1} \cdot (K-r) \cdot \eta_g \\ &= \left(1 - \frac{r}{K}\right) \cdot g \cdot N. \end{aligned} \tag{17}$$

Thus, by (15), (16) and (17), the computation load is

$$c = \frac{\sum_{k=1}^{K} |\mathcal{C}_k|}{NK} = \frac{r}{K} + \left(1 - \frac{r}{K}\right) g. \qquad (18)$$

3) *Communication Load:* The number of signals that each Node k transmits is $\binom{K-1}{r} \cdot \binom{r}{g}$, each of size $\frac{\eta_g \cdot T}{g}$ bits. Thus, the communication load is

$$\begin{aligned} L &= \frac{\sum_{k=1}^{K} |X_k|}{NKT} \\ &= \frac{1}{NKT} \cdot K \binom{K-1}{r} \binom{r}{g} \frac{NT}{\binom{K}{r}\binom{r}{g}} \cdot \frac{1}{g} \\ &= \frac{1}{g} \cdot \left(1 - \frac{r}{K}\right) \\ &\overset{(a)}{=} \frac{1}{c - r/K} \cdot \left(1 - \frac{r}{K}\right)^2, \end{aligned}$$

where (a) follows from (18).

This concludes the achievability proof for $r \in \{1, \cdots, K\}$, $g \in \{1, \cdots, r\}$. We complete the proof by proposing several extensions of the basic D3C scheme:

E1. For any $r \in [1, K)$, and $g \in \{1, \cdots, \lfloor r \rfloor\}$, there exist an unique $\alpha \in [0, 1)$ such that

$$r = (1 - \alpha)\lfloor r \rfloor + \alpha \lceil r \rceil.$$

We partition the N files into two groups with cardinalities[1] $(1 - \alpha)N$ and αN respectively. For the first group, implement the basic D3C scheme with a storage space $\lfloor r \rfloor$, and computing load $c_1 \triangleq \frac{\lfloor r \rfloor}{K} + \left(1 - \frac{\lfloor r \rfloor}{K}\right) g$. For the second group, implement the basic D3C scheme with a storage space $\lceil r \rceil$, and computing load $c_2 \triangleq \frac{\lceil r \rceil}{K} + \left(1 - \frac{\lceil r \rceil}{K}\right) g$. This results the achievability of (9), when $r \in [1, K)$, $g \in \{1, \cdots, \lfloor r \rfloor\}$.

E2. For any $r \in [1, K)$ and for $g \in \big[1, \lfloor r \rfloor\big)$, there exists an unique $\beta \in [0, 1)$ such that

$$g = (1 - \beta)\lfloor g \rfloor + \beta \lceil g \rceil.$$

We partition the N files into two groups with cardinalities $(1 - \beta)N$ and βN respectively. For the first group, implement the scheme in E1 with a storage space r, and computing load $c_1' \triangleq \frac{r}{K} + \left(1 - \frac{r}{K}\right) \lfloor g \rfloor$. For the second group, implement the scheme in E1 with a storage space r, and computation load $c_2' \triangleq \frac{r}{K} + \left(1 - \frac{r}{K}\right) \lceil g \rceil$. Notice that, β satisfies $c = (1 - \beta)c_1' + \beta c_2'$. Thus, this results the achievability of the lower convex envelop in (9) of Theorem 1.

E3. When $r \notin \mathbb{N}$, $g \in [\lfloor r \rfloor, g_r]$, then there exists an unique $\theta \in (0, r - \lfloor r \rfloor]$ such that

$$c = \frac{r}{K} + \left(1 - \frac{r}{K}\right) \cdot \left(\lfloor r \rfloor + \theta \cdot \frac{K - \lceil r \rceil}{K - r}\right).$$

Define $r'(\theta) \triangleq r - \frac{\theta}{1 - \theta} \cdot (\lceil r \rceil - r)$, or equivalently,

$$r = (1 - \theta) \cdot r'(\theta) + \theta \cdot \lceil r \rceil.$$

Notice that $r'(\theta)$ decreases with θ, so

$$r'(\theta) \geq r'(\theta)|_{\theta = r - \lfloor r \rfloor} = \lfloor r \rfloor.$$

We partition the N files into two groups with cardinalities $(1 - \theta)N$ and θN respectively. For the first group, implement the scheme in E1 with a storage space $r'(\theta)$, and computation load $\frac{r'(\theta)}{K} + \left(1 - \frac{r'(\theta)}{K}\right) \lfloor r \rfloor$. For the second group, implement the basic D3C scheme with a storage space $\lceil r \rceil$, and computation load $\frac{\lceil r \rceil}{K} + \left(1 - \frac{\lceil r \rceil}{K}\right) \lceil r \rceil$. This results the achievability when $r \notin \mathbb{N}$, $\lfloor r \rfloor < g \leq g_r$.

Finally, when $g > g_r$, the load $L^*(r)$ can be achieved with the scheme at $g = g_r$.

V. Conclusion

We proposed a new scheme (the D3C scheme) for distributed computing. It leverages a flexible tradeoff between the three elementary resources *storage space*, *computation load*, and *communication load*. In fact, the D3C scheme achieves the best communication load for any feasible storage space and computation load, since a matching converse has been established in a following-up work [9].

Acknowledgment

The work of Q. Yan and M. Wigger has been supported by the ERC Grant *CTO Com*.

References

[1] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Sixth USENIX OSDI*, Dec. 2004.

[2] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly, "Dryad: distributed data-parallel programs from sequential building blocks," *in Proceedings of the 2nd ACM SIGOPS/EuroSys'07*, Mar. 2007.

[3] 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.

[4] 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.

[5] S. Li, M. A. Maddah-Ali, and A. S. Avestimehr, "A unified coding framework for distributed computing with straggling servers," in *Proc. IEEE Globecom Works (GC Wkshps),* Washington, DC, USA, pp. 1–6, 2016.

[6] Q. Yu, S. Li, M. A. Maddah-Ali, and A. S. Avestimehr, "How to optimally allocate resources for coded distributed computing," in *Proc. IEEE Internation Conference on Communications (ICC),* Paris, France, 21–25, May. 2017.

[7] S. Li, Q. Yu, M. A. Maddah-Ali, A. S. Avestimehr,"A scalable framework for wireless distributed computing," *IEEE/ACM Trans. Netw.,*vol. 25, no. 5, pp. 2643–2653, Oct. 2017.

[8] Y. H. Ezzeldin, M. Karmoose, and C. Fragouli, "Communication vs distributed computation: an alternative trade-off curve," in *Proc. IEEE Information Theory Workshop (ITW),* Kaohsiung, Taiwan, pp. 279–283, Nov. 2017.

[9] Q. Yan, S. Yang, and M. Wigger, "Storage, computation, and communication: a fundamental tradeoff in distributed computing," arXiv:1806.07565.

[1]While αN and $(1 - \alpha)N$ being integers requires that, α has to be a rational number, irrational α can be approached arbitrarily close by a series of rational numbers, since the rational numbers are dense in the real interval $[0, 1]$. As Definition 4 allows a ϵ-discrepancy, this does not affect the proof.