Storage, Computation, and Communication: A Fundamental Tradeoff in 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*—We consider a MapReduce-like distributed computing system. We derive a lower bound on the communication cost for any given storage and computation costs. This lower bound matches the achievable bound we proposed recently. As a result, we completely characterize the optimal tradeoff between the storage, the computation, and the communication. Our result generalizes the previous one by Li *et al.* to also account for the number of computed intermediate values.**

I. Introduction

Systems like MapReduce [1], Dryad [2] *etc.* have become popular platforms for distributed computing to perform data-parallel computations across distributed computing nodes. In such systems, the computations are typically decomposed into "Map" and "Reduce" functions as detailed in the following. Consider the task of computing K output functions of the form

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

Here, each output function ϕ_k depends on all N data blocks $w_1,\cdots,w_N$, but can be decomposed into:

- N *map functions* $g_{k,1},\cdots,g_{k,N}$, each only depending on one block;
- a *reduce function* h_k that combines the outcomes of the N map functions.

Computation of such functions can be performed in a distributed way following 3-phases: In the first *map phase*, each node locally stores a subset of the input data $\mathcal{M}_k \subseteq \{w_1,\cdots,w_N\}$, and calculates all *intermediate values* (IVAs) that depend on the stored data:

$$\{g_{l,n}(w_n): l\in\{1,\cdots,K\},\ w_n\in\mathcal{M}_k\}.$$

In the subsequent *shuffle phase*, the nodes exchange the IVAs computed during the map phase, so that each node k is aware of all the IVAs $g_{k,1}(w_1),\cdots,g_{k,N}(w_N)$ required to calculate its own output function ϕ_k. In the final *reduce phase*, each node k combines the IVAs with the reduce function h_k as indicated in (1).

Li *et al.* [3] proposed a scheme, termed *coded distributed computing* (CDC), that in the map phase stores files multiple times across users so as to enable multicast opportunities for the shuffle phase. This approach can significantly reduce the communication load over traditional schemes, and was proved in [3] to have the smallest communication load among all the distributed computing schemes with same total storage requirements. Some extensions have been made in follow-up works. For example, straggling nodes were investigated in [4]; [5] studied optimal allocation of computation resources; [6] considered distributed nodes in a wireless network.

It is worth mentioning that Li *et al.* in [3] used the term *computation-communication* tradeoff, because they assumed 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 actually a measure of the total storage space consumed across the nodes. This is why we would rather refer to it as the *storage-communication* tradeoff.

Naturally, if an IVA is not used subsequently, there is no need to compute it, which can save computation resources (e.g., power) and shorten calculation latency. Therefore, it is natural to investigate a more general framework, where each node is allowed to choose to calculate or not the IVA for each output function from the data stored locally. The number of IVAs that each node needs to calculate normalized by the total number of IVAs is then used to measure the real computation load. In this sense, we extend the storage-communication tradeoff in [3] to a *storage-computation-communication* tradeoff. In particular, we wish to characterize the smallest communication load required in the shuffle phase for a given storage space and a given number of IVAs calculated during the map phase. Ezzeldin *et al.* proposed a modification on the CDC scheme in [7], that compute IVAs only if they are used subsequently. Recently, we also proposed a new scheme named *distributed computing and coded communication (D3C)* [8], and derived the tradeoff achieved by this scheme. In this paper, we provide a matching converse, and thereby characterize completely the optimal storage-computation-communication tradeoff.

Notations: Let $\mathbb{N}^+$ denote the set of positive integers, and for $m,n\in\mathbb{N}^+$, let $\mathbb{F}_{2^m}^n$ denote the n-dimensional vector space over the finite field $\mathbb{F}_{2^m}$. We also abbreviate $\{1,\cdots,n\}$ by $[n]$. For scalar quantities we use (upper or lower case) standard font, for sets calligraphic font, and for collections (sets of sets) bold font. The cardinality of a set $\mathcal{A}$ is denoted $|\mathcal{A}|$. The indicator function of an event is written as $\mathbb{I}(\cdot)$.

II. System Model

Consider a system with K distributed computing nodes and N files. Specifically, given any N files

$$\mathcal{W} = \{w_1, \cdots, w_N\}, \quad w_i \in \mathbb{F}_{2^F}, \forall\ i \in [N].$$

Node k ($k \in [K]$) wishes 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, where $B \in \mathbb{N}^+$.

Following the MapReduce framework [3], [6], we assume that the computation of the output functions ϕ_k can be decomposed as in (1), where

- 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 length T, i.e., $v_{k,n} \triangleq g_{k,n}(w_n) \in \mathbb{F}_{2^T}$, where $T \in \mathbb{N}$.
- The "Reduce" function

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

maps the intermediate values

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

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

The computations are carried out in three phases.

1) **Map Phase:** Each node k stores a subset of files $\mathcal{M}_k \subseteq \mathcal{W}$, $k \in [K]$, and then for each file $w_n \in \mathcal{M}_k$, computes a subset of IVAs $\mathcal{C}_{k,n} = \{v_{q,n} : q \in \Lambda_{k,n}\}$, where $\Lambda_{k,n} \subseteq [K]$. Denote the set of IVAs computed at node k by $\mathcal{C}_k$, i.e.,

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

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 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 c, as the total number of map functions computed across the K nodes, normalized by the total number of map functions NK, i.e.,*

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

2) **Shuffle Phase:** To compute the output function ϕ_k, node k needs to collect the IVAs of ϕ_k that are not computed locally in the map phase, i.e., $\mathcal{V}_k \backslash \mathcal{C}_k$. After the map phase, the K nodes exchange the computed IVAs. Particularly, each node k creates and multicasts 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, namely,

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

to all the other nodes for some encoding function

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

All the nodes receive the signals $X_1, \cdots, X_K$ error-free.

Definition 3 (Communication Load)**.** *We define the communication load L, as the total number 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 $\{X_i\}_{i=1}^K$ exchanged during the shuffle phase and the IVAs $\mathcal{C}_k$ computed locally in map phase, node k restores all the IVAs in $\mathcal{V}_k$, i.e.,

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

with the function

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

Finally, it proceeds to compute

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

Definition 4. *A distributed computing system is said to achieve 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 load, and communication load do not exceed $r + \epsilon$, $c + \epsilon$, and $L + \epsilon$, respectively. In particular, we define the optimal communication load by*

$$L^*(r, c) \triangleq \inf\left\{L : (r, c, L)\ \textit{is achievable}\right\}.$$

Without loss of generality (W.L.O.G), we assume $1 \leq c \leq r < K$. In fact, $|\mathcal{C}_k| \leq |\mathcal{M}_k|K$ is implied by (2), and thus $c \leq r$ by (3) and (4). Moreover, since each IVA needs to be computed at least once somewhere, we have $c \geq 1$. Furthermore, if $r \geq K$, each node trivially stores all the files and locally computes all the IVAs required for its output function.

III. Main Result

Define

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

where

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

Notice that, $L^*(r)$ is the optimal *storage-communication* trade-off derived in [3].

Theorem 1. *For any storage space $r \in [1, K)$, and*

$$c \in \left\{\frac{r}{K} + \left(1 - \frac{r}{K}\right) g\ :\ g = 1, \cdots, \lfloor r \rfloor\right\}, \tag{5}$$

the optimal communication load $L^*(r,c)$ is given by

$$L^*(r,c) = \frac{1}{c - r/K} \cdot \left(1 - \frac{r}{K}\right)^2. \tag{6}$$

For general $1 \le c \le c^*(r)$, the optimal communication load $L^*(r,c)$ is given by the lower convex envelope of the points in (5) *and* (6) *and the point $(c^*(r), L^*(r))$. Moreover,*

$$L^*(r,c) = L^*(r), \quad c^*(r) \le c \le r. \tag{7}$$

Proof: The tradeoff in Theorem 1 is achieved by the D3C scheme, see [8]. Equality (7) has been shown in [8, Corollary 1]. The converse for the case $0 \le c \le c^*(r)$ is proved in Section IV. ■

Notice that, $L^*(r,c)$ is piecewise linear in (r,c). In the *storage-computation-communication (r-c-L)* space, where the coordinates are associated with r, c, and L, respectively, Fig. 1 illustrates the surface $L^*(r,c)$ characterized by Theorem 1 when $K = 10$. In particular,

1) The line
$$\left(r, 1, 1 - \frac{r}{K}\right), \quad r \in [1, K)$$
is the *optimal computation curve (OCP)*, and characterizes the optimal storage-communication tradeoff at the lowest computation load ($c = 1$).
2) The curve
$$(r, c^*(r), L^*(r)), \quad r \in [1, K)$$
is the *optimal communication curve (OCM)*, and characterizes the optimal storage-computation tradeoff at the lowest communication load ($L = L^*(r)$).
3) The pareto-optimal surface is given by the triangles between the OCP and OCM curves.

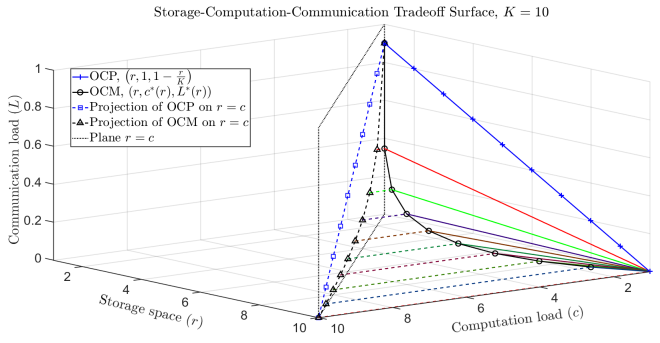

Fig. 1: The storage-computation-communication tradeoff surface for a system with $K = 10$ nodes. The dashed lines are the projections of OCP and OCM to the plane $c = r$.

Remark 1. We briefly sketch the D3C scheme in [8], which achieves the optimal tradeoff in Theorem 1. For integers r, g such that $1 \le g \le r < K$, the files are partitioned into $\binom{K}{r}\binom{r}{g}$ batches. Each batch is associated with a tuple $(\mathcal{S}, \mathcal{T})$ where $\mathcal{T} \subseteq \mathcal{S} \subset \mathcal{K}$, $|\mathcal{S}| = r, |\mathcal{T}| = g$. Let $\mathcal{W}_{\mathcal{S},\mathcal{T}}$ be the batch associated with $(\mathcal{S}, \mathcal{T})$, all nodes in $\mathcal{S}$ store $\mathcal{W}_{\mathcal{S},\mathcal{T}}$, and compute their own IVAs from $\mathcal{W}_{\mathcal{S},\mathcal{T}}$. Only the nodes in $\mathcal{T}$ compute the IVAs from $\mathcal{W}_{\mathcal{S},\mathcal{T}}$ that are needed by the nodes in $\mathcal{K}\backslash\mathcal{S}$. In the shuffle phase, for each pair $(\mathcal{I}, \mathcal{J})$ such that $\mathcal{I} = r + 1, \mathcal{J} = g + 1$, each node k in $\mathcal{J}$ creates a coded multicast signal useful for all nodes in $\mathcal{J}\backslash\{k\}$. Based on the received multicast signals and the IVAs it computed locally, each node can then compute the desired output function in the reduce phase.

When $g = r$, the D3C degrades to the modified CDC (M-CDC) scheme in [7]. The M-CDC scheme achieves the K corner points of the optimal tradeoff surface. Time- and memory-sharing the M-CDC scheme with different parameters can thus achieve all pareto-optimal points on the tradeoff surface, see [8] and [9] for details.

One may observe that, in both the D3C and the M-CDC scheme, the required number of input files increases very fast with the number of nodes. This may prevent implementation in practice. In the longer version of this paper [9], we propose ways to decrease the required number of files via placement delivery arrays [10].

IV. Converse

Fix $r \in [1, K)$, and $c \in [1, r]$. Consider a file allocation $\mathcal{M} = \{\mathcal{M}_k\}_{k=1}^K$ and its feasible IVA sets $\mathcal{C} = \{\mathcal{C}_k\}_{k=1}^K$, so that[1]

$$\frac{\sum_{k=1}^K |\mathcal{M}_k|}{N} \le r, \tag{8}$$

$$\frac{\sum_{k=1}^K |\mathcal{C}_k|}{NK} \le c. \tag{9}$$

For any nonempty set $\mathcal{S} \subseteq [K]$, denote $X_\mathcal{S} \triangleq \{X_k\}_{k\in\mathcal{S}}, \mathcal{V}_\mathcal{S} \triangleq \cup_{k\in\mathcal{S}}\mathcal{V}_k, \mathcal{C}_\mathcal{S} \triangleq \cup_{k\in\mathcal{S}}\mathcal{C}_k$. For any $k \in \mathcal{S}$ and $j \in [|\mathcal{S}| - 1]$, define

$$\mathcal{B}^k_{\mathcal{S},j} \triangleq \{v_{k,n} : v_{k,n} \text{ is only computed by } j \text{ nodes in } \mathcal{S}\backslash\{k\}\}.$$

Let $b^k_{\mathcal{S},j}$ be the cardinality of $\mathcal{B}^k_{\mathcal{S},j}$. Then the cardinality of

$$\mathcal{B}_{\mathcal{S},j} \triangleq \bigcup_{k\in\mathcal{S}} \mathcal{B}^k_{\mathcal{S},j}$$

is given by

$$b_{\mathcal{S},j} \triangleq \sum_{k\in\mathcal{S}} b^k_{\mathcal{S},j}. \tag{10}$$

A. Auxiliary Lemmas

To prove the converse, we need the following two lemmas, where Lemma 1 is proved in Section IV-C.

Lemma 1. *For any nonempty set $\mathcal{S} \subseteq [K]$,*

$$H(X_\mathcal{S}|\mathcal{V}_{\mathcal{S}^c}, \mathcal{C}_{\mathcal{S}^c}) \ge T \sum_{j=1}^{|\mathcal{S}|-1} b_{\mathcal{S},j} \cdot \frac{1}{j}, \tag{11}$$

where $\mathcal{S}^c \triangleq [K]\backslash\mathcal{S}$.

[1]As $\epsilon > 0$ can be arbitrarily close to 0 in Definition 4, to derive the lower bound for $L^*(r,c)$, we need to consider the case $\epsilon \to 0$.

Lemma 2. *Consider set $\mathcal{S} = [K]$ and define $b_j \triangleq b_{[K],j}$. Then,*

$$\sum_{j=1}^{K-1} b_j \geq N(K-r), \tag{12}$$

$$\sum_{j=1}^{K-1} (j-1) b_j \leq (c-1)NK. \tag{13}$$

Proof: For any $k \in [K]$, define

$$\mathcal{A}_k \triangleq \{v_{k,n} : v_{k,n} \text{ is computed by node } k, n \in [N]\}.$$

Set $a_k = |\mathcal{A}_k|$. Notice that

$$\mathcal{A}_1, \cdots, \mathcal{A}_K, \mathcal{B}_{[K],1}, \cdots, \mathcal{B}_{[K],K-1}$$

form a partition of all IVAs, and therefore

$$\sum_{k=1}^{K} a_k + \sum_{j=1}^{K-1} b_j = NK. \tag{14}$$

Moreover, since node k must store w_n if it has computed $v_{k,n}$, it must hold that $a_k \leq |\mathcal{M}_k|$, and thus by (8),

$$\sum_{k=1}^{K} a_k \leq \sum_{k=1}^{K} |\mathcal{M}_k| \leq rN. \tag{15}$$

Finally, for each $k \in [K]$, $j \in [K-1]$, the IVAs in $\mathcal{A}_k$ must be computed at node k and IVAs $\mathcal{B}_{[K],j}$ must be computed at j nodes, and by (9),

$$\sum_{k=1}^{K} a_k + \sum_{j=1}^{K-1} j b_j \leq \sum_{k=1}^{K} |\mathcal{C}_k| \leq cNK. \tag{16}$$

From (14)–(16), we obtain (12) and (13). ∎

B. Proof of the Converse to Theorem 1

For each $c \in [1, r]$, define

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

Notice that $g \geq 1$ since we assume $c \geq 1$. Let $g_1 \triangleq \lfloor g \rfloor$, $g_2 \triangleq \lceil g \rceil$, and

$$c_1 = \frac{r}{K} + \left(1 - \frac{r}{K}\right) g_1, \tag{17}$$

$$c_2 = \frac{r}{K} + \left(1 - \frac{r}{K}\right) g_2. \tag{18}$$

Notice that by these definitions,

$$c_1 \leq c \leq c_2. \tag{19}$$

Choose $\lambda, \mu \in \mathbb{R}$ so that

$$\lambda x + \mu|_{x=c_1} = \frac{1}{c_1 - r/K} \cdot \left(1 - \frac{r}{K}\right)^2, \tag{20}$$

$$\lambda x + \mu|_{x=c_2} = \frac{1}{c_2 - r/K} \cdot \left(1 - \frac{r}{K}\right)^2. \tag{21}$$

Then from (17)–(21), and the fact $g_2 - g_1 = 1$, we conclude that λ and μ satisfy:

$$\lambda = \frac{1}{g_2} - \frac{1}{g_1} < 0, \tag{22}$$

$$\mu = \frac{c_2}{g_1} - \frac{c_1}{g_2} > 0,$$

$$\lambda + \mu = \frac{c_2 - 1}{g_1} - \frac{c_1 - 1}{g_2} > 0. \tag{23}$$

By the convexity of the function $\frac{1}{x - r/K}\left(1 - \frac{r}{K}\right)^2$ over $x \in [1, +\infty)$, we then obtain:

$$\frac{1}{x - r/K}\left(1 - \frac{r}{K}\right)^2 \geq \lambda x + \mu,$$
$$\forall\, x \in \left\{\frac{r}{K} + \left(1 - \frac{r}{K}\right) g : g = 1, \cdots, K-1\right\}.$$

Therefore,

$$\begin{aligned}
L &\geq \frac{H\left(X_{[K]}\right)}{NKT} \\
&\geq \sum_{j=1}^{K-1} \frac{b_j}{NK} \cdot \frac{1}{j} \\
&\geq \frac{1}{N(K-r)} \sum_{j=1}^{K-1} \frac{b_j}{\left(1 - \frac{r}{K}\right) j + \frac{r}{K} - \frac{r}{K}} \left(1 - \frac{r}{K}\right)^2 \\
&\geq \frac{1}{N(K-r)} \sum_{j=1}^{K-1} b_j \left(\lambda \left(\left(1 - \frac{r}{K}\right) j + \frac{r}{K}\right) + \mu\right) \\
&= \frac{\lambda}{NK} \cdot \sum_{j=1}^{K-1} (j-1) b_j + \frac{\lambda + \mu}{N(K-r)} \cdot \sum_{j=1}^{K-1} b_j \\
&\overset{(a)}{\geq} \frac{\lambda}{NK} \cdot (c-1)NK + \frac{\lambda + \mu}{N(K-r)} \cdot N(K-r) \\
&= \lambda c + \mu,
\end{aligned} \tag{24}$$

where (a) follows from (12), (13), (22) and (23). This implies that for any storage space $r \in [1, K)$ and computation load $c \in [c_1, c_2]$, the optimal communication load $L^*(r, c)$ is lower bounded by the lower convex envelope of $(c_1, L^*(r, c_1))$ and $(c_2, L^*(r, c_2))$. Noting that also the point $(c^*(r), L^*(r))$ is on the line (24) concludes the converse proof.

C. Proof of Lemma 1

For notational brevity, we denote the tuple $(\mathcal{V}_\mathcal{S}, \mathcal{C}_\mathcal{S})$ by $Y_\mathcal{S}$ for any $\mathcal{S} \subseteq [K]$. We prove Lemma 1 by mathematical induction on the size of $\mathcal{S}$:

When $|\mathcal{S}| = 1$, without loss of generality, assume $\mathcal{S} = \{k\}$, then (11) becomes $H\left(X_k | Y_{[K]\setminus\{k\}}\right) \geq 0$, which is trivial.

Suppose that, the statement is true for all subsets of $[K]$ with size s, $1 \leq s < K$. Consider a set $\mathcal{S} \subseteq [K]$ such that $|\mathcal{S}| = s + 1$. Then

$$\begin{aligned}
&H(X_\mathcal{S} | Y_{\mathcal{S}^c}) \\
&\quad = \frac{1}{|\mathcal{S}|} \sum_{k \in \mathcal{S}} H(X_\mathcal{S}, X_k | Y_{\mathcal{S}^c}) \\
&\quad = \frac{1}{|\mathcal{S}|} \sum_{k \in \mathcal{S}} \left(H(X_k | Y_{\mathcal{S}^c}) + H(X_\mathcal{S} | X_k, Y_{\mathcal{S}^c})\right)
\end{aligned}$$

$$\geq \frac{1}{|\mathcal{S}|}H(X_{\mathcal{S}}|Y_{\mathcal{S}^c}) + \frac{1}{|\mathcal{S}|}\sum_{k\in\mathcal{S}} H(X_{\mathcal{S}}|X_k, Y_{\mathcal{S}^c}). \quad (25)$$

Then from (25), we have

$$\begin{aligned}
&H(X_{\mathcal{S}}|Y_{\mathcal{S}^c})\\
&\geq \frac{1}{|\mathcal{S}|-1}\sum_{k\in\mathcal{S}} H(X_{\mathcal{S}}|X_k, Y_{\mathcal{S}^c})\\
&\geq \frac{1}{s}\sum_{k\in\mathcal{S}} H(X_{\mathcal{S}}|X_k, \mathcal{C}_k, Y_{\mathcal{S}^c})\\
&\overset{(a)}{=} \frac{1}{s}\sum_{k\in\mathcal{S}} H(X_{\mathcal{S}}|\mathcal{C}_k, Y_{\mathcal{S}^c})\\
&\overset{(b)}{=} \frac{1}{s}\sum_{k\in\mathcal{S}} \big(H(X_{\mathcal{S}}|\mathcal{C}_k, Y_{\mathcal{S}^c}) + H(\mathcal{V}_k|X_{\mathcal{S}}, \mathcal{C}_k, Y_{\mathcal{S}^c})\big)\\
&\overset{(c)}{=} \frac{1}{s}\sum_{k\in\mathcal{S}} H(X_{\mathcal{S}}, \mathcal{V}_k|\mathcal{C}_k, Y_{\mathcal{S}^c})\\
&\overset{(d)}{=} \frac{1}{s}\sum_{k\in\mathcal{S}} (H(\mathcal{V}_k|\mathcal{C}_k, Y_{\mathcal{S}^c}) + H(X_{\mathcal{S}}|\mathcal{V}_k, \mathcal{C}_k, Y_{\mathcal{S}^c}))\\
&\overset{(e)}{=} \frac{1}{s}\sum_{k\in\mathcal{S}} \big(H(\mathcal{V}_k|\mathcal{C}_{(\mathcal{S}\backslash\{k\})^c}) + H(X_{\mathcal{S}\backslash\{k\}}|Y_{(\mathcal{S}\backslash\{k\})^c})\big)\\
&\overset{(f)}{\geq} \frac{T}{s}\sum_{k\in\mathcal{S}}\sum_{j=1}^{s} b^k_{\mathcal{S},j} + \frac{T}{s}\sum_{k\in\mathcal{S}}\sum_{j=1}^{s-1} b_{\mathcal{S}\backslash\{k\},j}\cdot\frac{1}{j}\\
&= \frac{T}{s}\sum_{j=1}^{s}\sum_{k\in\mathcal{S}} b^k_{\mathcal{S},j} + \frac{T}{s}\sum_{j=1}^{s-1}\sum_{k\in\mathcal{S}} b_{\mathcal{S}\backslash\{k\},j}\cdot\frac{1}{j}\\
&\overset{(g)}{=} \frac{T}{s}\sum_{j=1}^{s} b_{\mathcal{S},j} + \frac{T}{s}\sum_{j=1}^{s-1}\sum_{k\in\mathcal{S}}\sum_{l\in\mathcal{S}\backslash\{k\}} b^l_{\mathcal{S}\backslash\{k\},j}\cdot\frac{1}{j}\\
&= \frac{T}{s}\sum_{j=1}^{s} b_{\mathcal{S},j} + \frac{T}{s}\sum_{j=1}^{s-1}\sum_{l\in\mathcal{S}}\sum_{k\in\mathcal{S}\backslash\{l\}} b^l_{\mathcal{S}\backslash\{k\},j}\cdot\frac{1}{j}, \quad (26)
\end{aligned}$$

where (a) holds because X_k is a function of $\mathcal{C}_k$; (b) holds because by $H(\mathcal{V}_k|X_{\mathcal{S}}, \mathcal{C}_k, Y_{\mathcal{S}^c}) = 0$, since $\mathcal{V}_k$ can be decoded using $\mathcal{C}_k, X_{\mathcal{S}}$ and $X_{\mathcal{S}^c}$, which is a function of $Y_{\mathcal{S}^c}$; (c) and (d) follow from the chain rule; (e) holds because $Y_{\mathcal{S}^c} = (\mathcal{V}_{\mathcal{S}^c}, \mathcal{C}_{\mathcal{S}^c})$ and by the independence between $\mathcal{V}_k$ and $\mathcal{V}_{\mathcal{S}^c}$; (f) holds by the definition of $b^k_{\mathcal{S},j}$ and the induction assumption; and (g) holds by (10).

Notice that, in (26),

$$\begin{aligned}
&\sum_{k\in\mathcal{S}\backslash\{l\}} b^l_{\mathcal{S}\backslash\{k\},j}\\
&\overset{(a)}{=} \sum_{k\in\mathcal{S}\backslash\{l\}}\sum_{n=1}^{N} \mathbb{I}(v_{l,n} \text{ is only computed by } j \text{ nodes in } \mathcal{S}\backslash\{l\}) \cdot \mathbb{I}(v_{l,n} \text{ is not computed by node } k)\\
&= \sum_{n=1}^{N} \mathbb{I}(v_{l,n} \text{ is only computed by } j \text{ nodes in } \mathcal{S}\backslash\{l\}) \cdot \sum_{k\in\mathcal{S}\backslash\{l\}} \mathbb{I}(v_{l,n} \text{ is not computed by node } k)\\
&= \sum_{n=1}^{N} \mathbb{I}(v_{l,n} \text{ is only computed by } j \text{ nodes in } \mathcal{S}\backslash\{l\}) \cdot (s-j)\\
&\overset{(b)}{=} b^l_{\mathcal{S},j}(s-j),
\end{aligned}$$

where (a) and (b) follow from the definition of $b^l_{\mathcal{S},j}$. Thus, with (26),

$$\begin{aligned}
H(X_{\mathcal{S}}|Y_{\mathcal{S}^c}) &\geq \frac{T}{s}\sum_{j=1}^{s} b_{\mathcal{S},j} + \frac{T}{s}\sum_{j=1}^{s-1}\sum_{l\in\mathcal{S}} b^l_{\mathcal{S},j}\cdot\frac{s-j}{j}\\
&\overset{(a)}{=} \frac{T}{s}\sum_{j=1}^{s} b_{\mathcal{S},j} + \frac{T}{s}\sum_{j=1}^{s-1} b_{\mathcal{S},j}\cdot\frac{s-j}{j}\\
&= T\sum_{j=1}^{|\mathcal{S}|-1}\frac{b_{\mathcal{S},j}}{j}.
\end{aligned}$$

where (a) follows from (10).

Notice that, we have proved that (11) holds for all $\mathcal{S}\subseteq[K]$ with $|\mathcal{S}| = s+1$. By induction, we conclude that (11) holds for all nonempty subsets $\mathcal{S}\subseteq[K]$.

V. Conclusion

We proved a converse matching the performance of our recently proposed D3C [8]. As a result, the pareto-optimal storage-computation-communication tradeoff surface of all achievable storage-computation-communication triples is characterized.

Acknowledgement

The work of Q. Yan and M. Wigger has been supported by the ERC under grant agreement 715111.

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

[6] 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.

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

[8] Q. Yan, S. Yang, and M. Wigger, "A storage-computation-communication tradeoff for distributed computing," in *Proc. Int. Symp. Wireless Commun. Systems*, Lisbon, Portugal, Aug. 2018.

[9] Q. Yan, S. Sheng, and M. Wigger, "Storage, computation, and communication: A fundamental tradeoff in distributed computing," arXiv: 1806:07565.

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