Storage-Computation-Communication Tradeoff in Distributed Computing: Fundamental Limits and Complexity

Qifa Yan, Sheng Yang, Michèle Wigger

Storage-Computation-Communication Tradeoff in Distributed Computing: Fundamental Limits and Complexity

Qifa Yan, Sheng Yang, and Michèle Wigger

Abstract

Distributed computing has become one of the most important frameworks in dealing with large computation tasks. In this paper, we propose a systematic construction of coded computing schemes for MapReduce-type distributed systems. The construction builds upon placement delivery arrays (PDA), originally proposed by Yan *et al.* for coded caching schemes. The main contributions of our work are three-fold. First, we identify a class of PDAs, called *Comp-PDAs*, and show how to obtain a coded computing scheme from any Comp-PDA. We also characterize the normalized number of stored files (*storage load*), computed intermediate values (*computation load*), and communicated bits (*communication load*), of the obtained schemes in terms of the Comp-PDA parameters. Then, we show that the performance achieved by Comp-PDAs describing Maddah-Ali and Niesen's coded caching schemes matches a new information-theoretic converse, thus establishing the fundamental region of all achievable performance triples. In particular, we characterize *all* the Comp-PDAs achieving the pareto-optimal storage, computation, and communication (SCC) loads of the fundamental region. Finally, we investigate the file complexity of the proposed schemes, i.e., the smallest number of files required for implementation. In particular, we describe Comp-PDAs that achieve pareto-optimal SCC triples with significantly lower file complexity than the originally proposed Comp-PDAs.

Index Terms

Distributed computing, storage, communication, MapReduce, placement delivery array

I. Introduction

Massively large distributed systems have emerged as one of the most important forms to run big data and machine learning algorithms, so that data-parallel computations can be executed accross clusters of many individual computing nodes. In particular, distributed programs like MapReduce [2] and Dryad [3] have become popular and

Q. Yan and M. Wigger are with LTCI, Télécom ParisTech, Université Paris-Saclay, 75013 Paris, France. E-mails: qifa.yan@telecom-paristech.fr, michele.wigger@telecom-paristech.fr.

S. Yang is with L2S, (UMR CNRS 8506), CentraleSupélec-CNRS-Université Paris-Sud, 91192 Gif-sur-Yvette, France. Email: sheng.yang@centralesupelec.fr.

This paper was presented in part in 2018 IEEE Information Theory Workshop (ITW) [1].

can handle computing tasks involving data sizes as large as tens of terabytes. As illustrated in Fig. 1 and detailed in the following, computations in these systems are typically decomposed into "map" functions and "reduce" functions.

Consider the task of computing K output functions at K nodes and that each output function is of the form

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

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

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

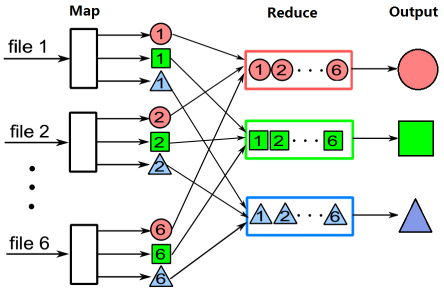


Fig. 1: A computing task with $N = 6$ files and $K = 3$ output functions. The small and big red circles, green squares, and blue triangles denote IVAs and results belonging to different output functions.

Computation of such functions can be performed in a distributed way following three phases: In the first phase, the *map phase*, each node $k = 1, \ldots, K$ 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:

$$\mathcal{C}_k \triangleq \{g_{q,n}(w_n) : q \in \{1, \ldots, K\},\ n \in \mathcal{M}_k\}. \tag{2}$$

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 $g_{k,1}(w_1), \ldots, g_{k,N}(w_N)$ 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).

Recently, Li *et al.* [4] proposed a so-called *coded distributed computing (CDC)* that stores files multiple times across different nodes in the map phase so as to create multicast opportunities for the shuffle phase. This approach can significantly reduce the communication load over traditional uncoded schemes, and was proved in [4] to have the smallest communication load among all coded computing schemes with the same total storage requirements. It is worth mentioning that Li *et al.* in [4] 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 load consumed across the nodes. This is why we would rather refer to it as the *storage-communication* tradeoff.

In this paper, we investigate a more general setup, where each node is allowed to choose for each IVA that it can

calculate from its locally stored data, whether or not to perform this calculation. The number of IVAs effectively calculated at all the nodes, normalized by the total number of IVAs, is then used to measure the real computation load. Thus, we extend the storage-communication tradeoff in [4] to a *storage-computation-communication* tradeoff. Notice that other interesting extensions have recently been proposed. For example, [6]–[15] included straggler nodes but restricted to map functions that are matrix-vector or matrix-matrix multiplications; straggler nodes with general linear map functions were considered in [16]; [17] studied optimal allocation of computation resources; [18]–[21] investigated distributed computing in wireless networks; [22]–[24] investigated the iterative procedures of data computing and shuffling; [25] studied the case when each node has been randomly allocated files; [26] investigated the case with random connectivity between nodes; [27]–[30] designed codes for computing gradient distributedely, which is particularly useful in machine learning.

One of our main contributions is a framework to construct a coded computing scheme from a given *placement delivery array (PDA)* [32], and to characterize the storage, computation, and communication (SCC) loads of the resulting scheme in terms of the PDA parameters. In this paper we focus on a class of PDAs that we call *PDAs for distributed computing*, for short *Comp-PDA*. Notice that PDAs were introduced in [32] to describe placement and delivery phases in a shared-link caching network. The connections between this caching network and the proposed distributed computing systems have been noticed and exploited in various previous works [4], [35], [36]. Here, we make the connection precise in the case of Comp-PDA based schemes, by exactly characterizing the SCC loads of these schemes for distributed computing. Notice that in contrast to shared-link caching systems, for the proposed distributed computing system, Comp-PDA based schemes turn out to be optimal. It means that they can attain all achievable SCC loads, and in particular the pareto-optimal SCC surface. Such optimality is proved in this paper by means of an information-theoretic converse that is not restricted to Comp-PDA based schemes.

Our results show that the (3-dimensional) pareto-optimal tradeoff surface can be obtained by sequentially pasting $K-2$ triangles next to each other. The corner points of these triangles are achieved by Comp-PDAs that also describe Maddah-Ali and Niesen's coded caching scheme [31][1] and the corresponding coded computing schemes coincide with the scheme proposed by Ezzeldin *et al.* [5] and with Li *et al.*'s CDC scheme if the unused IVAs are removed. These schemes all require a minimum number of $N \geq \binom{K}{g}$ files, where g is an integer between 1 and K and depends on the corner point under consideration. In this paper, we show that no Comp-PDA based scheme can achieve the corner points with a smaller number of files for $g \geq 2$. However, pareto-optimal SCC points that are close to the corner points can be achieved with a significantly smaller number of files. We prove this through new explicit Comp-PDA constructions, which include the PDAs proposed in [32, Construction A] as a special case.

Finally, we present necessary and sufficient conditions for a Comp-PDA scheme to achieve pareto-optimal SCC loads. Our results implies in particular that most of the Comp-PDA schemes based on existing PDA constructions [32], [33], and [34] have pareto-optimal SCC loads.

[1]The connection between these PDAs and the coded caching scheme in [31] was formalized in [32]. As explained previously, Comp-PDAs are also PDAs.

Paper Organization: Section II presents the system model. Section III introduces Comp-PDAs and explains at hand of an example how to obtain a distributed coded computing scheme from a Comp-PDA. The main results are summarized in Section IV. Proofs of the main results are provided in Section V–VII, where the more technical details are deferred to the appendices. Finally, Section VIII concludes the paper.

Notations: Let $\mathbb{N}^+$ be the set of positive integers, and $\mathbb{F}_2$ be the binary field. For $m, n \in \mathbb{N}^+$, denote the n-dimensional vector space over $\mathbb{F}_2$ by $\mathbb{F}_2^n$, and the integer set $\{1, \ldots, n\}$ by $[n]$. If $m < n$, we use $[m : n]$ to denote the set $\{m, m+1, \ldots, n\}$. We also use interval notations, e.g., $[a, b] \triangleq \{x : a \leq x \leq b\}$ and $[a, b) \triangleq \{x : a \leq x < b\}$ for real numbers a, b such that $a < b$. The notation $(a)^+$ is used to denote the number $\max\{a, 0\}$. The bitwise exclusive OR (XOR) operation is denoted by $\oplus$. To denote scalar or vector quantities, we use the standard font, e.g., a or A, for arrays we use upper case bold font, e.g., $\mathbf{A}$, for sets we use upper case calligraphic font, e.g., $\mathcal{A}$.

A line segment with end points A_1, A_2 or a line through the points A_1, A_2 is denoted by $\overline{A_1A_2}$. A triangle with vertices A_1, A_2, A_3 is denoted by $\triangle A_1A_2A_3$. A trapezoid with the four edges $\overline{A_1A_2}$, $\overline{A_2A_3}$, $\overline{A_3A_4}$, and $\overline{A_4A_1}$, where $\overline{A_1A_2}$ is parallel to $\overline{A_3A_4}$, is denoted by $\boxminus A_1A_2A_3A_4$. Let $\mathcal{F}$ be a set of facets, if the facets in $\mathcal{F}$ form a continuous surface, then we refer to this surface simply as $\mathcal{F}$.

II. System Model

Consider a system consisting of K distributed computing nodes

$$\mathcal{K} \triangleq \{1, \ldots, K\} \tag{3}$$

and N files,

$$\mathcal{W} = \{w_1, \ldots, w_N\}, \quad w_n \in \mathbb{F}_2^W, \forall\ n \in [N], \tag{4}$$

each of size W bits, where $K, N, W \in \mathbb{N}$. The goal of node k, $k \in \mathcal{K}$, is to compute an output function[2]

$$\phi_k \colon \mathbb{F}_2^{NW} \to \mathbb{F}_2^U, \tag{5}$$

which maps all the files to a bit stream

$$u_k = \phi_k(w_1, \ldots, w_N) \in \mathbb{F}_2^U \tag{6}$$

of length U, for a given $U \in \mathbb{N}$.

Following the conventions in [4], we assume that each output function ϕ_k decomposes as:

$$\phi_k(w_1, \ldots, w_N) = h_k(f_{k,1}(w_1), \ldots, f_{k,N}(w_N)), \tag{7}$$

where:

[2]See Remark 2 for a relaxed assumption.

- Each *"map" function* $f_{k,n}$ is of the form

$$f_{k,n} : \mathbb{F}_2^W \to \mathbb{F}_2^V, \tag{8}$$

and maps the file w_n into the IVA

$$v_{k,n} \triangleq g_{k,n}(w_n) \in \mathbb{F}_2^V, \tag{9}$$

for a given $V \in \mathbb{N}$.

- The *"reduce" function* h_k is of the form

$$h_k : \mathbb{F}_2^{NV} \to \mathbb{F}_2^U, \tag{10}$$

and maps the IVAs

$$\mathcal{V}_k \triangleq \{v_{k,n} : n \in [N]\} \tag{11}$$

into the output stream

$$u_k = h_k(v_{k,1}, \ldots, v_{k,N}). \tag{12}$$

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

The described structure of the output functions $\phi_1, \ldots, \phi_K$, allows the nodes to perform their computation in the following three-phase procedure.

1) **Map Phase:** Each node $k \in \mathcal{K}$ chooses to store a subset of files $\mathcal{M}_k \subseteq \mathcal{W}$. For each file $w_n \in \mathcal{M}_k$, node k computes a subset of IVAs

$$\mathcal{C}_{k,n} = \{v_{q,n} : q \in \mathcal{Z}_{k,n}\}, \tag{13}$$

where $\mathcal{Z}_{k,n} \subseteq \mathcal{K}$. Denote the set of IVAs computed at node k by $\mathcal{C}_k$, i.e.,

$$\mathcal{C}_k \triangleq \bigcup_{n:w_n \in \mathcal{M}_k} \mathcal{C}_{k,n}. \tag{14}$$

2) **Shuffle Phase:** The K nodes exchange some of their computed IVAs. In particular, node k creates a signal

$$X_k = \varphi_k\left(\mathcal{C}_k\right) \tag{15}$$

of some length $l_k \in \mathbb{N}$, using a function

$$\varphi_k : \mathbb{F}_2^{|\mathcal{C}_k|V} \to \mathbb{F}_2^{l_k}. \tag{16}$$

It then multicasts this signal to all the other nodes, which receive it error-free.

3) **Reduce Phase:** Using the shuffled signals $X_1, \ldots, X_K$ and the IVAs $\mathcal{C}_k$ it computed locally in the map phase, node k now computes the IVAs

$$(v_{k,1}, \ldots, v_{k,N}) = \psi_k \left(X_1, \ldots, X_K, \mathcal{C}_k \right), \tag{17}$$

for some function

$$\psi_k : \mathbb{F}_2^{l_1} \times \mathbb{F}_2^{l_2} \times \ldots \mathbb{F}_2^{l_K} \times \mathbb{F}_2^{|\mathcal{C}_k|V} \to \mathbb{F}_2^{NV}. \tag{18}$$

Finally, it computes

$$u_k = h_k(v_{k,1}, \ldots, v_{k,N}). \tag{19}$$

To measure the storage, computation, and communication costs of the described procedure, we introduce the following definitions.

Definition 1 (Storage load)**.** Storage load r *is defined as the total number of files stored across the* K *nodes normalized by the total number of files* N*:*

$$r \triangleq \frac{\sum_{k=1}^{K} |\mathcal{M}_k|}{N}. \tag{20}$$

Definition 2 (Computation load)**.** Computation load c *is defined as the total number of map functions computed across the* K *nodes, normalized by the total number of map functions* NK*:*

$$c \triangleq \frac{\sum_{k=1}^{K} |\mathcal{C}_k|}{NK}. \tag{21}$$

Definition 3 (Communication load)**.** Communication load L *is defined as the total number of the bits sent by the* K *nodes during the shuffle phase normalized by the total length of all intermediate values* NKV*:*

$$L = \frac{\sum_{k=1}^{K} l_k}{NKV}. \tag{22}$$

Remark 1. These measures were first defined in [5], where the storage load was called "load redundancy", and the computation load therein was the total number of computed IVAs (not normalized by NK). We use the term "storage load" because it actually captures the memory size constraint. We used the normalized version for computation load to keep symmetric definitions with storage load and communication load.

Note that the nontrivial regime of the parameters is:

$$1 \leq c \leq r \leq K, \tag{23a}$$

$$0 \leq L \leq 1 - \frac{r}{K}. \tag{23b}$$

Firstly, we argue that the regime of interest for L is $[0, 1 - r/K]$. By definition, $L \geq 0$. Moreover, each node k can trivially compute $|\mathcal{M}_k|$ of its desired IVAs locally and thus only needs to receive $N - |\mathcal{M}_k|$ IVAs from other nodes.

Uncoded shuffling of these missing IVAs requires a communication load of $L = \frac{\sum_{k=1}^{K}(N-|\mathcal{M}_k|)V}{NKV} = 1 - \frac{r}{K}$. The question of interest is whether a coded shuffling procedure allows to reduce this communication load. Secondly, we argue that we can restrict attention to values of c and r satisfying (23a). Since each IVA needs to be computed at least once somewhere, we have $c \geq 1$. Moreover, the definition of $\mathcal{C}_k$ in (14) implies that $|\mathcal{C}_k| \leq |\mathcal{M}_k|K$, and thus by (20) and (21) that $c \leq r$. Finally, the regime $r > K$ is not interesting, because in this case each node stores all the files, $\mathcal{M}_k = \{1, \ldots, N\}$, and can thus locally compute all the IVAs required to compute its output function. In this case, $c \geq 1$ and $L \geq 0$ can be arbitrary.

Definition 4 (Fundamental SCC region)**.** *An SCC-triple (r, c, L) as in* (23) *is called* feasible*, if for any $\epsilon > 0$ and sufficiently large N, there exist map, shuffle, and reduce procedures with storage load, computation load, and communication load less than $r+\epsilon$, $c+\epsilon$, and $L+\epsilon$. The set of all feasible SCC triples $\mathcal{R}$ is called the* fundamental SCC region*:*

$$\mathcal{R} \triangleq \{(r, c, L) \colon (r, c, L) \text{ is feasible}\}. \tag{24}$$

Definition 5 (Optimal tradeoff surface)**.** *A SCC triple (r, c, L) is called* pareto-optimal *if it is feasible and if no feasible SCC triple (r', c', L') exists so that $r' \leq r, c' \leq c$ and $L' \leq L$ with one or more of the inequalities being strict. Define the* optimal tradeoff surface *as*

$$\mathcal{O} \triangleq \{(r, c, L) \colon (r, c, L) \text{ is pareto-optimal}\}. \tag{25}$$

In order to achieve a certain SCC triple with a given scheme, it is implicitly assumed that the number of files is larger than some value. We refer to this value as the *file complexity*.

Definition 6 (File complexity)**.** *The smallest number of files N required to implement a given scheme is called the* file complexity *of this scheme.*

Remark 2. All our conclusions in this paper remain valid in an extended setup with Q output functions as in [4], where $K|Q$ and each node is supposed to compute $\frac{Q}{K}$ functions. In fact, in this setup, the K in definitions (21) and (22) will be replaced by Q. Achievability proofs can be shown by executing $\frac{Q}{K}$ times the coded computing schemes as explained in Section V. The converse can be derived by adjusting the definitions in (88), (90), (208), and following the same steps in Section VI-A and Appendix B.

III. Placement Delivery Arrays for Distributed Computing (Comp-PDA)

A. Definitions

In the following, we recall the definition of a PDA, define Comp-PDAs, and two subclasses thereof.

Definition 7 (PDA)**.** *For positive integers K, F, T and a nonnegative integer S, an $F \times K$ array $\mathbf{A} = [a_{j,k}]$, $j \in [F], k \in [K]$, composed of T specific symbols "$*$" and some ordinary symbols $1, \ldots, S$, each occurring at*

least once, is called a (K, F, T, S) PDA, if, for any two distinct entries $a_{j,k}$ and $a_{j',k'}$, we have $a_{j,k} = a_{j',k'} = s$, for some ordinary symbol s only if

a) *$j \neq j'$, $k \neq k'$, i.e., they lie in distinct rows and distinct columns; and*

b) *$a_{j,k'} = a_{j',k} = *$, i.e., the corresponding 2×2 sub-array formed by rows j, j' and columns k, k' must be of the following form*

$$\begin{bmatrix} s & * \\ * & s \end{bmatrix} \text{ or } \begin{bmatrix} * & s \\ s & * \end{bmatrix}. \tag{26}$$

A PDA with all "$*$" entries is called trivial. Notice that in this case $S = 0$ and $KF = T$.

The above PDA definition is more general than the original version in [32] in the sense that different columns can have different numbers of "$*$" symbols. In the original definition [32], each column had to contain the same number of "$*$" symbols and this number was one of the four parameters of the PDA. In this new definition, a PDA is parametrized by T, the *total* number of "$*$" symbols in *all* the columns. The motivation for this change is as follows. PDAs were originally proposed for the shared-link coded computing scheme where all users have same cache memory size. In such a setup, the number of "$*$" symbols in a column was proportional to the cache memory size at the corresponding user. By the equal memory-size assumption, each column thus had to contain the same number of "$*$" symbols. As we will see, for distributed computing, the number of "$*$" symbols in a column is proportional to the number of files stored at the corresponding node. Moreover, different nodes can have different memory sizes and we are only interested in the *total* memory size across *all* users. As a consequence, different columns of the PDA can have different numbers of "$*$" symbols and the PDA is parametrized by the *total* number of "$*$" symbols across *all* columns. Another generalization in the PDA definition is that we allow for arrays with only "$*$" symbols but no ordinary symbols.

In this work, we are interested in PDAs with at least one "$*$" symbol in each row.

Definition 8 (PDA for distributed computing (Comp-PDA))**.** *A Comp-PDA is a PDA with at least one "$*$" in each row.*

In particular a trivial PDA is a Comp-PDA. Two other Comp-PDAs are presented in the following example.

Example 1. The following $\mathbf{A}$ is a $(5, 4, 10, 4)$ Comp-PDA and $\mathbf{A}'$ is a $(3, 3, 6, 1)$ Comp-PDA.

$$\mathbf{A} = \begin{bmatrix} * & 2 & * & 3 & * \\ 1 & * & * & 4 & 2 \\ * & 4 & 1 & * & 3 \\ 3 & * & 2 & * & * \end{bmatrix} \quad \text{and} \quad \mathbf{A}' = \begin{bmatrix} * & 1 & * \\ * & * & 1 \\ 1 & * & * \end{bmatrix}. \tag{27}$$

As we will see, the performance of our Comp-PDA based schemes does not depend on the number of ordinary symbols S, but only on the relative frequencies with which they appear in the Comp-PDA.

Definition 9 (Symbol frequencies)**.** *For a given nontrivial (K, F, T, S) Comp-PDA, let S_t denote the number of ordinary symbols that occur exactly t times, for $t \in [K]$. The* symbol frequencies $\theta_1, \theta_2, \ldots, \theta_K$ *of the Comp-PDA are then defined as*

$$\theta_t \triangleq \frac{S_t t}{KF - T}, \quad t \in [K]. \tag{28}$$

They indicate the fractions of ordinary entries *of the Comp-PDA that occur exactly $1, 2, \ldots, K$ times, respectively. For completeness, we also define $\theta_t \triangleq 0$ for $t > K$.*

The following two classes of Comp-PDAs will be of particular interest.

Definition 10 (Almost-regular Comp-PDAs & regular Comp-PDAs)**.** *For $g \in [K]$, a (K, F, T, S) Comp-PDA is called* almost g-regular *if each ordinary symbol appears either g or $g + 1$ times with $\theta_{g+1} < 1$. If each ordinary symbol appears exactly g times, the Comp-PDA is called g-regular.*

Therefore, a g-regular Comp-PDA is also an almost g-regular Comp-PDA. An almost K-regular Comp-PDA is also a K-regular Comp-PDA. In particular, the Comp-PDAs $\mathbf{A}'$ and $\mathbf{A}$ in (27) are 3-regular and almost 2-regular, respectively.

Notice that a (K, F, T, S) Comp-PDA can be almost g-regular only if

$$g = \left\lfloor \frac{KF - T}{S} \right\rfloor, \tag{29}$$

and it can be g-regular only if

$$g = \frac{KF - T}{S}. \tag{30}$$

B. Constructing a Coded Computing Scheme from a Comp-PDA: A Toy Example

To have an idea of how to use Comp-PDAs to construct coded computing schemes, let us consider the following toy example with the above 3-regular $(3, 3, 6, 1)$ Comp-PDA $\mathbf{A}'$ in (27). We can derive a coded computing scheme for the computation task in Fig. 1 with $K = 3$ nodes. The scheme is depicted in Fig. 2. The top-most line in each of the three boxes indicates the files stored at the node. Below this line, is a rectangle indicating the map functions. The computed IVAs are depicted below the rectangle, 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 $\{v_{3,1}, \ldots, v_{3,6}\}$. The dashed circles/squares/triangles stand for the IVAs that are not computed from the stored files. The last line of each box indicates the IVAs that the node needs to learn during the shuffle phase.

The $N = 6$ files are first partitioned into $F = 3$ batches

$$\mathcal{W}_1 = \{w_1, w_2\}, \quad \mathcal{W}_2 = \{w_3, w_4\}, \quad \text{and } \mathcal{W}_3 = \{w_5, w_6\}, \tag{31}$$

which are associated to the rows 1, 2, and 3, respectively. The three nodes 1, 2, and 3 are associated with the columns 1, 2, and 3, respectively. The "$*$"-symbols in the Comp-PDA describe the storage operations. Each node

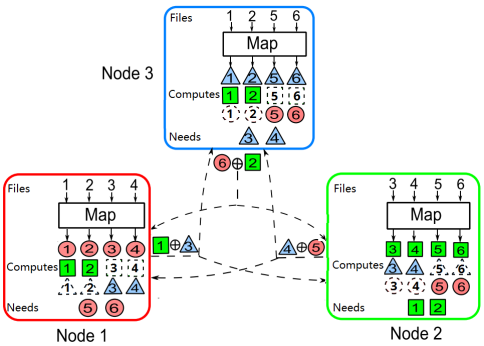


Fig. 2: An coded computing scheme from $\mathbf{A}$ in (27).

stores all the files of the batches that have a "$*$"-symbol in the corresponding column. For example, node 1, which is associated with the first column of the Comp-PDA, stores the files in batches $\mathcal{W}_1$ and $\mathcal{W}_2$, because they are associated with the first two rows. Node 2, which is associated with the second column, stores the files in batches $\mathcal{W}_2$ and $\mathcal{W}_3$; and node 3, which is associated with the third column, stores the files in batches $\mathcal{W}_1$ and $\mathcal{W}_3$.

The ordinary symbols in the Comp-PDA describe the shuffling operations. And indirectly also some of the computations of IVAs during the map phase. Each ordinary symbol entry in the array denotes the IVAs computed from the files in the batch corresponding to its row index for the output function computed by the node corresponding to its column index. In fact, during the map phase, each node first computes all its desired IVAs which it can obtain from its locally stored batches. Then, it computes all the IVAs indicated by the ordinary symbol entries except for the ones in its own column. Specifically, node 1 first computes the circle IVAs of files $1, 2, 3, 4$ pertaining to batches $\mathcal{W}_1$ and $\mathcal{W}_2$; node 2 first computes the square IVAs of files $3, 4, 5, 6$ pertaining to batches $\mathcal{W}_2$ and $\mathcal{W}_3$; and node 3 first computes the triangle IVAs of files $1, 2, 5, 6$ pertaining to batches $\mathcal{W}_1$ and $\mathcal{W}_3$. Then, they also compute the IVAs needed to form the XOR messages exchanged in the shuffle phase.

These XOR messages are described by the ordinary symbol 1 in the Comp-PDA $\mathbf{A}$. Each node considers the subarray of $\mathbf{A}$ that is formed by the columns associated with the other two nodes, and sends the XOR packet that the ordinary symbol 1 indicates for this subarray. For example, node 1 considers the subarray formed by the second and third columns of $\mathbf{A}$, where the ordinary symbol 1 appears in the first row of the second column and in the second row of the third column. These positions indicate that node 1 should multicast the XOR of a square IVA (i.e., an IVA for node 2) that can be computed from batch $\mathcal{W}_1$ and a triangle IVA (i.e., an IVA for node 3) that can be computed from batch $\mathcal{W}_2$. Notice that node 1 has computed both of these IVAs by the above description. We can verify that in Fig. 2, node 1 indeed sends the XOR between square IVA 1 and triangle IVA 3. From the described XOR, node 2 can recover its desired square IVA, because it has computed the triangle IVA locally, and node 3 can recover its desired triangle IVA, because it has computed the square IVA locally.

To create their own multicast messages for the shuffle phase, nodes 2 and 3 consider the subarrays of $\mathbf{A}$ formed by the first and third columns, and by the first and second columns, respectively. The positions of the symbol 1 in these subarrays indicate that node 2 should multicast the XOR of a circle IVA (an IVA for node 1) that can

be computed from batch $\mathcal{W}_3$ and of a triangle IVA (an IVA for node 3) that can be computed from batch $\mathcal{W}_2$. Similarly, node 3 should multicast the XOR of a circle IVA (an IVA for node 1) that can be computed from $\mathcal{W}_3$ and of a square IVA (an IVA for node 2) that can be computed from $\mathcal{W}_1$. We can verify again that in Fig. 2, node 2 and node 3 indeed multicast the described XOR messages. Moreover, given the signals they sent and the IVAs they computed locally, node 1 can recover all the circle IVAs, node 2 can recover all square IVAs, and node 3 can recover all triangle IVAs.

Each node k then terminates the reduce phase by applying the reduce function h_k to all its recovered IVAs.

IV. Main Results

A. Coded Computing Schemes from Comp-PDAs

At the end of the preceding section, we presented an example on how to obtain a Comp-PDA from a coded computing scheme. In Section V, we describe this procedure in general. For brevity, we say that a *Comp-PDA* $\mathbf{A}$ *achieves an SCC triple* (r, c, L) *with file complexity* γ if the coded computing scheme obtained by applying the procedure in Section V to $\mathbf{A}$ has file complexity γ and achieves the SCC triple (r, c, L). For convenience, we define $\{\theta'_t\}$ as follows.

$$\theta'_t = \begin{cases} 0, & t = 1 \\ \theta_1 + \theta_2, & t = 2 \\ \theta_t, & t > 2. \end{cases} \tag{32}$$

Theorem 1. *A (K, F, T, S) Comp-PDA $\mathbf{A}$ with symbol frequencies $\{\theta_t\}_{t=1}^K$ achieves the SCC triple*

$$(r, c, L) = \left(\frac{T}{F}, \ \frac{T}{KF} + \left(1 - \frac{T}{KF}\right) \cdot \sum_{t=2}^{K} \theta'_t(t-1), \ \left(1 - \frac{T}{KF}\right) \cdot \sum_{t=2}^{K} \frac{\theta'_t}{t-1} \right), \tag{33}$$

with file complexity F.

We notice that the file complexity of a Comp-PDA is simply the number of rows F. We therefore will call this parameter of a Comp-PDA its file complexity.

The theorem simplifies for almost-regular and regular Comp-PDAs.

Corollary 1. *An almost g-regular (K, F, T, S) Comp-PDA achieves the SCC triple*

$$(r, c, L) = \left(\frac{T}{F}, \ \frac{T}{KF} + \left(1 - \frac{T}{KF}\right) \cdot (g - 1 + \theta'_{g+1}), \ \left(1 - \frac{T}{KF}\right) \cdot \frac{(g - \theta'_{g+1} - 1)^+ + 1}{g \cdot ((g-2)^+ + 1)} \right). \tag{34}$$

In particular, a g-regular (K, F, T, S) Comp-PDA achieves the SCC triple

$$(r, c, L) = \left(\frac{T}{F}, \ \frac{T}{KF} + \left(1 - \frac{T}{KF}\right) \cdot ((g-2)^+ + 1), \ \frac{1}{(g-2)^+ + 1} \cdot \left(1 - \frac{T}{KF}\right) \right). \tag{35}$$

Proof: Notice that for almost g-regular Comp-PDAs, $\theta'_g = 0$ and $\theta'_{g+1} = 1$ when $g = 1$; $\theta'_g = 1$ and $\theta'_{g+1} = 0$ when $g = K$; and $\theta'_g + \theta'_{g+1} = 1$ when $1 < g < K$. As such, the equality (34) follows from Theorem 1

straightforwardly. Further, for g-regular Comp-PDAs, we have $\theta'_{g+1} = 1$ when $g = 1$; and $\theta'_{g+1} = 0$ when $g \geq 2$. Equality (35) follows readily from (34). ■

Corollary 1 is of particular interest since there are several explicit regular PDA constructions for coded caching in the literature [32]–[34]. See for example the Comp-PDA in the following Definition 11.

B. Achieving the Fundamental SCC Region

The following Comp-PDAs achieve points on the optimal tradeoff surface $\mathcal{O}$. They are obtained from the coded caching scheme proposed by Maddah-Ali and Niesen [31].

Definition 11 (Maddah-Ali Niesen PDA (MAN-PDA))**.** *Fix any integer $i \in [K]$, and let $\{\mathcal{T}_j\}_{j=1}^{\binom{K}{i}}$ denote all subsets of $[K]$ of size i. Also, choose an arbitrary bijective function κ from the collection of all subsets of $[K]$ with cardinality $i+1$ to the set $\left[\binom{K}{i+1}\right]$. Then, define the PDA $\mathbf{P}_i = [p_{j,k}]$ as*

$$p_{j,k} \triangleq \begin{cases} *, & \text{if } k \in \mathcal{T}_j \\ \kappa(\{k\} \cup \mathcal{T}_j), & \text{if } k \notin \mathcal{T}_j \end{cases}. \tag{36}$$

We observe that for any $i \in [K-1]$, the PDA $\mathbf{P}_i$ is an $(i+1)$-regular $\left(K, \binom{K}{i}, K\binom{K-1}{i-1}, \binom{K}{i+1}\right)$ Comp-PDA. For $i = K$, the PDA $\mathbf{P}_i$ consists only of "$*$"-entries and is thus a trivial PDA.

We can evaluate Corollary 1 for the Comp-PDAs $\mathbf{P}_1, \ldots, \mathbf{P}_K$.

Corollary 2. *For any $i \in [K]$, the Comp-PDA $\mathbf{P}_i$ achieves the SCC triple*

$$P_i \triangleq (r_{\mathbf{P}_i}, c_{\mathbf{P}_i}, L_{\mathbf{P}_i}) = \left(i,\ \ i\left(1 - \frac{i-1}{K}\right),\ \ \frac{1}{i}\left(1 - \frac{i}{K}\right)\right). \tag{37}$$

As the following Theorem 2 shows, the points $\{P_i\}$ lie on the optimal tradeoff surface $\mathcal{O}$. Let us also define the projection of point P_i to the surface $r = c$ in the SCC space as Q_i:

$$Q_i \triangleq \left(i,\ \ i,\ \ \frac{1}{i}\left(1 - \frac{i}{K}\right)\right), \quad i \in [K]. \tag{38}$$

Theorem 2. *Let $\mathcal{F}$ be the surface formed by the following triangles and trapezoids*

$$\mathcal{F} \triangleq \{\triangle P_1 P_2 Q_2\} \cup \{\triangle P_{i-1} P_i P_K \colon\ i = 2, \ldots, K-1\} \cup \{\boxminus P_i Q_i Q_{i+1} P_{i+1} \colon\ i = 2, \ldots, K-1\}. \tag{39}$$

where P_i and Q_i are defined in (37) *and* (38)*, respectively. Then, the optimal tradeoff surface $\mathcal{O}$ and the fundamental SCC region $\mathcal{R}$ are given by*

$$\mathcal{O} = \{\, \triangle P_{i-1} P_i P_K : \ i = 2, \ldots, K-1\}, \tag{40}$$

$$\mathcal{R} = \{\, (r,c,L) \colon\ (r,c,L) \ \text{ is on or above the surface } \mathcal{F} \text{ and satisfies (23)}\}. \tag{41}$$

Furthermore, a Comp-PDA achieves the optimal surface $\mathcal{O}$ if and only if it satisfies one of the following three conditions: it is almost g-regular, for any $g \in [K]$; it is trivial (i.e., consists only of "$*$" symbols); or each ordinary

symbol occurs at most three times.

Remark 3. A close inspection of the proof of Theorem 2 reveals that for $i \in \{3, \ldots, K-1\}$, a Comp-PDA achieves a point on the triangle $\triangle P_{i-1}P_iP_K$ if and only if it is almost i-regular. It achieves a point on the triangle $\triangle P_1P_2P_K$ if and only if it all the ordinary symbols occur either $1, 2$, or 3 times. This implies in particular that for any $i \in \{2, \ldots, K-2\}$ a Comp-PDA achieves a point on the line segment $\overline{P_iP_K}$ if and only if it is $(i+1)$-regular.

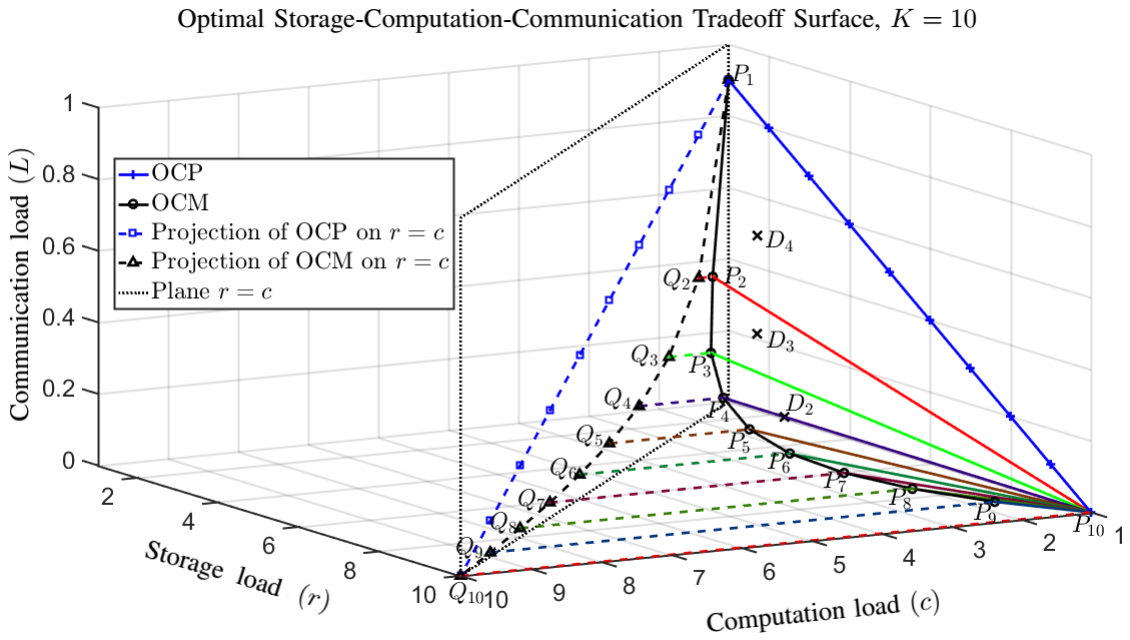


Fig. 3: The fundamental SCC region $\mathcal{R}$ for a system with $K = 10$ nodes. The figure illustrates the delimiting surface $\mathcal{F}$ formed by the triangles $\triangle P_1P_2Q_2$ and $\{\triangle P_{i-1}P_iP_K\}$ and the trapezoids $\{\boxminus P_iQ_iQ_{i+1}P_{i+1}\}$. The three points D_2, D_3, D_4 can be achieved by the new PDA design.

Note that setting $r = c$, we recover exactly the case investigated in [4] where the fundamental storage-communication tradeoff is characterized by

$$L^*(r) \triangleq \frac{1}{r}\left(1 - \frac{r}{K}\right), \tag{42}$$

for integer r, and for general r in the interval $[1, K]$

$$L^*(r) \triangleq \max_{i \in [K-1]} \left\{ -\frac{1}{i(i+1)}r + \frac{1}{i} + \frac{1}{i+1} - \frac{1}{K} \right\}. \tag{43}$$

An example of the fundamental SCC region for $K = 10$ is given in Fig. 3, where we can identify the surface $\mathcal{F}$ that is formed by the triangles $\triangle P_1P_2Q_2$ and $\{\triangle P_{i-1}P_iP_K\}$ and the trapezoids $\{\boxminus P_iQ_iQ_{i+1}P_{i+1}\}$. In particular, the boundary of the optimal tradeoff surface $\mathcal{O}$ is formed by the line segment $\overline{P_1P_K}$ and the sequence of line segments $\overline{P_1P_2}$, $\overline{P_2P_3}$, $\ldots$, $\overline{P_{K-1}P_K}$:

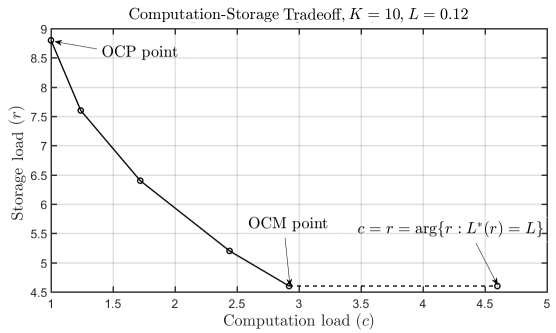
1) The computation load on the line segment $\overline{P_1P_K}$ is $c = 1$ for any given storage load r, which by (23) is

minimal and thus is referred to as the *optimal computation curve (OCP)*. It implies that during the map phase each IVA is calculated at a single node.

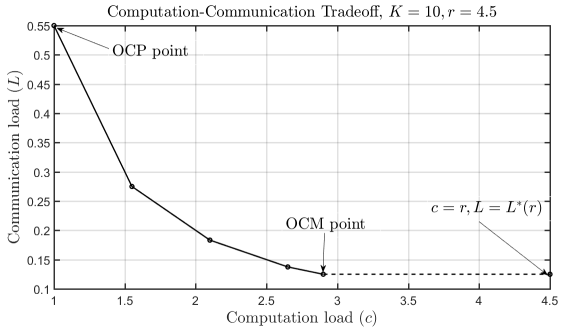
2) The points on the line segments $\overline{P_1P_2}$, $\overline{P_2P_3}$, $\ldots$, $\overline{P_{K-1}P_K}$ have minimum communication load L for any given storage load r among all pareto-optimal points, thus we refer to it as the *optimal communication curve (OCM)*.

Note that the projections of OCP and OCM curves on the surface $r = c$ correspond to the curves of the uncoded scheme and the CDC scheme in [4]. In this sense, our optimal tradeoff surface $\mathcal{O}$ is a natural extension of the tradeoff established in [4] with the additional dimension given by the computation load. From the SCC region, we can obtain straightforwardly the optimal tradeoff between computation and storage for a given communication load (Fig. 4(a)), as well as the tradeoff between computation and communication for a given storage load (Fig. 4(b)).

It is worth mentioning that our computation load that counts the exact number of IVAs that are necessary for the reduce phase may not reflect the actual computation load in practical systems. In some practical systems, the actual computation time might not decrease even with a reduced number of IVAs. Nevertheless, such a measure provides an performance indication that can be very different from the storage load. Specifically, we can observe that at high storage load, the computation load can be actually close to the lower bound 1, i.e., almost no extra computation is needed.



(a)



(b)

Fig. 4: Two-dimensional tradeoff curves for $K = 10$: (a) Computation-storage tradeoff with fixed communication load $L = 0.12$; (b) Computation-communication tradeoff with fixed storage load $r = 4.5$.

Remark 4. The idea of measuring the effectively computed IVAs is from [5]. In that paper, the authors also show the achievability of the corner points $P_1, P_2, \ldots, P_K$. The distributed computing scheme proposed in [4] coincides with the schemes obtained for Comp-PDAs $\mathbf{P}_1, \mathbf{P}_2, \ldots \mathbf{P}_K$. Our new contributions are an information-theoretic converse that allows to characterize the entire fundamental SCC region and necessary and sufficient conditions for any Comp-PDA to achieve the optimal surface of the fundamental SCC region.

C. Reducing the File Complexity to Attain the Optimal Tradeoff Surface

Recall that for any $i \in [K]$, the Comp-PDA $\mathbf{P}_i$ in Definition 11 achieves point P_i on the optimal tradeoff surface $\mathcal{O}$. The Comp-PDA $\mathbf{P}_i$ is of file complexity

$$F_{P_i} \triangleq \binom{K}{i}, \tag{44}$$

and the following theorem indicates that this is the minimum file complexity that can achieve P_i when $i \geq 2$.

Theorem 3. *Any Comp-PDA that achieves the corner point P_i, for $i \in [2:K]$, is of file complexity at least $\binom{K}{i}$.*

A required file size of $\binom{K}{i}$ can be prohibitively large and may prevent practical implementation of the Comp-PDA based schemes achieving the corner point P_i. However, as the following theorem shows, one can achieve points on the triangle $\Delta P_{i-1}P_iP_K$ close to the corner point P_i with significantly fewer files. (Notice that the simple approach of time- and memory- sharing the schemes achieving the points P_{i-1}, P_i, and P_K would require even more files.)

Theorem 4. *For any positive integers K and q such that $q < \frac{K}{2}$, and $m \triangleq \lceil \frac{K}{q} \rceil - 1$, there exists a $\left(K, q^m, Kq^{m-1}, (q-1)q^m\right)$ Comp-PDA achieving the triple*

$$\begin{aligned} D_q &= \left(r_{D_q}, c_{D_q}, L_{D_q}\right) && (45)\\ &\triangleq \left(\frac{K}{q},\ \frac{1}{q} + m\left(1-\frac{1}{q}\right)\left(2-\frac{(m+1)q}{K}\right),\ \left(\frac{1}{m} + \frac{(m+1)q-K}{K(m-1)}\right)\cdot\left(1-\frac{1}{q}\right)\right), && (46)\end{aligned}$$

with file complexity

$$F_{D_q} \triangleq q^m. \tag{47}$$

Moreover, the SCC triple D_q lies on the optimal tradeoff triangle $\triangle P_{m-1}P_mP_K$, and is close to P_m in the following sense:

$$-1 \leq r_{P_m} - r_{D_q} \leq 0, \tag{48a}$$

$$\frac{1}{q} - \frac{2}{K} \leq c_{P_m} - c_{D_q} \leq 1 - \frac{1}{q}, \tag{48b}$$

$$-\frac{q(q-1)}{K(K-q)} \leq L_{P_m} - L_{D_q} \leq \frac{q}{K(K-q)}. \tag{48c}$$

Furthermore, its file complexity satisfies

$$\frac{F_{P_m}}{F_{D_q}} \geq \frac{\sqrt{2\pi}}{e^2}\sqrt{\frac{q}{m(q-1)}}\left(\frac{q}{q-1}\right)^{m(q-1)}. \tag{49}$$

For example, for $K = 50$ and $q = 9$, we have $m = 5$. According to the above theorem, D_9 is close to P_5 but with file complexity $F_{D_q} = q^m \approx 6 \times 10^4$ instead of $F_{P_m} = \binom{K}{m} \approx 10^{10}$. In Fig. 3, we depict the new points D_2, D_3, D_4 for $K = 10$ nodes.

V. Coded Computing Schemes from Comp-PDAs (Proof of Theorem 1)

The proof of Theorem 1 has three parts.

A. Obtaining a Coded Computing Scheme from a Comp-PDA

In this section, we explain how to obtain a coded computing scheme from any (K, F, T, S) Comp-PDA.

Fix a (K, F, T, S) Comp-PDA $\mathbf{A} = [a_{i,j}]$. Partition the N files into F batches $\mathcal{W}_1, \ldots, \mathcal{W}_F$, each containing

$$\eta \triangleq \frac{N}{F} \tag{50}$$

files and so that $\mathcal{W}_1, \ldots, \mathcal{W}_F$ form a partition for $\mathcal{W}$. It is implicitly assumed here that η is an integer number.

Let $\mathcal{I}$ be the set of ordinary symbols that occur only once. Then the symbols in $\mathcal{I}$ can be partitioned into K subsets $\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_K$ as follows. For each $s \in \mathcal{I}$, let (i, j) be the unique tuple in $[F] \times [K]$ such that $a_{i,j} = s$. By Definition 8, there exists at least one $k \in [K]\backslash\{j\}$ such that $a_{i,k} = *$. Arbitrarily choose such a k and assign s into $\mathcal{I}_k$.

Let $\mathcal{U}_{i,j}$ denote the set of IVAs for the output function ϕ_j that can be computed from the files in $\mathcal{W}_i$, i.e.,

$$\mathcal{U}_{i,j} \triangleq \{v_{j,n} : w_n \in \mathcal{W}_i\}, \tag{51}$$

and let $\mathcal{A}_k$ denote the set of ordinary symbols in column k having occurrence at least two:

$$\mathcal{A}_k \triangleq \{s \in [S] : a_{i,k} = s \text{ for some } i \in [F]\}\backslash\mathcal{I}, \quad k \in [K]. \tag{52}$$

1) *Map Phase:* Each node k stores

$$\mathcal{M}_k = \bigcup_{\substack{i\in[F]:\\ a_{i,k}=*}} \mathcal{W}_i, \tag{53}$$

and computes the IVAs

$$\mathcal{C}_k = \mathcal{C}_k^{(1)} \cup \mathcal{C}_k^{(2)}, \tag{54}$$

where

$$\mathcal{C}_k^{(1)} = \bigcup_{\substack{i\in[F]:\\ a_{i,k}=*}} \mathcal{U}_{i,k}, \tag{55}$$

$$\mathcal{C}_k^{(2)} = \bigcup_{s\in\mathcal{I}_k\cup\mathcal{A}_k} \bigcup_{\substack{(l,d)\in[F]\times([K]\backslash\{k\}):\\ a_{l,d}=s}} \mathcal{U}_{l,d}, \tag{56}$$

Notice that node k can compute the IVAs in $\mathcal{C}_k^{(1)}$ from the files in $\mathcal{M}_k$, because of (51), (53), and (55). To show that it can also compute the IVAs in $\mathcal{C}_k^{(2)}$ from $\mathcal{M}_k$, we show that if for some $s \in \mathcal{I}_k \cup \mathcal{A}_k$ there exist $(l,d) \in [F] \times ([K]\backslash\{k\})$ so that $a_{l,d} = s$, then

$$a_{l,k} = *. \tag{57}$$

From this follows that $\mathcal{W}_l \subseteq \mathcal{M}_k$. To prove (57), we distinguish the cases $s \in \mathcal{I}_k$ and $s \in \mathcal{A}_k$. In the former case, the proof follows simply by the construction of the set $\mathcal{I}_k$, which implies that if $a_{l,d} = s$ and $s \in \mathcal{I}_k$, then $a_{l,k} = *$. In the latter case, the proof holds because by the definition of the set $\mathcal{A}_k$, if $a_{l,d} = s$ and $s \in \mathcal{A}_k$, then there exists an index $i \in [F]$ so that $a_{i,k} = s$. But by the PDA property C3, $a_{l,d} = a_{i,k} = s$ and $d \neq k$ imply that $l \neq i$ and $a_{l,k} = a_{i,d} = *$.

2) *Shuffle Phase:* For each pair $(i,k) \in [F] \times [K]$ such that $a_{i,k} \neq *$ do the following. Set $s = a_{i,k}$, and let g_s denote the occurrence of the ordinary symbol s in $\mathbf{A}$. If $g_s \geq 2$, or equivalently, $s \in [S]\backslash\mathcal{I}$, partition the IVAs in $\mathcal{U}_{i,k}$ into $g_s - 1$ smaller blocks of equal size. Let $(l_1, j_1), (l_2, j_2), \ldots, (l_{g_s-1}, j_{g_s-1})$ indicate all the other $g_s - 1$ occurrences of the ordinary symbol s other than (i,k):

$$a_{l_1,j_1} = a_{l_2,j_2} = \ldots = a_{l_{g_s-1},j_{g_s-1}} = s. \tag{58}$$

We denote the $g_s - 1$ subblocks of $\mathcal{U}_{i,k}$ by $\mathcal{U}_{i,k}^{j_1}, \ \ldots, \ \mathcal{U}_{i,k}^{j_{g_s-1}}$:

$$\mathcal{U}_{i,k} = \left\{\mathcal{U}_{i,k}^{j_1}, \ \ldots, \ \mathcal{U}_{i,k}^{j_{g_s-1}}\right\}. \tag{59}$$

Node $k \in \mathcal{K}$ then computes X_s^k from $\mathcal{C}_k$ for each $s \in \mathcal{I}_k \cup \mathcal{A}_k$ as defined in the following. For $s \in \mathcal{I}_k$, then there exists unique $(i,j) \in [F] \times [K]\backslash\{k\}$ such that $a_{i,j} = s$, then

$$X_s^k \triangleq \mathcal{U}_{i,j}. \tag{60}$$

For $s \in \mathcal{A}_k$, then

$$X_s^k \triangleq \bigoplus_{\substack{(i,j)\in[F]\times([K]\backslash\{k\}):\\ a_{i,j}=s}} \mathcal{U}_{i,j}^k. \tag{61}$$

Then node k multicasts the signal

$$X_k = \left\{ X_s^k : s \in \mathcal{I}_k \cup \mathcal{A}_k \right\}. \tag{62}$$

3) *Reduce Phase:* Node k has to compute all IVAs in

$$\bigcup_{i \in [F]} \mathcal{U}_{i,k}. \tag{63}$$

In the map phase, node k has already computed all IVAs in $\mathcal{C}_k^{(1)}$. It thus remains to compute all IVAs in

$$\bigcup_{\substack{i \in [F]: \\ a_{i,k} \neq *}} \mathcal{U}_{i,k}. \tag{64}$$

Fix an arbitrary $i \in [F]$ such that $a_{i,k} \neq *$. Set $s = a_{i,k}$. If $s \in \mathcal{A}_k$, each subset $\mathcal{U}_{i,k}^j$ in (59) can be restored by node k from the signal X_s^j sent by node j (see (61)):

$$X_s^j = \bigoplus_{\substack{(l,d) \in [F] \times ([K] \backslash \{j\}): \\ a_{l,d} = s}} \mathcal{U}_{l,d}^j. \tag{65}$$

In fact, for each $\mathcal{U}_{l,d}^j$ in (65), if $d = k$, then $a_{l,d} = a_{i,k} = s$ implies $l = i$ by the PDA property a); if $d \neq k$, then $a_{l,d} = a_{i,k} = s \in \mathcal{A}_k$. This indicates that, the IVAs in $\mathcal{U}_{l,d}^j$ have been computed by node k according to (56) and (59). Therefore, $\mathcal{U}_{i,k}^j$ can be decoded from (65). If $s \notin \mathcal{A}_k$, then $s \in \mathcal{I}$ by (52). There exists thus an index $j \in [K] \backslash \{k\}$ such that $s \in \mathcal{I}_j$ and therefore, by (60), the subset $\mathcal{U}_{i,k}$ can be recovered from the signal X_s^j sent by node j.

B. Performance Analysis

We analyze the performance of the scheme proposed in the preceding subsection.

1) *Storage Load:* Since the Comp-PDA $\mathbf{A}$ has T entries that are "$*$" symbols and each "$*$" symbol indicates that a batch of $\eta = \frac{N}{F}$ files is stored at a given node, see (53), the storage load of the proposed scheme is:

$$r = \frac{\sum_{k=1}^K |\mathcal{M}_k|}{N} = \frac{T \cdot \eta}{N} = \frac{T}{F}. \tag{66}$$

2) *Computation Load:* Since $\mathcal{C}_k^{(1)} \cap \mathcal{C}_k^{(2)} = \emptyset$, we have $|\mathcal{C}_k| = |\mathcal{C}_k^{(1)}| + |\mathcal{C}_k^{(2)}|$. By (55) and (56),

$$\sum_{k=1}^K |\mathcal{C}_k^{(1)}| = T \cdot \eta, \tag{67}$$

$$\sum_{k=1}^K |\mathcal{C}_k^{(2)}| = \sum_{k=1}^K \left[|\mathcal{I}_k| \cdot \eta + \sum_{s \in \mathcal{A}_k} (g_s - 1) \cdot \eta \right] \tag{68}$$

$$= |\mathcal{I}| \cdot \eta + \sum_{s \in [S] \backslash \mathcal{I}} g_s (g_s - 1) \eta, \tag{69}$$

where recall that g_s stands for the number s symbols in $\mathbf{A}$. The computation load of the proposed scheme is then

$$c = \frac{\sum_{k=1}^{K} |\mathcal{C}_k|}{NK} \tag{70}$$

$$= \frac{T \cdot \eta + |\mathcal{I}| \cdot \eta + \sum_{s \in [S] \backslash \mathcal{I}} g_s(g_s - 1) \cdot \eta}{NK} \tag{71}$$

$$= \frac{T}{KF} + \frac{|\mathcal{I}|}{KF} + \sum_{s \in [S] \backslash \mathcal{I}} \frac{g_s(g_s - 1)}{KF}. \tag{72}$$

Recall also that S_t, $t \in [K]$, stands for the number of ordinary symbols that occur t times in $\mathbf{A}$ and that θ_t stands for the fraction of ordinary symbols that occur t times, i.e.,

$$\theta_t = \frac{S_t t}{KF - T}, \quad \forall\, t \in [K]. \tag{73}$$

Then by (72), the computation load of the proposed scheme is

$$c = \frac{T}{KF} + \frac{S_1}{KF} + \sum_{t=2}^{K} \frac{S_t t(t-1)}{KF}$$

$$= \frac{T}{KF} + \frac{S_1}{KF - T} \cdot \frac{KF - T}{KF} + \sum_{t=2}^{K} \frac{S_t t}{KF - T} \cdot \frac{KF - T}{KF} \cdot (t-1) \tag{74}$$

$$= \frac{T}{KF} + \left(1 - \frac{T}{KF}\right) \cdot \left(\theta_1 + \sum_{t=2}^{K} \theta_t (t-1)\right) \tag{75}$$

$$= \frac{T}{KF} + \left(1 - \frac{T}{KF}\right) \cdot \sum_{t=2}^{K} \theta'_t (t-1), \tag{76}$$

where θ'_t was defined in (32).

3) *Communication Load:* Each set of IVAs $\mathcal{U}_{i,j}$ consists of $\frac{N}{F} V = \eta V$ bits. For each $k \in [K]$, node k sends a signal X_s^k for each $s \in \mathcal{I}_k \cup \mathcal{A}_k$. For each $s \in \mathcal{I}_k$, by (60), X_s^k consists of ηV bits. For each $s \in \mathcal{A}_k$, consider now a pair (i, j) where the entry $a_{i,j}$ is an ordinary symbol and occurs g_s times in the Comp-PDA $\mathbf{A}$, where $g_s \geq 2$ by definition of $\mathcal{A}_k$. Then, each subblock $\mathcal{U}_{i,j}^k$ consists of $\frac{\eta V}{g_s - 1}$ bits, and by (61) the signal X_s^k also consists of $\frac{\eta V}{g_s - 1}$ bits. The total length of the signal X_k is thus $l_k = |\mathcal{I}_k| \cdot \eta \cdot V + \sum_{s \in \mathcal{A}_k} \frac{\eta \cdot V}{g_s - 1}$, and the communication load of the proposed scheme:

$$L_{\mathbf{A}} = \frac{\sum_{k=1}^{K} l_k}{NKV} \tag{77}$$

$$= \frac{1}{NKV} \cdot \sum_{k=1}^{K} \left[|\mathcal{I}_k| \cdot \eta \cdot V + \sum_{k \in \mathcal{A}_k} \frac{\eta \cdot V}{g_s - 1} \right] \tag{78}$$

$$= \frac{1}{KF} \cdot \left[|\mathcal{I}| + \sum_{s \in [S] \backslash \mathcal{I}} \frac{g_s}{g_s - 1} \right] \tag{79}$$

$$= \frac{1}{KF} \cdot \left(S_1 + \sum_{t=2}^{K} \frac{S_t t}{t-1} \right) \tag{80}$$

$$= \frac{KF-T}{KF} \cdot \left(\frac{S_1}{KF-T} + \sum_{t=2}^{K} \frac{S_t t}{KF-T} \cdot \frac{1}{t-1} \right) \tag{81}$$

$$= \left(1 - \frac{T}{KF}\right) \cdot \left(\theta_1 + \sum_{t=2}^{K} \frac{\theta_t}{t-1} \right) \tag{82}$$

$$= \left(1 - \frac{T}{KF}\right) \cdot \sum_{t=2}^{K} \frac{\theta'_t}{t-1}. \tag{83}$$

C. File Complexity of the Proposed Schemes

To implement the scheme in Subsection V-A, the files are partitioned into F batches so that each batch contains $\eta = \frac{N}{F} > 0$ files. It is assumed that η is a positive integer. The smallest number of files N where this assumption can be met is F. Therefore, the file complexity of the scheme is F.

VI. Achieving the Fundamental SCC Region (Proof of Theorem 2)

In Corollary 2, we have shown that the SCC triple P_i ($i \in [K]$) is achievable. Therefore, any point on the triangle $\triangle P_{i-1}P_iP_K$ ($i \in [2:K-1]$) can be also be achieved by memory- and time- sharing between the points P_{i-1}, P_i and P_K. This proves the achievability of the surface $\mathcal{O}$. In the following, we only need to prove the converse and identify the Comp-PDAs that achieve the optimal tradeoff surface.

A. Converse

Fix a map-shuffle-reduce procedure, and let $\mathcal{M} = \{\mathcal{M}_k\}_{k=1}^K$, $\mathcal{C} = \{\mathcal{C}_k\}_{k=1}^K$ be their file and IVA allocation. Let further (r, c, L) denote the corresponding storage load, computation load, and communication load, then

$$r = \frac{\sum_{k=1}^K |\mathcal{M}_k|}{N}, \tag{84}$$

$$c = \frac{\sum_{k=1}^K |\mathcal{C}_k|}{NK}, \tag{85}$$

$$L \geq \frac{\sum_{k=1}^K H(X_k)}{NKV}. \tag{86}$$

For any nonempty set $\mathcal{S} \subseteq \mathcal{K}$, denote

$$X_{\mathcal{S}} \triangleq \bigcup_{k \in \mathcal{S}} \{X_k\}, \tag{87}$$

$$\mathcal{V}_{\mathcal{S}} \triangleq \bigcup_{k \in \mathcal{S}} \mathcal{V}_k, \tag{88}$$

$$\mathcal{C}_{\mathcal{S}} \triangleq \bigcup_{k \in \mathcal{S}} \mathcal{C}_k. \tag{89}$$

For any $k \in \mathcal{S}$, $j \in [|\mathcal{S}| - 1]$, define

$$\mathcal{B}_{\mathcal{S},j}^k \triangleq \{v_{k,n} : v_{k,n} \text{ is exclusively computed by } j \text{ nodes in } \mathcal{S} \backslash \{k\}\}. \tag{90}$$

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

$$\mathcal{B}_{\mathcal{S},j} \triangleq \bigcup_{k\in\mathcal{S}} \mathcal{B}^k_{\mathcal{S},j} \tag{91}$$

is

$$b_{\mathcal{S},j} \triangleq |\mathcal{B}_{\mathcal{S},j}| = \sum_{k\in\mathcal{S}} b^k_{\mathcal{S},j}. \tag{92}$$

To prove the converse, we need the following two lemmas, the proofs of which are deferred to Appendix B.

Lemma 1. *For any nonempty set $\mathcal{S} \subseteq \mathcal{K}$ and $\mathcal{S}^c \triangleq \mathcal{K}\backslash\mathcal{S}$,*

$$H(X_{\mathcal{S}}|\mathcal{V}_{\mathcal{S}^c}, \mathcal{C}_{\mathcal{S}^c}) \geq V \sum_{j=1}^{|\mathcal{S}|-1} b_{\mathcal{S},j} \cdot \frac{1}{j}. \tag{93}$$

Lemma 1 includes [4, Claim 1] and [17, Lemma 1] as special cases when one imposes that each node computes *all* IVAs pertaining its stored files. The proof of Lemma 1 follows the same steps as the proof of [4, Claim 1] except that in places the set of IVAs that *can* be computed at a given node k has to be replaced by the set of IVAs that *are effectively* computed at this node. Remarkably the proof steps remain valid, and the so obtained converse is also tight in our more general setup.

Lemma 2. *In* (91) *and* (92)*, let $b_j \triangleq b_{\mathcal{K},j}$ be the cardinality of the set in* (91) *when $\mathcal{S} = \mathcal{K}$. Then*

$$\sum_{j=1}^{K-1} b_j \geq N(K-r), \tag{94}$$

$$\sum_{j=1}^{K-1} (j-1) b_j \leq (c-1)NK. \tag{95}$$

Now, let us define, for each $i \in [K]$,

$$c_i \triangleq \frac{r}{K} + \left(1 - \frac{r}{K}\right) i, \tag{96}$$

and let for a fixed $i \in \{2, \ldots, K-1\}$, the parameters $\lambda_i, \mu_i \in \mathbb{R}^+$ be such that

$$\lambda_i x + \mu_i|_{x=c_{i-1}} = \frac{1}{c_{i-1} - r/K} \left(1 - \frac{r}{K}\right)^2 \tag{97}$$

$$= \frac{1}{i-1} \left(1 - \frac{r}{K}\right), \tag{98}$$

$$\lambda_i x + \mu_i|_{x=c_i} = \frac{1}{c_i - r/K} \left(1 - \frac{r}{K}\right)^2 \tag{99}$$

$$= \frac{1}{i} \left(1 - \frac{r}{K}\right). \tag{100}$$

Notice that by (96), (98), and (100), the following three relationships hold:

$$\lambda_i = -\frac{1}{i(i-1)} < 0, \tag{101}$$

$$\mu_i = \frac{2}{i}\left(1-\frac{r}{K}\right) + \frac{1}{i(i-1)} > 0, \tag{102}$$

$$\lambda_i + \mu_i = \frac{2}{i}\left(1-\frac{r}{K}\right) > 0. \tag{103}$$

Moreover, by its convexity over $x \in [1, +\infty)$, the function $\frac{1}{x-r/K}\left(1-\frac{r}{K}\right)^2 - (\lambda_i x + \mu_i)$ must be nonnegative outside of the interval formed by the two zeros, i.e.,

$$\frac{1}{x-r/K}\left(1-\frac{r}{K}\right)^2 \geq \lambda_i x + \mu_i, \quad \forall x \in [1, c_{i-1}] \cup [c_i, \infty). \tag{104}$$

Therefore,

$$\frac{1}{c_j-r/K}\left(1-\frac{r}{K}\right)^2 \geq \lambda_i c_j + \mu_i, \quad \forall j \in [K-1]. \tag{105}$$

Back to the converse, from (86), the communication load L is lower bounded as

$$L \geq \frac{H(X_{\mathcal{K}})}{NKT} \tag{106}$$

$$\overset{(a)}{\geq} \sum_{j=1}^{K-1} \frac{b_j}{NK} \cdot \frac{1}{j} \tag{107}$$

$$\overset{(b)}{=} \frac{1}{N(K-r)} \cdot \sum_{j=1}^{K-1} b_j \cdot \frac{1}{c_j - r/K}\left(1-\frac{r}{K}\right)^2 \tag{108}$$

$$\overset{(c)}{\geq} \frac{1}{N(K-r)} \sum_{j=1}^{K-1} b_j \left(\lambda_i c_j + \mu_i\right) \tag{109}$$

$$= \frac{1}{N(K-r)} \sum_{j=1}^{K-1} b_j \left(\lambda_i \left(\left(1-\frac{r}{K}\right) j + \frac{r}{K}\right) + \mu_i\right) \tag{110}$$

$$= \frac{1}{N(K-r)} \cdot \left(\lambda_i \left(1-\frac{r}{K}\right) \cdot \sum_{j=1}^{K-1} j b_j + \left(\lambda_i \frac{r}{K} + \mu_i\right) \cdot \sum_{j=1}^{K-1} b_j\right) \tag{111}$$

$$= \frac{\lambda_i}{NK} \cdot \sum_{j=1}^{K-1} (j-1) b_j + \frac{\lambda_i + \mu_i}{N(K-r)} \cdot \sum_{j=1}^{K-1} b_j \tag{112}$$

$$\overset{(d)}{\geq} \frac{\lambda_i}{NK} \cdot (c-1)NK + \frac{\lambda_i + \mu_i}{N(K-r)} \cdot N(K-r) \tag{113}$$

$$= \lambda_i c + \mu_i \tag{114}$$

$$\overset{(e)}{=} -\frac{1}{i(i-1)} c - \frac{2}{Ki} r + \frac{2i-1}{i(i-1)}, \tag{115}$$

where (a) follows from Lemma 1 by setting $\mathcal{S} = \mathcal{K}$; (b) follows from (96); (c) follows from (105); (d) follows from (94), (95) and (101), (103); and (e) follows from (101), (102). Since the SCC triples P_{i-1}, P_i, and P_K defined in (37) satisfy inequality (115) with equality, the above inequalities indicate that all feasible triples (r, c, L) must

lie above the plane containing $\triangle P_{i-1}P_iP_K$. Furthermore, the converse in [4] (cf. (43)) implies that, for any c, we have

$$L \geq -\frac{1}{i(i+1)}r + \frac{1}{i} + \frac{1}{i+1} - \frac{1}{K}, \quad i \in [K-1]. \tag{116}$$

Therefore, all feasible triples (r, c, L) must lie above the plane containing P_1, P_2, Q_2 and the planes containing $P_i, P_{i+1}, Q_{i+1}, Q_i$, for $i = 2, \ldots, K-1$, respectively. In conclusion, all feasible (r, c, L) must lie above the surface $\mathcal{F}$.

B. Comp-PDAs Achieving the Optimal SCC Tradeoff Surface

Fix a Comp-PDA $\mathbf{A}$, and let $r_{\mathbf{A}}$, $c_{\mathbf{A}}$, $L_{\mathbf{A}}$ denote respectively the storage, computation, and communication loads of the associated coded computing scheme. Obviously, $\mathbf{A}$ achieves the point P_K if, and only if, it is trivial. In the following, we assume that $\mathbf{A}$ is a non-trivial Comp-PDA, in which case $r_{\mathbf{A}} < K$.

As before, let θ_t denote the fraction of ordinary symbols that occur exactly t times, for each $t \in [K]$ and define θ'_t as in (32). Then for each $t \in [2:K]$, define

$$c_{\mathbf{A}}^{(t)} \triangleq \frac{T}{KF} + \left(1 - \frac{T}{KF}\right)(t-1) \tag{117a}$$

$$L_{\mathbf{A}}^{(t)} \triangleq \frac{1}{t-1}\left(1 - \frac{T}{KF}\right) \tag{117b}$$

and notice that by Theorem 1:

$$r_{\mathbf{A}} = \frac{T}{F} \tag{118a}$$

$$c_{\mathbf{A}} = \sum_{t=2}^{K} \theta'_i c_{\mathbf{A}}^{(t)} \tag{118b}$$

$$L_{\mathbf{A}} = \sum_{t=2}^{K} \theta'_i L_{\mathbf{A}}^{(t)}. \tag{118c}$$

Fix $i \in [2:K-1]$ and define

$$\beta_i \triangleq -\frac{1}{i(i-1)}, \tag{119a}$$

$$\gamma_i \triangleq \frac{2}{i}\left(1 - \frac{T}{KF}\right) + \frac{1}{i(i-1)}. \tag{119b}$$

Now, recall that $(r_{\mathbf{A}}, c_{\mathbf{A}}, L_{\mathbf{A}})$ lies on the triangle $\triangle P_{i-1}P_iP_K$ if, and only if,

$$L_{\mathbf{A}} = -\frac{1}{i(i-1)}c_{\mathbf{A}} - \frac{2}{Ki}r_{\mathbf{A}} + \frac{2i-1}{i(i-1)}. \tag{120}$$

In the following, we show that (120) holds if, and only if,

$$\theta'_t = 0, \ \forall\ t \in [2:K] \backslash \{i, i+1\}. \tag{121}$$

With the definition of θ'_t in (32), this proves that a Comp-PDA achieves a point on the triangle $\triangle P_{i-1}P_iP_K$ if, and only if, the following condition holds:

- if $i = 2$, then each ordinary symbol of the Comp-PDA occurs at most 3 times;
- if $i \in [3 : K-1]$, then the Comp-PDA is almost i-regular.

This concludes the proof of Theorem 2 and also proves Remark 3.

Next, we write

$$L_{\mathbf{A}} = \sum_{t=2}^{K} \theta'_t L_{\mathbf{A}}^{(t)} \tag{122}$$

$$\overset{(a)}{=} \sum_{t=2}^{K} \theta'_t \cdot \frac{1}{c_{\mathbf{A}}^{(t)} - T/(KF)} \left(1 - \frac{T}{KF}\right)^2 \tag{123}$$

$$\overset{(b)}{\geq} \sum_{t=2}^{K} \theta'_t \cdot \left(\beta_i c_{\mathbf{A}}^{(t)} + \gamma_i\right) \tag{124}$$

$$\overset{(c)}{=} \beta_i c_{\mathbf{A}} + \gamma_i \tag{125}$$

$$\overset{(d)}{=} -\frac{1}{i(i-1)} c_{\mathbf{A}} + \frac{2}{i}\left(1 - \frac{T}{KF}\right) + \frac{1}{i(i-1)} \tag{126}$$

$$\overset{(e)}{=} -\frac{1}{i(i-1)} c_{\mathbf{A}} - \frac{2}{Ki} r_{\mathbf{A}} + \frac{2i-1}{i(i-1)}, \tag{127}$$

where (a) holds by simple algebraic manipulations on (117); (b) is proved below; (c) holds by (118) and because $\sum_{t=2}^{K} \theta'_t = \sum_{t=1}^{K} \theta_t = 1$; (d) holds by (119); and (e) holds by (118a).

To see why step (b) holds, define the two functions over the interval $\left(\frac{T}{KF}, +\infty\right)$,

$$f_1 \colon c \mapsto \beta_i c + \gamma_i, \tag{128}$$

$$f_2 \colon c \mapsto \frac{1}{c - T/(KF)} \left(1 - \frac{T}{KF}\right)^2. \tag{129}$$

Notice that f_2 is strictly convex and it intersects f_1 at the points $\left(c_{\mathbf{A}}^{(i)}, L_{\mathbf{A}}^{(i)}\right)$ and $\left(c_{\mathbf{A}}^{(i+1)}, L_{\mathbf{A}}^{(i+1)}\right)$. Therefore,

$$\frac{1}{c - T/(KF)} \left(1 - \frac{T}{KF}\right)^2 \geq \beta_i c + \gamma_i, \qquad \forall\ c \in \left[1, c_{\mathbf{A}}^{(i)}\right] \cup \left[c_{\mathbf{A}}^{(i+1)}, \infty\right), \tag{130}$$

with equality if and only if $c \in \left\{c_{\mathbf{A}}^{(i)}, c_{\mathbf{A}}^{(i+1)}\right\}$. In particular,

$$\frac{1}{c_{\mathbf{A}}^{(t)} - T/(KF)} \left(1 - \frac{T}{KF}\right)^2 \geq \beta_i c_{\mathbf{A}}^{(t)} + \gamma_i, \qquad \forall\ t \in [2 : K], \tag{131}$$

with equality if and only if $t \in \{i, i+1\}$.

Combining this with (127), we conclude that (120) holds if and only if $\theta'_t = 0$ for all $t \in [2 : K] \backslash \{i, i+1\}$, i.e., (121) holds.

VII. Reducing the File Complexity (Proof of Theorems 3 and 4)

A. Lowest File Complexity of P_i (Proof of Theorem 3)

For $i = K$, the conclusion $F \geq \binom{K}{K} = 1$ is trivial.

We thus assume in the following that $i \in [2 : K-1]$. Fix such an i and choose a (K, F, T, S) Comp-PDA $\mathbf{A}$ that achieves the point P_i. Notice that by Remark 3, $\mathbf{A}$ needs to be $(i+1)$-regular. In the following, we show that it $\mathbf{A}$ needs to have exactly i "$*$" symbols in each row. By Lemma 3 at the end of this subsection, this will conclude the proof.

Notice first that if an ordinary symbol occurs $i+1$ times, then each row where it occurs must contain at least i "$*$" entries. (Namely in the columns where this symbol occurs in the other rows.) Thus, if t_j denotes the number of "$*$" entries in the j-th row of $\mathbf{A}$, then

$$t_j \geq i, \quad \forall\, j \in [F], \tag{132}$$

and by summing over $j \in [F]$:

$$\sum_{j=1}^{F} t_j \geq iF. \tag{133}$$

However, since by Theorem 1 and Corollary 2 for the point P_i the storage load is $r_{P_i} = \frac{T}{F} = i$:

$$\sum_{j=1}^{F} t_j = T = iF, \tag{134}$$

and thus both the inequalities (132) and (133) must hold with equality. Therefore, each row of $\mathbf{A}$ has exactly i "$*$" entries and the following lemma (which rephrases [32, Lemma 2]) concludes the proof.

Lemma 3 (From [32])**.** *Consider a g-regular (K, F, T, S) Comp-PDA with exactly $g-1$ "$*$" entries in each row where $g \geq 2$. Then $F \geq \binom{K}{g-1}$.*

B. New Comp-PDAs with Reduced File Complexity (Proof of Theorem 4)

First, the case for $q = 1$ is trivial, since $D_q = P_K = (K, 1, 0)$. In this case, the Comp-PDA is the trivial $(K, 1, K, 0)$ PDA with only "$*$" entries.

In the following, we consider an arbitrary integer q such that

$$1 \leq q < \frac{K}{2} \tag{135a}$$

and set

$$m \triangleq \left\lceil \frac{K}{q} \right\rceil - 1. \tag{135b}$$

We construct an almost $\lfloor \frac{K}{q} \rfloor$-regular (K, F, T, S) Comp-PDA with

$$F = q^m \tag{135c}$$

that achieves a point on the triangle $\triangle P_{m-1}P_mP_K$.

To present the new construction, we first introduce some notations. For a given $j \in [q^m]$, let $(j_{m-1}, j_{m-2}, \ldots, j_0) \in \mathbb{Z}_q^m$ be the unique tuple that satisfies:

$$j - 1 = j_{m-1}q^{m-1} + j_{m-2}q^{m-2} + \ldots + j_0. \tag{136}$$

For convenience, we will write

$$j = (j_{m-1}, j_{m-2}, \ldots, j_0)_q. \tag{137}$$

Similarly, for a given $s \in [(q-1)q^m]$, let $(s_m, s_{m-1}, s_{m-2}, \ldots, s_0) \in \mathbb{Z}_{q-1} \times \mathbb{Z}_q^m$ be the unique tuple that satisfies:

$$s - 1 = s_m q^m + s_{m-1}q^{m-1} + s_{m-2}q^{m-2} + \ldots + s_0, \tag{138}$$

For convenience of notation, we will write

$$s = (s_m, s_{m-1}, \ldots, s_0)_q^{q-1}. \tag{139}$$

For a given $k \in [K]$, let (k_1, k_0) be the unique pair that satisfies

$$k - 1 = k_1 q + k_0 \tag{140}$$

for some $k_1 \in [0 : m]$ and $k_0 \in [0 : q-1]$. We will write

$$k = (k_1, k_0)_q^{m+1}. \tag{141}$$

Construction 1. Consider a fixed positive integers K and let q, m, F be given as in (135). Construct the array $\mathbf{A}_q^K = [a_{j,k}]$ as follows

- If $k \leq qm$:

$$a_{j,k} = \begin{cases} *, & \text{if } j_{k_1} = k_0 \\ \big(j_{k_1} \ominus_q k_0 \ominus_q 1, j_{m-1}, j_{m-2}, \ldots, j_{k_1+1}, k_0, j_{k_1-1}, \ldots, j_0\big)_q^{q-1}, & \text{if } j_{k_1} \neq k_0 \end{cases}, \tag{142}$$

- and if $k > qm$:

$$a_{j,k} = \begin{cases} *, & \text{if } \sum_{l=0}^{m-1} j_l = k_0 \\ \big(k_0 \ominus_q \sum_{l=0}^{m-1} j_l \ominus_q 1, j_{m-1}, j_{m-2}, \ldots, j_0\big)_q^{m+1}, & \text{if } \sum_{l=0}^{m-1} j_l \neq k_0 \end{cases}, \tag{143}$$

where "$\ominus_q$" denotes minus modulo q, and the sum operation "$\sum$" is in modulo q.

Table I depicts $\mathbf{A}_2^5$. It coincides with the Comp-PDA $\mathbf{A}$ in (27), but uses a different notation for the ordinary symbols.

TABLE I: The construction of array $\mathbf{A}_2^5$.

$(j_1,j_0)_2 \backslash (k_1,k_0)_2^3$	$(0,0)_2^3$	$(0,1)_2^3$	$(1,0)_2^3$	$(1,1)_2^3$	$(2,0)_2^3$
$(0,0)_2$	$*$	$(0,0,1)_2^1$	$*$	$(0,1,0)_2^1$	$*$
$(0,1)_2$	$(0,0,0)_2^1$	$*$	$*$	$(0,1,1)_2^1$	$(0,0,1)_2^1$
$(1,0)_2$	$*$	$(0,1,1)_2^1$	$(0,0,0)_2^1$	$*$	$(0,1,0)_2^1$
$(1,1)_2$	$(0,1,0)_2^1$	$*$	$(0,0,1)_2^1$	$*$	$*$

In the following Lemma, we prove that all arrays from Construction 1 are indeed PDA.

Lemma 4. *For any given positive integers K and q such that $2 \leq q < \frac{K}{2}$, the array $\mathbf{A}_q^K$ is an almost g-regular $\left(K, q^m, Kq^{m-1}, (q-1)q^m\right)$ Comp-PDA for $g \triangleq \left\lfloor \frac{K}{q} \right\rfloor$ and $m \triangleq \lceil \frac{K}{q} \rceil - 1$.*

Proof: See Appendix C. ∎

For $q|K$ the Comp-PDA $\mathbf{A}_q^K$ specializes to the PDA proposed in [32, Theorem 4].

We now prove that the proposed Comp-PDA $\mathbf{A}_q^K$ satisfies the properties claimed in the theorem. Lemma 4 and Theorem 1 readily yield (46). Next, we prove (48), i.e., that D_q is close to the SCC triple

$$P_m = \left(m, m\left(1 - \frac{m-1}{K}\right), \frac{1}{m}\left(1 - \frac{m}{K}\right)\right). \tag{144}$$

Combining this with (46), yields

$$r_{P_m} - r_{D_q} = m - \frac{K}{q}, \tag{145}$$

$$c_{P_m} - c_{D_q} = m\left(1 - \frac{m-1}{K}\right) - \frac{1}{q} - m\left(1 - \frac{1}{q}\right)\left(2 - \frac{(m+1)q}{K}\right) \tag{146}$$

$$= -\frac{1}{K}m^2 + \left(1 + \frac{1}{K}\right)m - \frac{1}{q} + \frac{q-1}{K}m^2 - \frac{(2K-q)(q-1)}{Kq}m \tag{147}$$

$$= \frac{q-2}{K}m^2 + \frac{q^2 - Kq + 2K}{Kq}m - \frac{1}{q}, \tag{148}$$

$$L_{P_m} - L_{D_q} = \frac{1}{m}\left(1 - \frac{m}{K}\right) - \left(\frac{1}{m} + \frac{(m+1)q - K}{K(m-1)}\right) \cdot \left(1 - \frac{1}{q}\right) \tag{149}$$

$$= \frac{1}{m} - \frac{1}{K} - \frac{1}{m}\left(1 - \frac{1}{q}\right) - \frac{(m+1)q - K}{K(m-1)}\left(1 - \frac{1}{q}\right) \tag{150}$$

$$= \frac{1}{mq} - \frac{1}{K} - \frac{(m-1)q + 2q - K}{K(m-1)} \cdot \frac{q-1}{q} \tag{151}$$

$$= \frac{1}{mq} + \frac{(K-2q)(q-1)}{K(m-1)q} - \frac{q}{K}. \tag{152}$$

Therefore, from $\frac{K}{q} - 1 \leq m \leq \frac{K}{q}$ and the above evaluations,

1) (48a) follows immediately from (145);
2) (148) is quadratic in m and increases with m over the interval $\left[\frac{K}{q} - 1, \frac{K}{q}\right]$;

3) since $L_{P_m} - L_{D_q}$, given by (152), decreases with m, we obtain (48c).

Finally, we compare the file complexities of D_q and P_m:

$$F_{P_m} = \binom{K}{m} \overset{(a)}{\geq} \binom{mq}{m} \tag{153}$$

$$= \frac{(mq)!}{m!\,(m(q-1))!} \tag{154}$$

$$\overset{(b)}{\geq} \frac{\sqrt{2\pi}}{e^2} \frac{(mq)^{mq}\sqrt{mq}}{(m)^m\sqrt{m}(m(q-1))^{m(q-1)}\sqrt{m(q-1)}} \tag{155}$$

$$= \frac{\sqrt{2\pi}}{e^2}\sqrt{\frac{q}{m(q-1)}} \cdot \left(\frac{q^q}{(q-1)^{q-1}}\right)^m \tag{156}$$

$$= \frac{\sqrt{2\pi}}{e^2}\sqrt{\frac{q}{m(q-1)}} \left(\frac{q}{q-1}\right)^{m(q-1)} F_{D_q}, \tag{157}$$

where (a) follows since $K \geq mq$, (b) by applying Stirling's approximation $\sqrt{2\pi}\ n^{n+\frac{1}{2}}e^{-n} \leq n! \leq e\ n^{n+\frac{1}{2}}e^{-n}$ to both the numerator and the denominator.

VIII. Conclusion

We presented a framework for designing schemes from Comp-PDAs (placement delivery arrays for coded computing) for map-reduce like distributed computing systems, and expressed the storage, computation, communication (SCC) loads of the schemes in terms of the Comp-PDA parameters. The pareto optimal SCC tradeoff surface and the set of Comp-PDAs achieving these surface points were completely characterized. Moreover, we showed that while the corner points of the pareto optimal SCC surface can only be achieved with a large number of files, other points on this surface, which lie close to the corner points, can be achieved with a significantly smaller number of files.

Acknowledgement

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

Appendix A
The Properties of the Surfaces $\mathcal{F}$ and $\mathcal{O}$

A. Proof of Properties of Surface $\mathcal{F}$

That $\mathcal{F}$ is connected and continuous, follows simply because it can be obtained by successively pasting a triangle or a trapezoid to the boundary of the previously obtained region.

We turn to prove that for each pair (r, c) satisfying (23a), there exists exactly one point $(r, c, L) \in \mathcal{F}$. That there exists *at leaset one* such point follows by the continuity of $\mathcal{F}$ and because the triangle obtained by projecting the line segments $\overline{P_1Q_2}$, $\overline{Q_2Q_3}$, $\overline{Q_3Q_4}$, $\overline{Q_4Q_5}$, …, $\overline{Q_{K-1}Q_K}$, $\overline{Q_KP_K}$, $\overline{P_KP_1}$ onto the (r, c)-plane, contains all extreme points (r, c) that satisfy (23a). On the other hand, for each (r, c) there is not more than one point $(r, c, L) \in \mathcal{F}$, because none of the triangles and trapezoids that build $\mathcal{F}$ is vertical and the projections of any two facets in $\mathcal{F}$ onto the (r, c)-plane have nonoverlapping interiors.

B. Proof of Optimal Tradeoff Surface in Theorem 2

We now prove that $\mathcal{O}$ is the optimal tradeoff surface of the region $\mathcal{R}$. Obviously, all pareto-optimal points must lie on the surface $\mathcal{F}$. Since the triangle $\triangle P_1P_2Q_2$ and the trapezoids $\boxminus P_iQ_iQ_{i+1}P_{i+1}$ ($i \in [2:K-1]$) are parallel to $\overrightarrow{\mathbf{e}_2}$, all points in the interior of these facets cannot be pareto-optimal. In the following, we prove that, all the points on the triangles $\triangle P_{i-1}P_iP_K$ ($i \in [2:K-1]$) must be pareto-optimal.

For any (r,c) satisfying (23a), let $L^*(r,c)$ be the function such that $(r,c,L^*(r,c)) \in \mathcal{F}$. Then by (115), it has strictly positive directional derivative in any direction $(r \leq 0, c \leq 0)$ in the interior of the projection of $\triangle P_{i-1}P_iP_K$ on the (r,c) plane.

Fix now a triple $(r,c,L^*(r,c)) \in \mathcal{O}$. We show that it is pareto-optimal. To this end, consider any other triple $(r',c',L') \in \mathcal{R}$ that satisfies

$$r' \leq r, \quad c' \leq c, \quad L' \leq L^*(r,c). \tag{158}$$

We show by contradiction that all three inequalities must hold with equality. We distinguish between triples (r',c',L') for which

$$(r',c',L^*(r',c')) \in \mathcal{O}, \tag{159}$$

and triples where this is not the case.

1) Assume that (159) holds. If $r' = r$ and $c' = c$, then obviously, $L' \geq L^*(r,c)$, thus all equalities in (158) hold. If $r' < r$ or $c' < c$, then

$$L^*(r',c') > L^*(r,c), \tag{160}$$

simply because the directional derivative along $(r'-r, c'-c)$ is strictly positive by (115). Since $(r',c',L') \in \mathcal{R}$, we have $L' \geq L^*(r',c')$ and thus by (160), $L' > L^*(r,c)$, which contradicts (158).

2) Assume now that (159) is violated. Then, $(r',c',L^*(r',c'))$ must lie on at least one of the $K-1$ facets

$$\triangle P_1P_2Q_2 \quad \text{or} \quad \boxminus P_iQ_iQ_{i+1}P_{i+1}, \quad i = 2,\ldots,K-1. \tag{161}$$

As they are all parallel to $\overrightarrow{\mathbf{e}_2}$, there exists $c'' < c' \leq c$ such that, $(r',c'',L^*(r',c'')) \in \mathcal{O}$ and $L^*(r',c'') = L^*(r',c')$. Therefore,

$$L' \geq L^*(r',c') = L^*(r',c'') \overset{(a)}{>} L^*(r,c), \tag{162}$$

where (a) follows by proof step 1). But (162) contradicts with (158).

From the above analysis, we conclude that, any point on $\mathcal{O}$ is pareto-optimal.

APPENDIX B
PROOF OF LEMMAS 1 AND 2

A. 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 $\mathcal{S} \subseteq \mathcal{K}$. We prove Lemma 1 by mathematical induction on the size of $\mathcal{S}$:

When $|\mathcal{S}| = 1$, W.L.O.G, assume $\mathcal{S} = \{k\}$, then (93) becomes

$$H\left(X_k|\mathcal{V}_{\mathcal{K}\setminus\{k\}}, \mathcal{C}_{\mathcal{K}\setminus\{k\}}\right) \geq 0, \tag{163}$$

which is trivial.

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

$$\begin{aligned}
&H(X_{\mathcal{S}}|Y_{\mathcal{S}^c}) && (164)\\
&= \frac{1}{|\mathcal{S}|}\sum_{k\in\mathcal{S}} H(X_{\mathcal{S}}, X_k|Y_{\mathcal{S}^c}) && (165)\\
&= \frac{1}{|\mathcal{S}|}\sum_{k\in\mathcal{S}} \left(H(X_k|Y_{\mathcal{S}^c}) + H(X_{\mathcal{S}}|X_k, Y_{\mathcal{S}^c})\right) && (166)\\
&= \frac{1}{|\mathcal{S}|}\sum_{k\in\mathcal{S}} H(X_k|Y_{\mathcal{S}^c}) + \frac{1}{|\mathcal{S}|}\sum_{k\in\mathcal{S}} H(X_{\mathcal{S}}|X_k, Y_{\mathcal{S}^c}) && (167)\\
&\geq \frac{1}{|\mathcal{S}|}H(X_{\mathcal{S}}|Y_{\mathcal{S}^c}) + \frac{1}{|\mathcal{S}|}\sum_{k\in\mathcal{S}} H(X_{\mathcal{S}}|X_k, Y_{\mathcal{S}^c}). && (168)
\end{aligned}$$

Then from (168), we have

$$\begin{aligned}
&H(X_{\mathcal{S}}|Y_{\mathcal{S}^c}) && (169)\\
&= \frac{1}{|\mathcal{S}| - 1}\sum_{k\in\mathcal{S}} H(X_{\mathcal{S}}|X_k, Y_{\mathcal{S}^c}) && (170)\\
&\geq \frac{1}{s}\sum_{k\in\mathcal{S}} H(X_{\mathcal{S}}|X_k, \mathcal{C}_k, Y_{\mathcal{S}^c}) && (171)\\
&\overset{(a)}{=} \frac{1}{s}\sum_{k\in\mathcal{S}} H(X_{\mathcal{S}}|\mathcal{C}_k, Y_{\mathcal{S}^c}) && (172)\\
&\overset{(b)}{=} \frac{1}{s}\sum_{k\in\mathcal{S}} \left(H(X_{\mathcal{S}}|\mathcal{C}_k, Y_{\mathcal{S}^c}) + H(\mathcal{V}_k|X_{\mathcal{S}}, \mathcal{C}_k, Y_{\mathcal{S}^c})\right) && (173)\\
&\overset{(c)}{=} \frac{1}{s}\sum_{k\in\mathcal{S}} H(X_{\mathcal{S}}, \mathcal{V}_k|\mathcal{C}_k, Y_{\mathcal{S}^c}) && (174)\\
&\overset{(d)}{=} \frac{1}{s}\sum_{k\in\mathcal{S}} \left(H(\mathcal{V}_k|\mathcal{C}_k, Y_{\mathcal{S}^c}) + H(X_{\mathcal{S}}|\mathcal{V}_k, \mathcal{C}_k, Y_{\mathcal{S}^c})\right), && (175)
\end{aligned}$$

where (a) follows from the fact that X_k is a function of $\mathcal{C}_k$; (b) holds because

$$H(\mathcal{V}_k|X_{\mathcal{S}}, \mathcal{C}_k, Y_{\mathcal{S}^c}) = 0, \tag{176}$$

since $\mathcal{V}_k$ can be decoded by $\mathcal{C}_k, X_{\mathcal{S}}$ and $X_{\mathcal{S}^c}$, which is a function of $Y_{\mathcal{S}^c}$; and $(c), (d)$ follows from chain rule.

We proceed to derive a lower bound for each term in (175). Firstly,

$$\begin{aligned} H(\mathcal{V}_k|\mathcal{C}_k, Y_{\mathcal{S}^c}) &= H(\mathcal{V}_k|\mathcal{C}_k, \mathcal{V}_{\mathcal{S}^c}, \mathcal{C}_{\mathcal{S}^c}) && (177)\\ &\overset{(a)}{=} H(\mathcal{V}_k|\mathcal{C}_k, \mathcal{C}_{\mathcal{S}^c}) && (178)\\ &= H(\mathcal{V}_k|\mathcal{C}_{(\mathcal{S}\backslash\{k\})^c}) && (179)\\ &\overset{(b)}{=} V \cdot \sum_{j=1}^{|\mathcal{S}|-1} b^k_{\mathcal{S},j} && (180)\\ &= V \cdot \sum_{j=1}^{s} b^k_{\mathcal{S},j}, && (181) \end{aligned}$$

where (a) follows from the independence between $\mathcal{V}_k$ and $\mathcal{V}_{\mathcal{S}^c}$; and (b) holds because

$$\mathcal{B}^k_{\mathcal{S},1}, \ldots, \mathcal{B}^k_{\mathcal{S},|\mathcal{S}|-1} \tag{182}$$

form a partition of those IVAs of $\mathcal{V}_k$ that are not computed by any node in $(\mathcal{S}\backslash\{k\})^c$. Secondly,

$$\begin{aligned} & H(X_{\mathcal{S}}|\mathcal{V}_k, \mathcal{C}_k, Y_{\mathcal{S}^c}) && (183)\\ &= H(X_{\mathcal{S}\backslash\{k\}}, X_k|\mathcal{V}_k, \mathcal{C}_k, Y_{\mathcal{S}^c}) && (184)\\ &\overset{(a)}{=} H(X_{\mathcal{S}\backslash\{k\}}|\mathcal{V}_k, \mathcal{C}_k, Y_{\mathcal{S}^c}) && (185)\\ &= H(X_{\mathcal{S}\backslash\{k\}}|Y_{(\mathcal{S}\backslash\{k\})^c}) && (186)\\ &\overset{(b)}{\geq} V \cdot \sum_{j=1}^{s-1} b_{\mathcal{S}\backslash\{k\},j} \cdot \frac{1}{j}, && (187) \end{aligned}$$

where (a) follows from the fact that X_k is a function of $\mathcal{C}_k$; and (b) follows from the induction assumption.

Finally, combining (175), (181) and (187), we have

$$\begin{aligned} & H(X_{\mathcal{S}}|Y_{\mathcal{S}^c}) && (188)\\ &\geq \frac{1}{s}\sum_{k\in\mathcal{S}} H(\mathcal{V}_k|\mathcal{C}_k, Y_{\mathcal{S}^c}) + \frac{1}{s}\sum_{k\in\mathcal{S}} H(X_{\mathcal{S}}|\mathcal{V}_k, \mathcal{C}_k, Y_{\mathcal{S}^c}) && (189)\\ &\geq \frac{V}{s}\sum_{k\in\mathcal{S}}\sum_{j=1}^{s} b^k_{\mathcal{S},j} + \frac{V}{s}\sum_{k\in\mathcal{S}}\sum_{j=1}^{s-1} b_{\mathcal{S}\backslash\{k\},j} \cdot \frac{1}{j} && (190)\\ &= \frac{V}{s}\sum_{j=1}^{s}\sum_{k\in\mathcal{S}} b^k_{\mathcal{S},j} + \frac{V}{s}\sum_{j=1}^{s-1}\sum_{k\in\mathcal{S}} b_{\mathcal{S}\backslash\{k\},j} \cdot \frac{1}{j} && (191)\\ &\overset{(a)}{=} \frac{V}{s}\sum_{j=1}^{s} b_{\mathcal{S},j} + \frac{V}{s}\sum_{j=1}^{s-1}\sum_{k\in\mathcal{S}} b_{\mathcal{S}\backslash\{k\},j} \cdot \frac{1}{j}, && (192) \end{aligned}$$

where (a) follows from (92).

Let $\mathbb{I}(A)$ be the indicator function of an event A, i.e.,

$$\mathbb{I}(A) = \begin{cases} 1, & \text{if } A \text{ is true} \\ 0, & \text{if } A \text{ is false} \end{cases}. \tag{193}$$

Then,

$$\sum_{k\in\mathcal{S}} b_{\mathcal{S}\backslash\{k\},j} \tag{194}$$

$$\overset{(a)}{=} \sum_{k\in\mathcal{S}} \sum_{l\in\mathcal{S}\backslash\{k\}} b^l_{\mathcal{S}\backslash\{k\},j} \tag{195}$$

$$= \sum_{l\in\mathcal{S}} \sum_{k\in\mathcal{S}\backslash\{l\}} b^l_{\mathcal{S}\backslash\{k\},j} \tag{196}$$

$$\overset{(b)}{=} \sum_{l\in\mathcal{S}} \sum_{k\in\mathcal{S}\backslash\{l\}} \sum_{n=1}^{N} \mathbb{I}(v_{l,n} \text{ is only computed by } j \text{ nodes in } \mathcal{S}\backslash\{k,l\}) \tag{197}$$

$$= \sum_{l\in\mathcal{S}} \sum_{k\in\mathcal{S}\backslash\{l\}} \sum_{n=1}^{N} \mathbb{I}(v_{l,n} \text{ is only computed by } j \text{ nodes in } \mathcal{S}\backslash\{l\}) \cdot \mathbb{I}(v_{l,n} \text{ is not computed by node } k) \tag{198}$$

$$= \sum_{l\in\mathcal{S}} \sum_{n=1}^{N} \mathbb{I}(v_{l,n} \text{ is only computed by } j \text{ nodes in } \mathcal{S}\backslash\{l\}) \cdot \sum_{k\in\mathcal{S}\backslash\{l\}} \mathbb{I}(v_{l,n} \text{ is not computed by node } k) \tag{199}$$

$$= \sum_{l\in\mathcal{S}} \sum_{n=1}^{N} \mathbb{I}(v_{l,n} \text{ is only computed by } j \text{ nodes in } \mathcal{S}\backslash\{l\}) \cdot \ (s-j) \tag{200}$$

$$\overset{(c)}{=} \sum_{l\in\mathcal{S}} b^l_{\mathcal{S},j}(s-j) \tag{201}$$

$$\overset{(d)}{=} b_{\mathcal{S},j}(s-j), \tag{202}$$

where $(a), (d)$ follows from (92), and $(b), (c)$ follows from the definition of $b^l_{\mathcal{S}\backslash\{k\},j}$ and $b^l_{\mathcal{S},j}$ respectively.

Therefore, from (192) and (202),

$$H(X_{\mathcal{S}}|Y_{\mathcal{S}^c}) \geq \frac{V}{s} \cdot \sum_{j=1}^{s} b_{\mathcal{S},j} + \frac{V}{s} \cdot \sum_{j=1}^{s-1} \sum_{k\in\mathcal{S}} b_{\mathcal{S}\backslash\{k\},j} \cdot \frac{1}{j} \tag{203}$$

$$= \frac{V}{s} \cdot \sum_{j=1}^{s} b_{\mathcal{S},j} + \frac{V}{s} \cdot \sum_{j=1}^{s-1} b_{\mathcal{S},j} \cdot \frac{s-j}{j} \tag{204}$$

$$= \frac{V}{s} \cdot \sum_{j=1}^{s} b_{\mathcal{S},j} + V \cdot \sum_{j=1}^{s-1} \frac{b_{\mathcal{S},j}}{j} - \frac{V}{s} \cdot \sum_{j=1}^{s-1} b_{\mathcal{S},j} \tag{205}$$

$$= V \cdot \frac{b_{\mathcal{S},s}}{s} + V \cdot \sum_{j=1}^{s-1} \frac{b_{\mathcal{S},j}}{j} \tag{206}$$

$$= V \cdot \sum_{j=1}^{|\mathcal{S}|-1} \frac{b_{\mathcal{S},j}}{j}. \tag{207}$$

Notice that, we have proved that (93) holds for all $\mathcal{S} \subseteq \mathcal{K}$ with $|\mathcal{S}| = s+1$. By the principle of mathematical

induction, we conclude that (93) holds for all nonempty subsets $\mathcal{S} \subseteq \mathcal{K}$.

B. Proof of Lemma 2

For any $k \in \mathcal{K}$, define

$$\widetilde{\mathcal{B}}_k \triangleq \{v_{k,n} : v_{k,n} \text{ is computed by node } k\}, \tag{208}$$

i.e., $\widetilde{\mathcal{B}}_k$ is the set of IVAs for the output function ϕ_k that are computed by node k. Denote the cardinality of $\widetilde{\mathcal{B}}_k$ by $\widetilde{b}_k$. Notice that the sets

$$\widetilde{\mathcal{B}}_k, \mathcal{B}^k_{\mathcal{K},1}, \mathcal{B}^k_{\mathcal{K},2}, \ldots, \mathcal{B}^k_{\mathcal{K},K-1} \tag{209}$$

form a partition of the set of IVAs $\mathcal{V}_k$. Thus,

$$\widetilde{b}_k + \sum_{j=1}^{K-1} b^k_{\mathcal{K},j} = |\mathcal{V}_k| = N. \tag{210}$$

Therefore, summing over $k \in \mathcal{K}$ in (210), together with (92),

$$\sum_{k=1}^{K} \widetilde{b}_k + \sum_{j=1}^{K-1} b_j = NK. \tag{211}$$

Moreover, since each node k must store file w_n when $v_{k,n} \in \widetilde{\mathcal{B}}_k$, we have $\widetilde{b}_k \leq |\mathcal{M}_k|$, and by (84),

$$\sum_{k=1}^{K} \widetilde{b}_k \leq \sum_{k=1}^{K} |\mathcal{M}_k| = rN. \tag{212}$$

Also, for $k \in \mathcal{K}$ and $j \in [K-1]$, IVAs $\widetilde{\mathcal{B}}_k$ must be computed at node k, and IVAs $\mathcal{B}^k_{\mathcal{K},j}$ must be computed at j nodes. Thus by (85),

$$\sum_{k=1}^{K} \widetilde{b}_k + \sum_{j=1}^{K-1} j b_j \leq \sum_{k=1}^{K} |\mathcal{C}_k| = cNK. \tag{213}$$

Combining (211) with (212) and (213) respectively, yield (94) and (95).

APPENDIX C
PROOF OF LEMMA 4

It is easy to verify that $\mathbf{A}^K_q$ is a $q^m \times K$ array for any allowed choice of K and q. Each column contains exactly q^{m-1} "$*$" symbols and each ordinary symbol takes value in $[(q-1)q^m]$. Thus,

$$F_{\mathbf{A}^K_q} = q^m \tag{214}$$

$$T_{\mathbf{A}^K_q} = q^{m-1}K \tag{215}$$

$$S_{\mathbf{A}^K_q} = (q-1)q^m. \tag{216}$$

We now prove that $\mathbf{A}_q^K$ is indeed a PDA, i.e., we show that properties a) and b) in Definition 7 hold. By (142) and (143), the entry of $\mathbf{A}_q^K$ in row $j = (j_{m-1}, j_{m-2}, \ldots, j_0)_q \in [q^m]$ and column $k = (k_1, k_0)_q^{m+1} \in [K]$ equals $s = (s_m, s_{m-1}, \ldots, s_0)_q^{q-1} \in [(q-1)q^m]$ if, and only if, the following two conditions hold:

1) when $0 \leq k_1 < m$,

$$j = (s_{m-1}, \ldots, s_{k_1+1}, s_{k_1} \oplus_q s_m \oplus_q 1, s_{k_1-1}, \ldots, s_0)_q, \tag{217}$$

$$k = (k_1, s_{k_1})_q^{m+1}. \tag{218}$$

2) when $k_1 = m$,

$$j = (s_{m-1}, s_{m-2}, \ldots, s_0)_q, \tag{219}$$

$$k = \big(k_1, \sum_{l=0}^{m} s_l \oplus_q 1\big)_q^{m+1}, \tag{220}$$

where in this section "$\oplus_q$" denotes addition modulo q.

Assume now two pairs $(j,k), (j',k') \in [F] \times [K]$ so that the corresponding entries of $\mathbf{A}_q^K$ are both equal to the same ordinary symbol s:

$$a_{j,k} = a_{j',k'} = s. \tag{221}$$

Let $k = (k_1, k_0)$ and $k' = (k_1', k_0')$. Notice now that by (217)–(220), if $k_1 = k_1'$ then also $k_0 = k_0'$, $k = k'$, and $j = j'$.

We therefore restrict to the case $k_1 \neq k_1'$. Assume without loss of generality that $0 \leq k_1 < k_1' \leq m$. In this case it can be shown that j_1 and j_1' differ in their k_1' -th components, thus establishing that $j \neq j'$. Distinguish the cases $k_1' < m$ or $k_1' = m$ (equivalently, $k' \leq qm$ or $k' > qm$).

1) Consider the case $k_1' < m$, and notice that $0 \leq s_m \leq q-2$ and $s_m \oplus_q 1 \in \{1, 2, \ldots, q-1\}$. Therefore, by (217):

$$j_{k_1'} = s_{k_1'} \neq s_{k_1'} \oplus_q s_m \oplus_q 1 = j'_{k_1'}, \tag{222}$$

and hence $j' \neq j$. Notice also that (218) implies $s_{k_1'} = k_0'$. Since $j_{k_1'} = s_{k_1'}$ by (222), we conclude $j_{k_1'} = k_0'$, and thus the entry $a_{j,k'} = *$ by (142). Similar arguments where we replace (j,k) with (j',k') yields that also $a_{j',k} = *$.

2) Consider now the case $k_1' = m$. By (217)–(220) and since $s_m \oplus_q 1 \neq 0$:

$$j'_{k_1} = s_{k_1} \neq s_{k_1} \oplus_q s_m \oplus_q 1 = j_{k_1}, \tag{223}$$

and thus $j \neq j'$. We now argue that $a_{j,k'} = *$. To this end, notice that since $k_1' = m$, (220) implies $k_0' = \sum_{l=0}^{m} s_l \oplus_q 1$; since $k_1 < m$, (217) implies $\sum_{l=0}^{m-1} j_l = \sum_{l=0}^{m} s_l \oplus_q 1$. Therefore, we conclude $\sum_{l=0}^{m-1} j_l = k_0'$ and thus $a_{j,k'} = *$ by (143). The proof that also entry $a_{j',k} = *$, is similar to the case when $k_1' < m$ and omitted.

The above analysis indicates that the properties a) and b) in Definition 7 hold in all cases and $\mathbf{A}_q^K$ must be a PDA. Moreover, by (142)–(142), it is obvious that $\mathbf{A}_q^K$ is a Comp-PDA.

It remains to prove that $\mathbf{A}_q^K$ is g-regular for $g = \lfloor \frac{K}{q} \rfloor$. In fact, for a given $s = (s_m, s_{m-1}, \ldots, s_0)_q^{q-1} \in [(q-1)q^m]$, if it occurs in the row $j = (j_{m-1}, \ldots, j_0)_q \in [q^m]$ and column $k = (k_1, k_0)_q^{m+1} \in [K]$, then

1) If $k_1 \in \{0, 1, \ldots, m-1\}$, (j, k) can be uniquely determined by (217) and (218). Therefore, each $s \in [(q-1)q^m]$ occurs exactly m times in the first qm columns of $\mathbf{A}_q^K$;
2) If $k_1 = m$, by (219) and (220), and the fact $0 \leq k_0 < K - qm$, (j, k) can be uniquely determined if, and only if,

$$0 \leq \sum_{l=0}^{m} s_l \oplus_q 1 < K - qm. \tag{224}$$

It is easy to enumerate that the number of $(s_m, s_{m-1}, \ldots, s_0) \in \mathbb{Z}_{q-1} \times \mathbb{Z}_q^m$ satisfying (224) is $(K - qm)(q - 1)q^{m-1}$.

From the above analysis, we conclude that, there are $(K - qm)(q-1)q^{m-1}$ of the $(q-1)q^m$ ordinary symbols having occurrence $m+1$. There are $Kq^m - Kq^{m-1}$ ordinary entries, thus the fraction of entries having occurrence $m+1$ is

$$\theta_{m+1} = \frac{(K - qm)(q-1)q^{m-1}}{Kq^m - Kq^{m-1}} \tag{225}$$

$$= 1 - \frac{m((m+1)q - K)}{K} \tag{226}$$

$$\leq 1. \tag{227}$$

If q does not divide K then $\theta_{m+1} < 1$ and $\mathbf{A}_q^K$ is an almost-regular Comp-PDA with $g = m = \lceil \frac{K}{q} \rceil - 1 = \lfloor \frac{K}{q} \rfloor$; if q divides K, $\theta_{m+1} = 1$, then all ordinary symbols have occurrence $m+1$, thus, $\mathbf{A}_q^K$ is a regular Comp-PDA with $g = m + 1 = \lceil \frac{K}{q} \rceil = \lfloor \frac{K}{q} \rfloor$. Therefore, $\mathbf{A}_q^K$ is an almost-regular Comp-PDA with $g = \lfloor \frac{K}{q} \rfloor$.

REFERENCES

[1] Q. Yan, S. Yang, and M. Wigger, "Storage, computation and communication: A fundamental tradeoff in distributed computing," in *Proc. IEEE Inf. Thoery Workshop (ITW)*, Guangzhou, China, Nov. 2018.

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

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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, Dec. 2016.

[8] A. Reisizadeh, S. Prakash, R. Pedarsani, and S. Avestimehr, "Coded computation over heterogeneous clusters," in *Proc. IEEE Int. Symp. Inf. Theory,* Aachen, Germany, pp. 2408–2412, Jun. 2017.

[9] Q. Yu, M. Maddah-Ali, and S. Avestimehr, "Polynomial codes: An optimal design for high-dimensional coded matrix multiplication," in *Proc. The 31st Annual Conf. Neural Inf. Processing System (NIPS)*, Long Beach, CA, USA, May 2017.

[10] Q. Yu, M. Maddah-Ali, and S. Avestimehr, "Straggler mitigation in distributed matrix multiplication: Fundamental limits and optimal coding," in *Proc. IEEE Int. Symp. Inf. Theory,* Vail, CO, USA, pp. 2022–2026, Jun. 2018.

[11] K. Lee, C. Suh, and K. Ramchandran, "High-dimensional coded matrix multiplication," in *Proc. IEEE Int. Symp. Inf. Theory,* Aachen, Germany, pp. 2418–2422, Jun. 2017.

[12] H. Park, K. Lee, J. Sohn, C. Suh, and J. Moon, "Hierarchical coding for distributed computing," arXiv:1801.04686.

[13] F. Haddadpour and V. R. Cadambe, "Codes for distributed finite alphabet matrix-vector multiplication," in *Proc. IEEE Int. Symp. Inf. Theory,* Vail, CO, USA, pp. 1625–1629, Jun. 2018.

[14] S. Kiani, N. Ferdinand and S. C. Draper, "Exploitation of stragglers in coded computation," in *Proc. IEEE Int. Symp. Inf. Theory,* Vail, CO, USA, pp. 1988–1992, Jun. 2018.

[15] T. Baharav, K. Lee, O. Ocal, and K. Ramchandran, "Straggler-proofing massive-scale distributed matrix multiplication with d-dimensional product codes," in *Proc. IEEE Int. Symp. Inf. Theory,* Vail, CO, USA, pp. 1993–1997, Jun. 2018.

[16] N. Ferdinand and S. C. Draper, "Hierarchical coded computation," in *Proc. IEEE Int. Symp. Inf. Theory,* Vail, CO, USA, pp. 1620–1624, Jun. 2018.

[17] 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.

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

[19] S. Li, Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Edge-facilitated wireless distributed computing," in *Proc. IEEE Glob. Commun. Conf. (Globlcom),* Washington, DC, USA, Dec. 2016.

[20] F. Li, J. Chen, and Z. Wang, "Wireless Map-Reduce distributed computing," in *Proc. IEEE Int. Symp. Inf. Theory,* Vail, CO, USA, pp. 1286–1290, Jun. 2018.

[21] E. Parrinello, E. Lampiris, and P. Elia, "Coded distributed computing with node cooperation substantially increases speedup factors," in *Proc. IEEE Int. Symp. Inf. Theory,* Vail, CO, USA, pp. 1291–1295, Jun. 2018.

[22] M. A. Attia and R. Tandon, "On the worst-case communication overhead for distributed data shuffling," in *Proc. 54th Allerton Conf. Commun., Control, Comput.,* Monticello, IL, USA, pp. 961–968, Sep. 2016.

[23] M. A. Attia and R. Tandon, "Information theoretic limits of data shuffling for distributed learning," in *Proc. IEEE Glob. Commun. Conf. (Globlcom),* Washington, DC, USA, Dec. 2016.

[24] A. Elmahdy and S. Mohajer, "On the fundamental limits of coded data shuffling," in *Proc. IEEE Int. Symp. Inf. Theory,* Vail, CO, USA, pp. 716–720, Jun. 2018.

[25] L. Song, S. R. Srinivasavaradhan, and C. Fragouli, "The benefit of being flexible in distributed computation," in *Proc. IEEE Inf. Theory Workshop (ITW)*, Kaohsiung, Taiwan, pp. 289–293, Nov. 2017.

[26] S. R. Srinivasavaradhan, L. Song, and C. Fragouli, "Distributed computing trade-offs with random connectivity," in *Proc. IEEE Int. Symp. Inf. Theory,* Vail, CO, USA, pp. 1281–1285, Jun. 2018.

[27] R. Tandon, Q. Lei, A. G. Dimakis, and N. Karampatziakis, "Gradient coding: Avoiding stragglers in synchronous gradient descent," arXiv: 1612.03301.

[28] N. Raviv, I. Tamo, R. Tandon, and A. G. Dimakis, "Gradient coding from cyclic MDS codes and expander graphs," arXiv: 1707.03858.

[29] Z. Charles and D. Papailiopoulos, "Gradient coding using the stochastic block model," in *Proc. IEEE Int. Symp. Inf. Theory,* Vail, CO, USA, pp. 1998–2002, Jun. 2018.

[30] W. Halbawi, N. Azizan, F. Salehi, and B. Hassil, "Improving distributed gradient descent using Reed-Solomon codes," in *Proc. IEEE Int. Symp. Inf. Theory,* Vail, CO, USA, pp. 2027–2031, Jun. 2018.

[31] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *IEEE Trans. Inf. Theory,* vol. 60, no. 5, pp. 2856–2867, May 2014.

[32] 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.

[33] C. Shangguan, Y. Zhang, and G. Ge, "Centralized coded caching schemes: A hypergraph theoretical approach." *IEEE Trans. Inf. Theory,* vol. 64, no. 8, pp. 5755-5766, Aug. 2018.

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

[35] K. Konstantinidis and A. Ramamoorthy, "Leveraging coding techniques for speeding up distributed computing," arXiv:1802.03049

[36] N. Woolsey, R. R. Chen, and M. Ji,"A new combinatorial design of coded distributed computing," in *Proc. IEEE Int. Symp. Inf. Theory,* Vail, CO, USA, pp. 726–730, Jun. 2018.