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

Qifa Yan, *Member, IEEE*, Sheng Yang, *Member, IEEE*, and Michèle Wigger, *Senior Member, IEEE*

***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 across clusters of many individual computing nodes. In particular, distributed programs like MapReduce [2] and Dryad [3] have become popular and 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.

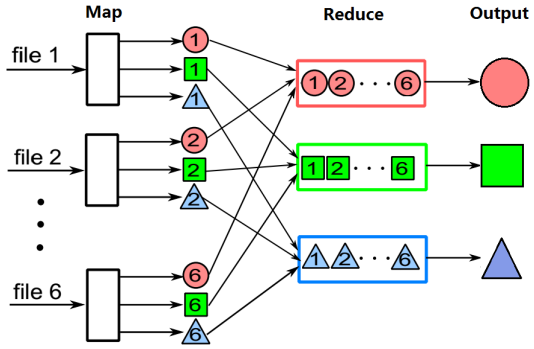

Fig. 1. A computing task with $N = 6$ files and $K = 3$ output functions (original image from [4]). The small and big red circles, green squares, and blue triangles denote IVAs and results belonging to different output functions.

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

$$\phi_k(w_1, \ldots, w_N) \;=\; h_k(f_{k,1}(w_1), \ldots, f_{k,N}(w_N)), \quad k = 1, \ldots, K. \tag{1}$$

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

Computation of such functions can be performed in a distributed way following 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:

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

In the subsequent *shuffle phase*, the nodes exchange the IVAs computed during the map phase, so that each node k is aware of all the IVAs $f_{k,1}(w_1), \ldots, f_{k,N}(w_N)$ required to calculate

Manuscript received 9 September 2019; revised 12 February 2022; accepted 23 February 2022. Date of publication 18 March 2022; date of current version 13 July 2022. The work of Qifa Yan and Michèle Wigger was supported by the European Union's Horizon 2020 Research and Innovation Program under Grant 715111. The work of Qifa Yan was supported in part by the National Natural Science Foundation of China under Grant 61941106 and Grant 62101464. An earlier version of this paper was presented in part at 2018 IEEE Information Theory Workshop (ITW) [1] [DOI: 10.1109/ITW.2018.8613519]. *(Corresponding author: Qifa Yan.)*

Qifa Yan was with LTCI, Tèlècom Paris, IP Paris, 91120 Palaiseau, France, and also with L2S, (UMR CNRS 8506), CentraleSupèlec, Paris-Saclay University, 91192 Gif-sur-Yvette, France. He is now with the Information Coding and Transmission Key Laboratory of Sichuan Province, CSNMT International Cooperation Research Centre (MoST), Southwest Jiaotong University, Chengdu 611756, China (e-mail: qifayan@swjtu.edu.cn).

Sheng Yang is with L2S, (UMR CNRS 8506), CentraleSupèlec, Paris-Saclay University, 91192 Gif-sur-Yvette, France (e-mail: sheng.yang@centralesupelec.fr).

Michèle Wigger is with LTCI, Tèlècom Paris, IP Paris, 91120 Palaiseau, France (e-mail: michele.wigger@telecom-paristech.fr).

Communicated by A. Jiang, Associate Editor for Coding Theory.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TIT.2022.3158828.

Digital Object Identifier 10.1109/TIT.2022.3158828

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.* [5] 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 [5] 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 [5] 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 [5] to a *storage-computation-communication* tradeoff. Notice that other interesting extensions have recently been proposed. For example, [7]–[16] 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 [17]; [18] studied optimal allocation of computation resources; [19]–[22] investigated distributed computing in wireless networks; [23]–[25] investigated the iterative procedures of data computing and shuffling; [26] studied the case when each node has been randomly allocated files; [27] investigated the case with random connectivity between nodes; [28]–[31] designed codes for computing gradient distributedly, 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)* [33], 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 [33] 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 [5], [36]–[38]. In particular, PDAs were used to characterize the storage and communication loads in [39], [40]. 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. Moreover, in a follow-up work [41], we show that Comp-PDAs allow us to design coded computing schemes for systems with straggling nodes.

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 [32][1] and the corresponding coded computing schemes coincide with the scheme proposed by Ezzeldin *et al.* [6] 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 [33, 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 [33], [34], and [35] have pareto-optimal SCC loads.

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

[1]The connection between these PDAs and the coded caching scheme in [32] was formalized in [33]. As explained previously, Comp-PDAs are also PDAs.

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 $\{1,\ldots,K\}$ and N files,

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

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

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

which maps all the files to a bit stream

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

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

Following the conventions in [5], we assume that each output function ϕ_k decomposes as:

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

where:

- Each *"map" function* $f_{k,n}$ is of the form
$$f_{k,n}:\mathbb{F}_2^W\to\mathbb{F}_2^V,$$
and maps the file w_n into the IVA
$$v_{k,n}\triangleq f_{k,n}(w_n)\in\mathbb{F}_2^V,$$
for a given $V\in\mathbb{N}$.
- The *"reduce" function* h_k is of the form
$$h_k:\mathbb{F}_2^{NV}\to\mathbb{F}_2^U,$$
and maps the IVAs
$$\mathcal{V}_k\triangleq\{v_{k,n}:n\in[N]\}\tag{2}$$
into the output stream
$$u_k=h_k(v_{k,1},\ldots,v_{k,N}).$$

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 $h_k=\phi_k,\ \forall\, n\in[N],\ k\in[K]$.

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.

A. Map Phase

Each node $k\in[K]$ 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

$$\mathcal{C}_{k,n}=\{v_{q,n}:q\in\mathcal{Z}_{k,n}\},\tag{3}$$

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

$$\mathcal{C}_k\triangleq\bigcup_{n:w_n\in\mathcal{M}_k}\mathcal{C}_{k,n}.\tag{4}$$

[2]See Remark 3 for a relaxed assumption.

B. Shuffle Phase

The K nodes exchange some of their computed IVAs. In particular, node k creates a signal

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

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

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

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

C. 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

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

for some function

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

Finally, it computes

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

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:

$$r\triangleq\frac{\sum_{k=1}^K|\mathcal{M}_k|}{N}.\tag{5}$$

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:

$$c\triangleq\frac{\sum_{k=1}^K|\mathcal{C}_k|}{NK}.\tag{6}$$

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:

$$L=\frac{\sum_{k=1}^K l_k}{NKV}.\tag{7}$$

Remark 1: These measures were first defined in [6], 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:

$$1\le c\le r\le K,\tag{8a}$$
$$0\le L\le 1-\frac{r}{K}.\tag{8b}$$

A scheme that sends all IVAs in an uncoded manner requires a communication load of $L = \frac{\sum_{k=1}^{K}(N-|\mathcal{M}_k|)V}{NKV} = 1 - \frac{r}{K}$ and computation load $c = 1$ because each IVA needs to be transmitted only once. In this sense the presented bounds in (8) are non-trivial. We proceed to justify them. 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. Secondly, we argue that we can restrict attention to values of c and r satisfying (8a). Since each IVA needs to be computed at least once somewhere, we have $c \geq 1$. Moreover, the definition of $\mathcal{C}_k$ in (4) implies that $|\mathcal{C}_k| \leq |\mathcal{M}_k|K$, and thus by (5) and (6) 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 (8) is called *feasible*, if for any $\epsilon > 0$ and sufficiently large N, W, V, 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*:

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

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

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

Remark 2: The pareto-optimal surface (see [44]) determines, e.g., the minimum required communication load for given storage and computation loads. (Or the minimum storage load for given communication and computation loads, or the minimum computation load for given storage and communication loads.) Moreover, it contains the minimizer (r, c, L) to any weighted sum $\alpha \cdot r + \beta \cdot c + \gamma \cdot L$ for given α, β, γ. As such, using appropriate weights α, β, γ, it can be used to minimize the total expected running time of the Map-Reduce system.

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.

Our schemes also require that the *size* of the files W and intermediate values V be sufficiently large. We will simply assume that this requirement is satisfied.

Remark 3: All our conclusions in this paper remain valid in an extended setup with Q output functions as in [5], where $K|Q$ and each node is supposed to compute $\frac{Q}{K}$ functions. In fact, in this setup, the K in definitions (6) and (7) 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 (2), (3), and (34) following the same steps in Section VI-A.

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

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

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

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 [33] in the sense that different columns can have different numbers of "$*$" symbols. In the original definition [33], 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. A non-trieval Comp-PDA is presented in the following example.

Example 1: The following $\mathbf{A}$ is a $(5, 4, 10, 4)$ Comp-PDA,

$$\mathbf{A} = \begin{bmatrix} * & 2 & * & 3 & * \\ 1 & * & * & 4 & 2 \\ * & 4 & 1 & * & 3 \\ 3 & * & 2 & * & * \end{bmatrix} \tag{9}$$

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

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

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

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

and it can be g-regular only if

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

Example 2: In the $(5, 4, 10, 4)$ Comp-PDA $\mathbf{A}$ in Example 1, there are $KF - T = 10$ ordinary entries in total, and $S_1 = S_4 = S_5 = 0$ (i.e., no ordinary symbol has occurences 1, 4, or 5) wheras $S_2 = S_3 = 2$ (i.e., two ordinary symbols 1, 4 occurr twice, and the other two ordinary symbols $2, 3$ occurr three times. The PDA $\mathbf{A}$ is thus an almost 2-regular Comp-PDA with frequencies $\theta_1 = \theta_4 = \theta_5 = 0$, $\theta_2 = \frac{2}{5}$ and $\theta_3 = \frac{3}{5}$.

IV. Main Results

A. Coded Computing Schemes From Comp-PDAs

In Section V, we will describe how to obtain a coded computing scheme from a Comp-PDA. 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.

$$\theta'_t = \begin{cases} 0, & t = 1 \\ \theta_1 + \theta_2, & t = 2 \\ \theta_t, & t > 2. \end{cases} \tag{10}$$

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

$$\begin{aligned}(r, c, L) \;=\; & \left(\frac{T}{F}, \; \frac{T}{KF} + \left(1 - \frac{T}{KF}\right) \cdot \sum_{t=2}^{K} \theta'_t(t-1), \right. \\ & \left. \left(1 - \frac{T}{KF}\right) \cdot \sum_{t=2}^{K} \frac{\theta'_t}{t-1} \right)\end{aligned}$$

with file complexity F.

We notice that the file complexity of a Comp-PDA is simply the number of rows F. It was observed in [36], [37] that large file complexity may cause large executation times and even dominate over the reductions achieved by coded computing [5]. This observation motivated the search for low file complexity coded computing schemes such as [36]–[38]. Theorem 1 indicates that, low file complexity coded computing schemes can be obtained via PDAs with a small number of rows F. We therefore will call the parameter F 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

$$\begin{aligned}(r, c, L) \;=\; & \left(\frac{T}{F}, \; \frac{T}{KF} + \left(1 - \frac{T}{KF}\right) \cdot (g - 1 + \theta'_{g+1}), \right. \\ & \left. \left(1 - \frac{T}{KF}\right) \cdot \frac{(g - \theta'_{g+1} - 1)^+ + 1}{g \cdot ((g-2)^+ + 1)} \right). \end{aligned} \tag{11}$$

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

$$\begin{aligned}(r, c, L) \;=\; & \left(\frac{T}{F}, \; \frac{T}{KF} + \left(1 - \frac{T}{KF}\right) \cdot ((g-2)^+ + 1), \right. \\ & \left. \frac{1}{(g-2)^+ + 1} \cdot \left(1 - \frac{T}{KF}\right) \right). \end{aligned} \tag{12}$$

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 (11) 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 (12) follows readily from (11). ∎

Corollary 1 is of particular interest since there are several explicit regular PDA constructions for coded caching in the literature [33]–[35]. 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 [32].

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

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

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.

Example 3: Let $K = 5$, in which case $\mathbf{P}_2$ is given by

$$\begin{array}{l} \{1,2\} \\ \{1,3\} \\ \{1,4\} \\ \{1,5\} \\ \{2,3\} \\ \{2,4\} \\ \{2,5\} \\ \{3,4\} \\ \{3,5\} \\ \{4,5\} \end{array} \begin{bmatrix} * & * & 1 & 2 & 3 \\ * & 1 & * & 4 & 5 \\ * & 2 & 4 & * & 6 \\ * & 3 & 5 & 6 & * \\ 1 & * & * & 7 & 8 \\ 2 & * & 7 & * & 9 \\ 3 & * & 8 & 9 & * \\ 4 & 7 & * & * & 10 \\ 5 & 8 & * & 10 & * \\ 6 & 9 & 10 & * & * \end{bmatrix},$$

where we label the rows by the subsets of size 2 as indicated by Definition 11.

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

$$\begin{aligned} P_i &\triangleq (r_{\mathbf{P}_i}, c_{\mathbf{P}_i}, L_{\mathbf{P}_i}) \\ &= \left(i,\ i\left(1 - \frac{i-1}{K}\right),\ \frac{1}{i}\left(1 - \frac{i}{K}\right)\right). \end{aligned} \quad (13)$$

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:

$$Q_i \triangleq \left(i,\ i,\ \frac{1}{i}\left(1 - \frac{i}{K}\right)\right), \quad i \in [K]. \quad (14)$$

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

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

where P_i and Q_i are defined in (13) and (14), respectively. Then, the optimal tradeoff surface $\mathcal{O}$ and the fundamental SCC region $\mathcal{R}$ are given by

$$\begin{aligned} \mathcal{O} &= \{\triangle P_{i-1} P_i P_K : i = 2, \ldots, K-1\}, \\ \mathcal{R} &= \{(r,c,L) \colon (r,c,L) \text{ *is on or above the surface* } \mathcal{F} \\ &\qquad \text{*and satisfies* (8)}\}. \end{aligned}$$

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 4: 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 $\triangle P_{i-1} P_i P_K$ if and only if it is almost i-regular. It achieves a point on the triangle $\triangle P_1 P_2 P_K$ if and only if 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 $\overline{P_i P_K}$ if and only if it is $(i+1)$-regular. As a consequence, the corner points $P_1, \ldots, P_K$ are achieved by a uniform file allocation (i.e., each node stores the same number of files).

Note that setting $r = c$, we recover exactly the case investigated in [5] where the fundamental storage-communication tradeoff is characterized by

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

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

$$L^*(r) \triangleq \max_{i \in [K-1]} \left\{ -\frac{1}{i(i+1)} r + \frac{1}{i} + \frac{1}{i+1} - \frac{1}{K} \right\}. \quad (15)$$

An example of the fundamental SCC region for $K = 10$ is given in Fig. 2, where we can identify the surface $\mathcal{F}$ that is formed by the triangles $\triangle P_1 P_2 Q_2$ and $\{\triangle P_{i-1} P_i P_K\}$ and the trapezoids $\{\boxminus P_i Q_i Q_{i+1} P_{i+1}\}$. In particular, the boundary of the optimal tradeoff surface $\mathcal{O}$ is formed by the line segment $\overline{P_1 P_K}$ and the sequence of line segments $\overline{P_1 P_2}$, $\overline{P_2 P_3}$, $\ldots, \overline{P_{K-1} P_K}$:

1) The computation load on the line segment $\overline{P_1 P_K}$ is $c = 1$ for any given storage load r, which by (8) 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 $\overline{P_1 P_2}$, $\overline{P_2 P_3}$, $\ldots, \overline{P_{K-1} P_K}$ 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 [5]. In this sense, our optimal tradeoff surface $\mathcal{O}$ is a natural extension of the tradeoff established in [5] 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. 3(a)), as well as the tradeoff between computation and communication for a given storage load (Fig. 3(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

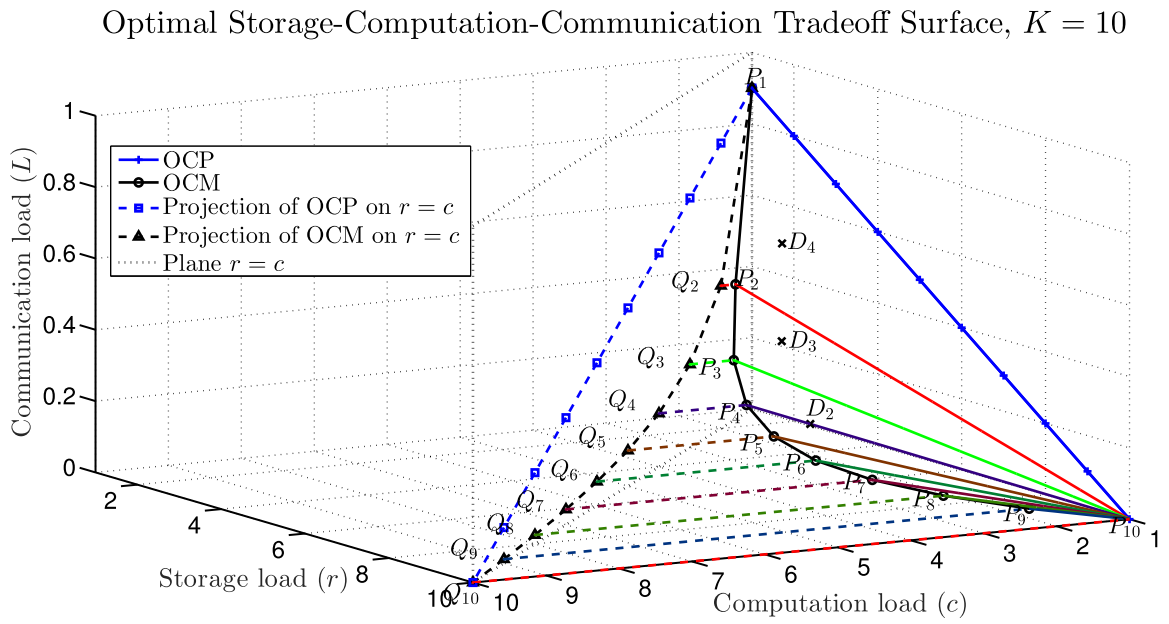

Fig. 2. The fundamental SCC region $\mathcal{R}$ for a system with $K = 10$ nodes. The figure illustrates the delimiting surface $\mathcal{F}$ formed by the triangles $\triangle P_1P_2Q_2$ and $\{\triangle P_{i-1}P_iP_K\}$ and the trapezoids $\{\boxminus P_iQ_iQ_{i+1}P_{i+1}\}$. The three points D_2, D_3, D_4 can be achieved by the new PDA design.

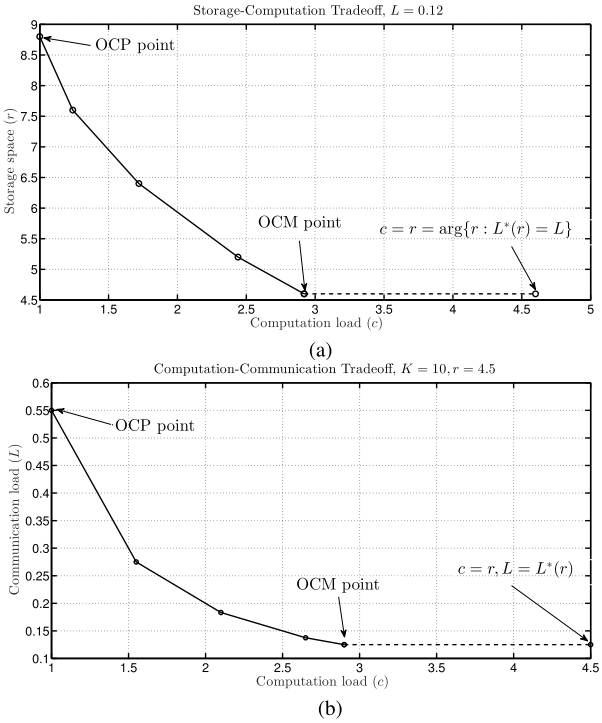

Fig. 3. Two-dimensional tradeoff curves for $K = 10$: (a) Computation-storage tradeoff with fixed communication load $L = 0.12$; (b) Computation-communication tradeoff with fixed storage load $r = 4.5$.

computation load can be actually close to the lower bound 1, i.e., almost no extra computation is needed.

Remark 5: The idea of measuring the effectively computed IVAs is from [6]. 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 [5] 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

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

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

$$\begin{aligned} D_q &= \left(r_{D_q}, c_{D_q}, L_{D_q}\right) \\ &\triangleq \left(\frac{K}{q},\ \frac{1}{q} + m\left(1 - \frac{1}{q}\right)\left(2 - \frac{(m+1)q}{K}\right),\right. \\ &\quad \left.\left(\frac{1}{m} + \frac{(m+1)q - K}{K(m-1)}\right) \cdot \left(1 - \frac{1}{q}\right)\right), \end{aligned} \quad (16)$$

with file complexity

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

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:

$$-1 \leq \quad r_{P_m} - r_{D_q} \quad \leq 0, \tag{17a}$$

$$\frac{1}{q} - \frac{2}{K} \leq \quad c_{P_m} - c_{D_q} \quad \leq 1 - \frac{1}{q}, \tag{17b}$$

$$-\frac{q(q-1)}{K(K-q)} \leq \quad L_{P_m} - L_{D_q} \quad \leq \frac{q}{K(K-q)}. \tag{17c}$$

Furthermore, its file complexity satisfies

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

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. 2, 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

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

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.,

$$\mathcal{U}_{i,j} \triangleq \{v_{j,n} : w_n \in \mathcal{W}_i\}, \tag{18}$$

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

$$\mathcal{A}_k \triangleq \{s \in [S] : a_{i,k} = s \text{ for some } i \in [F]\backslash\mathcal{I},\ k \in [K]. \tag{19}$$

1) *Map Phase:* Each node k stores

$$\mathcal{M}_k = \bigcup_{\substack{i \in [F]: \\ a_{i,k} = *}} \mathcal{W}_i, \tag{20}$$

and computes the IVAs

$$\mathcal{C}_k = \mathcal{C}_k^{(1)} \cup \mathcal{C}_k^{(2)}, \tag{21}$$

where

$$\mathcal{C}_k^{(1)} = \bigcup_{\substack{i \in [F]: \\ a_{i,k} = *}} \mathcal{U}_{i,k}, \tag{22}$$

$$\mathcal{C}_k^{(2)} = \bigcup_{s \in \mathcal{I}_k \cup \mathcal{A}_k} \bigcup_{\substack{(l,d) \in [F] \times ([K]\backslash\{k\}): \\ a_{l,d} = s}} \mathcal{U}_{l,d}, \tag{23}$$

Notice that node k can compute the IVAs in $\mathcal{C}_k^{(1)}$ from the files in $\mathcal{M}_k$, because of (18), (20), and (22). 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

$$a_{l,k} = *. \tag{24}$$

From this follows that $\mathcal{W}_l \subseteq \mathcal{M}_k$. To prove (24), 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:* Node $k \in [K]$ 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

$$X_s^k \triangleq \mathcal{U}_{i,j}. \tag{25}$$

For each $s \in \mathcal{A}_k \subseteq [S]\backslash\mathcal{I}$, and X_s^k is defined as follows. Firstly, for each $s \in [S]\backslash\mathcal{I}$, or equivalently, $g_s \geq 2$, let $(i_1, j_1), \ldots, (i_{g_s}, j_{g_s})$ indicate all the occurrences of the ordinary symbol s:

$$a_{i_1,j_1} = a_{i_2,j_2} = \ldots = a_{i_{g_s},j_{g_s}} = s.$$

For each $a \in [g_s]$, partition the IVAs $\mathcal{U}_{i_a,j_a}$ into $g_s - 1$ subblocks, and label them with the other $g_s - 1$ column indices, i.e.,

$$\mathcal{U}_{i_a,j_a} = \{\mathcal{U}_{i_a,j_a}^{j_b} : b \in [g_s]\backslash\{a\}\}, \quad \forall\, a \in [g_s]. \tag{26}$$

Intuitively, node j_a needs $\mathcal{U}_{i_a,j_a}$, which is partitioned into subblocks in (26). The subblock $\mathcal{U}_{i_a,j_a}^{j_b}$ will be retrieved from the signal sent by node j_b for each $b \in [g_s]\backslash\{a\}$. Thus, each node j_b needs to send all subblocks with superscript j_b. Specifically, node j_b will XOR all subblocks $\{\mathcal{U}_{i_a,j_a}^{j_b} : a \in [g_s]\backslash\{b\}\}$. In particular, for $s \in \mathcal{A}_k$, the signal X_s^k is formed accordingly by node k, i.e., let $s = a_{i,j}$, and

$$X_s^k \triangleq \bigoplus_{\substack{(i,j) \in [F] \times ([K]\backslash\{k\}): \\ a_{i,j} = s}} \mathcal{U}_{i,j}^k. \tag{27}$$

Then node k multicasts the signal

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

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

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

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

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

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 (26) can be restored by node k from the signal X_s^j sent by node j (see (27)):

$$X_s^j = \bigoplus_{\substack{(l,d)\in[F]\times([K]\backslash\{j\}):\\ a_{l,d}=s}} \mathcal{U}_{l,d}^j. \tag{28}$$

In fact, for each $\mathcal{U}_{l,d}^j$ in (28), 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 (23) and (26). Therefore, $\mathcal{U}_{i,k}^j$ can be decoded from (28). If $s \notin \mathcal{A}_k$, then $s \in \mathcal{I}$ by (19). There exists thus an index $j \in [K]\backslash\{k\}$ such that $s \in \mathcal{I}_j$ and therefore, by (25), the subset $\mathcal{U}_{i,k}$ can be recovered from the signal X_s^j sent by node j.

Example 4: To have an idea of how to use Comp-PDAs to construct coded computing schemes, let us consider a toy example with the following 3-regular $(3, 3, 6, 1)$ Comp-PDA $\mathbf{A}$.

$$\mathbf{A} = \begin{bmatrix} * & 1 & * \\ * & * & 1 \\ 1 & * & * \end{bmatrix} \tag{29}$$

We can derive a coded computing scheme for the computation task in Fig. 1 with $K = 3$ nodes. The scheme is depicted in Fig. 4. 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

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

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 stores all the files

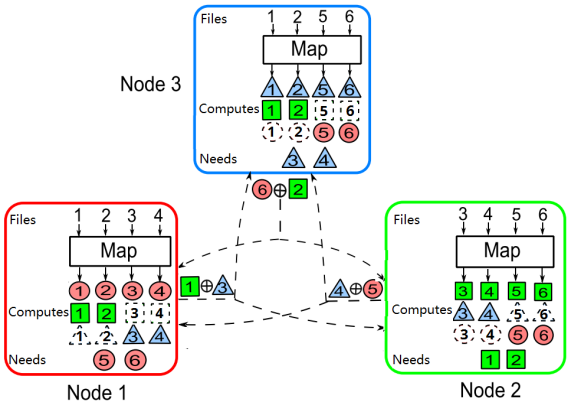

Fig. 4. A coded computing scheme from $\mathbf{A}$ in (29) (original image from [4]).

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. 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, it computes all the IVAs indicated by the ordinary symbol entries except for the ones in its own column. Specifically, node 1 computes the square IVAs $1, 2$ and the triangle IVAs $3, 4$, node 2 computes the triangle IVAs $3, 4$ and the circle IVAs $5, 6$, and node 3 computes the square IVAs $1, 2$ and the circle IVAs $5, 6$.

The signals are formed as follows. Notice that node 1 needs the circle IVAs $5, 6$, and they are expected to be retrieved from node 2 and 3 respectively. Similarly, node 2 (node 3 resp.) needs the square IVAs $1, 2$ (triangles $3, 4$ resp.) and hence they are expected to be retrieved from node 1 and 3 (node 1 and 2) respectively. Thus, each node sends the XOR of the IVAs it should provide to other nodes as illustrated in Fig. 4. 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.

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 (20), the storage load of the proposed scheme is:

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

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 (22) and (23),

$$\begin{aligned}\sum_{k=1}^{K} |\mathcal{C}_k^{(1)}| &= T \cdot \eta,\\ \sum_{k=1}^{K} |\mathcal{C}_k^{(2)}| &= \sum_{k=1}^{K} \left[|\mathcal{I}_k| \cdot \eta + \sum_{s \in \mathcal{A}_k} (g_s - 1) \cdot \eta \right]\\ &= |\mathcal{I}| \cdot \eta + \sum_{s \in [S] \backslash \mathcal{I}} g_s (g_s - 1) \eta,\end{aligned}$$

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

$$\begin{aligned}c &= \frac{\sum_{k=1}^{K} |\mathcal{C}_k|}{NK}\\ &= \frac{T \cdot \eta + |\mathcal{I}| \cdot \eta + \sum_{s \in [S] \backslash \mathcal{I}} g_s (g_s - 1) \cdot \eta}{NK}\\ &= \frac{T}{KF} + \frac{|\mathcal{I}|}{KF} + \sum_{s \in [S] \backslash \mathcal{I}} \frac{g_s (g_s - 1)}{KF}. \qquad (30)\end{aligned}$$

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.,

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

Then by (30), the computation load of the proposed scheme is

$$\begin{aligned}c &= \frac{T}{KF} + \frac{S_1}{KF} + \sum_{t=2}^{K} \frac{S_t t (t-1)}{KF}\\ &= \frac{T}{KF} + \frac{S_1}{KF - T} \cdot \frac{KF - T}{KF}\\ &\quad + \sum_{t=2}^{K} \frac{S_t t}{KF - T} \cdot \frac{KF - T}{KF} \cdot (t-1)\\ &= \frac{T}{KF} + \left(1 - \frac{T}{KF}\right) \cdot \left(\theta_1 + \sum_{t=2}^{K} \theta_t (t-1)\right)\\ &= \frac{T}{KF} + \left(1 - \frac{T}{KF}\right) \cdot \sum_{t=2}^{K} \theta_t' (t-1),\end{aligned}$$

where θ_t' was defined in (10).

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 (25), 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 (27) 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:

$$\begin{aligned}L_{\mathbf{A}} &= \frac{\sum_{k=1}^{K} l_k}{NKV}\\ &= \frac{1}{NKV} \cdot \sum_{k=1}^{K} \left[|\mathcal{I}_k| \cdot \eta \cdot V + \sum_{k \in \mathcal{A}_k} \frac{\eta \cdot V}{g_s - 1} \right]\\ &= \frac{1}{KF} \cdot \left[|\mathcal{I}| + \sum_{s \in [S] \backslash \mathcal{I}} \frac{g_s}{g_s - 1} \right]\\ &= \frac{1}{KF} \cdot \left(S_1 + \sum_{t=2}^{K} \frac{S_t t}{t-1} \right)\\ &= \frac{KF - T}{KF} \cdot \left(\frac{S_1}{KF - T} + \sum_{t=2}^{K} \frac{S_t t}{KF - T} \cdot \frac{1}{t-1} \right)\\ &= \left(1 - \frac{T}{KF}\right) \cdot \left(\theta_1 + \sum_{t=2}^{K} \frac{\theta_t}{t-1} \right)\\ &= \left(1 - \frac{T}{KF}\right) \cdot \sum_{t=2}^{K} \frac{\theta_t'}{t-1}.\end{aligned}$$

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 $\triangle P_{i-1} P_i P_K$ ($i \in [2 : K-1]$) can 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

$$r = \frac{\sum_{k=1}^{K} |\mathcal{M}_k|}{N}, \qquad (31)$$

$$c = \frac{\sum_{k=1}^{K} |\mathcal{C}_k|}{NK}, \qquad (32)$$

$$L \geq \frac{\sum_{k=1}^{K} H(X_k)}{NKV}. \qquad (33)$$

For any $k \in [K]$ and nonempty $\mathcal{S} \subseteq [K]\backslash\{k\}$, define

$$\mathcal{B}_{k,\mathcal{S}} := \{v_{k,n} : v_{k,n} \text{ is exclusively computed by the nodes in } \mathcal{S}\}, \quad (34a)$$

$$\widetilde{\mathcal{B}}_k := \{v_{k,n} : v_{k,n} \text{ is computed by node } k\}. \quad (34b)$$

Let $b_{k,\mathcal{S}}$ be the cardinality of the set $\mathcal{B}_{k,\mathcal{S}}$ and $\widetilde{b}_k$ be the cardinality of $\widetilde{\mathcal{B}}_k$. Notice that, the subsets $\{\mathcal{B}_{k,\mathcal{S}} : \mathcal{S} \subseteq [K]\backslash\{k\}, \mathcal{S} \neq \emptyset\}$ and $\widetilde{\mathcal{B}}_k$ form a partition of the IVAs $\mathcal{V}_k$, thus

$$\widetilde{b}_k + \sum_{\mathcal{S}\subseteq[K],\mathcal{S}\neq\emptyset} b_{k,\mathcal{S}} = N. \quad (35)$$

For each $j \in [K-1]$, the set of IVAs not computed locally but exclusively computed by j other nodes are

$$\mathcal{B}_j = \bigcup_{k\in[K]} \bigcup_{\mathcal{S}\subseteq[K]\backslash\{k\},|\mathcal{S}|=j} \mathcal{B}_{k,\mathcal{S}}.$$

Then the cardinality of set $\mathcal{B}_j$ is given by

$$b_j \triangleq \sum_{k\in[K]} \sum_{\mathcal{S}\subseteq[K],|\mathcal{S}|=j} b_{k,\mathcal{S}}, \quad \forall\, j \in [K-1]. \quad (36)$$

We need the following two lemmas.

Lemma 1: The sum of entropies of the signals have the following bound:

$$\sum_{k=1}^{K} H(X_k) \geq V \cdot \sum_{j=1}^{K-1} \frac{b_j}{j}.$$

Proof: By the lower bound of data exchange problem [43, Theorem 1], the sum of the entropies is lower bounded by

$$\begin{aligned}
&\sum_{k=1}^{K} H(X_k) \\
\geq\ & V \cdot \sum_{k\in[K]} \sum_{\mathcal{S}\subseteq[K]\backslash\{k\},\mathcal{S}\neq\emptyset} b_{k,\mathcal{S}} \cdot \frac{1}{|\mathcal{S}|} \\
=\ & V \cdot \sum_{k\in[K]} \sum_{j=1}^{K-1} \sum_{\mathcal{S}\subseteq[K]\backslash\{k\},|\mathcal{S}|=j} b_{k,\mathcal{S}} \cdot \frac{1}{|\mathcal{S}|} \\
=\ & V \cdot \sum_{j=1}^{K-1} \frac{1}{j} \cdot \sum_{k\in[K]} \sum_{\mathcal{S}\subseteq[K]\backslash\{k\},|\mathcal{S}|=j} b_{k,\mathcal{S}} \\
\overset{(a)}{=}\ & V \cdot \sum_{j=1}^{K-1} \frac{b_j}{j},
\end{aligned}$$

where (a) follows from (36). ■

Remark 6: Lemma 1 is proved by following the steps in the proof of [5, Lemma 1] but here we replace the set of IVAs that can be computed at a given node k by the set of IVAs that are effectively computed at this node. The proof steps remain valid, and the so obtained converse is also tight in our more general setup. After the initial submission of this article, a reviewer pointed out that the computational constraint "each node computes all IVAs it can compute" does not affect the lower bound. An observation subsequently also made in [42]. For conciseness and completeness, here we proved Lemma 1 based on the lower bound in [43, Theorem 1].

Lemma 2: The parameters $b_1, \ldots, b_{K-1}$ defined in (36) satisfy

$$\sum_{j=1}^{K-1} b_j \geq N(K-r), \quad (37)$$

$$\sum_{j=1}^{K-1} (j-1) b_j \leq (c-1)NK. \quad (38)$$

Proof: Summing over $k \in [K]$ in (35), together with (36),

$$\sum_{k=1}^{K} \widetilde{b}_k + \sum_{j=1}^{K-1} b_j = NK. \quad (39)$$

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 (31),

$$\sum_{k=1}^{K} \widetilde{b}_k \leq \sum_{k=1}^{K} |\mathcal{M}_k| = rN. \quad (40)$$

Also, for $k \in [K]$ and $j \in [K-1]$, IVAs $\widetilde{\mathcal{B}}_k$ must be computed at node k, and IVAs $\mathcal{B}_j$ must be computed at j nodes. Thus by (32),

$$\sum_{k=1}^{K} \widetilde{b}_k + \sum_{j=1}^{K-1} j b_j \leq \sum_{k=1}^{K} |\mathcal{C}_k| = cNK. \quad (41)$$

Combining (39) with (40) and (41) yields (37) and (38), respectively. ■

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

$$c_i \triangleq \frac{r}{K} + \left(1 - \frac{r}{K}\right) i, \quad (42)$$

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

$$\begin{aligned}
\lambda_i x + \mu_i|_{x=c_{i-1}} &= \frac{1}{c_{i-1} - r/K} \left(1 - \frac{r}{K}\right)^2 \\
&= \frac{1}{i-1} \left(1 - \frac{r}{K}\right), \quad (43) \\
\lambda_i x + \mu_i|_{x=c_i} &= \frac{1}{c_i - r/K} \left(1 - \frac{r}{K}\right)^2 \\
&= \frac{1}{i} \left(1 - \frac{r}{K}\right). \quad (44)
\end{aligned}$$

Notice that by (42), (43), and (44), the following three relationships hold:

$$\lambda_i = -\frac{1}{i(i-1)} < 0, \quad (45)$$

$$\mu_i = \frac{2}{i} \left(1 - \frac{r}{K}\right) + \frac{1}{i(i-1)} > 0, \quad (46)$$

$$\lambda_i + \mu_i = \frac{2}{i} \left(1 - \frac{r}{K}\right) > 0. \quad (47)$$

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.,

$$\frac{1}{x - r/K} \left(1 - \frac{r}{K}\right)^2 \geq \lambda_i x + \mu_i, \quad \forall\, x \in [1, c_{i-1}] \cup [c_i, \infty). \quad (48)$$

Therefore,

$$\frac{1}{c_j - r/K}\left(1-\frac{r}{K}\right)^2 \geq \lambda_i c_j + \mu_i, \quad \forall j \in [K-1]. \tag{49}$$

Back to the converse, from (33), the communication load L is lower bounded as

$$\begin{aligned}
L &\geq \frac{\sum_{k=1}^{K} H(X_k)}{NKV} \\
&\overset{(a)}{\geq} \sum_{j=1}^{K-1} \frac{b_j}{NK}\cdot\frac{1}{j} \\
&\overset{(b)}{=} \frac{1}{N(K-r)}\cdot\sum_{j=1}^{K-1} b_j \cdot \frac{1}{c_j - r/K}\left(1-\frac{r}{K}\right)^2 \\
&\overset{(c)}{\geq} \frac{1}{N(K-r)}\sum_{j=1}^{K-1} b_j\left(\lambda_i c_j + \mu_i\right) \\
&= \frac{1}{N(K-r)}\sum_{j=1}^{K-1} b_j\left(\lambda_i\left(\left(1-\frac{r}{K}\right)j + \frac{r}{K}\right) + \mu_i\right) \\
&= \frac{1}{N(K-r)}\cdot\left(\lambda_i\left(1-\frac{r}{K}\right)\cdot\sum_{j=1}^{K-1} j b_j \right. \\
&\qquad \left. + \left(\lambda_i\frac{r}{K}+\mu_i\right)\cdot\sum_{j=1}^{K-1} b_j\right) \\
&= \frac{\lambda_i}{NK}\cdot\sum_{j=1}^{K-1}(j-1)b_j + \frac{\lambda_i+\mu_i}{N(K-r)}\cdot\sum_{j=1}^{K-1} b_j \\
&\overset{(d)}{\geq} \frac{\lambda_i}{NK}\cdot(c-1)NK + \frac{\lambda_i+\mu_i}{N(K-r)}\cdot N(K-r) \\
&= \lambda_i c + \mu_i \\
&\overset{(e)}{=} -\frac{1}{i(i-1)}c - \frac{2}{Ki}r + \frac{2i-1}{i(i-1)},
\end{aligned} \tag{50}$$

where (a) follows from Lemma 1; (b) follows from (42); (c) follows from (49); (d) follows from (37), (38) and (45), (47); and (e) follows from (45), (46). Since the SCC triples P_{i-1}, P_i, and P_K defined in (13) satisfy inequality (50) 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 [5] (cf. (15)) implies that, for any c, we have

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

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 (10). Then for each $t \in [2:K]$, define

$$c_{\mathbf{A}}^{(t)} \triangleq \frac{T}{KF} + \left(1-\frac{T}{KF}\right)(t-1) \tag{51a}$$

$$L_{\mathbf{A}}^{(t)} \triangleq \frac{1}{t-1}\left(1-\frac{T}{KF}\right) \tag{51b}$$

and notice that by Theorem 1:

$$r_{\mathbf{A}} = \frac{T}{F} \tag{52a}$$

$$c_{\mathbf{A}} = \sum_{t=2}^{K}\theta'_t c_{\mathbf{A}}^{(t)} \tag{52b}$$

$$L_{\mathbf{A}} = \sum_{t=2}^{K}\theta'_t L_{\mathbf{A}}^{(t)}. \tag{52c}$$

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

$$\beta_i \triangleq -\frac{1}{i(i-1)}, \tag{53a}$$

$$\gamma_i \triangleq \frac{2}{i}\left(1-\frac{T}{KF}\right) + \frac{1}{i(i-1)}. \tag{53b}$$

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,

$$L_{\mathbf{A}} = -\frac{1}{i(i-1)}c_{\mathbf{A}} - \frac{2}{Ki}r_{\mathbf{A}} + \frac{2i-1}{i(i-1)}. \tag{54}$$

In the following, we show that (54) holds if, and only if,

$$\theta'_t = 0, \ \forall\, t \in [2:K]\backslash\{i, i+1\}. \tag{55}$$

With the definition of θ'_t in (10), 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 4.

Next, we write

$$\begin{aligned}
L_{\mathbf{A}} &= \sum_{t=2}^{K}\theta'_t L_{\mathbf{A}}^{(t)} \\
&\overset{(a)}{=} \sum_{t=2}^{K}\theta'_t\cdot\frac{1}{c_{\mathbf{A}}^{(t)} - T/(KF)}\left(1-\frac{T}{KF}\right)^2 \\
&\overset{(b)}{\geq} \sum_{t=2}^{K}\theta'_t\cdot\left(\beta_i c_{\mathbf{A}}^{(t)} + \gamma_i\right) \\
&\overset{(c)}{=} \beta_i c_{\mathbf{A}} + \gamma_i \\
&\overset{(d)}{=} -\frac{1}{i(i-1)}c_{\mathbf{A}} + \frac{2}{i}\left(1-\frac{T}{KF}\right) + \frac{1}{i(i-1)} \\
&\overset{(e)}{=} -\frac{1}{i(i-1)}c_{\mathbf{A}} - \frac{2}{Ki}r_{\mathbf{A}} + \frac{2i-1}{i(i-1)},
\end{aligned} \tag{56}$$

where (a) holds by simple algebraic manipulations on (51); (b) is proved below; (c) holds by (52) and because $\sum_{t=2}^{K}\theta'_t = \sum_{t=1}^{K}\theta_t = 1$; (d) holds by (53); and (e) holds by (52a).

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

$$\begin{aligned} f_1 &: \quad c \mapsto \beta_i c + \gamma_i, \\ f_2 &: \quad c \mapsto \frac{1}{c - T/(KF)} \left(1 - \frac{T}{KF}\right)^2. \end{aligned}$$

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,

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

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

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

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

Combining this with (56), we conclude that (54) holds if and only if $\theta'_t = 0$ for all $t \in [2:K] \backslash \{i, i+1\}$, i.e., (55) 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 4, $\mathbf{A}$ needs to be $(i+1)$-regular. In the following, we show that $\mathbf{A}$ 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

$$t_j \geq i, \quad \forall\ j \in [F], \tag{57}$$

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

$$\sum_{j=1}^{F} t_j \geq iF. \tag{58}$$

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

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

and thus both the inequalities (57) and (58) must hold with equality. Therefore, each row of $\mathbf{A}$ has exactly i "$*$" entries and the following lemma (which rephrases [33, Lemma 2]) concludes the proof.

Lemma 3 (From [33]): 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

$$1 \leq q < \frac{K}{2}$$

and set

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

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

$$F = q^m \tag{59}$$

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

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:

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

For convenience, we will write

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

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

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

For convenience of notation, we will write

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

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

$$k - 1 = k_1 q + k_0$$

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

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

Construction 1: Consider a fixed positive integers K and let q, m, F be given as in (59). Construct the array $\mathbf{A}_q^K = [a_{j,k}]$ as follows

- If $k \leq qm$:

$$a_{j,k} = \begin{cases} *, & \text{if } j_{k_1} = k_0 \\ \left(j_{k_1} \ominus_q k_0 \ominus_q 1, j_{m-1}, j_{m-2}, \right. & \\ \quad \left. \ldots, j_{k_1+1}, k_0, j_{k_1-1}, \ldots, j_0\right)_q^{q-1}, & \text{if } j_{k_1} \neq k_0 \end{cases} \tag{60}$$

TABLE I
THE CONSTRUCTION OF ARRAY $\mathbf{A}_2^5$

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

- and if $k > qm$:

$$a_{j,k} = \begin{cases} *, & \text{if } \sum_{l=0}^{m-1} j_l = k_0 \\ \left(k_0 \ominus_q \sum_{l=0}^{m-1} j_l \ominus_q 1,\right. & \\ \quad \left. j_{m-1}, j_{m-2}, \ldots, j_0\right)_q^{m+1}, & \text{if } \sum_{l=0}^{m-1} j_l \neq k_0 \end{cases} \tag{61}$$

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 (9), but uses a different notation for the ordinary symbols.

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 $g \triangleq \lfloor \frac{K}{q} \rfloor$ and $m \triangleq \lceil \frac{K}{q} \rceil - 1$.

Proof: See Appendix B. ■

For $q|K$ the Comp-PDA $\mathbf{A}_q^K$ specializes to the PDA proposed in [33, 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 (16). Next, we prove (17), i.e., that D_q is close to the SCC triple

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

Combining this with (16), yields

$$r_{P_m} - r_{D_q} = m - \frac{K}{q}, \tag{62}$$

$$\begin{aligned} & c_{P_m} - c_{D_q} \\ = \;& m\left(1 - \frac{m-1}{K}\right) - \frac{1}{q} - m\left(1 - \frac{1}{q}\right)\left(2 - \frac{(m+1)q}{K}\right) \\ = \;& -\frac{1}{K}m^2 + \left(1 + \frac{1}{K}\right)m - \frac{1}{q} \\ & + \frac{q-1}{K}m^2 - \frac{(2K-q)(q-1)}{Kq}m \\ = \;& \frac{q-2}{K}m^2 + \frac{q^2 - Kq + 2K}{Kq}m - \frac{1}{q}, \end{aligned} \tag{63}$$

$$\begin{aligned} & L_{P_m} - L_{D_q} \\ = \;& \frac{1}{m}\left(1 - \frac{m}{K}\right) - \left(\frac{1}{m} + \frac{(m+1)q - K}{K(m-1)}\right) \cdot \left(1 - \frac{1}{q}\right) \\ = \;& \frac{1}{m} - \frac{1}{K} - \frac{1}{m}\left(1 - \frac{1}{q}\right) - \frac{(m+1)q - K}{K(m-1)}\left(1 - \frac{1}{q}\right) \\ = \;& \frac{1}{mq} - \frac{1}{K} - \frac{(m-1)q + 2q - K}{K(m-1)} \cdot \frac{q-1}{q} \\ = \;& \frac{1}{mq} + \frac{(K-2q)(q-1)}{K(m-1)q} - \frac{q}{K}. \end{aligned} \tag{64}$$

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

1) (17a) follows immediately from (62);
2) (63) 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 (64), decreases with m, we obtain (17c).

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

$$\begin{aligned} F_{P_m} &= \binom{K}{m} \\ &\overset{(a)}{\geq} \binom{mq}{m} \\ &= \frac{(mq)!}{m!\,(m(q-1))!} \\ &\overset{(b)}{\geq} \frac{\sqrt{2\pi}}{e^2} \frac{(mq)^{mq}\sqrt{mq}}{(m)^m\sqrt{m}(m(q-1))^{m(q-1)}\sqrt{m(q-1)}} \\ &= \frac{\sqrt{2\pi}}{e^2}\sqrt{\frac{q}{m(q-1)}} \cdot \left(\frac{q^q}{(q-1)^{q-1}}\right)^m \\ &= \frac{\sqrt{2\pi}}{e^2}\sqrt{\frac{q}{m(q-1)}}\left(\frac{q}{q-1}\right)^{m(q-1)} F_{D_q}, \end{aligned}$$

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.

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 (8a), 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 (8a). 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 (8a), let $L^*(r,c)$ be the function such that $(r,c,L^*(r,c))\in\mathcal{F}$. Then by (50), it has strictly positive directional derivative in any direction $(r\le 0, c\le 0)$ 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

$$r'\le r,\quad c'\le c,\quad L'\le L^*(r,c). \tag{65}$$

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

$$(r',c',L^*(r',c'))\in\mathcal{O}, \tag{66}$$

and triples where this is not the case.

1) Assume that (66) holds. If $r'=r$ and $c'=c$, then obviously, $L'\ge L^*(r,c)$, thus all equalities in (65) hold. If $r'<r$ or $c'<c$, then

$$L^*(r',c')>L^*(r,c), \tag{67}$$

simply because the directional derivative along $(r'-r,c'-c)$ is strictly positive by (50). Since $(r',c',L')\in\mathcal{R}$, we have $L'\ge L^*(r',c')$ and thus by (67), $L'>L^*(r,c)$, which contradicts (65).

2) Assume now that (66) is violated. Then, $(r',c',L^*(r',c'))$ must lie on at least one of the $K-1$ facets

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

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

$$L'\ge L^*(r',c')=L^*(r',c'')\overset{(a)}{>}L^*(r,c), \tag{68}$$

where (a) follows by proof step 1). But (68) contradicts with (65).

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

APPENDIX B
PROOF OF LEMMA 4

It is easy to verify that $\mathbf{A}_q^K$ 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,

$$\begin{aligned}F_{\mathbf{A}_q^K}&=q^m\\ T_{\mathbf{A}_q^K}&=q^{m-1}K\\ S_{\mathbf{A}_q^K}&=(q-1)q^m.\end{aligned}$$

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 (60) and (61), the entry of $\mathbf{A}_q^K$ in row $j=(j_{m-1},j_{m-2},\dots,j_0)_q\in[q^m]$ and column $k=(k_1,k_0)_q^{m+1}\in[K]$ equals $s=(s_m,s_{m-1},\dots,s_0)_q^{q-1}\in[(q-1)q^m]$ if, and only if, the following two conditions hold:

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

$$\begin{aligned}j&=(s_{m-1},\dots,s_{k_1+1},\\ &\qquad s_{k_1}\oplus_q s_m\oplus_q 1,s_{k_1-1},\dots,s_0)_q,\quad(69)\\ k&=(k_1,s_{k_1})_q^{m+1}.\quad(70)\end{aligned}$$

2) when $k_1=m$,

$$\begin{aligned}j&=(s_{m-1},s_{m-2},\dots,s_0)_q,\quad(71)\\ k&=\big(k_1,\sum_{l=0}^{m}s_l\oplus_q 1\big)_q^{m+1},\quad(72)\end{aligned}$$

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:

$$a_{j,k}=a_{j',k'}=s.$$

Let $k=(k_1,k_0)$ and $k'=(k_1',k_0')$. Notice now that by (69)–(72), if $k_1=k_1'$ then also $k_0=k_0'$, $k=k'$, and $j=j'$.

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

1) Consider the case $k_1'<m$, and notice that $0\le s_m\le q-2$ and $s_m\oplus_q 1\in\{1,2,\dots,q-1\}$. Therefore, by (69):

$$j_{k_1'}=s_{k_1'}\quad\ne\quad s_{k_1'}\oplus_q s_m\oplus_q 1=j'_{k_1'}, \tag{73}$$

and hence $j' \neq j$. Notice also that (70) implies $s_{k'_1} = k'_0$. Since $j_{k'_1} = s_{k'_1}$ by (73), we conclude $j_{k'_1} = k'_0$, and thus the entry $a_{j,k'} = *$ by (60). 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 (69)–(72) and since $s_m \oplus_q 1 \neq 0$:

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

and thus $j \neq j'$. We now argue that $a_{j,k'} = *$. To this end, notice that since $k'_1 = m$, (72) implies $k'_0 = \sum_{l=0}^{m} s_l \oplus_q 1$; since $k_1 < m$, (69) 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 (61). 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 (60) and (61), 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 (69) and (70). 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 (71) and (72), and the fact $0 \leq k_0 < K - qm$, (j,k) can be uniquely determined if, and only if,

$$0 \leq \sum_{l=0}^{m} s_l \oplus_q 1 < K - qm. \tag{74}$$

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 (74) 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

$$\begin{aligned} \theta_{m+1} &= \frac{(K - qm)(q-1)q^{m-1}}{Kq^m - Kq^{m-1}} \\ &= 1 - \frac{m((m+1)q - K)}{K} \\ &\leq 1. \end{aligned}$$

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. Theory Workshop (ITW)*, Guangzhou, China, Nov. 2018, pp. 1–5.

[2] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," in *Proc. 6th USENIX OSDI*, Dec. 2004, pp. 1–13.

[3] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly, "Dryad: Distributed data-parallel programs from sequential building blocks," in *Proc. 2nd ACM SIGOPS/EuroSys Eur. Conf. Comput. Syst.*, Mar. 2007, pp. 59–72.

[4] S. Li, M. A. Maddah-Ali, and A. S. Avestimehr, "Compressed coded distributed computing," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Vail, CO, USA, Jun. 2018, pp. 2032–2036.

[5] 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.

[6] 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, Nov. 2017, pp. 279–283.

[7] 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.

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

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

[10] Q. Yu, M. Maddah-Ali, and S. Avestimehr, "Polynomial codes: An optimal design for high-dimensional coded matrix multiplication," in *Proc. The 31st Annu. Conf. Neural Inf. Process. Syst. (NIPS)*, Long Beach, CA, USA, May 2017, pp. 1–11.

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

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

[13] H. Park, K. Lee, J.-Y. Sohn, C. Suh, and J. Moon, "Hierarchical coding for distributed computing," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Vail, CO, USA, Jun. 2018, pp. 1630–1634.

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

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

[16] 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, Jun. 2018, pp. 1993–1997.

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

[18] 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)*, Paris, France, May 2017, pp. 1–7.

[19] 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.

[20] S. Li, Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Edge-facilitated wireless distributed computing," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Washington, DC, USA, Dec. 2016, pp. 1–7.

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

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

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

[24] M. A. Attia and R. Tandon, "Information theoretic limits of data shuffling for distributed learning," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Washington, DC, USA, Dec. 2016, pp. 1–6.

[25] A. Elmahdy and S. Mohajer, "On the fundamental limits of coded data shuffling," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Vail, CO, USA, Jun. 2018, pp. 716–720.
[26] 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, Nov. 2017, pp. 289–293.
[27] S. Srinivasavaradhan, L. Song, and C. Fragouli, "Distributed computing trade-offs with random connectivity," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Vail, CO, USA, Jun. 2018, pp. 1281–1285.
[28] R. Tandon, Q. Lei, A. G. Dimakis, and N. Karampatziakis, "Gradient coding: Avoiding stragglers in synchronous gradient descent," in *Proc. 34th Int. Conf. Mach. Learn. (ICML)*, Aug. 2017, pp. 3368–3376.
[29] N. Raviv, I. Tamo, R. Tandon, and A. G. Dimakis, "Gradient coding from cyclic MDS codes and expander graphs," *IEEE Trans. Inf. Theory*, vol. 66, no. 12, pp. 7475–7489, Dec. 2020.
[30] Z. Charles and D. Papailiopoulos, "Gradient coding using the stochastic block model," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Vail, CO, USA, Jun. 2018, pp. 1998–2002.
[31] W. Halbawi, N. Azizan, F. Salehi, and B. Hassibi, "Improving distributed gradient descent using Reed–Solomon codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Vail, CO, USA, Jun. 2018, pp. 2027–2031.
[32] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *IEEE Trans. Inf. Theory*, vol. 60, no. 5, pp. 2856–2867, May 2014.
[33] 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.
[34] 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.
[35] 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.
[36] K. Konstantinidis and A. Ramamoorthy, "Leveraging coding techniques for speeding up distributed computing," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Abu Dhabi, United Arab Emirates, Dec. 2018, pp. 1–6.
[37] K. Konstantinidis and A. Ramamoorthy, "Resolvable designs for speeding up distributed computing," *IEEE/ACM Trans. Netw.*, vol. 28, no. 4, pp. 1657–1670, Aug. 2020.
[38] N. Woolsey, R.-R. Chen, and M. Ji, "A new combinatorial design of coded distributed computing," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Vail, CO, USA, Jun. 2018, pp. 726–730.
[39] Q. Yan, X. Tang, and Q. Chen, "Placement delivery array and its applications," in *Proc. IEEE Inf. Theory Workshop (ITW)*, Guangzhou, China, Nov. 2018, pp. 1–5.
[40] V. Ramkumar and P. V. Kumar, "Coded MapReduce schemes based on placement delivery array," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Paris, France, Jul. 2019, pp. 3087–3091.
[41] Q. Yan, M. Wigger, S. Yang, and X. Tang, "A fundamental storage-communication tradeoff for distriubted computing with straggling nodes," *IEEE Trans. Commun.*, vol. 68, no. 12, pp. 7311–7327, Dec. 2020.
[42] Q. Yu, "Coded computing: A transformative framework for resilient, secure, private, and communication efficient large scale distributed computing," Ph.D. dissertation, Dept. Elect. Comput. Eng., University of Southern California, Los Angeles, CA, USA, Aug. 2020.
[43] P. Krishnan, L. Natarajan, and V. Lalitha, "An umbrella converse for data exchange: Applied to caching, computing, shuffling & rebalancing," in *Proc. IEEE Inf. Theory Workshop (ITW)*, Riva del Garda, Italy, Apr. 2021, pp. 1–5.
[44] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

Qifa Yan (Member, IEEE) received the B.S. degree in mathematics and applied mathematics from Shanxi University, Taiyuan, China, in 2010, and the Ph.D. degree in communication and information system from the School of Information Science and Technology, Southwest Jiaotong University, Chengdu, China, in 2017. From November 2017 to October 2019, he was a Joint Post-Doctoral Researcher with Télécom Paris, Institut Politechnique de Paris, and the CentraleSupélec, Paris-Saclay University, France. From January 2020 to January 2021, he was a Post-Doctoral Research Fellow with the Department of Electrical and Computer Engineering, University of Illinois, Chicago, USA. He is currently an Assistant Professor with the School of Information Science and Technology, Southwest Jiaotong University. His research interests include caching networks, distributed computing, other fields related to wireless networks, information theory, and coding theory.

Sheng Yang (Member, IEEE) received the B.E. degree in electrical engineering from Shanghai Jiao Tong University, Shanghai, China, in 2001, the diplôme d'ingénieur and the M.Sc. degree in electrical engineering from Telecom ParisTech, Paris, France, in 2004, and the Ph.D. degree from the Université de Pierre et Marie Curie (Paris VI) in 2007. From October 2007 to November 2008, he was with the Motorola Research Center, Gif-sur-Yvette, France, as a Senior Staff Research Engineer. Since December 2008, he has been with the CentraleSupélec, Paris-Saclay University, where he is currently a Full Professor. Since April 2015, he has been holding the Honorary Associate Professorship with the Department of Electrical and Electronic Engineering, The University of Hong Kong (HKU). He received the 2015 IEEE ComSoc Young Researcher Award for the Europe, Middle East, and Africa Region (EMEA). He was an Associate Editor of the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS from 2015 to 2020. He is an Associate Editor of the IEEE TRANSACTIONS ON INFORMATION THEORY.

Michèle Wigger (Senior Member, IEEE) received the M.Sc. degree (Hons.) in electrical engineering and the Ph.D. degree in electrical engineering from ETH Zurich in 2003 and 2008, respectively. In 2009, she was first a Post-Doctoral Fellow with the University of California, San Diego, USA, and then joined Telecom Paris, France, where she is currently a Full Professor. She has held a Visiting Professor appointments with the Technion-Israel Institute of Technology and ETH Zurich. Her research interests include multiterminal information theory, in particular in distributed source coding and in capacities of networks with states, feedback, and user cooperation or caching. She serves as an Associate Editor for *Shannon Theory* (2016–2019) and the IEEE TRANSACTIONS ON INFORMATION THEORY (since 2021) and as a Distinguished Lecturer of the IEEE Information Theory Society (2022–2023). Previously, she served as an Associate Editor for the IEEE COMMUNICATION LETTERS (2012–2015) and on the Board of Governors of the IEEE Information Theory Society (2017–2019).