On the Fundamental Limits of Coded Data Shuffling

Adel Elmahdy
Department of ECE, University of Minnesota
Email: adel@umn.edu

Soheil Mohajer
Department of ECE, University of Minnesota
Email: soheil@umn.edu

***Abstract*—We consider the data shuffling problem, in which a master node is connected to a set of worker nodes, via a shared link, in order to communicate a set of files to the worker nodes. The master node has access to a database of files. In every shuffling iteration, each worker node processes a new subset of files, and has excess storage to partially cache the remaining files. We characterize the exact rate-memory trade-off for the worst-case shuffling under the assumption that cached files are uncoded, by deriving the minimum communication rate for a given storage capacity per worker node. As a byproduct, the exact rate-memory trade-off for any random shuffling is characterized when the number of files is equal to the number of worker nodes. We propose a novel deterministic and systematic coded shuffling scheme, which improves the state of the art. Then, we prove the optimality of our proposed scheme by deriving a matching lower bound and showing that the placement phase of the proposed coded shuffling scheme is optimal over all shuffles.**

I. Introduction

Data Shuffling is one of the core components of distributed computing algorithms. In distributed learning algorithms, the data-set is randomly shuffled among a cluster of worker nodes to provide new batches of the data points at each learning epoch. It has been shown that data shuffling enhances the learning model quality and leads to significant statistical gains [1]. However, the statistical benefits of data shuffling in distributed computing systems come at a price. For each new shuffle, the entire data-set is communicated from the master node to the worker nodes. Consequently, this leads to performance bottlenecks due to the communication overhead.

Inspired by the coded caching introduced by Maddah-Ali et al. [2], Lee et al. [1] proposed the first coded shuffling algorithm, based on random storage placement, that leverages the excess storage of the local caches of the worker nodes to slash the communication bottlenecks. Several works investigated the data shuffling problem and proposed different shuffling algorithms [3]–[6]. A similar coding idea was proposed in MapReduce distributed computing framework, e.g. [7].

The most related work to ours was done by Attia and Tandon [3], [4], where a data shuffling problem, consisting of a master node that communicates data points to worker nodes with limited storage capacity, was studied. An information-theoretic formulation of the shuffling problem was proposed for data delivery and storage update phases. Furthermore, the worst-cast communication rate is defined to be the maximum communication load from the server to the worker nodes over all possible data shuffles. Accordingly, the authors characterized the optimal trade-off between the storage capacity per worker node and the worst-case communication rate for certain cases. This rate was characterized when the number of worker nodes is limited to $K \in \{2, 3\}$ in [3]. Furthermore, the special case of *no-excess storage* (arbitrary N and K, but $S = N/K$) was addressed in [4]. However, the proposed schemes in these works do not generalize for arbitrary parameters.

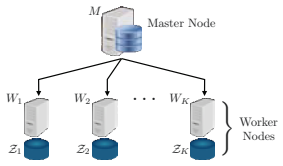

Fig. 1. Data shuffling in distributed computing systems.

In this paper, we propose a novel systematic and deterministic coded shuffling scheme which improves the state of the art and achieves a lower communication rate, for random shuffling when $N = K$. Furthermore, inspired by a recent work of [8], the optimality of the proposed coded shuffling scheme is demonstrated through a matching converse proof. We show that the placement phase, assuming uncoded prefetching, of the proposed scheme is optimal. We also use this canonical case as a building block, and propose a shuffling strategy for $N \geq K$, which is optimum for the worst-case shuffle.

II. Problem Formulation

Let $[K]$ denote $\{1, 2, \ldots, K\}$. Fig. 1 depicts a distributed computing system with a master node, denoted by M, and a set of K worker nodes, denoted by $\mathcal{W} = \{W_i : i \in [K]\}$. The master node is assumed to have access to the entire data-set of N files[1], denoted by $\mathcal{F} = \{F^j : j \in [N]\}$, where the size of each file is normalized to 1 unit. Each worker node W_i has a cache $\mathcal{Z}_i$ that can store up to S files. At every iteration, the master node randomly shuffles N files, and assigns N/K files[2] to each worker node to store in its cache and perform a local computational task. This imposes the constraint $S \geq N/K$ on the size of the cache at each worker node. For a given iteration t, we denote by $u(i)$ the set of indices of the files to be processed by W_i, and by $\mathcal{P}^i$ the portion of the cache of W_i dedicated to the *under-processing* files: $\mathcal{P}^i = \{F^j : j \in u(i)\}$. The subsets $\{u(i) : i \in [K]\}$ provide a partitioning for the set of files, i.e., $u(i) \cap u(j) = \varnothing$ for $i \neq j$, and $\bigcup_{i=1}^{K} u(i) = [N]$. Similarly, $d(i)$ denotes the subset of N/K files to be processed by W_i at iteration $t + 1$, where $\{d(i) : i \in [K]\}$ also form a partitioning for $\mathcal{F}$. When $S > N/K$, each worker node can

[1]We study the shuffling problem under the practical assumption $N \geq K$.
[2]Unless otherwise stated, we assume that N/K and $S/(N/K)$ are integers.

978-1-5386-4780-6/18/$31.00©2018 IEEE

cache (parts of) the other files in $\mathcal{F}$, in addition to the N/K files being processed. We denote by $\mathcal{E}_i = \mathcal{Z}_i \backslash \mathcal{P}^i$ the remaining space of the cache of W_i, which is called the *excess* cache and used to stores sub-files of other files. Therefore, $\mathcal{Z}_i = \mathcal{P}^i \cup \mathcal{E}_i$.

Filling the excess part of the cache is performed regardless of the new assigned subsets $\{d(i) : i \in [K]\}$. Between iterations t and $t+1$, the master node should compute and broadcast a message (a function of all files in $\mathcal{F}$), such that each worker node W_i can retrieve all files in $d(i)$ from its cached data $\mathcal{Z}_i$ and the broadcast message $\mathcal{X}$. The communication load R is defined as the size of the broadcast message $\mathcal{X}$. We interchangeably refer to R as *rate* and *communication load*. The goal is to minimize R for any $\{d(i) : i \in [K]\}$. For $S \geq N$, we have $R = 0$ since each worker node can store all the files and no communication is needed. Thus, we can focus on the regime $N/K \leq S \leq N$.

A demand profile is defined as a directed graph $\mathcal{G}(V, E)$, where V is the set of K vertices and E is the set of N directed edges. An edge $e_j = (i, \ell) \in E$ with $j \in [N]$ indicates $j \in u(i) \cap d(\ell)$, i.e., file F^j is processed by worker node W_i at iteration t, and by worker node W_ℓ at iteration $t+1$.

File Partitioning and Labeling: Let $\widehat{S} = S/(N/K)$. At every iteration, each file F^j, that is processed by W_i, is split into $\binom{K-1}{\widehat{S}-1}$ equal sub-files, and the subfiles are labeled as $F^j = \{F^j_\Gamma : \Gamma \subseteq [K]\backslash\{i\}, |\Gamma| = \widehat{S}-1\}$, for $i \in [K]$, $j \in [N]$. Since the size of each file is normalized to 1, the size of each sub-file will be $1/\binom{K-1}{\widehat{S}-1}$. We also define dummy sub-files $F^j_\Gamma = 0$, for every $\Gamma \subseteq [K]$ with $j \in \Gamma$ or $|\Gamma| \neq \widehat{S} - 1$.

Sub-file Placement: The cache $\mathcal{Z}_i$ of W_i consists of two parts: (1) the under-processing part $\mathcal{P}^i$, in which *all sub-files* of files $\left\{F^j : j \in u(i)\right\}$ are cached; (2) the excess storage part $\mathcal{E}_i$, which is equally distributed among all other files. We denote by $\mathcal{E}^\ell_i$ the portion of $\mathcal{E}_i$ dedicated to the file F^ℓ, in which all sub-files F^ℓ_Γ with $i \in \Gamma$ are cached. Hence, we have

$$\mathcal{Z}_i = \mathcal{P}^i \cup \mathcal{E}_i = \mathcal{P}^i \cup \left(\bigcup\nolimits_{\ell \in [N]\backslash u(i)} \mathcal{E}^\ell_i\right) \tag{1}$$

$$\mathcal{P}^i = \left\{F^j_\Gamma : j \in u(i),\, \Gamma \subseteq [K]\backslash\{i\},\, |\Gamma| = \widehat{S}-1\right\}, \tag{2}$$

$$\mathcal{E}^\ell_i = \left\{F^\ell_\Gamma : \ell \notin u(i),\, i \in \Gamma \subseteq [K],\, |\Gamma| = \widehat{S}-1\right\}, \tag{3}$$

for $i \in [K]$. For any worker node W_i, there are N/K complete files in $\mathcal{P}^i$. Moreover, for each of the remaining $N - N/K$ files, there are $\binom{K-2}{\widehat{S}-2}$ sub-files, out of a total $\binom{K-1}{\widehat{S}-1}$ sub-files, that are cached in the excess storage part. Thus, we have

$$|\mathcal{Z}_i| = |\mathcal{P}^i| + \sum_{\ell \in [N]\backslash u(i)} |\mathcal{E}^\ell_i| = \frac{N}{K} + \left(N - \frac{N}{K}\right)\frac{\binom{K-2}{\widehat{S}-2}}{\binom{K-1}{\widehat{S}-1}} = S,$$

which satisfies the memory constraints.

Let $\mathcal{Q}_i$, for $i \in [K]$, denote the set of sub-files to be processed by W_i at iteration $t+1$, which are not available in its cache $\mathcal{Z}_i$ at iteration t, that is,

$$\mathcal{Q}_i = \left\{F^\ell_\Gamma : \ell \in d(i),\, \ell \notin u(i),\, i \notin \Gamma,\, |\Gamma| = \widehat{S}-1\right\}. \tag{4}$$

It is easy to see that each worker node needs to decode at most $\binom{K-2}{\widehat{S}-1}$ sub-files for each of the N/K files in $d(i)$, in order to process them at iteration $t+1$.

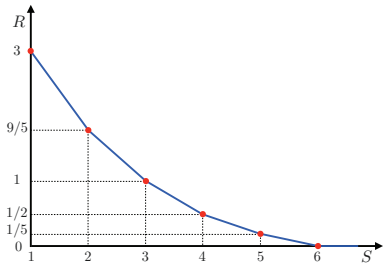

Fig. 2. Trade-off curve between broadcast communication rate R and storage capacity per worker node S, when $N = K = 6$ and $\gamma = 3$.

III. The Canonical Setting ($N = K$)

Since $N = K$, then $\widehat{S} = S$. Without loss of generality, we assume that W_i processes file F^i at iteration t, i.e., $u(i) = i$, for $i \in [K]$, otherwise we can relabel the files. The number of cycles in the underlying graph of the demand profile is denoted by γ, with cycle lengths $(\ell_1, \ell_2, \ldots, \ell_\gamma)$ where $\sum_{i=1}^{\gamma} \ell_i = K$. The following theorems summarize our main results.

Theorem 1. *For a distributed computing system with K worker nodes each with a cache of size $S \in \{1, 2, \ldots, N\}$ files*[3]*, the broadcast communication rate R required to shuffle $N = K$ files for any demand profile is upper bounded by*[4]

$$R \leq \frac{\binom{K-1}{S}}{\binom{K-1}{S-1}}. \tag{5}$$

The next theorem provides an achievable rate (depending on the demand profile) by an opportunistic coding scheme.

Theorem 2. *For a distributed computing system with K worker nodes each with a cache of size $S \in \{1, 2, \ldots, N\}$ files, the shuffling of $N = K$ files for a given demand profile that comprises γ cycles can be performed by broadcasting a message of size R, where*

$$R \leq \frac{\binom{K-1}{S} - \binom{\gamma-1}{S}}{\binom{K-1}{S-1}}. \tag{6}$$

Theorem 3. *For the distributed computing system introduced in Theorem 2, the broadcast communication rate R required to shuffle $N = K$ files for a given demand profile that comprises γ cycles is lower bounded by*

$$R \geq \frac{\binom{K-1}{S} - \binom{\gamma-1}{S}}{\binom{K-1}{S-1}}, \tag{7}$$

Theorems 2 and 3 prove the optimality of the proposed shuffling scheme. Fig. 2 captures the trade-off curve between R and S for $N = K = 6$ and $\gamma = 3$. Detailed proofs of the theorems are provided in the extended version of this paper [9].

IV. Coded Shuffling Scheme for any Random Shuffling ($N = K$)

A. Illustrative Example

Consider a shuffling system with a master node and $K = 6$ worker nodes. The size of the cache at each worker

[3]For non-integer values of S, the lower convex envelope of the N corner points is achievable by space-sharing.

[4]Note that $\binom{n}{k} = 0$ when $n < k$.

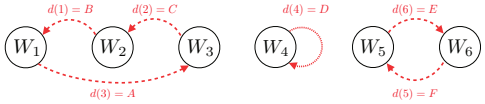

Fig. 3. Demand profile for a data shuffling system with $N = K = 6$ and $\gamma = 3$. Worker nodes W_1, W_2, W_3, W_4, W_5 and W_6 are processing files A, B, C, D, E, and F, respectively.

node is $S = 3$ files. There are $N = 6$ files, denoted by $\{F^1, F^2, F^3, F^4, F^5, F^6\}$. For simplicity, we rename the files as $\{A, B, C, D, E, F\}$, respectively. Assume worker nodes W_1, W_2, W_3, W_4, W_5 and W_6 are processing files A, B, C, D, E, and F, respectively. The demand profile of worker nodes is depicted by Fig. 3. It comprises $\gamma = 3$ cycles, with cycle lengths $(\ell_1, \ell_2, \ell_3) = (3, 1, 2)$, i.e., That is, $d(1) = B$, $d(2) = C$, $d(3) = A$, $d(4) = D$, $d(5) = F$, $d(6) = E$. Fig. 4 captures the cache organization of worker nodes, along with the sub-files assigned to them by the master node at the next iteration. Note that $\mathcal{P}^i$ and $\mathcal{E}_i$, for $i \in [K]$, are designed according to (2) and (3), respectively. Moreover, we use (9) and (10) to design the broadcast message $\mathcal{X}$. For example, X_{123} is expressed as

$$\begin{aligned} X_{123} &= \left(F^1_{23} \oplus F^2_{13} \oplus \left(F^2_{34} \oplus F^2_{35} \oplus F^2_{36}\right)\right) \oplus \left(F^2_{13} \oplus F^3_{12} \oplus \left(F^3_{14}\right.\right. \\ &\quad \left.\left. \oplus F^3_{15} \oplus F^3_{16}\right)\right) \oplus \left(F^3_{12} \oplus F^1_{23} \oplus \left(F^1_{24} \oplus F^1_{25} \oplus F^1_{26}\right)\right). \\ &= A_{24} \oplus A_{25} \oplus A_{26} \oplus B_{34} \oplus B_{35} \oplus B_{36} \oplus C_{14} \oplus C_{15} \oplus C_{16}. \end{aligned}$$

Similarly, the set of other functions transmitted by the master node to the worker nodes is expressed as

$$\begin{aligned} X_{124} &= A_{24} \oplus B_{34} \oplus B_{45} \oplus B_{46} \oplus C_{14}, \\ X_{125} &= A_{25} \oplus B_{35} \oplus B_{45} \oplus B_{56} \oplus C_{15} \oplus E_{12} \oplus F_{12}, \\ X_{134} &= A_{24} \oplus A_{45} \oplus A_{46} \oplus B_{34} \oplus C_{14}, \\ X_{135} &= A_{25} \oplus A_{45} \oplus A_{56} \oplus B_{35} \oplus C_{15} \oplus E_{13} \oplus F_{13}, \\ X_{145} &= A_{45} \oplus B_{45} \oplus E_{14} \oplus F_{14}, \\ X_{234} &= A_{24} \oplus B_{34} \oplus C_{14} \oplus C_{45} \oplus C_{46}, \\ X_{235} &= A_{25} \oplus B_{35} \oplus C_{15} \oplus C_{45} \oplus C_{56} \oplus E_{23} \oplus F_{23}, \\ X_{245} &= B_{45} \oplus C_{45} \oplus E_{24} \oplus F_{24}, \\ X_{345} &= A_{45} \oplus C_{45} \oplus E_{34} \oplus F_{34}. \end{aligned} \tag{8}$$

For each worker node, Fig. 6 shows the received functions from the master node after removing the sub-files that exist in its cache. The decoding procedure of the proposed coded shuffling scheme is analogous to interference mitigation techniques in wireless communications. To present this analogy, we focus on three different cases of the decoding procedure.

(1) Decoding C_{14} from X_{124} by W_2:

$$X_{124} = \underbrace{C_{14}}_{\text{Desired sub-file}} \oplus \underbrace{A_{24} \oplus B_{34} \oplus B_{45} \oplus B_{46}}_{\text{Cached sub-files in } \mathcal{Z}_2}.$$

The decoding procedure is analogous to the interference suppression technique. W_2 decodes C_{14} by canceling the interfering sub-files using its cache content $\mathcal{Z}_2$.

(2) Decoding C_{16} from X_{123} by W_2:

$$X_{123} = \underbrace{C_{16}}_{\text{Desired sub-file}} \oplus \underbrace{C_{14}}_{\substack{\text{Decoded sub-file} \\ \text{from } X_{124}}} \oplus \underbrace{C_{15}}_{\substack{\text{Decoded sub-file} \\ \text{from } X_{125}}} \oplus \underbrace{A_{24} \oplus A_{25} \oplus A_{26} \oplus B_{34} \oplus B_{35} \oplus B_{36}}_{\text{Cached sub-files in } \mathcal{Z}_2}.$$

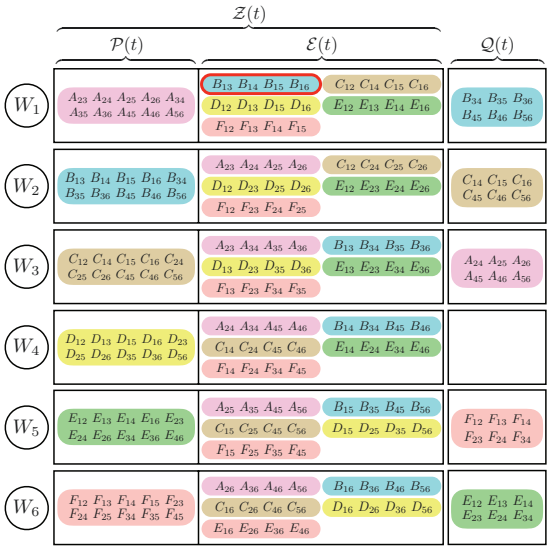

Fig. 4. Cache organization of worker nodes at iteration t, along with the set of sub-files to be processed at iteration $t+1$, which are not available in the caches at iteration t.

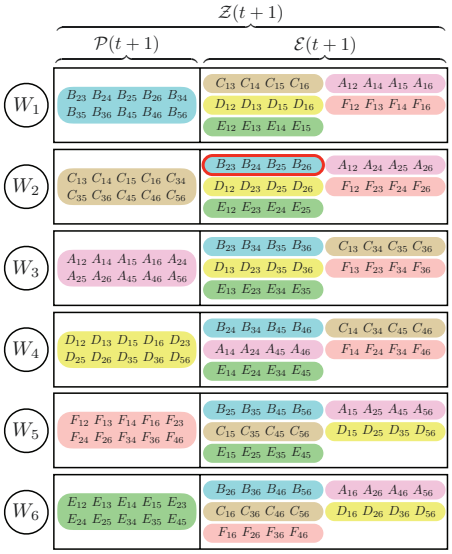

Fig. 5. Cache organization of worker nodes at iteration $t+1$ after updating the caches and relabeling the sub-files of Fig. 4. For instance, subfiles $\{B_{13}, B_{14}, B_{15}, B_{16}\}$ in $\mathcal{E}_1(t)$ are moved $\mathcal{E}_2(t+1)$ and relabeled to $\{B_{23}, B_{24}, B_{25}, B_{26}\}$, respectively.

The decoding procedure is analogous to successive interference cancellation (SIC) technique. W_2 decodes C_{16} by first canceling the sub-files that exist in $\mathcal{Z}_2$. Next, it exploits the sub-files decoded from X_{124} and X_{125} to successively cancel the interfering sub-files.

(3) Decoding E_{23} from X_{234} by W_6:

$$X_{235} = \underbrace{E_{23}}_{\text{Desired sub-file}} \oplus \underbrace{(C_{56} \oplus F_{23})}_{\substack{\text{Cached sub-files} \\ \text{in } \mathcal{Z}_6}} \oplus \underbrace{(A_{25} \oplus B_{35} \oplus C_{15} \oplus C_{45})}_{X_{123} \oplus X_{234}}.$$

The decoding procedure is analogous to aligned interference suppression technique. W_6 decodes E_{23} by first canceling the sub-files cached in $\mathcal{Z}_6$. Then, by the proposed data placement phase, the remaining interfering sub-files are the result of

	W_1	W_2	W_3	W_4	W_5	W_6
X_{123}	$B_{34} \oplus B_{35} \oplus B_{36}$	$C_{14} \oplus C_{15} \oplus C_{16}$	$A_{24} \oplus A_{25} \oplus A_{26}$	–	$A_{24} \oplus A_{26} \oplus B_{34} \oplus B_{36} \oplus C_{14} \oplus C_{16}$	$A_{24} \oplus A_{25} \oplus B_{34} \oplus B_{35} \oplus C_{14} \oplus C_{15}$
X_{124}	$B_{34} \oplus B_{45} \oplus B_{46}$	C_{14}	$A_{24} \oplus B_{45} \oplus B_{46}$	–	$A_{24} \oplus B_{34} \oplus B_{46} \oplus C_{14}$	$A_{24} \oplus B_{34} \oplus B_{45} \oplus C_{14}$
X_{125}	$B_{35} \oplus B_{45} \oplus B_{56}$	C_{15}	$A_{25} \oplus B_{45} \oplus B_{56} \oplus E_{12} \oplus F_{12}$	–	F_{12}	$A_{25} \oplus B_{35} \oplus B_{45} \oplus C_{15} \oplus E_{12}$
X_{134}	B_{34}	$A_{45} \oplus A_{46} \oplus C_{14}$	$A_{24} \oplus A_{45} \oplus A_{46}$	–	$A_{24} \oplus A_{46} \oplus B_{34} \oplus C_{14}$	$A_{24} \oplus A_{45} \oplus B_{34} \oplus C_{14}$
X_{135}	B_{35}	$A_{45} \oplus A_{56} \oplus C_{15} \oplus E_{13} \oplus F_{13}$	$A_{25} \oplus A_{45} \oplus A_{56}$	–	F_{13}	$A_{25} \oplus A_{45} \oplus B_{35} \oplus C_{15} \oplus E_{13}$
X_{145}	B_{45}	$A_{45} \oplus E_{14} \oplus F_{14}$	$A_{45} \oplus B_{45} \oplus E_{14} \oplus F_{14}$	–	F_{14}	$A_{45} \oplus B_{45} \oplus E_{14}$
X_{234}	$B_{34} \oplus C_{45} \oplus C_{46}$	$C_{14} \oplus C_{45} \oplus C_{46}$	A_{24}	–	$A_{24} \oplus B_{34} \oplus C_{14} \oplus C_{46}$	$A_{24} \oplus B_{34} \oplus C_{14} \oplus C_{45}$
X_{235}	$B_{35} \oplus C_{45} \oplus C_{56} \oplus E_{23} \oplus F_{23}$	$C_{15} \oplus C_{45} \oplus C_{56}$	A_{25}	–	F_{23}	$A_{25} \oplus B_{35} \oplus C_{15} \oplus C_{45} \oplus E_{23}$
X_{245}	$B_{45} \oplus C_{45} \oplus E_{24} \oplus F_{24}$	C_{45}	$B_{45} \oplus E_{24} \oplus F_{24}$	–	F_{24}	$B_{45} \oplus C_{45} \oplus E_{24}$
X_{345}	$C_{45} \oplus E_{34} \oplus F_{34}$	$A_{45} \oplus C_{45} \oplus E_{34} \oplus F_{34}$	A_{45}	–	F_{34}	$A_{45} \oplus C_{45} \oplus E_{34}$

Fig. 6. Received functions by worker nodes after removing the cached sub-files. The complete received functions at worker nodes are expressed in (8).

XORing some other received sub-messages, i.e., $X_{123} \oplus X_{234}$, and hence, they can be canceled accordingly.

As a result, the rate is $R_{\text{coded}} = \binom{5}{3}/\binom{5}{2} = 1$. On the other hand, the rate achieved by the uncoded shuffling scheme, under the same placement strategy, is $R_{\text{uncoded}} = (5 \times 6)/\binom{5}{2} = 3$. In other words, the proposed coded shuffling scheme can save around 66% of the communication rate, and thus, it speeds up the overall run-time of the data shuffling process. When each worker node decodes all missing subfiles at iteration t, Fig. 5 depicts the cache organization of each worker node after updating the cache and relabeling the subfiles in preparation for the following random shuffle at iteration $t+1$.

B. Proposed Achievable Scheme

When $S = 1$, each worker node only stores the file that it processes with no extra storage. The following set of encoded functions $\mathcal{X} = \{\mathcal{Z}_i \oplus \mathcal{Z}_{i+1} : i \in [K-1]\}$ ensures that each worker node accesses all the files and stores what it needs to process at the next iteration. Thus, the achieved rate under this scheme is $R = K - 1$. On the other hand, when $S = N$, each worker node can store all the files in its cache. Therefore, there is no communication needed between the master node and the worker nodes, and hence, $R = 0$. The challenging regime of S is when $1 < S < N$ where $N = K$. In what follows, we propose an achievable scheme to show the rate is $R = \binom{K-1}{S}/\binom{K-1}{S-1}$ in this regime.

Encoding: Given all cache contents $\{\mathcal{Z}_i : i \in [K]\}$ and $\{d(i) : i \in [K]\}$, the broadcast message $\mathcal{X}$ is obtained at the master node by the concatenation of several sub-messages X_Δ, each specified for a group of worker nodes Δ, that is,

$$\mathcal{X} = \{X_\Delta : \Delta \subseteq [K-1], |\Delta| = S\}, \text{ where} \tag{9}$$

$$X_\Delta \triangleq \bigoplus_{i \in \Delta} \left(F^i_{\Delta \setminus \{i\}} \oplus F^{d(i)}_{\Delta \setminus \{d(i)\}} \oplus \bigoplus_{j \in [K] \setminus \Delta} F^{d(i)}_{(\{j\} \cup \Delta) \setminus \{i, d(i)\}} \right). \tag{10}$$

The encoding design hinges on $(K-1)$ worker nodes. Without loss of generality, we consider $W_1, W_2, \ldots, W_{K-1}$ for whom the broadcast sub-messages are designed, and designate W_K as the ignored worker node. Accordingly, there is a total of $\binom{K-1}{S}$ sub-messages, each of size of the unit partition $1/\binom{K-1}{S-1}$ that is transmitted by the master node to the worker nodes. Hence, the overall broadcast communication rate is given by (5).

Decoding: Each worker node utilizes the cache content, along with the functions received from the master node, in order to decode the missing sub-files that constitute the file to be processed at iteration $t+1$. We demonstrate a decoding scheme that allows each worker node to decode its intended sub-files. More specifically, we divide the set of worker nodes into two different groups, namely, W_ℓ with $\ell \in [K-1]$ and a single element group W_K. The following lemmas demonstrate how the intended sub-files can be decoded from the broadcast sub-messages and the cache contents of worker nodes. Detailed proofs of the lemmas are reported in [9].

Lemma 1. *If $K \notin \Gamma$, then a sub-file $F^{d(\ell)}_\Gamma \in \mathcal{Q}_\ell$ intended for W_ℓ, for $\ell \in [K-1]$, can be decoded from the cache content $\mathcal{Z}_\ell$ and the broadcast sub-message $X_{\{\ell\} \cup \Gamma}$. On the other hand, if $K \in \Gamma$, then $F^{d(\ell)}_\Gamma$ can be decoded from the cache content $\mathcal{Z}_\ell$ and the broadcast sub-message $X_{(\Gamma \setminus \{K\}) \cup \{\ell, d(\ell)\}}$, and other sub-files previously decoded by W_ℓ.*

Lemma 2. *Any sub-file $F^{d(K)}_\Gamma \in \mathcal{Q}_K$ intended for W_K can be decoded from the cache content $\mathcal{Z}_K$ and the broadcast sub-message $\bigoplus_{\ell \in [K-1] \setminus \Gamma} X_{\{\ell\} \cup \Gamma}$.*

Cache Updating and Subfile Relabeling: After the worker nodes decode the missing subfiles characterized by (4), the caches of worker nodes need to be updated and the subfiles needs to be relabeled before processing the files at iteration $t+1$. The precise procedure can be found in the extended version of this paper [9]. Finally, the shuffling procedure at iteration t terminates, and the master node commences another random data shuffling at iteration $t+1$.

V. Optimal Coded Shuffling Scheme for any Random Shuffling ($N = K$)

The scheme proposed in Section IV provides the worker nodes with their missing sub-files using the broadcast message and cached sub-files. However, depending on the demand profile, the rate obtained by that scheme may be sub-optimum. In the extended version of this paper [9], and based on the scheme proposed in Section IV, we propose an opportunistic coded shuffling scheme that yields a lower rate. More concretely, when $\gamma - 1 \geq S$, there are precisely $\binom{\gamma-1}{S}$ sub-messages in (9) that are linearly dependent on the other sub-messages. Thus, by refraining from broadcasting these sub-messages, the reduced rate is given by (6).

VI. Converse Proof for Data Shuffling ($N = K$)

We prove the optimality of the proposed coded shuffling scheme in Section V. Let us assign, for a given demand profile, a unique index pair $(c(i), p(i))$ to W_i, where $c(i)$ is the cycle number that W_i belongs to, and $p(i)$ is the position of W_i within cycle $c(i)$. More formally, there is a function w that maps the worker node index i and the unique index pair $(c(i), p(i))$ as follows:

$$i = w((c(i), p(i))), \text{ for } i \in [K],\ c(i) \in [\gamma],\ p(i) \in [\ell_c].$$

Note that there is a one-to-one mapping between the worker node index and the introduced index pair, and hence, we can interchangeably use them without confusion. Let us denote by $\tilde{F}_j^i$ the part of file F^i cached at W_j. Furthermore, for a subset of worker nodes $\mathcal{J} \subseteq [K] \setminus \{i\}$, we define the cardinality of a union of sub-files of F^i that are cached at $\mathcal{J}$ as $\mu_{\mathcal{J}}^i = \left| \bigcup_{j \in \mathcal{J}} \tilde{F}_j^i \right|$. For a given cardinality η of a set of worker nodes, let μ_η denote the average cardinality of a set of sub-files that are cached by η worker nodes, that is,

$$\mu_\eta = \frac{1}{K\binom{K-1}{\eta}} \sum_{i \in [K]} \sum_{\substack{\mathcal{J} \subseteq [K] \setminus i, \\ |\mathcal{J}| = \eta}} \mu_{\mathcal{J}}^i. \tag{11}$$

Accordingly, the following properties hold: $\mu_0 = 0$, $\mu_1 = \frac{S-1}{K-1}$, $\mu_{K-1} = \min\{1, S-1\}$. Let us assume a virtual worker node $W_\star$ equipped with a cache $\mathcal{Z}_\star$ that is characterized as

$$\mathcal{Z}_\star = \mathcal{Z}_\star^{(1)} \cup \mathcal{Z}_\star^{(2)}, \text{ where } \mathcal{Z}_\star^{(1)} = \left(\bigcup_{\substack{i: c(i) \in [\gamma], \\ p(i) = 1}} F^{u(i)} \right),$$

$$\mathcal{Z}_\star^{(2)} = \left(\bigcup_{\substack{j: c(j) \in [\gamma], \\ p(j) > 1}} \left(\left(\bigcup_{\substack{r: c(r) < c(j), \\ p(r) < \ell_{c(r)}}} \tilde{F}_{u(r)}^{u(j)} \right) \cup \left(\bigcup_{\substack{r: c(r) = c(j), \\ p(r) < p(j)}} \tilde{F}_{u(r)}^{u(j)} \right) \right) \right).$$

Lemma 3. *Given the broadcast message $\mathcal{X}$, and the cache content $\mathcal{Z}_\star$ of the virtual worker node $W_\star$, then*

$$H(\{F^i\}_{i=1}^K | X, \mathcal{Z}_\star) = 0. \tag{12}$$

Therefore, $W_\star$ can decode all the files of the shuffling system.

Next, the communication rate is lower bounded as follows.

Lemma 4. *The communication rate R, over all files and all sets of worker nodes with given cardinalities, is lower bounded by*

$$R \geq N - \gamma - \sum_{i=1}^{K-\gamma} \mu_i. \tag{13}$$

Finally, we seek the tightest lower bound on R as follows.

Lemma 5. *The optimal $\{\mu_j^\star\}_{j=0}^{K-1}$, that maximizes $\sum_{i=1}^{K-\gamma} \mu_i$ over its feasible region, is characterized by*

$$\mu_j^\star = 1 - \frac{\binom{K-j-1}{S-1}}{\binom{K-1}{S-1}}. \tag{14}$$

Substituting (14) into (13) yields the tightest lower bound on the communication rate R given by (7). Detailed proofs of the lemmas are reported in [9].

VII. The General Setting ($N \geq K$)

We investigate the general setting of data shuffling problem when $N \geq K$, using the results obtained in Section III when $N = K$. The following theorems summarize our main results.

Theorem 4. *For a distributed computing system that processes N files, and consists of K worker nodes each with a normalized storage capacity of $\widehat{S} = S/(N/K) \in \{1, 2, \ldots, K\}$ files per worker node, the achievable communication rate R required to shuffle N files for any demand profile is upper bounded by*

$$R \leq \frac{N}{K} \frac{\binom{K-1}{\widehat{S}}}{\binom{K-1}{\widehat{S}-1}}. \tag{15}$$

Theorem 5. *For the distributed computing system introduced in Theorem 4, the worst-case communication rate $R_{\text{worst-case}}$ required to shuffle N files for any demand profile is lower bounded by*

$$R_{\text{worst-case}} \geq \frac{N}{K} \frac{\binom{K-1}{\widehat{S}}}{\binom{K-1}{\widehat{S}-1}}. \tag{16}$$

Theorems 4 and 5 prove the optimality of the proposed shuffling scheme in the worst-case shuffle. The main idea is to decompose the demand profile into N/K sub-graphs on the same set of nodes, such that the in-degree and out-degree of each node in a sub-graph are 1. Each resulting sub-graph reduces to a data shuffling system with K files (corresponding to the edges appear in the sub-graph), K worker nodes, and storage capacity per worker node $S/(N/K)$. Hence, we can apply the optimal coded shuffling scheme proposed in Section V to achieve the minimum communication rate within each sub-graph. This decomposition is shown to be optimum for the worst-case shuffle when $N \geq K$, and hence, it yields an exact rate-memory trade-off characterization, reported in [9]. The characterization of the optimum trade-off for a given profile demand is still an open problem.

References

[1] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Speeding up distributed machine learning using codes," *IEEE Transactions on Information Theory*, 2017.

[2] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *IEEE Transactions on Information Theory*, vol. 60, no. 5, pp. 2856–2867, 2014.

[3] M. A. Attia and R. Tandon, "Information theoretic limits of data shuffling for distributed learning," *IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, 2016.

[4] ——, "On the worst-case communication overhead for distributed data shuffling," *54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 961–968, 2016.

[5] L. Song, C. Fragouli, and T. Zhao, "A pliable index coding approach to data shuffling," *IEEE International Symposium on Information Theory (ISIT)*, pp. 2558–2562, 2017.

[6] J. Chung, K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Ubershuffle: Communication-efficient data shuffling for SGD via coding theory," *Advances in NIPS*, 2017.

[7] S. Li, M. A. Maddah-Ali, Q. Yu, and A. S. Avestimehr, "A fundamental tradeoff between computation and communication in distributed computing," *IEEE Transactions on Information Theory*, vol. 64, no. 1, pp. 109–128, 2018.

[8] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "The exact rate-memory tradeoff for caching with uncoded prefetching," *IEEE Transactions on Information Theory*, vol. 64, no. 2, pp. 1281–1296, 2018.

[9] A. Elmahdy and S. Mohajer, "On the fundamental limits of coded data shuffling," *http://people.ece.umn.edu/~soheil/pubs/Shuffling.pdf*.