Approximately Optimal Distributed Data Shuffling

Mohamed Adel Attia Ravi Tandon
Department of Electrical and Computer Engineering
University of Arizona, Tucson, AZ
Email: {*madel, tandonr*}@email.arizona.edu

***Abstract*—Data shuffling between distributed workers is one of the critical steps in implementing large-scale learning algorithms. The focus of this work is to understand the fundamental trade-off between the amount of storage and the communication overhead for distributed data shuffling. We first present an information theoretic formulation for the data shuffling problem, accounting for the underlying problem parameters (i.e., number of workers, K, number of data points, N, and the available storage, S per node). Then, we derive an information theoretic lower bound on the communication overhead for data shuffling as a function of these parameters. Next, we present a novel coded communication scheme and show that the resulting communication overhead of the proposed scheme is within a multiplicative factor of at most 2 from the lower bound. Furthermore, we introduce an improved *aligned coded shuffling scheme*, which achieves the optimal storage vs communication trade-off for $K < 5$, and further reduces the maximum multiplicative gap down to $7/6$, for $K \geq 5$.**

I. Introduction

Distributed computing comes at the unavoidable communication cost due to data shuffling among distributed workers. Data shuffling can arise in many applications such as: a) random shuffling of the data-set across different points before each learning epoch so that each worker is assigned new training data, which provides statistical benefits, e.g., distributed gradient descent algorithm and its stochastic variations [1]; b) shuffling the data-set across attributes to assign different features to each worker, e.g., in mobile cloud gaming systems; and c) shuffling the data between the mappers and the reducers in the MapReduce framework. The application of coding theory to the data shuffling problem was first considered in [2], using excess storage at the workers in a probabilistic fashion, to create coded multicasting opportunities. In [3], the authors presented coded data shuffling in the MapReduce setting, with more redundant computations at the mappers, leading to a communication vs computation trade-off.

In this paper, we focus on the distributed data shuffling problem in a master-worker setting. At each learning epoch, the data set is randomly shuffled at the master node, and different data chunks need to be sent to the workers for distributed computation, which leads to the communication overhead. On one extreme, when all the workers can store the whole data set, no communication is needed for any random shuffle. On the other hand, when the storage is just enough to store the assigned data, also referred to as the *no excess storage case*, the communication is expected to be maximal. The goal of this work is to characterize the fundamental information-theoretic trade-off between the communication overhead and the available storage at the distributed workers.

This work was supported by the NSF grant CAREER-1651492.

In our prior work [4], we characterized the optimal tradeoff for $K = 2$ and $K = 3$ workers. In [5], it was shown that even for the no excess storage case, coding opportunities can still be leveraged. In [6], a pliable index coding approach for data shuffling was considered for semi-random shuffles. In this work however, we consider the tradeoff between worst-case rate over all possible shuffles and storage. The data shuffling problem is also related to the index coding problem [7]. The key difference is that the *side information* in the data shuffling problem (i.e., data stored at distributed workers) is generally not static, and the side information itself can change over time.

Summary of contributions: We first derive an information theoretic lower bound on the worst-case communication overhead for the data shuffling problem, based on a novel bounding methodology, similar in spirit to the recent converse proofs in the coded caching literature [8], [9]. Next, we introduce our achievable scheme based on a *"structurally invariant placement and update"* procedure that maintains the structure of the storage and allows the use of coded data delivery similar to [10]. We show that the rate of this scheme is within a multiplicative gap of 2 for all problem parameters. In order to close this gap, we then introduce a novel *aligned coded shuffling* scheme combining the ideas of coding and interference alignment. This scheme matches with the lower bound for all $K < 5$, and reduces the multiplicative gap to $7/6$, for all $K \geq 5$.

II. System Model

We assume a master node which has a data-set $\mathcal{A}$, containing N data points, $D_1, \ldots, D_N$, of size d bits each, and K distributed workers connected to the master node via a shared link. Treating the data points D_n as i.i.d. random variables, we therefore have $H(\mathcal{A}) = N \times H(D_n) = Nd$. At each iteration, indexed by t, the master node divides the data-set $\mathcal{A}$ into K equal sized data batches (assuming N is divisible by K), which correspond to a random permutation of the data-set, $\pi^t : \mathcal{A} \rightarrow \{\mathcal{A}^t(1), \ldots, \mathcal{A}^t(K)\}$, where $\mathcal{A}^t(k)$ denotes the data partition designated to be processed by worker w_k at time t. Since the data batches are disjoint and span the whole data-set, i.e., $\cup_{k\in[1:K]}\mathcal{A}^t(k) = \mathcal{A}$, and $\mathcal{A}^t(i) \cap \mathcal{A}^t(j) = \phi$, for all $i \neq j$, we have $H(\mathcal{A}^t(k)) = \frac{N}{K}d$, for all $k \in [1 : K]$.

After getting the data batch, each worker locally computes a function (such as the gradient or sub-gradients of the assigned

978-1-5386-4780-6/18/$31.00©2018 IEEE

data points) to be processed subsequently at the master node. We assume that each worker w_k has a storage Z_k^t of size Sd bits, which is used to store some function of the data-set, and S denotes the storage parameter. Considering Z_k^t as a random variable, we then have

$$H(Z_k^t) = Sd, \quad H(Z_k^t|\mathcal{A}) = 0, \quad \forall k \in [1:K]. \tag{1}$$

For processing purposes, each worker w_k must at least store the assigned data batch $\mathcal{A}^t(k)$ (of size $\frac{N}{K}d$ bits) at time t in Z_k^t, which gives the range of storage parameter as $N/K \le S \le N$. Also, we get the *processing constraint* as

$$H(\mathcal{A}^t(k)|Z_k^t) = 0, \quad \forall k \in [1:K]. \tag{2}$$

In the next epoch $t+1$, the data-set is randomly reshuffled at the master node according to a random permutation π^{t+1} : $\mathcal{A} \to \{\mathcal{A}^{t+1}(1), \ldots, \mathcal{A}^{t+1}(K)\}$. The two phases of the overall process, namely *Data Delivery* of the new data batches and *Storage Update*, are described next.

A. Data Delivery Phase

At time $t+1$, the master node sends a function of the data batches for the subsequent shuffles (π_t, π_{t+1}), $X_{(\pi_t,\pi_{t+1})} = \phi(\mathcal{A}^t(1), \ldots, \mathcal{A}^t(K), \mathcal{A}^{t+1}(1), \ldots, \mathcal{A}^{t+1}(K))$ over the shared link, where $\phi(\cdot)$ is the data delivery encoding function. We also define $R_{(\pi_t,\pi_{t+1})}$ as the normalized rate of the shared link based on the shuffles (π_t, π_{t+1}). We then have

$$H\left(X_{(\pi_t,\pi_{t+1})}|\mathcal{A}\right) = 0, \quad H\left(X_{(\pi_t,\pi_{t+1})}\right) = R_{(\pi_t,\pi_{t+1})}d. \tag{3}$$

Each worker w_k should reliably decode the desired batch $\mathcal{A}^{t+1}(k)$ out of the transmitted function $X_{(\pi_t,\pi_{t+1})}$, as well as the data stored in the previous time slot Z_k^t, i.e., $\mathcal{A}^{t+1}(k) = \psi(X_{(\pi_t,\pi_{t+1})}, Z_k^t)$, where $\psi(\cdot)$ is the decoding function at the workers. Therefore, for reliable decoding, we have the following *decodability constraint* at each worker:

$$H\left(\mathcal{A}^{t+1}(k)|Z_k^t, X_{(\pi_t,\pi_{t+1})}\right) = 0, \quad \forall k \in [1:K]. \tag{4}$$

B. Storage Update Phase

At the next iteration $t+1$, the storage for each worker w_k is updated to Z_k^{t+1}, which is a function of the old storage content Z_k^t as well as transmitted function $X_{(\pi_t,\pi_{t+1})}$, i.e., $Z_k^{t+1} = \mu(X_{(\pi_t,\pi_{t+1})}, Z_k^t)$, where μ is the update function. Therefore, we have the following *storage-update constraint*:

$$H(Z_k^{t+1}|Z_k^t, X_{(\pi_t,\pi_{t+1})}) = 0, \quad \forall k \in [1:K]. \tag{5}$$

The excess storage after storing $\mathcal{A}^{t+1}(k)$ in Z_k^{t+1}, given by $(S - \frac{N}{K})$, can be used to opportunistically store a function of the remaining $K-1$ data batches. For the scope of this work, we focus on *uncoded storage schemes*, meaning that the excess storage is dedicated to store uncoded functions of the remaining $K-1$ batches. We give the notation $\mathcal{A}^{t+1}(i,k)$, where $i \neq k$, as that part of $\mathcal{A}^{t+1}(i)$ which worker w_k stores in it's excess storage at time $t+1$. As a result, we can write the content of Z_k^{t+1} for uncoded storage placement as

$$Z_k^{t+1} = \left\{\mathcal{A}^{t+1}(k), \cup_{i\in[1:K]\setminus k}\mathcal{A}^{t+1}(i,k)\right\}. \tag{6}$$

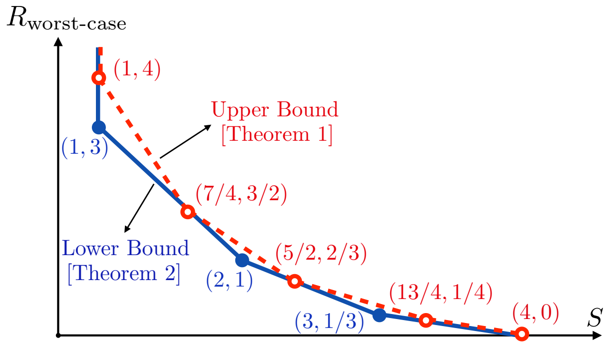

Fig. 1. Lower and upper bounds on $R^*_{\text{worst-case}}$ for $N = K = 4$.

Our goal in this work is to characterize the optimal worst-case communication $R^*_{\text{worst-case}}(S) \triangleq \min_{(\phi,\psi,\mu)} \max_{(\pi_t,\pi_{t+1})} R^{(\phi,\psi,\mu)}_{(\pi_t,\pi_{t+1})}(S)$, where $R^{(\phi,\psi,\mu)}_{(\pi_t,\pi_{t+1})}(S)$ is the rate of a scheme (ϕ, ψ, μ), based on the shuffle (π_t, π_{t+1}).

III. Main Results and Discussions

The first Theorem presents an achievable scheme, which yields an upper bound on $R^*_{\text{worst-case}}(S)$.

Theorem 1: For the data shuffling problem, the lower convex envelope of the following $K+1$ storage-rate pairs is achievable: for $i \in [0:K]$,

$$\left(S = \left(1 + i\frac{K-1}{K}\right)\frac{N}{K}, \; R^{\text{upper}}_{\text{worst-case}} = \frac{N(K-i)}{K(i+1)}\right). \tag{7}$$

The complete proof of Theorem 1 is presented in [11, Appendix A]. One of the crucial steps in the achievability proof is the *structural invariant placement and update* of the storage at the workers over time. This allows us to leverage coded data delivery, which achieves the above rate-storage pairs. We present an illustrative example for $K = N = 4$ to introduce the main elements of the achievability proof.

Example 1: Consider the case of $K = N = 4$. We consider the cyclic shuffle $\pi_t = (1,2,3,4)$, and $\pi_{t+1} = (2,3,4,1)$. A similar data delivery mechanism can be devised for any arbitrary shuffle (see [11]). According to Theorem 1, the achievable worst-case storage-rate trade-off is given by the lower convex envelope of the 5 storage-rate pairs shown by the red dashed curve in Figure 1. Using memory sharing (see [11, Claim 1] for more details), once we achieve these pairs, the lower convex envelope is also achievable. Due to space limitation, we only consider here the achievability for the rate pair $(S = 7/4,\ R = 3/2)$, while the rest of the storage points can be found in [11, Example 1].

Storage Placement: The storage placement for $S = 7/4$ is shown in Figure 2a. First, each data point D_i is partitioned into 4 sub-points. For instance, D_1 is partitioned as $\{D_{1,\{1\}}, D_{1,\{2\}}, D_{1,\{3\}}, D_{1,\{4\}}\}$, each of size $d/4$. At time t, every worker w_k fully stores the assigned data point, D_k, as well as the sub-points labeled with k. For instance, w_1 stores D_1 and the sub-points $\{D_{2,\{1\}}, D_{3,\{1\}}, D_{4,\{1\}}\}$.

Data Delivery: Since the shuffle at time $t+1$ is $\pi_{t+1} = (2,3,4,1)$, every worker needs 3 sub-points of the assigned data point, e.g., w_1 needs the sub-points

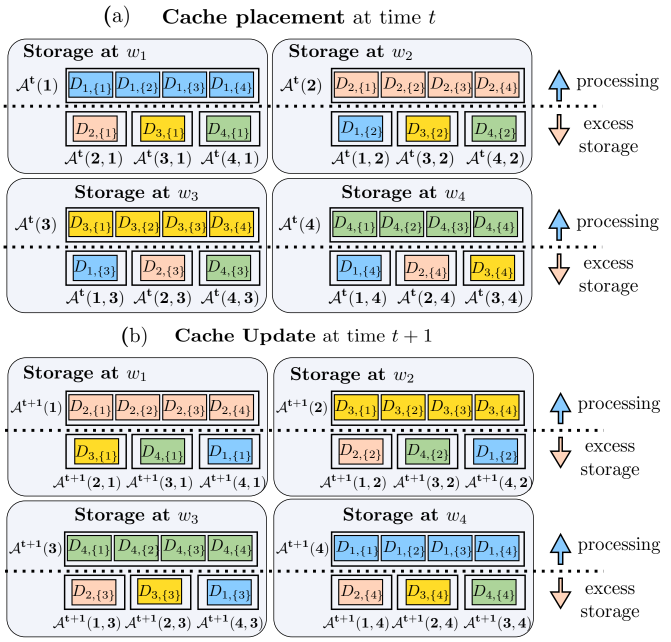

Fig. 2. Structural invariant storage placement, (a), and update, (b), for $K = N = 4$, and $S = 7/4$. Above the dotted line are assigned data points, and below is the excess storage used to store the sub-points labeled with the worker's index.

$\{D_{2,\{2\}}, D_{2,\{3\}}, D_{2,\{4\}}\}$, and every sub-point is available at least in one of the remaining workers. Therefore, the master node sends six coded symbols, each being useful for two workers at the same time as follows: $D_{2,\{2\}} \oplus D_{3,\{1\}}$, for (w_1, w_2); $D_{2,\{3\}} \oplus D_{4,\{1\}}$, for (w_1, w_3); $D_{2,\{4\}} \oplus D_{1,\{1\}}$, for (w_1, w_4); $D_{3,\{3\}} \oplus D_{4,\{2\}}$, for (w_2, w_3); $D_{3,\{4\}} \oplus D_{1,\{2\}}$, for (w_2, w_4); and $D_{4,\{4\}} \oplus D_{1,\{3\}}$, for (w_3, w_4). The rate of this transmission is $6 \times d/4 = 3d/2$ bits, and the pair $(S = 7/4, R = 3/2)$ is achieved.

Storage Update: The storage update at time $t + 1$ follows the storage placement at time t, and is shown Figure 2b. For example, w_1 stores D_2 completely, and keeps from D_1 only the sub-point $D_{1,\{1\}}$.

Our second main result in Theorem 2 gives an information theoretic lower bound on $R^*_{\text{worst-case}}$.

Theorem 2: For the data shuffling problem, a lower bound on $R^*_{\text{worst-case}}$ is given by the lower convex envelope of the following K storage-rate pairs: for $m \in [1 : K]$,

$$\left(S = m\frac{N}{K},\, R^{\text{lower}}_{\text{worst-case}} = \frac{N(K-m)}{Km}\right). \tag{8}$$

The complete proof of Theorem 2 is in [11, Appendix B]. We present the key ideas behind the converse proof in Example 2 which is presented after stating Theorem 3.

*Remark 1 (**Basic idea for the converse**):* A lower bound over the optimal rate $R^*_{(\pi_t,\pi_{t+1})}$ of a shuffle (π_t, π_{t+1}) also lower bounds the worst-case since $R^*_{\text{worst-case}} \geq R^*_{(\pi_t,\pi_{t+1})}$. Therefore, we get lower bounds over $R^*_{\text{worst-case}}$ by averaging out a set of lower bounds for a sequence of shuffles. The novel part in our proof is to carefully choose the right shuffles which lead to the highest lower bound. In addition, we leverage a novel bounding methodology similar to [8], [9], where the optimal uncoded cache placement system is considered. The difference in the data shuffling problem, is that the workers in addition to storing the data under processing, have excess storage and can also update their storage over time.

In our next result, we compare the upper and lower bounds in Theorems 1 and 2, respectively.

Theorem 3: For the data shuffling problem, the multiplicative gap between upper and lower bounds on $R^*_{\text{worst-case}}$ given by Theorems 1, and 2, respectively, is bounded as follows:

$$\frac{R^{\text{upper}}_{\text{worst-case}}}{R^{\text{lower}}_{\text{worst-case}}} \leq \frac{K}{K-1} \leq 2. \tag{9}$$

The complete proof of Theorem 3 can be found in [11, Appendix C]. This Theorem shows that the gap between the bounds vanishes as K increases, i.e., $\lim_{K\to\infty} \left(\frac{K}{K-1}\right) = 1$.

Next, we discuss the example of $N = K = 4$ to present the ideas behind the converse proof of Theorem 2.

Example 2: Assume the $N = 4$ data points are assigned at time t according to $\pi_t = (1, 2, 3, 4)$. From (2), each worker should store the assigned data point at time t, therefore,

$$H(\mathcal{A}^t(k)|Z_k^t) = H(D_k|Z_k^t) = 0, \quad \forall k \in [1 : 4]. \tag{10}$$

Following Remark 1, we consider the following cyclic shuffle: for a permutation $\sigma : (1, 2, 3, 4) \to (\sigma_1, \sigma_2, \sigma_3, \sigma_4)$, we have $\mathcal{A}^{t+1}(\sigma_k) = \mathcal{A}^t(\sigma_{k-1}) = D_{\sigma_{k-1}}$. Using the decodability constraint in (4), worker w_{σ_k} must decode $\mathcal{A}^{t+1}(\sigma_k) = D_{\sigma_{k-1}}$ from its cache content $Z^t_{\sigma_k}$ as well as the transmission $X_{(\pi_t,\pi_{t+1})}$ which gives the following condition:

$$H(D_{\sigma_{k-1}}|Z^t_{\sigma_k}, X_{(\pi_t,\pi_{t+1})}) = 0, \quad \forall k \in [1 : 4]. \tag{11}$$

Consequently, we get the following bound using (10) and (11):

$$\begin{aligned} &H(\mathcal{A}|Z^t_{\sigma_2}, Z^t_{\sigma_3}, Z^t_{\sigma_4}, X_{(\pi_t,\pi_{t+1})}) \\ &\overset{(a)}{\leq} H(D_{\sigma_1}|Z^t_{\sigma_2}, X_{(\pi_t,\pi_{t+1})}) + \sum_{k=2}^{4} H(D_{\sigma_k}|Z^t_{\sigma_k}) = 0, \end{aligned} \tag{12}$$

where (a) follows since $\mathcal{A} = \{D_1, D_2, D_3, D_4\}$, and from the fact that $H(A, B) \leq H(A) + H(B)$ and that conditioning reduces entropy. Next, we obtain the following bound:

$$\begin{aligned} 4d = H(\mathcal{A}) &\overset{(a)}{=} I(\mathcal{A}; Z^t_{\sigma_2}, Z^t_{\sigma_3}, Z^t_{\sigma_4}, X_{(\pi_t,\pi_{t+1})}) \\ &\overset{(b)}{\leq} H(Z^t_{\sigma_2}, Z^t_{\sigma_3}, Z^t_{\sigma_4}, X_{(\pi_t,\pi_{t+1})}) \\ &\leq H(X_{(\pi_t,\pi_{t+1})}) + H(Z^t_{\sigma_4}) + H(Z^t_{\sigma_3}|Z^t_{\sigma_4}, X_{(\pi_t,\pi_{t+1})}) \\ &\quad + H(Z^t_{\sigma_2}|Z^t_{\sigma_3}, Z^t_{\sigma_4}, X_{(\pi_t,\pi_{t+1})}) \\ &\overset{(c)}{\leq} R^*_{\text{worst-case}} d + H(Z^t_{\sigma_4}) + H(Z^t_{\sigma_3}|Z^t_{\sigma_4}, D_{\sigma_3}, D_{\sigma_4}) \\ &\quad + H(Z^t_{\sigma_2}|Z^t_{\sigma_3}, Z^t_{\sigma_4}, D_{\sigma_2}, D_{\sigma_3}, D_{\sigma_4}) \\ &\overset{(d)}{=} R^*_{\text{worst-case}} d + H(D_{\sigma_4}, D_{\sigma_1}(\sigma_4), D_{\sigma_2}(\sigma_4), D_{\sigma_3}(\sigma_4)) + \\ &\quad + H(D_{\sigma_1}(\sigma_3), D_{\sigma_2}(\sigma_3)|Z^t_{\sigma_4}) + H(D_{\sigma_1}(\sigma_2)|Z^t_{\sigma_3}, Z^t_{\sigma_4}) \\ &\overset{(e)}{=} R^*_{\text{worst-case}} d + H(D_{\sigma_4}) + H(D_{\sigma_3}(\sigma_4)) + [H(D_{\sigma_2}(\sigma_4)) \\ &\quad + H(D_{\sigma_2}(\sigma_3)|D_{\sigma_2}(\sigma_4))] + [H(D_{\sigma_1}(\sigma_4)) \\ &+ H(D_{\sigma_1}(\sigma_3)|D_{\sigma_1}(\sigma_4)) + H(D_{\sigma_1}(\sigma_2)|D_{\sigma_1}(\sigma_3), D_{\sigma_1}(\sigma_4))] \end{aligned}$$

$$\overset{(f)}{\leq} R^*_{\text{worst-case}} d + d + H(D_{\sigma_3}(\sigma_4)) + H(D_{\sigma_2}(\sigma_3, \sigma_4)) + H(D_{\sigma_1}(\sigma_2, \sigma_3, \sigma_4)), \quad (13)$$

where (a) follows from (12), (b) follows from (1), and (3), where $X_{(\pi_t, \pi_{t+1})}$ and Z_k^t for $k \in [1:4]$ are deterministic functions of the data-set $\mathcal{A}$, (c) follows from Remark 1, (10), (11), and because conditioning reduces entropy, (d) follows from the storage content in (6), where $D_i(j)$ is the part of D_i stored in the excess storage of worker w_j at time t, (e) follows due to the data points are independent and since out of the cache contents Z_j^t, the data sub-point $D_k(i)$ only depends on the sub-point $D_k(j)$, for any $i \neq j \neq k$, (f) follows from the chain rule of entropy where $D_i(\mathcal{W})$ is the part of D_i stored in the excess storage of the workers whose indexes are in $\mathcal{W}$.

Summing up over all possible $4! = 24$ permutations σ of the ordered set $(1, 2, 3, 4)$, we arrive at the following bound,

$$R^*_{\text{worst-case}} d \geq 3d - \frac{1}{24} \sum_{\sigma \in [4!]} [H(D_{\sigma_1}(\sigma_2)) + H(D_{\sigma_1}(\sigma_2, \sigma_3)) + H(D_{\sigma_1}(\sigma_2, \sigma_3, \sigma_4))], \quad (14)$$

where $[4!]$ is the set of all possible permutations of the set $(1, 2, 3, 4)$, and we changed the summation indexes due to the symmetry in the summation.

We assume a generic placement strategy for storing the data point D_k, at time t, in the excess storage of the workers w_i, where $i \neq k$, by defining $D_{k,\mathcal{W}}$ as the partition of D_k stored exclusively in the excess storage of the workers whose labels are in the set $\mathcal{W} \subseteq [1:4] \backslash k$. Defining $|D_{k,\mathcal{W}}| = H(D_{k,\mathcal{W}})/d$, we can express the following entropies in terms of $D_{k,\mathcal{W}}$:

$$H(D_k) = \sum_{\mathcal{W} \subseteq [1:4] \backslash k} |D_{k,\mathcal{W}}| d, \quad H(D_k(i)) = \sum_{\substack{\mathcal{W} \subseteq [1:4] \backslash k \\ i \in \mathcal{W}}} |D_{k,\mathcal{W}}| d. \quad (15)$$

The following two constraints are obtained in terms of $D_{k,\mathcal{W}}$:

- **Data size constraint:** The total size of the data is given by

$$4 = \frac{1}{d} \sum_{k=1}^{4} H(D_k) \overset{(a)}{=} \sum_{\ell=0}^{3} \sum_{k=1}^{4} \sum_{\substack{\mathcal{W} \subseteq [1:4] \backslash k \\ |\mathcal{W}| = \ell}} |D_{k,\mathcal{W}}| = \sum_{\ell=0}^{3} x_\ell, \quad (16)$$

where (a) follows from (15), and $x_\ell \geq 0$ is defined as $x_\ell \triangleq \sum_{k=1}^{4} \sum_{\mathcal{W} \subseteq [1:4] \backslash k: \, |\mathcal{W}| = \ell} |D_{k,\mathcal{W}}|$, for $\ell \in [0:3]$.

- **Excess storage size constraint:** The size of the total excess storage of all the workers cannot exceed $4(S-1)d$ bits,

$$4(S-1) \overset{(a)}{\geq} \sum_{k=1}^{4} \sum_{\substack{\mathcal{W} \subseteq [1:4] \backslash k \\ i \in \mathcal{W}}} |\mathcal{W}| |D_{k,\mathcal{W}}| \overset{(c)}{=} \sum_{\ell=0}^{3} \ell x_\ell, \quad (17)$$

where (a) is true since a unique partition $|D_{k,\mathcal{W}}(i)|$ is counted $|\mathcal{W}|$ number of times, which is the number of workers storing this partition as excess storage.

In the summation term of (14), we obtain the term $|D_{k,\mathcal{W}}|$ only for $|\mathcal{W}| \in \{1, 2, 3\}$. Moreover, due to the symmetry, the coefficient of $|D_{k,\mathcal{W}}|$ for any $k \in [1:K]$ and $\mathcal{W} \in [1:K] \setminus k$ is equal for every value of $|\mathcal{W}|$. Assume this constant coefficient is c_ℓ for $|\mathcal{W}| = \ell$, where $\ell \in \{1, 2, 3\}$. Therefore, we can simplify the bound in (14) as follows:

$$R^*_{\text{worst-case}} \geq 3 - c_1 x_1 - c_2 x_2 - c_3 x_3. \quad (18)$$

Obtaining the coefficient c_ℓ is equivalent to obtaining the coefficient of $|D_{k,\mathcal{W}}|$ for any $k \in [1:K]$ and $|\mathcal{W}| = \ell$. For example, we obtain c_1 by finding the coefficient of $|D_{1,\{2\}}|$ in (14). We get $|D_{1,\{2\}}|$ in the first term of the summation in (14), i.e., $H(D_{\sigma_1}(\sigma_2, \sigma_3, \sigma_4))$, only if $\sigma_1 = 1$ which is satisfied in 6 out of the 24 permutations. In the second term, we obtain $|D_{1,\{2\}}|$ only if $\sigma_1 = 1$ and $\sigma_4 \neq 2$ in total number of 4 permutations. In the third term, we obtain $|D_{1,\{2\}}|$ only if $\sigma_1 = 1$ and $\sigma_2 = 2$ in total number of 2 permutations. Therefore, the coefficient of $|D_{1,\{2\}}|$ is $c_1 = \frac{6+4+2}{24} = \frac{1}{2}$. Following the same steps, and as discussed in detail in [11, Example 2], we obtain $c_2 = \frac{2}{3}$ and $c_3 = \frac{3}{4}$. Therefore, we can write the bound in (18) as follows:

$$R^*_{\text{worst-case}} \geq 3 - \frac{x_1}{2} - \frac{2x_2}{3} - \frac{3x_3}{4}. \quad (19)$$

We get three bounds over $R^*_{\text{worst-case}}$ by selected elimination of some of the variables x_ℓ, where $\ell \in [0:3]$, from (19) using the constraints in (16) and (17), as follows:

$$R^*_{\text{worst-case}} \geq 5 - 2S + \frac{x_2}{3} + \frac{3x_3}{4} \overset{(a)}{\geq} 5 - 2S, \quad (20)$$

$$R^*_{\text{worst-case}} \geq \frac{7}{3} - \frac{2S}{3} + \frac{x_0}{3} + \frac{x_3}{12} \overset{(b)}{\geq} \frac{7 - 2S}{3}, \quad (21)$$

$$R^*_{\text{worst-case}} \geq \frac{4}{3} - \frac{S}{3} + \frac{5x_0}{6} + \frac{x_1}{12} \overset{(c)}{\geq} \frac{4 - S}{3}, \quad (22)$$

where (a), (b) and (c) are true since $x_\ell \geq 0$, for $\ell \in [0:3]$. The intersection of the three bounds in (20), (21), and (22) is the lower convex hull of the 4 storage-rate pairs, $(S = m, R = \frac{4-m}{m})$ for $m \in [1:4]$, which is the lower bound on $R^*_{\text{worst-case}}$ shown in Figure 1, satisfying Theorem 2 for $K = N = 4$. We can also observe that the storage-rate pairs achieved in Theorem 1, i.e., $(7/4, 3/2)$, $(5/2, 2/3)$, and $(13/4, 1/4)$, are optimal. The maximum gap between the bounds is at $S = 1$, and is given by $4/3$, satisfying the bound in Theorem 3.

The next Theorem provides an improved gap through a new scheme, which we call as *"aligned coded shuffling"*.

Theorem 4: For the data shuffling problem, the lower bound over $R^*_{\text{worst-case}}$ in Theorem 2 is achievable for $K < 5$ (hence leads to the optimal tradeoff), while for $K \geq 5$ is achievable within a multiplicative gap which is bounded as

$$\frac{R^{\text{upper}}_{\text{worst-case}}}{R^{\text{lower}}_{\text{worst-case}}} \leq \frac{K - \frac{1}{3}}{K - 1} \leq \frac{7}{6}. \quad (23)$$

In [11, Appendix D], we present the complete proof of Theorem 4 by closing the gap between the two bounds in Theorems 1 and 2 for the storage values $S = m\frac{N}{K}$, and $m \in \{1, K-2, K-1\}$. To illustrate the new ideas, we revisit Example 1 with $K = 4$ workers and $N = 4$ data points.

Example 3: From Figure 1, we notice that if we achieve the storage-rate pairs, $(1, 3)$, $(2, 1)$, and $(3, 1/3)$, then we can fully characterize $R^*_{\text{worst-case}}$ using memory sharing (see [11, Claim 1]). To this end, we present the aligned coded shuffling scheme for data delivery, which combines the ideas of coded shuffling and interference alignment. Let us consider the same shuffle as in Example 1, i.e., $\pi_t = \{1, 2, 3, 4\}$, and $\pi_{t+1} = \{2, 3, 4, 1\}$. Due to space limitation, we only describe here the achievability for the storage-rate pair (2, 1), while the rest of the storage points can be found in [11, Example 3].

Storage Placement: At time t, every data point D_i is partitioned into 3 sub-points, each labeled by the indexes in the set $[1 : 4]$ excluding the index of the worker being assigned to D_i. For example, the data point D_1 (assigned to worker w_1) is partitioned as $D_1 = \{D_{1,\{2\}}, D_{1,\{3\}}, D_{1,\{4\}}\}$. The storage placement at time t is shown in Figure 3a. First, every worker stores the assigned data point. Then, every worker w_k stores the sub-points labeled with k from the remaining data points.

Aligned Coded Shuffling: According to the storage placement, every worker needs 2 sub-points of the new assigned point at time $t+1$. From an interference perspective, each one of the needed sub-points is an interference to exactly one worker. For example, at time $t+1$, $D_{3,\{4\}}$ is (i) needed by w_2, (ii) available at (w_3, w_4), and (iii) interference at w_1. Worker w_1 thus faces interference from exactly two sub-points, $D_{3,\{4\}}$, and $D_{4,\{2\}}$ needed by w_2, and w_3, respectively. Therefore, we first create the aligned coded symbol $D_{3,\{4\}} \oplus D_{4,\{2\}}$ which is: (i) available at w_4, (ii) useful for (w_2, w_3), and (iii) the only source of *aligned* interference for w_1. Similarly, we can produce 4 aligned coded symbols, summarized as follows:

Coded Symbol	Interference at	Available at	Useful for
$D_{3,\{4\}} \oplus D_{4,\{2\}}$	w_1	w_4	w_2 & w_3
$D_{1,\{3\}} \oplus D_{4,\{1\}}$	w_2	w_1	w_3 & w_4
$D_{1,\{2\}} \oplus D_{2,\{4\}}$	w_3	w_2	w_1 & w_4
$D_{2,\{3\}} \oplus D_{3,\{1\}}$	w_4	w_3	w_1 & w_2

Therefore, these 4 coded symbols provide every worker with the 2 needed sub-points. Moreover, it suffices to send only three independent linear combinations of these 4 coded symbols, since every worker already has one of them available locally at its storage. The rate of this transmission is $R = 3 \times 1/3 = 1$, and the pair $(S = 2, R = 1)$ is achievable which closes the gap in Figure 1 for $S = 2$.

Storage update and sub-points relabeling: The storage update at time $t+1$ is needed to preserve the structure of the stored data points. This involves relabeling some of the data sub-points shown by red dashed frames in Figure 3b. For example, the data point D_1 is processed by w_1 at time t, and w_4 at time $t+1$. Thus, to maintain the structural properties of storage at time $t+1$, worker w_1 keeps from D_1, the sub-point $D_{1,\{4\}}$ and relabels it to $D_{1,\{1\}}$, and the same relabeling for $D_{1,\{4\}}$ also happens at worker w_4. Relabeling of other points is done similarly (see [11, Example 3]).

IV. Conclusion

In this paper, we characterized the approximately optimal worst-case communication vs storage tradeoff for the data shuffling problem, within a constant multiplicative gap of $7/6$. We also characterized the optimal tradeoff for all $K < 5$ through a novel aligned coded data shuffling scheme. Future directions include closing the gap for $K \geq 5$, and adding a sub-divisibility constraint over the data points.

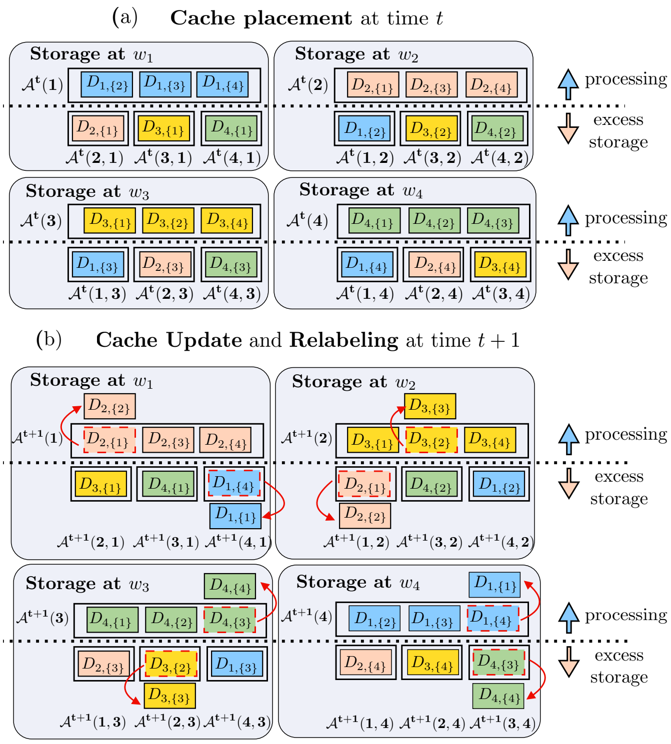

Fig. 3. Closing the gap for $K = N = 4$, and $S = 2$: (a) structural invariant storage placement, (b) storage update and sub-point relabeling to maintain storage structure.

References

[1] L. Bottou, "Stochastic gradient descent tricks," *Neural Networks: Tricks of the Trade*, pp. 421–436, 2012.

[2] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Speeding up distributed machine learning using codes," *IEEE Transactions on Information Theory*, vol. 64, no. 3, pp. 1514–1529, 2018.

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

[4] M. Attia and R. Tandon, "Information theoretic limits of data shuffling for distributed learning," in *Proc. of the IEEE Global Communications Conference (GLOBECOM)*, 2016.

[5] ——, "On the worst-case communication overhead for distributed data shuffling," in *Proc. of the Allerton conference*, Sep. 2016.

[6] L. Song, C. Fragouli, and T. Zhao, "A pliable index coding approach to data shuffling," in *Proc. of the IEEE International Symposium on Information Theory (ISIT)*, 2017.

[7] Z. Bar-Yossef, Y. Birk, T. Jayram, and T. Kol, "Index coding with side information," *IEEE Transactions on Information Theory*, vol. 57, no. 3, pp. 1479–1494, 2011.

[8] K. Wan, D. Tuninetti, and P. Piantanida, "On the optimality of uncoded cache placement," in *Proc. of the IEEE Information Theory Workshop (ITW)*, Sept 2016, pp. 161–165.

[9] Q. Yu, M. Maddah-Ali, and A. Avestimehr, "The exact rate-memory tradeoff for caching with uncoded prefetching," in *Proc. of the IEEE International Symposium on Information Theory (ISIT)*, 2017.

[10] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *IEEE Transactions on Information Theory*, vol. 60, no. 5, pp. 2856–2867, Feb. 2015.

[11] M. Attia and R. Tandon, "Near optimal coded data shuffling for distributed learning," *arXiv preprint arXiv:1801.01875*, 2018. [Online]. Available: https://arxiv.org/pdf/1801.01875.pdf