On the Worst-case Communication Overhead for Distributed Data Shuffling

Mohamed Adel Attia Ravi Tandon
Department of Electrical and Computer Engineering
University of Arizona, Tucson, AZ 85721
E-mail:{*madel, tandonr*}*@email.arizona.edu*

***Abstract*— Distributed learning platforms for processing large scale data-sets are becoming increasingly prevalent. In typical distributed implementations, a centralized master node breaks the data-set into smaller batches for parallel processing across distributed workers to achieve speed-up and efficiency. Several computational tasks are of sequential nature, and involve multiple passes over the data. At each iteration over the data, it is common practice to randomly re-shuffle the data at the master node, assigning different batches for each worker to process. This random re-shuffling operation comes at the cost of extra communication overhead, since at each shuffle, new data points need to be delivered to the distributed workers.**

In this paper, we focus on characterizing the information theoretically optimal communication overhead for the distributed data shuffling problem. We propose a novel coded data delivery scheme for the case of no excess storage, where every worker can only store the assigned data batches under processing. Our scheme exploits a new type of coding opportunity and is applicable to any arbitrary shuffle, and for any number of workers. We also present information theoretic lower bounds on the minimum communication overhead for data shuffling, and show that the proposed scheme matches this lower bound for the worst-case communication overhead.

I. Introduction

Processing of large scale data-sets over a large number of distributed servers is becoming increasingly prevalent. The parallel nature of distributed computational platforms such as Apache Spark [1], Apache Hadoop [2], and MapReduce [3] enables the processing of data-intensive tasks common in machine learning and empirical risk analysis. In typical distributed systems, a centralized node which has the entire data-set assigns different parts of the data to distributed workers for iterative processing.

Several practical computational tasks are inherently sequential in nature, in which the next iteration (or pass over the data) is dependent on the previous iteration. Of particular relevance are sequential optimization algorithms such as incremental gradient descent, stochastic gradient descent, and random reshuffling. The convergence of such iterative algorithms depends on the order in which the data-points are processed, which in turn depends on the skewness of the data. However, the *preferred ordering* of data points is unknown apriori and application dependent. One commonly employed practice is to perform *random reshuffling*, which involves multiple passes over the whole data set with different orderings at each iteration. Random reshuffling has recently been shown to have better convergence rates than stochastic gradient descent [4], [5].

Implementing random reshuffling in a distributed setting comes at the cost of an extra communication overhead, since at each iteration random data assignment is done for the distributed workers, and these data points need to be communicated to the distributed workers. This leads to a fundamental trade-off between the communication overhead, and storage at each worker. On one extreme case when each worker can store the whole data-set, no communication is necessary for any shuffle. On the other extreme, when the workers are just able to store the batches under processing, which is refereed to as the *no-excess storage* case, the communication overhead is expected to be maximum.

Main Contributions: The main focus of this work is characterizing the information theoretic optimal communication overhead for the *no-excess storage* case. The main contributions of this paper are summarized as follows:

- We present an information theoretic formulation of the problem, and develop a novel approach of describing the communication problem through a shuffling matrix which describes the data-flow across the workers.

- We next present a novel coded-shuffling scheme which exploits a new type of coding opportunity in order to reduce the communication overhead, in contrast to existing approaches. Our scheme is applicable to any arbitrary shuffle, and for any number of distributed workers.

- We present information theoretic lower bounds on the communication overhead as a function of the shuffle matrix. Moreover, we show that the proposed scheme matches this lower bound on the worst-case communication overhead, thus characterizing the information theoretically optimal worst-case communication necessary for data shuffling.

Related work: The benefits of coding to reduce communication overhead of shuffling were recently investigated in [6], which proposes a probabilistic coding scheme. However, [6] focuses on using the excess storage at the workers to increase the coding opportunities and reduce the average communication overhead. In our recent work [7], we presented the optimal worst-case communication overhead for any value of storage for two and three distributed workers. In another interesting line of work, Coded MapReduce has been proposed in [8], to reduce the communication between the mappers and reducers. However, the focus of this paper is significantly different, where we study the communication between the centralized master node and the distributed workers, motivated by the random reshuffling problem as initiated in [6].

II. System Model

We consider a master-worker distributed system, where a master node possesses the entire data-set. The master node sends batches of the data-set to the distributed workers over a shared link in order to locally calculate some function or train a model in a parallel manner. The local results are then fed-back to the master node, for iterative processing. In order to enhance the statistical performance of the learning algorithm, the data-set is randomly permuted at the master node before each epoch of the distributed algorithm, and then the shuffled data-points are transmitted to the workers.

We assume a master node which has access to the entire data-set $A = [x_1^T, x_2^T, \dots, x_N^T]^T$ of size Nd bits, i.e., A is a matrix containing N data points, denoted by $x_1, x_2, \dots, x_N$, where d is the dimensionality of each data point. Treating the data points $\{x_n\}$ as independent and identically distributed (i.i.d.) random variables, we have

$$H(x_n) = d, \ \ \forall n \in \{1, \dots, N\}, \quad H(A) = Nd. \tag{1a}$$

At each iteration, indexed by t, the master node divides the data-set A among K distributed workers, given as $A_1^t, A_2^t, \dots, A_K^t$, where the batch A_k^t is designated to be processed by worker w_k, and these batches correspond to the random permutation of the data-set, $\pi^t : A \to \{A_1^t, \dots, A_K^t\}$. Note that these data chunks are disjoint, and span the whole data-set, i.e.,

$$A_i^t \cap A_j^t = \phi, \quad \forall i \neq j, \tag{2a}$$

$$A_1^t \cup A_2^t \cup \ldots \cup A_K^t = A, \quad \forall t. \tag{2b}$$

Hence, the entropy of any batch A_k^t is given as

$$H(A_k^t) = \frac{1}{K} H(A) = \frac{N}{K} d \quad , \forall k \in \{1, \dots, K\}. \tag{3}$$

After getting the data batch, each worker locally computes a function (as an example, this function could correspond to the gradient or sub-gradients of the data points assigned to the kth worker) $f_k(A_k^t)$, in iteration t,. The local functions from the K workers are processed later at the master node, to get an estimate of the function $f_t(A)$. For processing purposes, the data block A_k^t is needed to be stored by the worker while processing, therefore, we assume that worker w_k has a cache Z_k^t with storage capability of size sd bits, for some real number s, that must at least store the data block A_k^t at time t, i.e., if we consider Z_k^t and A_k^t as random variables then the storage constraint is given by

$$H(Z_k^t) = sd \geq H(A_k^t), \qquad \forall k \in \{1, \dots, K\}. \tag{4}$$

For the scope of this paper, we focus on the setting of *no-excess storage*, corresponding to $s = N/K$, in which each worker can exactly store $1/K$ fraction of the entire data, i.e., it only stores $s = N/K$ data points which are assigned to it in that iteration, therefore, the cache content at time t for worker w_k is given by $Z_k^t = A_k^t$, and the relationship in (4) is satisfied with equality. Henceforth, we drop the notation Z_k^t as the cache content and use the notation for the data batch A_k^t instead since they are the same for the no-excess storage setting. In the next epoch $t+1$, the data-set is randomly reshuffled at the master node according to the random permutation $\pi^{t+1} : A \to \{A_1^{t+1}, A_2^{t+1}, \dots, A_K^{t+1}\}$.

The main communication bottleneck occurs during ***Data Delivery*** since the master node needs to communicate some function of the data to all the workers $X_{(\pi_t, \pi_{t+1})}$ of size $R_{(\pi_t, \pi_{t+1})} d$ bits, where $R_{(\pi_t, \pi_{t+1})}$ is the rate of the shared link based on the shuffle (π_t, π_{t+1}). Each worker w_k should be able to extract the data points designated for it out of the incoming data, $X_{(\pi_t, \pi_{t+1})}$ from the master node as well as its locally stored data, i.e., A_k^t.

We next proceed to describe the data delivery mechanism, and the associated encoding and decoding functions. The main process then can be divided into 2 phases, namely the data delivery phase and the storage update phase as described next: in the *data delivery phase*, the master node sends some function of the data to all the workers. Each worker should be able to extract the data points designated for it out of the incoming data from the master node as well as the data stored in its local cache storage. In the *cache update phase*, each worker stores the required data points for processing purposes, that can also be useful in reducing the communication overhead in subsequent epochs.

At time $t+1$, the master node sends a function of the data batches for the subsequent shuffles (π_t, π_{t+1}), $X_{(\pi_t, \pi_{t+1})} = \phi(A_1^t, \dots, A_K^t, A_1^{t+1}, \dots, A_K^{t+1}) = \phi_{(\pi_t, \pi_{t+1})}(A)$ over the shared link, where ϕ is the data delivery encoding function

$$\phi : \left[2^{\frac{N}{K} d}\right]^{2K} \to [2^{R_{(\pi_t, \pi_{t+1})} d}]. \tag{5}$$

Since $X_{(\pi_t, \pi_{t+1})}$ is a function of the data set A, we have

$$H\left(X_{(\pi_t, \pi_{t+1})} | A\right) = 0, \tag{6a}$$

$$H\left(X_{(\pi_t, \pi_{t+1})}\right) = R_{(\pi_t, \pi_{t+1})} d. \tag{6b}$$

Each worker w_k should decode the desired batch A_k^{t+1} out of the transmitted function $X_{(\pi_t, \pi_{t+1})}$, and the data stored in the previous time slot denoted as A_k^t. Therefore, the desired data is given by $A_k^{t+1} = \psi(X_{(\pi_t, \pi_{t+1})}, A_k^t)$, where ψ is the decoding function at the workers

$$\psi : [2^{R_{(\pi_t, \pi_{t+1})} d}] \times [2^{sd}] \to [2^{\frac{N}{K} d}], \tag{7}$$

which also gives us the *decodability constraint* as follows

$$H\left(A_k^{t+1} | A_k^t, X_{(\pi_t, \pi_{t+1})}\right) = 0 \quad , \forall k \in \{1, \dots, K\}. \tag{8}$$

The update procedure for the no-excess storage setting is rather straightforward: worker w_k keeps the part that does not change in the new shuffle, i.e., $A_k^{t+1} \cap A_k^t$. Then it removes the remaining part of its previously stored content, i.e., $A_k^t \setminus A_k^{t+1}$, and stores instead the new part, i.e, $A_k^{t+1} \setminus A_k^t$.

Our goal in this work is to characterize the information theoretic bounds for optimal communication overhead $R^*_{(\pi_t, \pi_{t+1})}(K)$ for any arbitrary number of workers K, and any arbitrary shuffle (π_t, π_{t+1}), defined as

$$R^*_{(\pi_t, \pi_{t+1})}(K) = \min_{(\phi, \psi)} \quad R^{(\phi, \psi)}_{(\pi_t, \pi_{t+1})}(K), \tag{9}$$

where $R^{(\phi, \psi)}_{(\pi_t, \pi_{t+1})}(K)$ is the rate of an achievable scheme

defined by the encoding, and decoding functions (ϕ, ψ). Subsequently, the optimal worst-case overhead is defined as

$$R^*_{\text{worst-case}}(K) = \max_{(\pi_t, \pi_{t+1})} R^*_{(\pi_t, \pi_{t+1})}(K). \quad (10)$$

III. Properties of Distributed Data Shuffling

Before presenting our main results on the communication overhead of shuffling, we present some fundamental properties that are satisfied for any two consecutive data shuffles give by $\pi_t : A \to \{A_1^t, \ldots, A_K^t\}$, and $\pi_{t+1} : A \to \{A_1^{t+1}, \ldots, A_K^{t+1}\}$. We start with the following definitions.

Definition 1 (Shuffle Index): We define

$$S_{i,j}^{(\pi_t, \pi_{t+1})} \triangleq |A_i^t \cap A_j^{t+1}|, \quad (11)$$

as the shuffle index representing the number of data points that are needed by worker w_j at time $t+1$, and are available at worker w_i from the previous shuffle t.

Definition 2 (Shuffle Matrix): We also define the $K \times K$ shuffle matrix for the permutation pair (π_t, π_{t+1}) as

$$S^{(\pi_t, \pi_{t+1})} \triangleq \left[S_{i,j}^{(\pi_t, \pi_{t+1})}\right], \quad i, j \in \{1, \ldots, K\}. \quad (12)$$

Remark 1: The significance of $S_{i,i}^{\pi_t, \pi_{t+1}}$ is that it is the number of common data points between A_i^t, and A_i^{t+1}. Thus, these number of data points do not need to be transmitted to worker w_i, and are not involved in the data delivery process. Using the definition in (11), together with (2), it follows readily that

$$\sum_{i=1}^{K} S_{i,j}^{(\pi_t, \pi_{t+1})} = \sum_{i=1}^{K} |A_i^t \cap A_j^{t+1}| = |A_j^{t+1}| = \frac{N}{K},$$
$$\sum_{j=1}^{K} S_{i,j}^{(\pi_t, \pi_{t+1})} = \sum_{j=1}^{K} |A_i^t \cap A_j^{t+1}| = |A_i^t| = \frac{N}{K}. \quad (13)$$

The properties in (13) imply that the sum of elements across any row (or column) for the shuffling matrix $S^{\pi_t, \pi_{t+1}}$ is constant for any shuffle (π_t, π_{t+1}) and is equal to $\frac{N}{K}$.

Remark 2 (Data-flow Conservation Property): We next state an important property satisfied by any shuffle, namely the data-flow conservation property:

$$\sum_{j \in \{1,\ldots,K\} \setminus i} S_{j,i}^{(\pi_t, \pi_{t+1})} = \sum_{j \in \{1,\ldots,K\} \setminus i} S_{i,j}^{(\pi_t, \pi_{t+1})}. \quad (14)$$

The proof of this property follows directly from (13), and has the following interesting interpretation: the total number of new data points that need to be delivered to worker w_i (and are present elsewhere), i.e., $\sum_{j \neq i} S_{j,i}^{(\pi_t, \pi_{t+1})}$ is exactly equal to the total number of data points that worker w_i has that are desired by the other workers, which is $\sum_{j \neq i} S_{i,j}^{(\pi_t, \pi_{t+1})}$.

Definition 3 (Leftover Index and Leftover Matrix): We define the leftover index as the number of leftover datapoints needed by worker w_j at time $t+1$ and available at w_i at time t as

$$\Omega_{i,j}^{\pi_t, \pi_{t+1}} \triangleq S_{i,j}^{\pi_t, \pi_{t+1}} - \min(S_{i,j}^{\pi_t, \pi_{t+1}}, S_{j,i}^{\pi_t, \pi_{t+1}}). \quad (15)$$

The leftover matrix for the permutation pair (π_t, π_{t+1}) is defined as

$$\Omega^{\pi_t, \pi_{t+1}} \triangleq [\Omega_{i,j}^{\pi_t, \pi_{t+1}}], \quad i, j \in \{1, \ldots, K\}. \quad (16)$$

This definition and the significance of the leftover matrix will become clear in the subsequent sections, when we describe our proposed coded data delivery scheme. From the definition in (15), we note that the diagonal entries of the leftover matrix are all zero.

Remark 3 (Leftover Conservation Property): Analogous to the data-flow conservation property, we next show that the leftover indices also satisfy a similar leftover conservation property, as follows

$$\sum_{j \in \{1,\ldots,K\} \setminus i} \Omega_{i,j}^{(\pi_t, \pi_{t+1})} = \sum_{j \in \{1,\ldots,K\} \setminus i} \Omega_{j,i}^{(\pi_t, \pi_{t+1})}. \quad (17)$$

To prove the above property, we use the definition of leftovers in (15), to first compute the total leftovers at a worker w_i as follows

$$\sum_{j \in \{1,\ldots,K\} \setminus i} \Omega_{i,j}^{(\pi_t, \pi_{t+1})} = \sum_{j \in \{1,\ldots,K\} \setminus i} S_{i,j}^{(\pi_t, \pi_{t+1})} - \sum_{j \in \{1,\ldots,K\} \setminus i} \min(S_{i,j}^{(\pi_t, \pi_{t+1})}, S_{j,i}^{(\pi_t, \pi_{t+1})}). \quad (18)$$

Similarly, we can also write the total number of leftover data points coming from all other workers to worker w_i

$$\sum_{j \in \{1,\ldots,K\} \setminus i} \Omega_{j,i}^{(\pi_t, \pi_{t+1})} = \sum_{j \in \{1,\ldots,K\} \setminus i} S_{j,i}^{(\pi_t, \pi_{t+1})} - \sum_{j \in \{1,\ldots,K\} \setminus i} \min(S_{i,j}^{(\pi_t, \pi_{t+1})}, S_{j,i}^{(\pi_t, \pi_{t+1})}). \quad (19)$$

From the property in (14), we notice that the quantities in (18), and (19) are equal and hence we arrive at the proof of (17). Using the leftover conservation property in (17), we can show that the sum across rows or columns for the leftover matrix Ω is constant for any shuffle (π_t, π_{t+1}).

Subsequently, we refer to $R_{(\pi_t, \pi_{t+1})}$ as the rate for any achievable scheme (ϕ, ψ). We also drop the index (π_t, π_{t+1}) from $S^{(\pi_t, \pi_{t+1})}$, $\Omega^{(\pi_t, \pi_{t+1})}$, $R_{(\pi_t, \pi_{t+1})}$, and $X_{(\pi_t, \pi_{t+1})}$.

IV. Main Results

The main contributions of this paper are presented next in the following three Theorems.

Theorem 1: The optimal communication overhead $R^*(K)$ for a shuffle characterized by a shuffle matrix $S = [S_{i,j}]$ is upper bounded as

$$R^*(K) \leq \sum_{i=1}^{K-1} \sum_{j=i+1}^{K} \max(S_{i,j}, S_{j,i}) - \max_k \sum_{j \in \{1,\ldots,K\} \setminus \{k\}} \Omega_{k,j}. \quad (20)$$

Theorem 2: The optimal communication overhead $R^*(K)$, for any arbitrary shuffle matrix $S = [S_{i,j}]$ is lower bounded as

$$R^*(K) \geq \sum_{i=1}^{K-1} \sum_{j=i+1}^{K} S_{\sigma_i, \sigma_j}, \quad (21)$$

for any permutation σ: $\{1,\dots,K\} \to \{\sigma_1,\dots,\sigma_K\}$ of the K workers.

Theorem 3: The information theoretically optimal worst-case communication overhead for data shuffling is given by

$$R^*_{\text{worst-case}}(K) = \left(\frac{K-1}{K}\right) N. \tag{22}$$

V. Proof of Theorem 1 (Upper bound)

In this section, we present an achievable scheme for the shuffling process, which gives an upper bound on the communication overhead as stated in Theorem 1. We consider the random reshuffling process (π_t, π_{t+1}), characterized by a shuffle matrix $S = [S_{i,j}]$, from time t given by the data batches $A_1^t, A_2^t, \dots, A_K^t$, to time $t+1$ given by the data batches $A_1^{t+1}, A_2^{t+1}, \dots, A_K^{t+1}$.

We first describe the main idea of our scheme through a representative example.

Example 1: Consider $K = 3$ workers (denoted as $\{w_1, w_2, w_3\}$) and $N = 15$ be the total number of data points. Consider the following shuffle matrix $S = [S_{i,j}]$:

$$S = \begin{bmatrix} S_{1,1} & S_{1,2} & S_{1,3} \\ S_{2,1} & S_{2,2} & S_{2,3} \\ S_{3,1} & S_{3,2} & S_{3,3} \end{bmatrix} = \begin{bmatrix} 2 & 1 & 2 \\ 2 & 1 & 2 \\ 1 & 3 & 1 \end{bmatrix} \tag{23}$$

The numbers in the diagonal represents the data points that remains unchanged across the workers, therefore, they do not participate in the communication process (see Remark 1). For uncoded communication, the number of transmitted data points would be the sum of all non-diagonal entries, i.e., $R_{\text{uncoded}} = 11$.

We first show how coding can be utilized to further reduce the communication overhead. For this example, worker w_1 needs $S_{2,1} = 2$ data points from w_2. Let us denote these points as $\{x_{2,1}^{(1)}, x_{2,1}^{(2)}\}$. At the same time, w_2 needs $S_{1,2} = 1$ data point from w_1 (denoted as $x_{1,2}$). Instead of uncoded transmission, the master node can send a coded symbol $x_{2,1}^{(1)} + x_{1,2}$ which is simultaneously useful for both w_1, and w_2 as follows: w_1 has $x_{1,2}$, then it subtracts from the coded symbol to get the needed data-point $x_{2,1}^{(1)}$. Similarly, w_2 gets $x_{1,2}$ using $x_{2,1}^{(1)}$ and $x_{2,1}^{(1)}+x_{1,2}$. This coded symbol is refereed to as an order-2 symbol, since it is useful for two workers at the same time.

By exploiting all such pairwise coding opportunities, we can send a total of 4 order 2 symbols as follows: one coded symbol for $\{w_1, w_2\}$, one for $\{w_1, w_3\}$, and two for $\{w_2, w_3\}$. After having exhausted all pairwise coding opportunities, there are still some remaining data points, which we call as leftovers. The leftover matrix (defined in (15) and (16)), contains the number of leftover symbols after combining the order 2 symbols, is given as

$$\Omega = \begin{bmatrix} \Omega_{1,1} & \Omega_{1,2} & \Omega_{1,3} \\ \Omega_{2,1} & \Omega_{2,2} & \Omega_{2,3} \\ \Omega_{3,1} & \Omega_{3,2} & \Omega_{3,3} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{24}$$

If the remaining 3 leftover symbols (sum of all non-zero elements of Ω) are sent uncoded, then, the total rate would be $R_{\text{paired-coding}} = R_{\text{coded-order2}} + R_{\text{uncoded-leftovers}} = 4 + 3 = 7$, therefore, $R_{\text{paired-coding}} < R_{\text{uncoded}}$.

We now describe the main idea behind our proposed coding scheme which exploits a new type of coding opportunity as follows. Till this end, for each worker, we combine its incoming leftover symbols with its outgoing leftover symbols. By the leftover conservation property, these two are equal. Then, we have the three coded symbols as follows

$$\{x_{3,1} + x_{1,2}, \quad x_{1,2} + x_{2,3}, \quad x_{2,3} + x_{3,1}\}. \tag{25}$$

The key observation is that *any two out of these three* coded symbols are enough for all the workers to get the remaining leftovers. Two workers decode the needed points in one step, while the ignored worker decodes in two steps.

For example, if the master node transmits the first two coded symbols, i.e., $x_{3,1} + x_{1,2}$ and $x_{1,2} + x_{2,3}$, then the decoding works as follows: w_1, and w_2 have $x_{1,2}$, and $x_{2,3}$, respectively, then they can get the needed ones, $x_{3,1}$, and $x_{1,2}$, respectively. Worker w_3, however, decodes its desired symbol through a two step procedure as follows: since it has $x_{3,1}$, then it can get $x_{1,2}$ from the first symbol $x_{3,1} + x_{1,2}$ in the first step. In the second step, from the second symbol $x_{1,2} + x_{2,3}$, it then uses $x_{1,2}$ to finally obtain the needed data point $x_{2,3}$. As a summary, we are able to send 3 leftovers in 2 coded symbols only. Therefore, communication overhead of the proposed scheme reduces to $R_{\text{proposed-coded}} = R_{\text{coded-order2}} + R_{\text{coded-leftovers}} = 4 + 2 = 6$, i.e., $R_{\text{proposed-coded}} < R_{\text{paired-coding}}$.

We next present our proposed scheme for a general shuffle matrix and arbitrary number of workers K, which can be described in the following two phases, namely the first phase of transmitting order-2 symbols, and the second phase, which is what we call the leftover combining phase.

A. Phase 1: Order-2 symbols

First we start by transmitting order-2 symbols, that are useful for two workers at the same time. If we consider two workers w_i, and w_j, then worker w_i has some data points for worker w_j, given by $A_i^t \cap A_j^{t+1}$, which are $S_{i,j} = |A_i^t \cap A_j^{t+1}|$ data points in total. Similarly, w_j has $S_{j,i}$ data points for w_i. Now, if we take all the data points $x_{i,j} \in A_i^t \cap A_j^{t+1}$, and combine them with the points $x_{j,i} \in A_j^t \cap A_i^{t+1}$ to transmit order-2 symbols jointly useful for w_i, and w_j, then we are limited by $\min(S_{i,j}, S_{j,i})$ number of order-2 symbols for the pair (i,j). Therefore, we can transmit total number of order-2 symbols for all possible (i,j) pairs of workers as follows

$$R_{\text{Phase 1}} = \sum_{i=1}^{K-1} \sum_{j=i+1}^{K} \min(S_{i,j}, S_{j,i}). \tag{26}$$

B. Phase 2: Coded Leftover Communication

Now, we consider a coded approach for sending the leftovers after combining the order-2 symbols at phase 1. For a pair of workers (i,j), after combining $\min(S_{i,j}, S_{j,i})$

symbols in phase 1, then we still have $\Omega_{i,j} = S_{i,j} - \min(S_{i,j}, S_{j,i})$ leftover symbols that are still needed to be transmitted from w_i to w_j. Similarly, the leftovers form w_j to w_i is given by $\Omega_{j,i} = S_{j,i} - \min(S_{i,j}, S_{j,i})$. We notice that if $S_{i,j} > S_{j,i}$, then $\Omega_{i,j} = S_{i,j} - S_{j,i} > 0$, and $\Omega_{j,i} = 0$, and vice versa. This gives us the following properties

$$\begin{aligned} &\Omega_{i,j} + \Omega_{j,i} = \max(\Omega_{i,j}, \Omega_{j,i}) = |S_{i,j} - S_{j,i}|, \\ &\min(\Omega_{i,j}, \Omega_{j,i}) = 0. \end{aligned} \tag{27}$$

Clearly, if $S_{i,j} = S_{j,i}$, then $\Omega_{i,j} = \Omega_{j,i} = 0$, and there are no leftover symbols for the pair (i, j). The property in (27) states that if a worker w_i has some data points for w_j in its leftovers ($\Omega_{i,j} \neq 0$), then w_j has nothing in its leftovers needed by w_i ($\Omega_{j,i} = 0$). Using the leftover data conservation property in (17), we first state the following claim:

Claim 1: After combining the order-2 symbols in phase 1, the total number of symbols at a worker w_i needed by other workers (outgoing leftovers) is equal to the total number of data points needed by the worker w_i from other workers (incoming needed points).

As a simple scheme, we can use Claim 1 to combine all the leftovers with the needed data points for every worker w_i. Therefore, each worker can use its own outgoing leftover data points to get the desired incoming points. However, it is obvious that this coded scheme achieves the same rate as if we are sending the leftovers uncoded.

We next present the following claim which is one of the novel contributions of this paper:

Claim 2: If we combine the leftovers with the needed data points for any $K-1$ workers, then under a certain combining condition (stated below) for the remaining ignored worker, say w_k, it can get its own needed data points without the need of being combined with its own leftovers.

Before presenting the proof of Claim 2, we first state the combining condition. In order to ignore a worker w_k from combining its leftovers with the needed points, the following condition must be satisfied while combining the leftovers with the needed points for other non-ignored workers:

Definition 4 (Combining Condition for Ignoring w_k): The needed data-points at the ignored worker w_k from leftovers of other workers $x_{i,k}$, and independently the leftovers at w_k needed by other workers $x_{k,j}$ should only be combined with the data-points $x_{j,i}$ as follows

$$\{x_{k,j} + x_{j,i},\ \ x_{j,i} + x_{i,k}\}. \tag{28}$$

In order to understand the combining condition, we use the following example. Let us consider the following three types of leftover data points: (i) a data point $x_{i,k}$ that is needed by an ignored worker w_k, and is available at worker w_i; (ii) a data point $x_{k,j}$ that is a leftover at w_k, and is needed by worker w_j; and (iii) a data point $x_{i,j}$ that is a leftover at w_i, and is needed by worker w_j.

In order for w_k to decode $x_{i,k}$ using the leftover $x_{k,j}$, the leftover coded combining condition should be satisfied as follows

- While combining the leftovers with the needed points of w_j at the master node, the needed data point $x_{k,j}$ (from w_j's perspective) should only be combined with the leftover data point $x_{j,i}$ as follows:

$$x_{k,j} + x_{j,i}. \tag{29}$$

- While combining the leftovers with the needed points of w_i at the master node, the leftover data point $x_{i,k}$ (from w_i's perspective) should only be combined with the needed data point $x_{j,i}$ as follows:

$$x_{j,i} + x_{i,k}. \tag{30}$$

From the above coded combining, we notice the following: 1) Workers w_i, and w_j still can decode the needed points $x_{j,i}$, and $x_{k,j}$, respectively. 2) Worker w_k decodes in two steps: First, it uses $x_{k,j}$ to get $x_{j,i}$ from the coded symbol in (29). In the next step, from the second coded symbol in (30) it uses $x_{j,i}$ to decode the needed data point $x_{i,k}$.

C. Proof of Claim 2

Now we need to prove formally the decodability at the ignored worker w_k. In order to complete the proof, we need to show that the number of intermediate points the ignored worker w_k can get in the first step of decoding; are enough to decode the needed points in the next step of the decoding process.

We start by partitioning the leftover data points $\Omega_{i,j}$ into non-overlapping $(K-2)$ parts $\Omega_{i,j}^{(\ell)},\ \ \ell \in \{1, 2, \ldots, K\} \setminus \{i, j\}$, where $\Omega_{i,j}^{(\ell)} \leq \Omega_{i,j}$ is defined as the number of intermediate (unintended since $\ell \neq \{i, j\}$) data points originally needed by w_j that w_ℓ can get using its own leftovers needed for w_i (through w_i).

Therefore, $\Omega_{i,j}$ can be written as

$$\Omega_{i,j} = \sum_{\ell \in \{1,\ldots,K\} \setminus \{i,j\}} \Omega_{i,j}^{(\ell)}. \tag{31}$$

As shown in Figure 1, w_K for example uses its own leftovers needed by w_1 (through w_1), i.e., $\Omega_{K,1}$ points, to get unintended points (labelled with blue) that are needed by the other workers $\{2, 3, \ldots, K-1\}$, i.e., $\Omega_{1,2}^{(K)}, \ldots, \Omega_{1,K-1}^{(K)}$. Therefore, the total number of unintended (intermediate) data points recovered by w_K using $\Omega_{K,1}$ data points is

$$\Omega_{K,1} = \sum_{j=2}^{K-1} \Omega_{1,j}^{(K)}. \tag{32}$$

Generally, through the combined symbols for w_i, the number of unintended data points which worker w_ℓ can obtain is

$$\Omega_{\ell,i} = \sum_{j=\{1,\ldots,K\} \setminus \{i,\ell\}} \Omega_{i,j}^{(\ell)}. \tag{33}$$

Let us assume now without loss of generality, that the ignored worker is the last worker w_K. As shown in Figure 1, the ignored worker w_K cannot get the needed data-points (colored chunks above the dotted lines) directly. Instead, w_K uses its leftovers $\sum_{i=1}^{K-1} \Omega_{K,i}$ to get first unintended intermediate points (blue labelled points $\Omega_{1,j}^{(K)}$ through w_1, red labelled

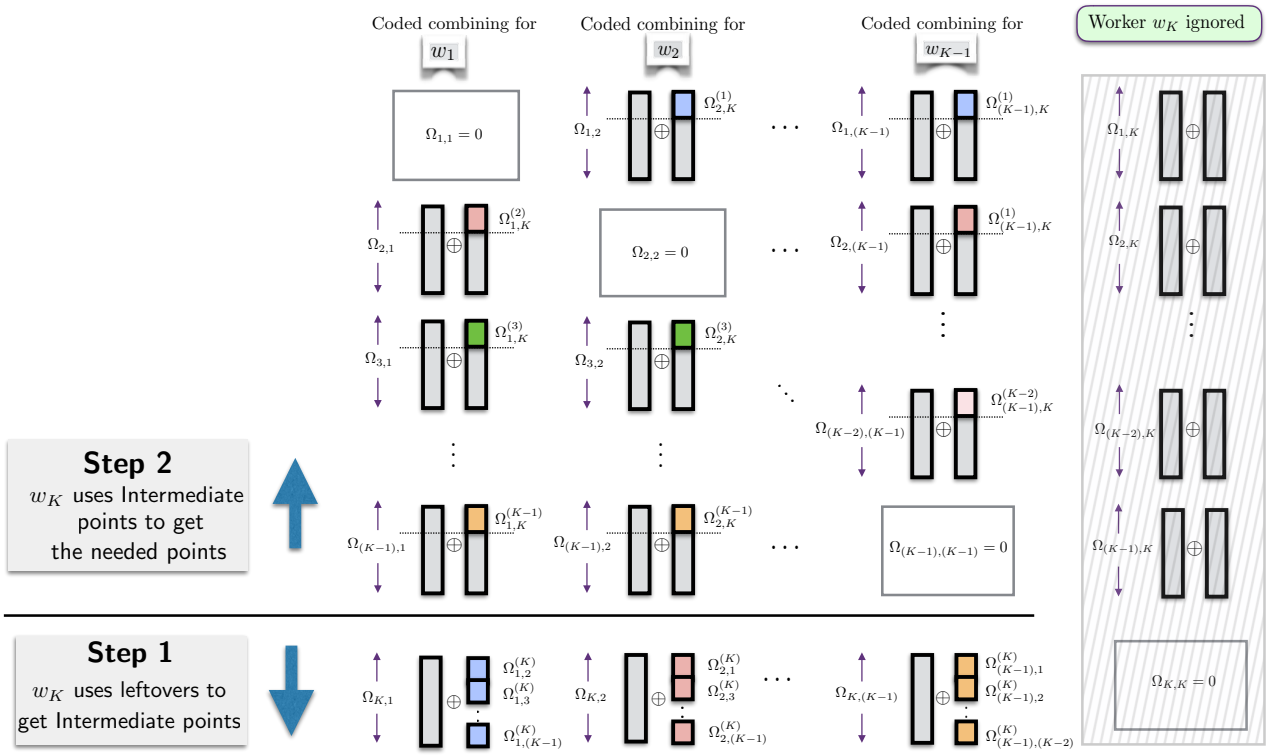

Fig. 1. The leftover combining process after ignoring w_K. Below the solid line is the first step of decoding for the ignored worker where w_K gets intermediate points using its leftover points. Above the solid line is the second step of decoding, where w_K uses the intermediate points to decode the needed points.

points $\Omega_{2,j}^{(K)}$ through w_2, etc.), which are shown below the solid line in the Figure.

In order for w_K to make use of the intermediate symbols $\Omega_{i,j}^{(K)}$, $\{(i,j) \in \{1,\ldots,K-1\}, i \neq j\}$, every symbol $x_{i,j}$ of them should be paired up with data points useful for w_K in the coded combining for w_j, i.e, $x_{i,j} + x_{j,K}$, which is satisfying the combining constraint in Definition 4. Following the relation in (33), the actual total number of unintended symbols w_K can get in the first step of decoding is given by

$$\sum_{i=1}^{K-1} \Omega_{K,i} = \sum_{i=1}^{K-1} \sum_{j=\{1,\ldots,K-1\}\setminus\{i\}} \Omega_{i,j}^{(K)} = \sum_{\substack{(i,j)\in\{1,\ldots,K-1\}\\ i\neq j}} \Omega_{i,j}^{(K)}. \tag{34}$$

Using the unintended symbols that w_K gets through w_i and are originally needed by w_j, i.e., $\Omega_{i,j}^{(K)}$, it should be able to decode the needed symbols $\Omega_{j,K}^{(i)}$. As an example, w_K gets the blue unintended data points $\Omega_{1,2}^{(K)},\ldots,\Omega_{1,K-1}^{(K)}$ through w_1, then these data points are used to get the blue labelled needed points $\Omega_{2,K}^{(1)},\ldots,\Omega_{K-1,K}^{(1)}$ as shown above the solid line in Figure 1.

The minimum number of unintended symbols w_K needs to decode out of $\Omega_{i,j}$ points in the first step, should be enough to decode (equal to) the needed part $\Omega_{j,K}^{(i)}$ in the next step of decoding. From the unintended data recovery condition in (33), $\Omega_{j,K}^{(i)}$ is given by

$$\Omega_{j,K}^{(i)} = \Omega_{i,j} - \sum_{\ell=\{1,\ldots,K-1\}\setminus\{i,j\}} \Omega_{j,\ell}^{(i)}. \tag{35}$$

Therefore, the total number of unintended symbols that the worker w_K should at least have in order to decode all the needed points in the next step is given by

$$\begin{aligned}
\sum_{j=1}^{K-1} \Omega_{j,K} &\overset{(a)}{=} \sum_{\substack{(i,j)\in\{1,\ldots,K-1\}\\ i\neq j}} \Omega_{j,K}^{(i)} \\
&\overset{(b)}{=} \sum_{\substack{(i,j)\in\{1,\ldots,K-1\}\\ i\neq j}} \Omega_{i,j} - \sum_{\substack{(i,j,\ell)\in\{1,\ldots,K-1\}\\ i\neq j\neq \ell}} \Omega_{j,\ell}^{(i)} \\
&\overset{(c)}{=} \sum_{\substack{(i,j)\in\{1,\ldots,K-1\}\\ i\neq j}} \Omega_{i,j} - \sum_{\substack{(i,j,\ell)\in\{1,\ldots,K-1\}\\ i\neq j\neq \ell}} \Omega_{i,j}^{(\ell)} \\
&= \sum_{\substack{(i,j)\in\{1,\ldots,K-1\}\\ i\neq j}} \left[\Omega_{i,j} - \sum_{\ell\in\{1,\ldots,K-1\}\setminus\{i,j\}} \Omega_{i,j}^{(\ell)}\right] \\
&\overset{(d)}{=} \sum_{\substack{(i,j)\in\{1,\ldots,K-1\}\\ i\neq j}} \Omega_{i,j}^{(K)},
\end{aligned} \tag{36}$$

where (a) follows from (31), (b) follows from the constraint

in (35), (c) by switching the sum indices, and (d) from the definition in (31). From (34) and (36), it now follows that the total number of intermediate points the ignored worker w_K can decode in the first step is exactly equal to the minimum number it must decode in order to get the needed points in the second step, which completes the proof of Claim 2.

Hence, the total communication overhead of phase 2 is the total of all leftover symbols (except the ignored worker k), and is given as:

$$R_{\text{Phase 2}} = \overbrace{\sum_{i\in\{1,\dots,K\}\setminus\{k\}}}^{\text{ignoring } w_k} \overbrace{\sum_{j\in\{1,\dots,K\}\setminus\{i\}} \Omega_{i,j}}^{\text{leftovers at } w_i} \tag{37}$$

$$= \sum_{i=1}^{K} \sum_{j\in\{1,\dots,K\}\setminus\{i\}} \Omega_{i,j} - \sum_{j\in\{1,\dots,K\}\setminus\{k\}} \Omega_{k,j}$$

$$= \sum_{i=2}^{K}\sum_{j=1}^{i-1}\Omega_{i,j} + \sum_{i=1}^{K-1}\sum_{j=i+1}^{K}\Omega_{i,j} - \sum_{j\in\{1,\dots,K\}\setminus\{k\}} \Omega_{k,j} \tag{38}$$

$$\overset{(a)}{=} \sum_{i=1}^{K-1}\sum_{j=i+1}^{K}\Omega_{j,i} + \sum_{i=1}^{K-1}\sum_{j=i+1}^{K}\Omega_{i,j} - \sum_{j\in\{1,\dots,K\}\setminus\{k\}} \Omega_{k,j} \tag{39}$$

$$= \sum_{i=1}^{K-1}\sum_{j=i+1}^{K}(\Omega_{i,j}+\Omega_{j,i}) - \sum_{j\in\{1,\dots,K\}\setminus\{k\}} \Omega_{k,j} \tag{40}$$

$$\overset{(b)}{=} \sum_{i=1}^{K-1}\sum_{j=i+1}^{K}\max(\Omega_{i,j},\Omega_{j,i}) - \sum_{j\in\{1,\dots,K\}\setminus\{k\}} \Omega_{k,j}, \tag{41}$$

where (a) follows by swapping the indices j and i in the first summand, and (b) follows from the property of leftovers in (27), which states that that $\min(\Omega_{i,j},\Omega_{j,i})=0$.

Hence, the total communication overhead of the proposed scheme is the total number of transmitted symbols over Phases 1 and 2, which is the sum of (26), and (41), and is given by

$$R(K) = R_{\text{Phase 2}} + R_{\text{Phase 2}}$$

$$= \sum_{i=1}^{K-1}\sum_{j=i+1}^{K}\min(S_{i,j},S_{j,i}) + \sum_{i=1}^{K-1}\sum_{j=i+1}^{K}\max(\Omega_{i,j},\Omega_{j,i}) - \sum_{j\in\{1,\dots,K\}\setminus\{k\}} \Omega_{k,j}$$

$$\overset{(a)}{=} \sum_{i=1}^{K-1}\sum_{j=i+1}^{K}\max(S_{i,j},S_{j,i}) - \sum_{j\in\{1,\dots,K\}\setminus\{k\}} \Omega_{k,j}, \tag{42}$$

where (a) follows from the property in (27). In order to get the lowest possible rate for this scheme, which is also an upper bound for the optimal communication overhead, the choice of the ignored worker w_k can be optimized to have the maximum number of leftovers, which is given by

$$R^*(K)$$

$$\leq \min_k \left(\sum_{i=1}^{K-1}\sum_{j=i+1}^{K}\max(S_{i,j},S_{j,i}) - \sum_{j\in\{1,\dots,K\}\setminus\{k\}} \Omega_{k,j} \right)$$

$$= \sum_{i=1}^{K-1}\sum_{j=i+1}^{K}\max(S_{i,j},S_{j,i}) - \max_k \left(\sum_{j\in\{1,\dots,K\}\setminus\{k\}} \Omega_{k,j} \right). \tag{43}$$

This completes the proof of Theorem 1.

VI. Proof of Theorem 2 (Lower bound)

In this section, we present the lower bound on the optimal communication overhead for any arbitrary random shuffle between two subsequent epochs t, and $t+1$ given by a shuffle matrix $S=[S_{i,j}]$, as stated in Theorem 2.

$$\begin{aligned}
Nd &\overset{(a)}{=} H(A) \\
&\overset{(b)}{=} I(A;A_1^t,\dots,A_K^t,X) + H(A|A_1^t,\dots,A_K^t,X) \\
&\overset{(c)}{=} H(A_1^t,\dots,A_K^t,X) - H(A_1^t,\dots,A_K^t,X|A) \\
&\overset{(d)}{=} H(A_{\sigma_1}^t,A_{\sigma_2}^t,\dots,A_{\sigma_K}^t,X) \\
&\overset{(e)}{=} H(A_{\sigma_K}^t,X) + \sum_{i=1}^{K-1} H(A_{\sigma_i}^t|A_{\sigma_{i+1}}^t,\dots,A_{\sigma_K}^t,X) \\
&\overset{(f)}{\leq} H(A_{\sigma_K}^t) + H(X) + \sum_{i=1}^{K-1} H(A_{\sigma_i}^t|A_{\sigma_{i+1}}^{t+1},\dots,A_{\sigma_K}^{t+1}) \\
&\overset{(g)}{\leq} \frac{Nd}{K} + Rd + \sum_{i=1}^{K-1}\left[\frac{Nd}{K} - I(A_{\sigma_i}^t;A_{\sigma_{i+1}}^{t+1},\dots,A_{\sigma_K}^{t+1})\right] \\
&= Nd + Rd - \sum_{i=1}^{K-1} I(A_{\sigma_i}^t;A_{\sigma_{i+1}}^{t+1},\dots,A_{\sigma_K}^{t+1}),
\end{aligned} \tag{44}$$

where (a) follows from (3), (b) and (c) are due to the fact that $I(A;B)=H(A)-H(A|B)=H(B)-H(B|A)$, and from (2b) where the data-batches at any time span A, (d) from (2b) and (6a), where the data-batches and X are all functions of the data-set A, and σ is any permutation of the the set $\{1,\dots,K\}$, (e) from the chain rule of entropy, (f) from the decoding constraint in (8), the fact that conditioning reduces entropy, and the fact $H(A,B)\leq H(A)+H(B)$, and (g) from (3), (6b), and the fact $H(A|B)=H(A)-I(A;B)$. By rearranging the inequality in (44), we arrive at

$$\begin{aligned}
Rd &\geq \sum_{i=1}^{K-1} I(A_{\sigma_i}^t;A_{\sigma_{i+1}}^{t+1},\dots,A_{\sigma_K}^{t+1}) \\
&= \sum_{i=1}^{K-1}\sum_{j=i+1}^{K} I(A_{\sigma_i}^t;A_{\sigma_j}^{t+1}) = \sum_{i=1}^{K-1}\sum_{j=i+1}^{K} S_{\sigma_i,\sigma_j} d.
\end{aligned} \tag{45}$$

Therefore, the lower bound on the communication overhead is given by $R^*(K) \geq \sum_{i=1}^{K-1}\sum_{j=i+1}^{K} S_{\sigma_i,\sigma_j}$, completing the proof of Theorem 2.

VII. Proof of Theorem 3

In this section, we prove the optimality of our proposed scheme for the worst-case shuffle, which describes the maximum communication overhead across all possible shuffles.

A. Achievability (Worst-case Shuffle)

We start by using the upper bound described in Theorem 1, where we use a variation of the expression in (20) by adding (26), and (37) as follows

$$\begin{aligned} R(K) &\overset{(a)}{=} \sum_{i=1}^{K-1}\sum_{j=i+1}^{K} \min(S_{i,j}, S_{j,i}) + \sum_{i\in\{1,\ldots,K\}\setminus\{k\}} \sum_{j=1}^{K} \Omega_{i,j} \\ &\overset{(b)}{=} \sum_{i=1}^{K-1}\sum_{j=i+1}^{K} \min(S_{\sigma_i,\sigma_j}, S_{\sigma_j,\sigma_i}) + \sum_{i=1}^{K-1}\sum_{j=1}^{K} \Omega_{\sigma_i,\sigma_j} \\ &\overset{(c)}{=} \sum_{i=1}^{K-1}\sum_{j=i+1}^{K} \min(S_{\sigma_i,\sigma_j}, S_{\sigma_j,\sigma_i}) + \sum_{i=1}^{K-1}\sum_{j=1}^{K} S_{\sigma_i,\sigma_j} \\ &\quad - \sum_{i=1}^{K-1}\sum_{j=1}^{K} \min(S_{\sigma_i,\sigma_j}, S_{\sigma_j,\sigma_i}) \\ &\overset{(d)}{\leq} \sum_{i=1}^{K-1}\sum_{j=1}^{K} S_{\sigma_i,\sigma_j} \overset{(e)}{=} \sum_{i=1}^{K-1} \frac{N}{K} = \left(\frac{K-1}{K}\right) N, \end{aligned} \tag{46}$$

where (a) holds because $\Omega_{i,i} = 0$, (b) follows by considering a permutation $\sigma = \{\sigma_1, \ldots \sigma_K\}$ of the workers, where $\sigma_K = k$ is the ignored worker, (c) follows from the definition of $\Omega_{i,j}$ in (15), (d) is due to the fact that $\min(S_{i,j}, S_{j,i}) \geq 0$, and (e) from the property in (13). Since this derived upper bound is found for any arbitrary shuffle, it is also an upper bound for the optimal worst-case communication overhead. Hence, we have

$$R^*_{\text{worst-case}}(K) \leq \left(\frac{K-1}{K}\right) N. \tag{47}$$

B. Converse (Information Theoretic lower bound)

We start by assuming a particular data shuffle, and then specialize our lower bound (obtained in Theorem 2) for this particular shuffle. We use the fact that the worst-case overhead $R^*_{\text{worst-case}}(K)$ is lower bounded by the overhead of any shuffle $R(K)$, therefore the lower bound found for this given shuffle works as a lower bound for the worst-case as well, i.e.,

$$R^*_{\text{worst-case}}(K) \geq R^*(K). \tag{48}$$

We assume a data shuffle matrix S described as follows: For some permutation of the K workers given by $\sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_K\}$, any worker $w_{\sigma_{i+1}}$ at time $t+1$ needs only all the data points that w_{σ_i} has from the previous shuffle at time t, which can be described as

$$S_{\sigma_i,\sigma_j} = \begin{cases} \frac{N}{K}, & j = i+1, \\ 0, & \text{otherwise.} \end{cases} \tag{49}$$

Therefore, using the lower bound in Theorem 2 given by (21), and using (48), the lower bound for this particular shuffle, and hence the optimal worst-case shuffle, can be found as

$$\begin{aligned} R^*_{\text{worst-case}}(K) \geq R^*(K) &\geq \sum_{i=1}^{K-1}\sum_{j=i+1}^{K} S_{\sigma_i,\sigma_j} \\ &= \sum_{i=1}^{K-1} S_{\sigma_i,\sigma_{i+1}} = \sum_{i=1}^{K-1} \frac{N}{K} = \left(\frac{K-1}{K}\right) N. \end{aligned} \tag{50}$$

From (47), and (50), it follows that the information theoretically optimal worst case communication overhead is

$$R^*_{\text{worst-case}}(K) = \left(\frac{K-1}{K}\right) N. \tag{51}$$

VIII. Conclusion

In this paper, we presented new results on the minimum necessary communication overhead for the data shuffling problem. We proposed a novel coded-shuffling scheme which exploits a new type of coding opportunity, namely coded leftover combining in order to reduce the communication overhead. Our scheme is applicable to any arbitrary shuffle, and for any number of distributed workers. We also presented an information theoretic lower bound on the optimal communication overhead that is also applicable for any arbitrary shuffle. Finally, we showed that the proposed scheme matches this lower bound for the worst-case communication overhead across all shuffles, and thus characterizes the information theoretically optimal worst-case overhead.

References

[1] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," in *Proceedings of the 2nd USENIX Workshop on Hot Topics in Cloud Computing (HotCloud)*, 2010.

[2] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system," in *IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, May 2010, pp. 1–10.

[3] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," in *Proceedings of the 6th Symposium on Operating System Design and Implementation (OSDI)*, 2004.

[4] M. Gurbuzbalaban, A. Ozdaglar, and P. Parrilo, "Why Random Reshuffling Beats Stochastic Gradient Descent," *CoRR*, vol. abs/1510.08560, 2015. [Online]. Available: http://arxiv.org/abs/1510.08560

[5] S. Ioffe and C. Szegedy, "Batch normalization accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015. [Online]. Available: http://arxiv.org/abs/1502.03167

[6] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Speeding Up Distributed Machine Learning Using Codes," *CoRR*, vol. abs/1512.02673, December 2015. [Online]. Available: http://arxiv.org/abs/1512.02673

[7] M. Attia and R. Tandon, "Information theoretic limits of data shuffling for distributed learning," in *Proceedings IEEE Global Communications Conference (GLOBECOM)*, Dec. 2016. [Online]. Available: Available: https://www.dropbox.com/s/lk00u2nuf7tiogr/GC2016.pdf?dl=0

[8] S. Li, M. A. Maddah-Ali, and S. Avestimehr, "Coded MapReduce," Presented at the 53rd Annual Allerton conference on Communication, Control, and Computing, Monticello, IL, Sep. 2015.