Mapping Heterogeneity Does Not Affect Wireless Coded MapReduce

Eleftherios Lampiris, Daniel Jiménez Zorrilla, Petros Elia
Communication Systems Department, EURECOM
Sophia Antipolis, France
Email: {lampiris, jimenez, elia}@eurecom.fr

Abstract— **The work considers a Coded MapReduce setting where computing nodes of different processing capabilities coexist. Motivated by scenarios where the mapping phase is performed by nodes of heterogeneous computing capabilities, we explore the setting with K_1 nodes that can each map a fraction $\gamma_1 \in \left[\frac{1}{K}, 1\right]$ of the dataset, and K_2 nodes that can each map a smaller fraction $\gamma_2 < \gamma_1$. For the standard wireless (single-antenna) device-to-device channel or its equivalent wired network with network-coding capabilities at the nodes, we propose a solution of assigning data to the nodes and a method of communicating intermediate values during the shuffling phase, that can be applied to any MapReduce problem and which entirely removes the affects of heterogeneity. The surprising outcome of this work is that the shuffling-phase delay is reduced by a factor of $K_1\gamma_1 + K_2\gamma_2$, matching the performance of the corresponding homogeneous setting, thus revealing for the first time that heterogeneity during the mapping phase does not inherently deteriorate the overall performance.**

I. Introduction

The MapReduce (MR) model [1] is a parallel computing framework that transforms a sequential problem into a parallel one. For a setting of K computing nodes connected through a bottleneck channel, the ultimate objective is the parallel computation of $Q \geq K$ functions on a dataset F comprized of f elements. To facilitate the execution of $\frac{Q}{K}$ functions at each node i.e., to allow a parallel execution of the final problem, the MR process takes place in three distinct steps, i) the *mapping*, ii) the *shuffling*, and iii) the *reduce* phases.

The required execution time of the MR model, assuming that phases happen in sequence, can be seen to be

$$T_{\text{tot}}^{\text{MR}} = T_{\text{map}}\left(\frac{f}{K}\right) + T_{\text{shuf}}\frac{K-1}{K} + \frac{Q}{K}T_{\text{red}} \tag{1}$$

where $T_{\text{map}}(df)$ is the time spent, by one node, to pre-process a fraction d of the dataset, T_{shuf} is the required time to communicate the entire amount of intermediate values between any two nodes, and T_{red} denotes the time required, by a single node, to complete the reduction part of a single function.

Coded MapReduce: By examining Eq. (1), it can be evident that while the mapping and reduce phases are scalable with the number of computing nodes, the real bottleneck becomes the communication (shuffling) phase. To tackle the above bottleneck, a novel method was introduced in [2], [3], named Coded MapReduce (CMR), that draws inspiration from the coded transmissions used in the cache-aided communications literature [4], where the redundant storage of data across the different caches translated in speeding up the communication phase. The main idea in [2], [3] is for each node to map an equal fraction $\gamma > \frac{1}{K}$ of the dataset, allowing for each element of the dataset to be mapped a total of $K\gamma$ times across the different nodes. This increased redundancy (from 1 to $K\gamma$) equips nodes with more intermediate values, thus impacting the shuffling phase in two ways. Firstly each node, having mapped a bigger part of the dataset, will have to receive less intermediate values from the other nodes; this reduction can be referred to as a *local gain*. Secondly, due to this same redundancy, each transmitted message can now contain a combination of several desired intermediate values (usually combined in the form of a XOR), because now decoding at receiving nodes is assisted by the computed intermediate values which are used to remove the interfering messages inside the XOR. This allows for a shuffling phase that serves more than one node at a time. This number of nodes served at a time is referred to as the *coding gain* and, when the nodes are able to map an equal fraction of data, this gain has been shown in [2], [3] to match the aforementioned redundancy $K\gamma$. Hence, we know from [2], [3] that the shuffling-phase delay can be reduced by a factor equal to the dataset redundancy, and thus the overall completion time of CMR takes the form

$$T_{\text{tot}}^{\text{CMR}} = T_{\text{map}}(\gamma f) + T_{\text{shuf}}\frac{1-\gamma}{K\gamma} + \frac{Q}{K}T_{\text{red}}. \tag{2}$$

Compared to MR, this shuffling delay reduction comes at the expense of an increase in the delay for the mapping phase, and thus a tradeoff is formed between the cost of increased mapping time and the associated gains that this increase entails.

Coding for Straggler Mitigation: Another line of work (see [5]–[10]), has considered a different bottleneck of distributed computing (not falling under the MR framework) that is now caused by some nodes experiencing delays during the computation, thus causing significant delays in the overall execution time of the algorithm. The main idea behind these efforts to alleviate this so-called straggling effect, is to split the dataset into some $L < K$ parts and assign to nodes a function of one or more parts (usually in the form of coded linear combinations), thus needing only a subset of nodes (in particular the faster ones) to recover the result.

The work is supported by the European Research Council under the EU Horizon 2020 research and innovation program / ERC grant agreement no. 725929. (ERC project DUALITY)

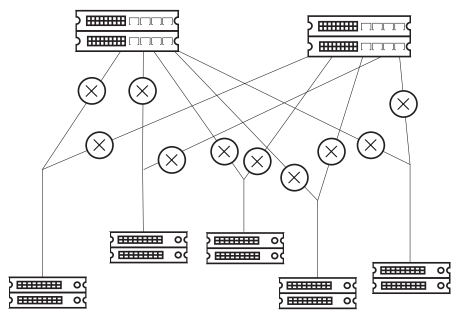

Fig. 1. Illustration of the wired setting. $\times$ denotes a network coding operation.

While coding for straggling nodes can provide an increased performance, nevertheless it suffers from two main limitations: i) an increased load assigned to each node (fraction $\frac{1}{L}$ instead of fraction $\frac{1}{K}$ of the dataset), and ii) a restriction to linear problems (with the notable extension to polynomial problems [11]) thus excluding other tasks such as sorting.

Stragglers in Coded MapReduce: It is interesting to note that the tools from the straggler mitigation literature have been imported in CMR to address straggling nodes [12], [13].

Wireless Coded MapReduce: Recent works on CMR have migrated from the assumption of the shared wired communication link, to also consider the equivalent shared wireless channel (or its equivalent wired network where intermediate nodes can perform linear operations (see Fig. 1)). This approach — while maintaining the fundamental shared-link limitation of the channel (i.e., while maintaining the exact same T_{shuf}) — has in some cases allowed for increased communication gains. For example, the work in [14] exploits the potential for full-duplex to double (for some parameters) the communication gains compared to the original CMR [2], while the work in [15] utilizes node-coordination among sets of some L nodes to increase communication gains by a factor of up to L in practical scenarios where the dataset size is bounded[1].

Overview of our Solution

In this work, we assume that nodes have heterogeneous computing capabilities during the mapping phase. Specifically, K_1 nodes (set $\mathcal{K}_1$) can perform mapping faster than the rest $K_2=K-K_1$ nodes (set $\mathcal{K}_2$), and as a result each node of the first type is tasked with mapping a fraction $\gamma_1 \in \left[\frac{1}{K_1}, 1\right]$ of the dataset, while each node of the second type is tasked with mapping a smaller fraction $\gamma_2 \in [0, \gamma_1)$. Our approach is applied to the CMR framework, under the assumption of a wireless channel between the nodes or its equivalent wired network with network-coding capabilities at the intermediate nodes and uses a solution that takes advantage of the fact that, while the rank of the channel to any one node is still the same (unit rank), the rank of the channel between two sets of nodes can be more than one.

Using this increased set-to-set dimensionality, we exploit the redundancy at all nodes and reduce the shuffling delay by a factor of $K_1\gamma_1 + K_2\gamma_2$ that matches exactly the cumulative computed redundancy. This solution will show that even though the system suffers from computational heterogeneity, it actually performs like an equivalent homogeneous system where each node could map a uniform fraction $\gamma = \frac{K_1\gamma_1+K_2\gamma_2}{K}$ of the dataset.

Data assignment uses a modification of the original CMR algorithm of [2]. The novel communication algorithm during the shuffling phase manages to consistently serve $K_1\gamma_1 + K_2\gamma_2$ nodes per transmission, a gain that appears either because a) nodes can transmit messages as if the two types were not interfering, or otherwise because b) nodes of set $\mathcal{K}_1$ can jointly transmit as if they were a single transmitter with $K_1\gamma_1$ antennas.

Setting Description

We assume K computing nodes that are tasked with calculating a total of $Q \geq K$ functions over dataset F, comprized of f elements. Specifically, node $k \in [K] \triangleq \{1, 2, ..., K\}$ will be responsible for calculating $\frac{Q}{K}$ functions $\{g_q,\ q \in Q_k \subset [Q]\}$.

To perform their tasks, nodes use the MR framework, in accordance to which it is assumed that function g_q, $q \in [Q]$ can be decomposed as $g_q(F) = r_q(m_q(F_1), ..., m_q(F_S))$, $\cup_{s=1}^{S} F_s = F$, where m_q and r_q represent, respectively, the mapping and the reduce functions.

The process commences with the *mapping phase*, where the dataset F is divided into S chunks, $F_1, ..., F_S$, which are distributed to the nodes, where each node is responsible for executing a set of mapping functions $\{m_q,\ \forall q \in [Q]\}$ on each of its assigned dataset chunks. The result of the mapping phase is the set of intermediate values $\{m_q(F_s), q \in [Q],\ s \in [S]\}$.

At the end of the mapping phase, all nodes need to communicate intermediate values guaranteeing that node $k \in [K]$ will gather $\mathcal{I}_q = \{m_q(F_s) \triangleq F_s^{(q)},\ s \in [S]\},\ \forall q \in Q_k$. We define as Category $q \in [Q]$ the collection of all intermediate values that, after reduction, can produce result $q \in [Q]$.

During the reduce phase the objective is for each node to perform functions r_q, $\forall q \in Q_k$ on the set $\mathcal{I}_q$ of intermediate values that it has gathered.

To reflect the heterogeneous capabilities of the computing nodes during the mapping phase – as stated before – we consider a set of K_1 computing nodes, $\{1, 2, ..., K_1\} \triangleq \mathcal{K}_1$, being able of mapping on time a fraction $\gamma_1 \in \left[\frac{1}{K}, 1\right]$ of the dataset, while the rest $K_2 = K - K_1$ nodes, $\{K_1+1, ..., K\} \triangleq \mathcal{K}_2$, are able to map a smaller fraction $\gamma_2 \in [0, \gamma_1)$ of the dataset.

The shuffling phase may take place over a wireless fully-connected channel or a wired, high-rank[2] network. During the

[1]This coordination requires no additional exchange of dataset information across the nodes (no dataset exchange overhead), and its gains are achieved by dramatically reducing the required number of subpackets of the original CMR approach where the coding gains came at the expense of having to split the dataset into a number of packets that grows exponentially with the gain. For a discussion on the impact of high subpacketization on the maximum coding gains that can be achieved under finite dataset sizes, see also [16].

[2]A wired setting can possess the same desirable properties of the wireless setting (see [15]) by having intermediary nodes perform linear operations on received data and then forward them (cf. Fig. I). The purpose of the linear operations is to form a high-rank matrix between transmitter and receiver sets. As will become clear later on, the matrix rank of the wired system needs to match the product $K_1\gamma_1$.

communication, if nodes in set $\tau \subset [K]$ are transmitting then the message at a receiving node $k \in [K] \setminus \tau$ takes the form

$$y_k = \sum_{m\in\tau} h_{m,k} x_m + w_k \tag{3}$$

where $h_{m,k} \in \mathbb{C}$ denotes the channel (or network coding) coefficient between nodes m and k, $\mathbb{E}(\|x_m\|^2) \leq P$, $\forall m \in \tau$ for some power constraint P, and $w_k \sim \mathbb{C}\mathcal{N}(0,1)$. We assume that during communication, channel state information at both the transmitters (CSIT) and receivers (CSIR) (or the network coding coefficients in the wired setting), are known with perfect[3] accuracy. Further, node cooperation will refer to clock synchronization, data selection and exchange of CSIT/network coding coefficients between the nodes and *not* intermediate values exchange.

II. Main Results

The main contribution of this work is the design of a new algorithm for dataset assignment, and a new transmission policy during the shuffling phase. The achievable performance for any values of γ_1, γ_2 yields

$$T_{\text{tot}}^{\text{hCMR}}(\gamma_1,\gamma_2) = \max\left\{T_{\text{map}}^{(1)}(\gamma_1 f), T_{\text{map}}^{(2)}(\gamma_2 f)\right\} + \frac{T_{\text{shuf}}}{K}\frac{1-\gamma_1}{\gamma_1} + \frac{T_{\text{shuf}}}{K}\frac{K_2\left(1-\frac{\gamma_2}{\gamma_1}\right)}{\min\{K_2, K_1\gamma_1+K_2\gamma_2\}} + \frac{Q}{K}T_{\text{red}} \tag{4}$$

where $T_{\text{map}}^{(j)}(\gamma_j F),\ j = 1,2$ corresponds to the time required for each node in group j to map fraction γ_j of the dataset, and where the delay reflects the fact that the shuffling phase will conclude when the mapping at any node has finished. In order to minimize Eq. (4), we choose values γ_1, γ_2 in such a way to reflect the relative computing capabilities[4] i.e., $T_{\text{map}}^{(1)}(\gamma_1 F) = T_{\text{map}}^{(2)}(\gamma_2 F)$. This is summarized in the following theorem.

Theorem 1. *The achievable time of a MapReduce algorithm in a heterogeneous distributed computing system with K_1 nodes each able to map fraction γ_1 of the total dataset, while K_2 nodes can map fraction γ_2 of the dataset each, takes the form*

$$T_{tot}^{hCMR}(\gamma_1,\gamma_2) = T_{map}^{(1)}(\gamma_1 f) + \frac{T_{shuf}}{K}\frac{1-\gamma_1}{\gamma_1} + \frac{T_{shuf}}{K}\frac{K_2\left(1-\frac{\gamma_2}{\gamma_1}\right)}{\min\{K_2, K_1\gamma_1+K_2\gamma_2\}} + \frac{Q}{K}T_{red} \tag{5}$$

where the communication (shuffling) cost can be simplified, in the case of $K_2 \geq K_1\gamma_1 + K_2\gamma_2$, to

$$T_{comm}^{hCMR}(\gamma_1,\gamma_2) = \frac{T_{shuf}}{K}\frac{K_1(1-\gamma_1)+K_2(1-\gamma_2)}{K_1\gamma_1+K_2\gamma_2} \tag{6}$$

Proof. The proof is constructive and presented in Sec. III. □

***Remark* 1.** *The effect of heterogeneity can be completely removed.*

[3]The feedback overhead can be expected to be very small because in realistic distributed computing settings, nodes are expected to have very low or no mobility.

[4]Thus, if nodes in $\mathcal{K}_1$ are twice as fast as nodes in $\mathcal{K}_2$, then $\gamma_1 = 2\gamma_2$.

The above holds directly from Eq. (6) where we see that the delay matches that of a homogeneous system with K nodes and a uniform $\gamma = \frac{K_1\gamma_1+K_2\gamma_2}{K}$.

***Remark* 2.** *Heterogeneous systems with interference are faster than multiple homogeneous systems even when the latter multiple systems experience no inter-system interference.*

This is because, by comparing the proposed solution with a second system where now the two sets $\mathcal{K}_1$ and $\mathcal{K}_2$ are 'magically' non-interfering (isolated), we conclude that while the mapping delay of the two systems would be the same, the shuffling phase delay $T_{\text{com}} = \frac{T_{\text{shuf}}}{K}\max\left\{\frac{1-\gamma_1}{\gamma_1}, \frac{1-\gamma_2}{\gamma_2}\right\} = \frac{T_{\text{shuf}}}{K}\frac{1-\gamma_2}{\gamma_2}$ of the second system would exceed that of our system. This reveals an important advantage of larger heterogeneous systems over multiple non-interfering homogeneous ones, and it suggests that the collaborative effects from mapping data to more nodes, despite heterogeneity, exceeds any effect of having parallel (smaller, homogeneous) systems. This surprising conclusion is mainly due to the fact that "strong" nodes have the potential to assist the transmission of intermediate values to the "weak" nodes, thus boosting the overall performance.

III. Scheme Description

Data Assignment: We begin by dividing the dataset into $S = \binom{K_1}{K_1\gamma_1}\cdot\binom{K_2}{K_2\gamma_2}$ chunks[5] F_{τ_1,τ_2}, $\tau_1 \subset \mathcal{K}_1$, $|\tau_1| = K_1\gamma_1$, $\tau_2 \subset \mathcal{K}_2$, $|\tau_2| = K_2\gamma_2$.

A. Mapping Phase

Nodes $k_1 \in \mathcal{K}_1, k_2 \in \mathcal{K}_2$ must respectively map[6]

$$\mathcal{M}_{k_1} = \{F_{\tau_1,\tau_2} : k_1 \in \tau_1,\ \forall \tau_2 \subset \mathcal{K}_2, |\tau_2| = K_2\gamma_2\}$$
$$\mathcal{M}_{k_2} = \{F_{\tau_1,\tau_2} : k_2 \in \tau_2,\ \forall \tau_1 \subset \mathcal{K}_1, |\tau_1| = K_1\gamma_1\}.$$

B. Shuffling Phase

After the mapping phase, the nodes exchange information so that each node $k \in [K]$ can collect the set

$$\mathcal{I}_k = \left\{F_{\tau_1,\tau_2}^{(q)},\ \tau_j \subseteq \mathcal{K}_j, |\tau_j| = K_j\gamma_j,\ j = \{1,2\},\ q \in Q_k\right\}$$

where in the above we used $F_{\tau_1,\tau_2}^{(q)} \triangleq m_q(F_{\tau_1,\tau_2})$.

We will employ a novel way of transmission, comprized of two sub-phases. The first sub-phase is reminiscent of the original CMR approach [2], where though now we will allow two nodes to transmit simultaneously, one node from each group; this can happen because we choose the transmitted messages to be completely known to the receiving nodes of the other group. During the second shuffling sub-phase, nodes from $\mathcal{K}_1$ will coordinate to act as a distributed multi-antenna system and transmit the remaining data to nodes of set $\mathcal{K}_2$.

[5]As is common in the literature, it is assumed that both $K_1\gamma_1$ and $K_2\gamma_2$ are integers, an assumption that does not interfere with the generality of the result, since in a different case memory-sharing can be performed, as in [4].

[6]The dataset fraction assigned to a node in each group yields

$$\frac{|\mathcal{M}_{k_i}|}{F} = \frac{\binom{K_i-1}{K_i\gamma_i-1}\cdot\binom{K_j}{K_j\gamma_j}}{\binom{K_i}{K_i\gamma_i}\cdot\binom{K_j}{K_j\gamma_j}} = \gamma_j,\ i,j \in \{1,2\}, i \neq j.$$

```
1 for all $\tau_1 \subset \mathcal{K}_1$, $|\tau_1| = K_1\gamma_1$ (pick Rx nodes from first set) do
2     for all $\tau_2 \subset \mathcal{K}_2$ , $|\tau_2| = K_2\gamma_2$ (pick Rx nodes from second set) do
3         for all $k_1 \in \mathcal{K}_1 \setminus \tau_1$ (pick transmitter 1) do
4             for all $k_2 \in \mathcal{K}_2 \setminus \tau_2$ (pick transmitter 2) do
5                 Transmit concurently:
                  Tx $k_1$: $x^{k_2,\tau_2}_{k_1,\tau_1} = \bigoplus_{m\in\tau_1} F^{(q_m),k_1,k_2}_{\{k_1\}\cup\tau_1\setminus\{m\},\tau_2}$
                  Tx $k_2$: $x^{k_1,\tau_1}_{k_2,\tau_2} = \bigoplus_{n\in\tau_2} F^{(q_n),k_1,k_2}_{\tau_1,\{k_2\}\cup\tau_2\setminus\{n\}}$
6             end
7         end
8     end
9 end
```

Algorithm 1: Shuffling Sub-Phase 1

1) First Shuffling Sub-Phase (Transmission): At the beginning of the first sub-phase, an intermediate value $F^{(q_k)}_{\tau_1,\tau_2}$, $q_k \in Q_k$ required by node $k \in [K]$, is further subpacketized as

$$F^{(q_k)}_{\tau_1,\tau_2} \rightarrow \begin{cases} \left\{F^{(q_k),k_1,k_2}_{\tau_1,\tau_2}, k_1 \in \tau_1,\ k_2 \in \mathcal{K}_2 \setminus \tau_2\right\} & k \in \mathcal{K}_1 \\ \left\{F^{(q_k),k_1,k_2}_{\tau_1,\tau_2}, k_1 \in \mathcal{K}_1 \setminus \tau_1,\ k_2 \in \tau_2\right\} & k \in \mathcal{K}_2 \end{cases} \tag{7}$$

yielding the sub-packet sizes

$$\frac{\left|F^{(q_k),k_1,k_2}_{\tau_1,\tau_2}\right|}{\left|F^{(q_k)}_{\tau_1,\tau_2}\right|} = \begin{cases} \dfrac{1}{K_1\gamma_1 K_2(1-\gamma_2)}, & \text{if } k \in \mathcal{K}_1 \\ \dfrac{1}{K_1(1-\gamma_1)K_2\gamma_2}, & \text{if } k \in \mathcal{K}_2 \end{cases} \tag{8}$$

meaning that sub-packets for nodes in set $\mathcal{K}_2$ are larger than sub-packets for nodes in set $\mathcal{K}_1$. To rectify this uneveness and allow for the simultaneous transmissions required by our algorithm, we will equalize the packet sizes transmitted from both $\mathcal{K}_1$ and $\mathcal{K}_2$ by further splitting the sub-packets intended for node $k \in \mathcal{K}_2$ into two parts, with respective sizes

$$\left|F'^{(q_k),k_1,k_2}_{\tau_1,\tau_2}\right| = \frac{\left|F^{(q_k)}_{\tau_1,\tau_2}\right|}{K_1\gamma_1 K_2(1-\gamma_2)} \tag{9}$$

$$\left|F''^{(q_k),k_1,k_2}_{\tau_1,\tau_2}\right| = \left|F^{(q_k)}_{\tau_1,\tau_2}\right| \frac{1-\gamma_2/\gamma_1}{K_1(1-\gamma_1)K_2(1-\gamma_2)\gamma_2}. \tag{10}$$

The process of transmitting packets in the first shuffling sub-phase is described in Algorithm 1, which successfully communicates a single category to each of the nodes.

In Algorithm 1, Steps 1 and 2 select, respectively, a subset of receiving nodes from set $\mathcal{K}_1$ and a subset of receiving nodes from set $\mathcal{K}_2$. Then, Steps 3 and 4 pick transmitting nodes from the sets $\mathcal{K}_1$ and $\mathcal{K}_2$, respectively such that these nodes are not in the already selected receiver sets from Steps 1 and 2. Then, in Step 5, the two selected transmitting nodes create and simultaneously transmit their packets.

Decoding in First Shuffling Sub-phase: In Algorithm 1, transmitted sub-packets for nodes of set $\mathcal{K}_1$ are picked to be known at all nodes of set $\mathcal{K}_2$, and vice versa. This allows nodes of a set to communicate as if not being interfered by nodes of the other set.

First Shuffling Sub-phase Performance: The above-described steps are arranged in nested loops that will, eventually, pick all possible combinations of $(k_1, k_2, \tau_1, \tau_2)$, thus delivering all needed sub-packets to nodes in $\mathcal{K}_1$. The normalized completion time of this first sub-phase is

$$T_{\text{comm},1} = \frac{Q}{K}\frac{K_1(1-\gamma_1)\binom{K_1}{K_1\gamma_1}K_2(1-\gamma_2)\binom{K_2}{K_2\gamma_2}}{K_1\gamma_1 K_2(1-\gamma_2)\binom{K_1}{K_1\gamma_1}\binom{K_2}{K_2\gamma_2}} = \frac{Q}{K}\frac{1-\gamma_1}{\gamma_1}$$

where factor $\frac{Q}{K}$ reflects a repetition of Algorithm 1 needed for delivering all categories, where the numerator reflects the four loops of the algorithm, and where the denominator uses the subpacketization level S and Eq. (8) to reflect the size of each transmitted packet.

2) Second Shuffling Sub-Phase: The second part of the shuffling phase initiates when the data required by the nodes in $\mathcal{K}_1$ have all been delivered. The transmission during this second sub-phase follows the ideas of the distributed antenna cache-aided literature (cf. [16]–[19]). Specifically, nodes of set $\mathcal{K}_1$ act as a distributed $K_1\gamma_1$-multi-antenna[7] system, while nodes of set $\mathcal{K}_2$ act as cache-aided receivers, cumulatively "storing" $K_2\gamma_2$ of the total "library". Directly from [16]–[19], we can conclude that in our setting, this setup allows a total of $\min\{K_2,\ K_1\gamma_1 + K_2\gamma_2\}$ nodes to decode successfully a desired intermediate value per transmission. The remaining messages to be sent during this sub-phase are

$$\left\{F''^{(q_k),k_1,k_2}_{\tau_1,\tau_2},\ k_1 \in \mathcal{K}_1 \setminus \tau_1,\ k_2 \in \tau_2, \forall\tau_1, \forall\tau_2\right\}.$$

By combining the remaining messages of category q_k i.e., $\{F''^{(q_k),k_1,k_2}_{\tau_1,\tau_2}\ \forall k_1 \in \mathcal{K}_1 \setminus \tau_1,\ \forall k_2 \in \mathcal{K}_2\} \rightarrow F''^{(q_k)}_{\tau_1,\tau_2}$, we have

$$\left|F''^{(q_k)}_{\tau_1,\tau_2}\right| = \frac{1-\gamma_2/\gamma_1}{1-\gamma_2}\left|F''^{(q_k),k_1,k_2}_{\tau_1,\tau_2}\right|, \quad \forall\tau_1,\tau_2. \tag{11}$$

With this in place, the delay of the second subphase is

$$\begin{aligned} T_{\text{comm},2} &= \overbrace{\frac{Q}{K}}^{t_2}\ \overbrace{\frac{1-\gamma_2/\gamma_1}{(1-\gamma_2)}}^{t_3}\ \underbrace{\frac{\overbrace{K_2}^{t_4}\ \overbrace{(1-\gamma_2)}^{t_5}}{\min\{K_2,\ K_2\gamma_2 + K_1\gamma_1\}}}_{t_1} \\ &= \frac{Q}{K}\frac{K_2(1-\gamma_2/\gamma_1)}{\min\{K_2,\ K_2\gamma_2 + K_1\gamma_1\}} \end{aligned} \tag{12}$$

where term t_1 corresponds to the aforementioned performance (in number of nodes served per transmission) that is achieved by multi-antenna Coded Caching algorithms [16]–[19]. In the above, the term t_2 corresponds to a repetition of the scheme to deliver multiple categories to each node. Finally terms t_3, t_4, t_5 represent the total amount of information that needs to be transmitted, where t_3 represents the size of each category, t_4 the number of nodes and t_5 corresponding to the (remaining) part that each node had not computed and needed to receive.

[7]We note to the reader that at this point is where the notion of the high-rank matrix is required in the equivalent wired system.

C. Reduce Phase

At this point, each node has the required intermediate values to perform reduction in a completely parallel way.

IV. Example

Let us consider a setting where the two sets are comprised of $K_1 = 3$ and $K_2 = 4$ nodes, and where any node in each group respectively maps fractions $\gamma_1 = \frac{2}{3}$ and $\gamma_2 = \frac{1}{2}$ of the dataset. The goal is the calculation of a total of $Q = 7$ functions. Initially, dataset F is divided into $S = \binom{3}{2}\binom{4}{2} = 18$ chunks, each defined by two indices[8], $\tau_1 \in \{12, 13, 23\}$ and $\tau_2 \in \{45, 46, 47, 56, 57, 67\}$. Node $k \in [7]$ is tasked with mapping F_{τ_1,τ_2} iff $k \in \tau_1 \cup \tau_2$.

Following the mapping phase, each chunk, F_{τ_1,τ_2}, has $Q = 7$ associated intermediate values $F^{(1)}_{\tau_1,\tau_2}, ..., F^{(7)}_{\tau_1,\tau_2}$, where each intermediate value $F^{(q_k)}_{\tau_1,\tau_2}$, is further divided into sub-packets according to Eq. (7), while those sub-packets meant for $\mathcal{K}_2$, are further divided according to Eq. (8)-(10). For example chunks $C_{12,45}$, $D_{12,56}$ are split as

$$C_{12,45} \rightarrow \{C^{1,6}_{12,45}, C^{1,7}_{12,45}, C^{2,6}_{12,45}, C^{2,7}_{12,45}\}$$
$$D_{12,56} \rightarrow \{D^{3,5}_{12,56}, D^{3,6}_{12,56}\} \rightarrow \{D'^{3,5}_{12,56}, D''^{3,5}_{12,56}, D'^{3,6}_{12,56}, D''^{3,6}_{12,56}\}.$$

Directly from Algorithm 1, the first 4 sets of transmissions that correspond to shuffling sub-phase 1 are

$$\begin{aligned}
x^{4,56}_{1,23} &= B^{1,4}_{13,56} \oplus C^{1,4}_{12,56}, & x^{1,23}_{4,56} &= E^{1,4}_{23,46} \oplus F^{1,4}_{23,45}\\
x^{4,57}_{1,23} &= B^{1,4}_{13,57} \oplus C^{1,4}_{12,57}, & x^{1,23}_{4,57} &= E^{1,4}_{23,47} \oplus G^{1,4}_{23,45}\\
x^{4,67}_{1,23} &= B^{1,4}_{13,67} \oplus C^{1,4}_{12,67}, & x^{1,23}_{4,67} &= F^{1,4}_{23,47} \oplus G^{1,4}_{23,46}\\
x^{5,46}_{1,23} &= B^{1,5}_{13,46} \oplus C^{1,5}_{12,46}, & x^{1,23}_{5,46} &= D^{1,5}_{23,56} \oplus F^{1,5}_{23,45}
\end{aligned}$$

During shuffling sub-phase 2, where now all information for the first set of nodes has been delivered, the first set of nodes will use their computed intermediate values so as to speed up the transmission of the remaining intermediate values toward the second set of nodes. To simplify the second shuffling sub-phase, we will combine together sub-parts of the same upper indices i.e., $\{F''^{k_1,k_2}_{\tau_1,\tau_2}, \forall k_1 \in \mathcal{K}_1 \backslash \tau_1, \forall k_2 \in \tau_2\} \rightarrow F_{\tau_1,\tau_2}$.

Using a scheme similar to that of [16] and denoting with $H^{-1}_{\tau,\mu}$ the normalized inverse of the channel matrix between transmitting nodes in set τ and receiving nodes in set μ, and with $\mathbf{h}_{\tau,k}$ the channel vector from nodes in set τ to node k, then the 9 transmissions that deliver all remaining data are

$$x^{\tau}_{45,67} = H^{-1}_{\tau,45}\begin{bmatrix} D_{\tau,67}\\ E_{\tau,67}\end{bmatrix} + H^{-1}_{\tau,67}\begin{bmatrix} F_{\tau,45}\\ G_{\tau,45}\end{bmatrix} \quad \tau \in \{12, 13, 23\}$$

$$x^{\tau}_{46,57} = H^{-1}_{\tau,46}\begin{bmatrix} D_{\tau,57}\\ F_{\tau,57}\end{bmatrix} + H^{-1}_{\tau,57}\begin{bmatrix} E_{\tau,46}\\ G_{\tau,46}\end{bmatrix} \quad \tau \in \{12, 13, 23\}$$

$$x^{\tau}_{47,56} = H^{-1}_{\tau,47}\begin{bmatrix} D_{\tau,56}\\ G_{\tau,56}\end{bmatrix} + H^{-1}_{\tau,56}\begin{bmatrix} E_{\tau,47}\\ F_{\tau,47}\end{bmatrix} \quad \tau \in \{12, 13, 23\}.$$

The message arriving at each receiving node takes the form of a linear combination of intermediate values. In order for a node to decode its desired intermediate value, first we can see that matrix $H^{-1}_{\tau,\mu}$ is designed in such a way so as to separate the transmitted messages at the two nodes, i.e.,

$$\mathbf{h}_{\tau,k\in\mu} \cdot H^{-1}_{\tau,\mu} \cdot [W^1, ..., W^k, ..., W^n]^T = W^k.$$

For example, Node 4 will receive, in the first transmission,

$$y^{12}_{45,67}(k = 4) = D_{12,67} + \mathbf{h}_{12,6} \cdot H^{-1}_{12,67}\begin{bmatrix} F_{12,45}\\ G_{12,45}\end{bmatrix}$$

and then, using CSIR and its computed intermediate values $F_{12,45}$ and $G_{12,45}$, Node 4 will decode its desired intermediate value $D_{12,67}$.

[8]For notational simplicity, for sets we use, for example $\{12\}$ instead of $\{1, 2\}$. We also rename as $A_{\tau_1,\tau_2} \triangleq F^{(1)}_{\tau_1,\tau_2}$, $B_{\tau_1,\tau_2} \triangleq F^{(2)}_{\tau_1,\tau_2}, ..., G_{\tau_1,\tau_2} \triangleq F^{(7)}_{\tau_1,\tau_2}$ and also $F'^{(q_k)}_{\tau_1,\tau_2}$ is denoted with $F^{(q_k)}_{\tau_1,\tau_2}$, since the corresponding transmission (either in the first or second sub-phase) is clearly associated with each of the two parts.

V. Conlusions

We studied the performance of a heterogeneous Coded MapReduce system and showed that mapping heterogeneity does not slow down the execution, and that rather, having slower nodes contributes to a further reduction of the completion time.

References

[1] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," *Communications of the ACM*, 2008.

[2] S. Li, M. A. Maddah-Ali, Q. Yu, and A. S. Avestimehr, "A fundamental tradeoff between computation and communication in distributed computing," *IEEE Trans. on Inf. Theory*, 2018.

[3] S. Li, M. A. Maddah-Ali, and A. S. Avestimehr, "Compressed coded distributed computing," in *Int. Symp. on Inf. Theory (ISIT)*, June 2018.

[4] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *IEEE Transactions on Information Theory*, 2014.

[5] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Speeding up distributed machine learning using codes," *IEEE Trans. on Inf. Theory*, 2018.

[6] R. Tandon, Q. Lei, A. G. Dimakis, and N. Karampatziakis, "Gradient coding," *arXiv preprint arXiv:1612.03301*, 2016.

[7] Q. Yu, M. Maddah-Ali, and S. Avestimehr, "Polynomial codes: an optimal design for high-dimensional coded matrix multiplication," in *Advances in Neural Information Processing Systems*, 2017.

[8] K. Lee, C. Suh, and K. Ramchandran, "High-dimensional coded matrix multiplication," in *Int. Symp. on Inf. Theory (ISIT)*, IEEE, 2017.

[9] A. B. Das, L. Tang, and A. Ramamoorthy, "C3LES: Codes for coded computation that leverage stragglers," *arXiv preprint arXiv:1809.06242*, 2018.

[10] C. Suh and K. Ramchandran, "Exact-repair MDS codes for distributed storage using interference alignment," in *Int. Symp. on Inf. Theory (ISIT)*, June 2010.

[11] Q. Yu, N. Raviv, J. So, and A. S. Avestimehr, "Lagrange coded computing: Optimal design for resiliency, security and privacy," *arXiv preprint arXiv:1806.00939*, 2018.

[12] S. Li, M. A. Maddah-Ali, and A. S. Avestimehr, "A unified coding framework for distributed computing with straggling servers," in *IEEE Globecom Workshops (GC Wkshps)*, Dec 2016.

[13] J. Zhang and O. Simeone, "Improved latency-communication trade-off for map-shuffle-reduce systems with stragglers," *arXiv preprint arXiv:1808.06583*, 2018.

[14] F. Li, J. Chen, and Z. Wang, "Wireless MapReduce distributed computing," in *Int. Symp. on Inf. Theory (ISIT)*, June 2018.

[15] E. Parrinello, E. Lampiris, and P. Elia, "Coded distributed computing with node cooperation substantially increases speedup factors," in *International Symposium on Information Theory (ISIT)*, June 2018.

[16] E. Lampiris and P. Elia, "Adding transmitters dramatically boosts coded-caching gains for finite file sizes," *J. on Selec. Areas in Comm.*, 2018.

[17] N. Naderializadeh, M. A. Maddah-Ali, and A. S. Avestimehr, "Fundamental limits of cache-aided interference management," *IEEE Transactions on Information Theory*, 2017.

[18] E. Lampiris and P. Elia, "Achieving full multiplexing and unbounded caching gains with bounded feedback resources," in *International Symposium on Information Theory (ISIT)*, IEEE, 2018.

[19] S. P. Shariatpanahi, S. A. Motahari, and B. H. Khalaj, "Multi-server Coded Caching," *IEEE Transactions on Information Theory*, 2016.