Coded Matrix Multiplication on a Group-Based Model

Muah Kim
School of Electrical Engineering
KAIST
Daejeon, Republic of Korea
02mu-a21@kaist.ac.kr

Jy-yong Sohn
School of Electrical Engineering
KAIST
Daejeon, Republic of Korea
jysohn1108@kaist.ac.kr

Jaekyun Moon
School of Electrical Engineering
KAIST
Daejeon, Republic of Korea
jmoon@kaist.edu

***Abstract*—Coded distributed computing has been considered as a promising technique which makes large-scale systems robust to the "straggler" workers. Yet, practical system models for distributed computing have not been available that reflect the clustered or grouped structure of real-world computing servers. Also, the large variations in the computing power and bandwidth capabilities across different servers have not been properly modeled. We suggest a *group-based model* to reflect practical conditions and develop an appropriate coding scheme for this model. The suggested code, called *group code*, employs parallel encoding for each group. We show that the suggested coding scheme can asymptotically achieve optimal computing time in the regime of infinite n, the number of workers. While theoretical analysis is conducted in the asymptotic regime, numerical results also show that the suggested scheme achieves near-optimal computing time for any finite but reasonably large n. Moreover, we demonstrate that decoding complexity of the suggested scheme is significantly reduced by the virtue of parallel decoding.**

A full version of this paper is accessible at: https://arxiv.org/abs/1901.05162

I. Introduction

In the era of big data, distributed computing has been recognized as a solution for realizing large-scale machine learning [1]. Unlike conventional centralized systems, a distributed computing system divides the computational work into subtasks and distributes them over multiple nodes. This system successfully supports large-scale machine learning by reducing the computing time via parallel computing.

Yet, there is still a significant room for improvement as the system is slowed down by the random nature of computing nodes, where certain nodes are inevitably slower than others. In particular, the distributed system is shown to be dramatically degraded by the slowest workers, the "stragglers", whose computational latency is realized by the tail probability [2]. *Lee* et al. suggested coded computation as a straggler-proof scheme, which speeds up matrix multiplication by employing redundancy with a maximum distance separable (MDS) code [3]. Since then it has been shown that coded computation can effectively improve the performance of computing system with regards to: matrix-matrix multiplication [4]–[6], distributed gradient descent [7], [8], convolution [9], Fourier transform [10], and matrix sparsification [11], [12]. Moreover, regarding the matrix multiplication, new models reflecting the practical environment of computing systems such as the tree structure and heterogeneity are suggested and analyzed [13], [14].

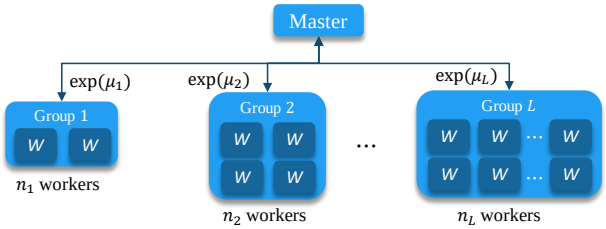

Figure 1: Computing Network Model: an $(\boldsymbol{n}, \boldsymbol{\mu})-$group system with L groups. Group i has n_i workers having i.i.d. completion time distribution with statistical parameter μ_i for $i \in [L]$.

In recent years, distributed cloud computing services such as Amazon EC2 have enabled customers to deal with large-scale computation [15]. The real distributed computing systems generally adopt the multi-rack structure, where the computing workers are grouped together in multiple racks [16]–[18]. Moreover, in the real world, the workers' latency statistics are heterogeneous due to a mixed use of hardwares with varying performances or the dynamics of multiple user requests over shared resources [19]. So far, the homogeneous grouped structure has been considered in [14], and the heterogeneous workers without grouped feature have been studied in [13]. However, system solutions which reflect both of the two practical conditions−grouped structure and heterogeneity (in terms of number of workers in each group as well as the bandwidth of the communication links associated with the groups)−are yet to be established.

A. Main Contributions

We design a group-based computing model as shown in Fig. 1, where n workers are dispersed into L groups, each having a different number of nodes and distinct computing time statistics. We assume that group i has n_i nodes, each of which has a computing time given by an exponential random variable with rate μ_i. This is a more practical model than the existing ones because it resembles the tree-shaped (grouped) distributed computing systems such as the Hadoop file system while also considering the heterogeneity of the groups.

Considering the scenario of computing k tasks in the suggested model, we show that an $(n, k)-$MDS code achieves the optimal computing time. Yet, this scheme requires a prohibitive decoding complexity as k increases. In addition,

it is hard to obtain a closed-form expression for the optimal computing time due to the heterogeneous nature of the model.

To address these issues, we propose a coding scheme called *group code* which divides the total k tasks into L partitions and then employs L distinct MDS codes. We show that a carefully designed group code can asymptotically achieve the optimal computing time as n goes to infinity. In addition, the suggested group code can reduce the decoding complexity by a factor of $(\frac{1}{L})^{\beta}$ compared to an $(n,k)-$MDS code, where $\beta > 1$. Furthermore, we obtain a closed-form expression for the expected optimal computing time, when the number of workers n goes to infinity.

B. Related Works

Previous works on coded computation either achieve the optimal computing time with a prohibitive decoding complexity, or reduce the decoding complexity at the cost of the optimality loss in terms of computing time. In addition, most of them assume homogeneous workers. Applying an $(n,k)-$MDS code in homogeneous systems is suggested by [3], which achieves the optimal computing time but requires a large decoding complexity as k increases. Considering a system model with heterogeneous workers, the authors of [13] suggested a coding scheme which achieves an asymptotically optimal computing time. However, the decoding process requires the computational complexity level of $\mathcal{O}(k^3)$. On the other hand, the coding schemes suggested in [4], [6], [14] encode the tasks along multiple dimensions, which can effectively reduce the decoding complexities by the virtue of parallel decoding or using a peeling decoder. However, these codes lose the MDS property and thereby cannot achieve the optimal computing time. Besides, these codes do not provide solutions for practical systems with heterogeneous groups. Compared to these existing works, our suggested scheme is shown to not only asymptotically achieve the optimal computing time, but also requires a low decoding complexity.

C. Notations

Here, we list mathematical notations used in this paper. For a positive integer n, a set of positive integers less than or equal to n is denoted by $[n] = \{1, 2, \dots, n\}$. For a matrix $\mathbf{A}$ with multiple rows, $\mathbf{A} = [\mathbf{A}_1; \mathbf{A}_2]$ represents row-wise division of $\mathbf{A}$, i.e. $\mathbf{A}^T = [\mathbf{A}_1^T \mathbf{A}_2^T]$. We use $C_G(\boldsymbol{n}, \boldsymbol{k})$ to denote an $(\boldsymbol{n}, \boldsymbol{k})-$group code and $C_{\mathrm{MDS}}(n,k)$ to denote an $(n,k)-$MDS code. The definition of $(\boldsymbol{n}, \boldsymbol{k})-$group code is in Section II-A.

II. System Model and Target Problem

A. System Model

Consider the n workers that are spread into L groups as shown in Fig. 1. Here, group i has n_i workers whose response times are described by i.i.d. random variables with a parameter μ_i. We define this system as an $(\boldsymbol{n}, \boldsymbol{\mu})-$*group system*, where $\boldsymbol{n} = [n_1, n_2, \dots, n_L], \boldsymbol{\mu} = [\mu_1, \mu_2, \dots, \mu_L]$. For simplicity, we call the j^{th} worker in group i as $w(i,j)$ for $i \in [L]$ and $j \in [n_i]$. We implement a matrix-vector multiplication $\mathbf{A}\boldsymbol{x}$ on this system, where $\mathbf{A} \in \mathbb{R}^{m \times d}$ is a work matrix, and $\boldsymbol{x} \in \mathbb{R}^{d \times 1}$ is an input vector for some positive integers m and d. The work matrix $\mathbf{A}$ is divided into equal-sized k submatrices as $\mathbf{A} = [\mathbf{A}_1; \mathbf{A}_2; \cdots; \mathbf{A}_k]$, where k is a positive integer that can divide m, and $\mathbf{A}_r \in \mathbb{R}^{\frac{m}{k} \times d}$ for $r \in [k]$.

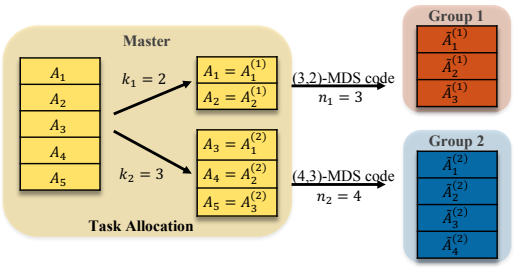

Figure 2: Illustration of $(\boldsymbol{n}, \boldsymbol{k}) = ([3,4],[2,3])-$group code. The matrix is split into two submatrices following the task allocation vector $\boldsymbol{k} = [2,3]$, and then the submatrices are encoded with MDS code group-wise.

The task of computing $\mathbf{A}\boldsymbol{x}$ is distributed to n workers as follows. First, we define $\boldsymbol{k} = [k_1, k_2, \dots, k_L]$ as a *task allocation* vector, where the elements are positive integers satisfying $\sum_{i=1}^{L} k_i = k$. The set of submatrices $\{\mathbf{A}_r\}_{r=1}^{k}$ is now partitioned into L disjoint subsets $\{\mathbb{S}_i\}_{i=1}^{L}$ such that $|\mathbb{S}_i| = k_i$ holds for $i \in [L]$. We denote the elements in set $\mathbb{S}_i$ as $\mathbb{S}_i = \{\mathbf{A}_j^{(i)}\}_{j=1}^{k_i}$. Afterwards, the k_i elements of $\mathbb{S}_i$ are encoded with an $(n_i, k_i)-$MDS code and we denote the set of n_i coded submatrices by $\widetilde{\mathbb{S}}_i = \{\widetilde{\mathbf{A}}_j^{(i)}\}_{j=1}^{n_i}$. Worker $w(i,j)$ now stores $\widetilde{\mathbf{A}}_j^{(i)}$ and computes $\widetilde{\mathbf{A}}_j^{(i)}\boldsymbol{x}$ when it receives the input vector $\boldsymbol{x}$ from the master. We call this coding scheme as an $(\boldsymbol{n}, \boldsymbol{k})-$*group code*, denoted by $C_{\mathrm{G}}(\boldsymbol{n}, \boldsymbol{k})$. Fig. 2 illustrates an example of an $(\boldsymbol{n}, \boldsymbol{k})-$group code when $\boldsymbol{n} = [3,4]$ and $\boldsymbol{k} = [2,3]$. The matrix $\mathbf{A} = [\mathbf{A}_1; \mathbf{A}_2; \dots; \mathbf{A}_5]$ is divided into two sets of submatrices, $\{\mathbf{A}_1, \mathbf{A}_2\}$ and $\{\mathbf{A}_3, \mathbf{A}_4, \mathbf{A}_5\}$. Then, by applying a $(3,2)-$MDS code and a $(4,3)-$MDS code, respectively, we obtain $\{\widetilde{\mathbf{A}}_1^{(1)}, \widetilde{\mathbf{A}}_2^{(1)}, \widetilde{\mathbf{A}}_3^{(1)}\}$ and $\{\widetilde{\mathbf{A}}_1^{(2)}, \widetilde{\mathbf{A}}_2^{(2)}, \widetilde{\mathbf{A}}_3^{(2)}, \widetilde{\mathbf{A}}_4^{(2)}\}$. Each worker individually transmits its computational result $\widetilde{\mathbf{A}}_j^{(i)}\boldsymbol{x}$ to the master when its computation is finished. To obtain the computational output $\mathbf{A}\boldsymbol{x}$, the master needs at least k_i computational results from each group i to decode the $(n_i, k_i)-$MDS code. Note that this model can be directly applied to the matrix-matrix multiplication, where the input vector $\boldsymbol{x}$ is replaced by a matrix $\mathbf{B} \in \mathbb{R}^{d \times c}$.

We adopt the exponential distribution model for the *completion time* of a worker, which is defined as the time taken for both computation and transmission of the computed result to the master. The exponential distribution closely reflects the runtime of an actual distributed computing system, as shown in [3]. This model has also been used in other papers on coded computation [4], [14]. Unlike these papers, however, a worker in group i has the distribution parameter μ_i, where μ_i varies among different groups. More precisely, the completion time $T_j^{(i)}$ of worker $w(i,j)$ is defined by its cumulative distribution function as $\Pr[T_j^{(i)} \leq t] = 1 - e^{k\mu_i t}$ for time $t \geq 0$. Here, the completion time has the rate of $k\mu_i$ since the number of rows in the submatrix $\mathbf{A}_r \in \mathbb{R}^{\frac{m}{k} \times d}$ becomes smaller as k increases.

B. Target Problem

This paper mainly aims at analyzing the total execution time T_{exec} of $(\boldsymbol{n}, \boldsymbol{k})-$group codes, which refers to the entire time taken for computing and decoding. The computing time T_{comp} is the time taken for the master to gather computational subtasks from the workers, while the decoding time T_{dec} is the time taken to recover the original task of computing $\mathbf{A}\boldsymbol{x}$ from the gathered subtasks. In this paper, we assume that the encoding time complexity is negligible compared to T_{comp} and T_{dec}. This is because we focus on the scenarios of multiplying varying input vectors with the same work matrix $\mathbf{A}$, which is encoded once prior to the computation. Thus, we have

$$T_{\text{exec}}(C) = T_{\text{comp}}(C) + T_{\text{dec}}(C),$$

when code C is applied to the system.

We focus on analyzing the computing time of $(\boldsymbol{n}, \boldsymbol{k})-$group codes, which is denoted by $T_{\text{comp}}(C_{\text{G}}(\boldsymbol{n}, \boldsymbol{k}))$. Recall that the computing time of an $(\boldsymbol{n}, \boldsymbol{k})-$group code is equivalent to the time when every group i has at least k_i workers which finish their tasks. Let $T^{(i)}_{k_i:n_i}$ be the k_i^{th} smallest value among $\{T_j^{(i)}\}_{j=1}^{n_i}$. Then, $T_{\text{comp}}(C_{\text{G}}(\boldsymbol{n}, \boldsymbol{k}))$ can be expressed as

$$T_{\text{comp}}(C_{\text{G}}(\boldsymbol{n}, \boldsymbol{k})) = \max(T^{(1)}_{k_1:n_1}, T^{(2)}_{k_2:n_2}, \ldots, T^{(L)}_{k_L:n_L}).$$

Since it is hard to find a closed-form expression for $\mathbb{E}[T_{\text{comp}}(C_{\text{G}}(\boldsymbol{n}, \boldsymbol{k}))]$ when n is finite, we set our main problem as that of obtaining the expected value as n goes to infinity, i.e.

$$\mathbb{P}_{\text{main}} : \text{compute} \lim_{n\to\infty} \mathbb{E}[T_{\text{comp}}(C_{\text{G}}(\boldsymbol{n}, \boldsymbol{k}))].$$

Here, we assume $k = \Theta(n)$ and $n_i = \Theta(n)$ for $i \in [L]$.

III. Optimal Computing Time Analysis

Here we find the optimal computing time of a given $(\boldsymbol{n}, \boldsymbol{\mu})-$group system. Theorem 1 states that applying an $(n, k)-$MDS code achieves the optimal computing time. We assume that an $(n, k)-$MDS code is applied to the k submatrices $\{\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_k\}$, resulting in n coded submatrices $\{\widetilde{\mathbf{A}}_1, \widetilde{\mathbf{A}}_2, \ldots, \widetilde{\mathbf{A}}_n\}$. Then, the n coded submatrices are distributed to n workers regardless of the groups they belong. Here we denote the computing time of an $(n, k)-$MDS code as $T_{\text{comp}}(C_{\text{MDS}}(n, k))$.

Theorem 1. *Consider computing k tasks on an $(\boldsymbol{n}, \boldsymbol{\mu})-$group systems. Then, an $(n, k)-$MDS code achieves the optimal computing time. In other words, for arbitrary (n, k) linear code $C \in \mathcal{C}(n, k)$,*

$$T_{\text{comp}}(C_{\text{MDS}}(n, k)) \leq T_{\text{comp}}(C).$$

Proof. Given an arbitrary realization of the completion times $\{T_j^{(i)}\}_{i\in[L], j\in[n_i]}$ of workers, we can think of their order statistics $T_{1:n} < T_{2:n} < \cdots < T_{n:n}$. Recall that (n, k) linear code C cannot recover the original message if there are more than $n - k$ erasures, which leads to $T_{\text{comp}}(C) \geq T_{k:n}$. By the MDS property, we have $T_{\text{comp}}(C_{\text{MDS}}(n, k)) = T_{k:n}$, which completes the proof. □

IV. Computing Time Analysis

In this section, we provide the computing time analysis when the workers are dispersed into $L = 2$ groups. The computing time for general L can be analyzed by the mathematical induction, which is given in [20] due to the lack of space.

A. Computing Time for an Arbitrary Task Allocation $\boldsymbol{k}$

For simplicity, we denote the task allocation vector as $\boldsymbol{k} = [k_1, k_2] = [k_1, k - k_1]$. The computing time of an $(\boldsymbol{n}, \boldsymbol{k})-$group code for $L = 2$ can be expressed as $T_{\text{comp}}(C_{\text{G}}(\boldsymbol{n}, \boldsymbol{k})) = \max(T^{(1)}_{k_1:n_1}, T^{(2)}_{k_2:n_2})$ by definition. Lemma 1 provides the expected computing time of an $(\boldsymbol{n}, \boldsymbol{k})-$group code when n goes to infinity.

Lemma 1. *Consider an $(\boldsymbol{n}, \boldsymbol{\mu})-$group system with $L = 2$ groups. Then, the expected computing time of an $(\boldsymbol{n}, \boldsymbol{k})-$group code satisfies the following:*

$$\begin{aligned}\lim_{n\to\infty} \mathbb{E}[T_{\text{comp}}(C_{\text{G}}(\boldsymbol{n}, \boldsymbol{k}))] &= \lim_{n\to\infty} \mathbb{E}[\max(T^{(1)}_{k_1:n_1}, T^{(2)}_{k_2:n_2})] \\ &= \max(\lim_{n\to\infty} \mathbb{E}[T^{(1)}_{k_1:n_1}], \lim_{n\to\infty} \mathbb{E}[T^{(2)}_{k_2:n_2}]) \\ &= \max\left(-\frac{1}{k\mu_1}\log(1 - \frac{k_1}{n_1}), -\frac{1}{k\mu_2}\log(1 - \frac{k_2}{n_2})\right). \quad (1)\end{aligned}$$

Proof. We set aside the proof in [20]. □

This lemma illustrates that in the asymptotic regime of large n, the expected computing time of an $(\boldsymbol{n}, \boldsymbol{k})-$group code can be easily obtained for given $\boldsymbol{n}$, $\boldsymbol{\mu}$ and k.

B. Optimizing the Task Allocation of a Group Code

Now, we aim at optimizing task allocation rule $\boldsymbol{k}$ which minimizes the computing time of an $(\boldsymbol{n}, \boldsymbol{k})-$group code. We define the optimal task allocation vector by

$$\boldsymbol{k}^* := \arg\min_{\boldsymbol{k}} \mathbb{E}[T_{\text{comp}}(C_{\text{G}}(\boldsymbol{n}, \boldsymbol{k}))], \quad (2)$$

whose elements are denoted by $\boldsymbol{k}^* = [k_1^*, k_2^*, \ldots, k_L^*]$. Before finding $\boldsymbol{k}^*$, we state a relationship between $T_{k:n}$ and $\{T^{(i)}_{k_i:n_i}\}_{i=1}^L$ in the following Lemma. Recall that $T_{k:n}$ is equivalent to $T_{\text{comp}}(C_{\text{MDS}}(n, k))$, and the maximum among $\{T^{(i)}_{k_i:n_i}\}_{i=1}^L$ corresponds to $T_{\text{comp}}(C_{\text{G}}(\boldsymbol{n}, \boldsymbol{k}))$ by definition.

Lemma 2. *Under the scenario of computing k tasks on an $(\boldsymbol{n}, \boldsymbol{\mu})-$group system with $L = 2$ groups, consider applying an $(\boldsymbol{n}, \boldsymbol{k})-$group code where $\boldsymbol{n} = [n_1, n_2]$ and $\boldsymbol{k} = [k_1, k - k_1]$. Given an arbitrary realization of completion time $\{T_j^{(i)}\}_{i\in[2], j\in[n_i]}$ of workers, let $T_{k:n}$ be the k^{th} smallest value among $\{T_j^{(i)}\}_{i\in[2], j\in[n_i]}$. Meanwhile, $T^{(i)}_{k_i:n_i}$ denotes the k_i^{th} smallest value among $\{T_j^{(i)}\}_{j=1}^{n_i}$. Then, we have*

$$\min(T^{(1)}_{k_1:n_1}, T^{(2)}_{k-k_1:n_2}) \leq T_{k:n} \leq \max(T^{(1)}_{k_1:n_1}, T^{(2)}_{k-k_1:n_2}). \quad (3)$$

Proof. Let $\mathbb{U} = \{T_j^{(i)} : T_j^{(i)} \leq T_{k:n} \text{for} i \in [2], j \in [n_i]\}$. Consider a subset $\mathbb{U}_1$ of set $\mathbb{U}$ such that $\mathbb{U}_1 = \{T_j^{(1)} : T_j^{(1)} \leq T_{k:n} \text{for} j \in [n_1]\}$ and its complementary set $\mathbb{U}_1^C = \{T_j^{(2)} : T_j^{(2)} \leq T_{k:n} \text{for} j \in [n_2]\}$. Here, we define $k_1' := |\mathbb{U}_1|$. Notice that $|\mathbb{U}_1^C| = k - k_1'$. Then, we may write $T^{(2)}_{k-k_1'-1:n_2} < T_{k:n} < T^{(1)}_{k_1'+1:n_1}$. When $k_1' < k_1$, we have $T_{k:n} < T^{(1)}_{k_1'+1:n_1} \leq T^{(1)}_{k_1:n_1}$. Similarly, we have $T_{k:n} > T^{(2)}_{k-k_1'-1:n_2} \geq T^{(2)}_{k-k_1:n_2}$, which

leads to $T^{(2)}_{k-k_1:n_2} \leq T_{k:n} \leq T^{(1)}_{k_1:n_1}$. When $k_1' > k_1$, we have $T^{(1)}_{k_1:n_1} \leq T_{k:n} \leq T^{(2)}_{k-k_1:n_2}$ using the same method as above. For $k_1' = k_1$, it is obvious that $\min(T^{(1)}_{k_1:n_1}, T^{(2)}_{k-k_1:n_2}) < T_{k:n} = \max(T^{(1)}_{k_1:n_1}, T^{(2)}_{k-k_1:n_2})$. This completes the proof. □

In the following theorem, we find the optimal task allocation $\boldsymbol{k}^*$, and show that the expected computing time of an $(\boldsymbol{n}, \boldsymbol{k}^*)-$group code converges to that of an $(n,k)-$MDS code for sufficiently large n.

Theorem 2. *Consider a scenario of computing k tasks on an $(\boldsymbol{n}, \boldsymbol{\mu})-$group system with $L = 2$ groups, where an $(\boldsymbol{n}, \boldsymbol{k})-$group code is applied. In the asymptotic regime of large n, the optimal task allocation $\boldsymbol{k}^* = [k_1^*, k - k_1^*]$ can be obtained[1] by solving*

$$k_1^* + n_2 - n_2 \left(1 - \frac{k_1^*}{n_1}\right)^{\frac{\mu_2}{\mu_1}} = k. \tag{4}$$

Moreover, the expected computing time of an $(\boldsymbol{n}, \boldsymbol{k}^*)-$group code satisfies the following:

$$\lim_{n\to\infty} \mathbb{E}[T_{\text{comp}}(C_{\text{G}}(\boldsymbol{n}, \boldsymbol{k}^*))] = \lim_{n\to\infty} \mathbb{E}[T_{\text{comp}}(C_{\text{MDS}}(n,k))]. \tag{5}$$

Proof. Combining (1) and (2), we obtain

$$\begin{aligned}\lim_{n\to\infty} k_1^* &= \arg\min_{k_1\in[k]} \Big\{ \lim_{n\to\infty} \max(\mathbb{E}[T^{(1)}_{k_1:n_1}], \mathbb{E}[T^{(2)}_{k-k_1:n_2}]) \Big\} \\ &= \arg\min_{k_1\in[k]} \Big\{ \max\Big(-\frac{1}{k\mu_1} \log(1 - \frac{k_1}{n_1}), \\ &\qquad -\frac{1}{k\mu_2} \log(1 - \frac{k-k_1}{n_2}) \Big) \Big\}.\end{aligned}$$

Note that the first variable of the max function is a strictly increasing convex function of k_1, while the second one is a strictly decreasing convex function. Thus, taking the maximum of the two variables results in a convex function of k_1. Therefore, as n grows to infinity, the minimizer k_1^* coincides with the intersection point of the two functions, i.e.,

$$\lim_{n\to\infty} \mathbb{E}[T^{(1)}_{k_1^*:n_1}] = \lim_{n\to\infty} \mathbb{E}[T^{(2)}_{k-k_1^*:n_2}]. \tag{6}$$

From (1) and (6), we obtain (4) by simple algebraic manipulations. Now we move on to the proof of (5). First, by taking $\lim_{n\to\infty} \mathbb{E}[\cdot]$ on (3) and applying Lemma 1, we obtain

$$\begin{aligned}\min\Big(\lim_{n\to\infty} \mathbb{E}[T^{(1)}_{k_1:n_1}], \lim_{n\to\infty} \mathbb{E}[T^{(2)}_{k-k_1:n_2}] \Big) &\leq \lim_{n\to\infty} \mathbb{E}[T_{k:n}] \\ &\leq \max\Big(\lim_{n\to\infty} \mathbb{E}[T^{(1)}_{k_1:n_1}], \lim_{n\to\infty} \mathbb{E}[T^{(2)}_{k-k_1:n_2}] \Big).\end{aligned}$$

When $k_1 = k_1^*$, the upper and lower bounds have the same value as in (6). Thus, bythe squeeze theorem, we have

$$\lim_{n\to\infty} \mathbb{E}[T_{k:n}] = \lim_{n\to\infty} \mathbb{E}[T^{(1)}_{k_1^*:n_1}] = \lim_{n\to\infty} \mathbb{E}[T^{(2)}_{k-k_1^*:n_2}].$$

Therefore, we obtain (5) by using $T_{\text{comp}}(C_{\text{MDS}}(n,k)) = T_{k:n}$ and $T_{\text{comp}}(C_{\text{G}}(\boldsymbol{n}, \boldsymbol{k})) = \max_{i\in[L]} T^{(i)}_{k_i:n_i}$. □

[1]Here we assume that k_1^* is an integer since the task allocation vector $\boldsymbol{k}$ consists of integers. However, in the case where k_1^* is not an integer, the optimal allocation rule is either $\boldsymbol{k} = [\lceil k_1^* \rceil, k - \lceil k_1^* \rceil]$ or $\boldsymbol{k} = [\lfloor k_1^* \rfloor, k - \lfloor k_1^* \rfloor]$, since $\mathbb{E}[T_{\text{comp}}(C_{\text{G}}(\boldsymbol{n}, \boldsymbol{k}))]$ is a convex function of k_1, as discussed in the proof.

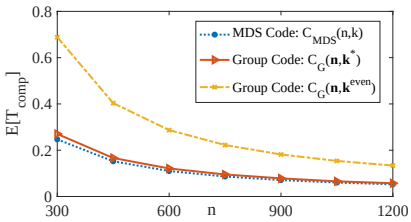

Figure 3: Simulated average computing time $\mathbb{E}[T_{\text{comp}}]$ of an MDS code and two types of group codes. Parameters are set to $(\boldsymbol{n}, \boldsymbol{\mu}) = \big([\frac{3}{4}n, \frac{1}{4}n], [1,2]\big)$ and $k = 100$.

Recall that an $(n,k)-$MDS code achieves the optimal computing time as stated in Theorem 1. The above theorem implies that an $(\boldsymbol{n}, \boldsymbol{k})-$group coded system can asymptotically achieve the optimal computing time by using the optimal task allocation rule $\boldsymbol{k} = \boldsymbol{k}^*$. Note that (4) can be easily solved when $\mu_1/\mu_2 = 2$ by using the quadratic formula. The following corollary provides the optimal task allocation $\boldsymbol{k}^*$ and the corresponding $\mathbb{E}[T_{\text{comp}}(C_{\text{G}}(\boldsymbol{n}, \boldsymbol{k}^*))]$ when $\mu_1 = 2\mu_2$.

Corollary 1. *Consider a scenario of computing k tasks on an $(\boldsymbol{n}, \boldsymbol{\mu})-$group system with $L = 2$ and $\boldsymbol{\mu} = [2\mu_2, \mu_2]$. Under the scenario, the optimal task allocation $\boldsymbol{k}^* = [k_1^*, k - k_1^*]$ is obtained as*

$$k_1^* = k - n_2 - \frac{n_2^2}{2n_1} + \sqrt{(n_2 + \frac{n_2^2}{2n_1})^2 - \frac{k}{n_1} n_2^2}. \tag{7}$$

Moreover, the expected value of the corresponding computing time $\mathbb{E}[T_{\text{comp}}(C_{\text{G}}(\boldsymbol{n}, \boldsymbol{k}^*))]$ can be calculated as

$$\begin{aligned}&\lim_{n\to\infty} \mathbb{E}[T_{\text{comp}}(C_{\text{G}}(\boldsymbol{n}, \boldsymbol{k}^*))] \\ &\qquad = \frac{1}{k\mu_2} \log\left(\sqrt{(1 + \frac{n_2}{2n_1})^2 - \frac{k}{n_1}} - \frac{n_2}{2n_1} \right)^{-1}.\end{aligned} \tag{8}$$

Proof. When $\mu_1 = 2\mu_2$, (4) reduces to (7). In addition, inserting (7) into (1) results in (8). □

V. Numerical Results for finite Number of Nodes

This section provides the simulated results of computing time and decoding time when the total number of workers n is finite. The combinded analysis is set aside in [20] due to lack of space. Here we first provide simulation results on the computing time of an $(\boldsymbol{n}, \boldsymbol{k})-$group code when the number of nodes n is finite. Fig. 3 illustrates the expected computing time of an $(n,k)-$MDS code $\mathbb{E}[T_{\text{comp}}(C_{\text{MDS}}(n,k))]$ and that of $(\boldsymbol{n}, \boldsymbol{k})-$group code $\mathbb{E}[T_{\text{comp}}(C_{\text{G}}(\boldsymbol{n}, \boldsymbol{k}))]$, for various n. We consider two types of group codes: one with the optimal task allocation $\boldsymbol{k}^* = [k_1^*, k - k_1^*]$, and the other with an even task allocation $\boldsymbol{k}^{\text{even}} = [\frac{1}{2}k, \frac{1}{2}k]$. For a fixed number of tasks $k = 100$, we assume that n workers are divided into two groups as $\boldsymbol{n} = [n_1, n_2] = [\frac{3}{4}n, \frac{1}{4}n]$. Moreover, the average computing time of a worker doubles in the first group, i.e., $\boldsymbol{\mu} = [\mu_1, \mu_2] = [1,2]$. For the estimation, we employ Monte Carlo methods with 10^4 random samples. The simulation result demonstrates

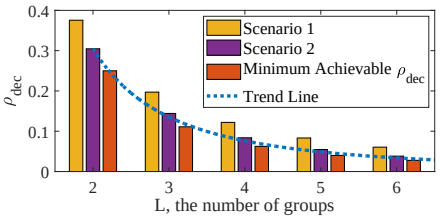

Figure 4: ρ_{dec} versus L for three different scenarios: imbalanced, balanced and minimum

that the expected computing time of an $(\boldsymbol{n}, \boldsymbol{k}^*)-$group code approaches that of an $(n, k)-$MDS code in the asymptotic regime of large n, as proved in Theorem 2. Moreover, the average computing times of two group codes $-$ the optimal group code $C_{\text{G}}(\boldsymbol{n}, \boldsymbol{k}^*)$ and a naive group code $C_{\text{G}}(\boldsymbol{n}, \boldsymbol{k}^{\text{even}})$ $-$ have a significant gap, which supports the need for a careful task allocation considering the heterogeneity of groups.

Now we compare the decoding complexity of the suggested $(\boldsymbol{n}, \boldsymbol{k})-$group code to that of an $(n, k)-$MDS code. We assume that the decoding complexity of an $(n, k)-$MDS code is $\mathcal{O}(k^\beta)$ for $\beta > 1$[2]. Then, the suggested $(\boldsymbol{n}, \boldsymbol{k})-$group code has a decoding complexity of $\mathcal{O}((k_{\max})^\beta)$ by the virtue of parallel decoding, where $k_{\max} = \max_{i \in [L]} k_i$. Note that decoding complexities of two schemes grow with different orders. For a comparison, we define the ratio of the two orders as

$$\rho_{\text{dec}} = \left(\frac{k_{\max}}{k} \right)^\beta.$$

Note that the ratio ρ_{dec} can be minimized down to $(1/L)^\beta$ when we have $k_{\max} = k/L$.

Fig. 4 illustrates ρ_{dec} under two different scenarios for $n = 240$ and $k = 120$. In both scenarios, $\boldsymbol{n}$ and $\boldsymbol{\mu}$ are randomly generated. Moreover, the task allocations for both scenarios are selected as the optimal $\boldsymbol{k}^*$, depending on the given parameters of $\boldsymbol{n}$ and $\boldsymbol{\mu}$. Motivated by the practical setting where the size of each group and the average computing time of each worker are bounded, we set $\boldsymbol{n} \sim \text{unif}(0.7\frac{n}{L}, 1.3\frac{n}{L})$ and $\boldsymbol{\mu} \sim \text{unif}(1, 2)$ with uniform distributions. Scenarios 1 and 2 differ in the rule of ordering the elements of $\boldsymbol{n}$ and $\boldsymbol{\mu}$, as illustrated below. For scenario 1, we sort the elements of $\boldsymbol{n}$ and $\boldsymbol{\mu}$ in ascending and descending order, respectively. In other words, $n_i \leq n_j$ and $\mu_i \geq \mu_j$ hold for all $i < j$. This is the scenario when a group with less average response time has less workers. In the case of scenario 2, both $\boldsymbol{n}$ and $\boldsymbol{\mu}$ are sorted in ascending order, i.e., $n_i \geq n_j$ and $\mu_i \geq \mu_j$ hold for $i < j$. This is the scenario when a group with less average response time has more workers. Under these scenarios, we obtain the average values of ρ_{dec} for 10^4 samples when $\beta = 2$. The simulations on two scenarios are compared to the minimum achievable $\rho_{\text{dec}} = (1/L)^\beta$. Moreover, we plot the trend line, which is set to stretch from the point of Scenario 2 for $L = 2$ and grow by a factor of $(1/L)^\beta$.

Fig. 4 delineates that ρ_{dec} diminishes along with the trend line under any scenarios as L grows, while there exists a noticeable gap between the two scenarios. Moreover, the proposed group code provides a significant decoding complexity reduction. For example, when $L = 4$, an $(\boldsymbol{n}, \boldsymbol{k})-$group code already achieves approximately 10x reduced decoding complexity compared to an $(n, k)-$MDS code.

[2]According to the recent works [21], [22] on decoding algorithms, practical scenarios satisfy $\beta > 1$.

REFERENCES

[1] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, A. Senior, P. Tucker, K. Yang, Q. V. Le *et al.*, "Large scale distributed deep networks," in *Advances in neural information processing systems*, 2012, pp. 1223–1231.

[2] J. Dean and L. A. Barroso, "The tail at scale," *Communications of the ACM*, vol. 56, no. 2, pp. 74–80, 2013.

[3] 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.

[4] K. Lee, C. Suh, and K. Ramchandran, "High-dimensional coded matrix multiplication," in *Information Theory (ISIT), 2017 IEEE International Symposium on*. IEEE, 2017, pp. 2418–2422.

[5] 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, pp. 4403–4413.

[6] T. Baharav, K. Lee, O. Ocal, and K. Ramchandran, "Straggler-proofing massive-scale distributed matrix multiplication with d-dimensional product codes," 2018.

[7] N. Raviv, R. Tandon, A. Dimakis, and I. Tamo, "Gradient coding from cyclic mds codes and expander graphs," in *International Conference on Machine Learning*, 2018, pp. 4302–4310.

[8] R. Tandon, Q. Lei, A. G. Dimakis, and N. Karampatziakis, "Gradient coding: Avoiding stragglers in distributed learning," in *International Conference on Machine Learning*, 2017, pp. 3368–3376.

[9] S. Dutta, V. Cadambe, and P. Grover, "Coded convolution for parallel and distributed computing within a deadline," in *Information Theory (ISIT), 2017 IEEE International Symposium on*, pp. 2403–2407.

[10] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Coded fourier transform," in *Communication, Control, and Computing (Allerton), 2017 55th Annual Allerton Conference on*. IEEE, 2017, pp. 494–501.

[11] S. Dutta, V. Cadambe, and P. Grover, "Short-dot: Computing large linear transforms distributedly using coded short dot products," in *Advances In Neural Information Processing Systems*, 2016, pp. 2100–2108.

[12] G. Suh, K. Lee, and C. Suh, "Matrix sparsification for coded matrix multiplication," in *Communication, Control, and Computing (Allerton), 2017 55th Annual Allerton Conference on*. IEEE, 2017, pp. 1271–1278.

[13] A. Reisizadeh, S. Prakash, R. Pedarsani, and A. S. Avestimehr, "Coded computation over heterogeneous clusters," *IEEE Transactions on Information Theory*, 2019.

[14] H. Park, K. Lee, J.-y. Sohn, C. Suh, and J. Moon, "Hierarchical coding for distributed computing," in *2018 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2018, pp. 1630–1634.

[15] [Online]. Available: https://aws.amazon.com/ec2/?nc1=h_ls

[16] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[17] F. Ahmad, S. T. Chakradhar, A. Raghunathan, and T. Vijaykumar, "Shufflewatcher: Shuffle-aware scheduling in multi-tenant mapreduce clusters." in *USENIX Annual Technical Conference*, 2014, pp. 1–12.

[18] A. Vahdat, M. Al-Fares, N. Farrington, R. N. Mysore, G. Porter, and S. Radhakrishnan, "Scale-out networking in the data center," *Ieee Micro*, vol. 30, no. 4, pp. 29–41, 2010.

[19] M. Zaharia, A. Konwinski, A. D. Joseph, R. H. Katz, and I. Stoica, "Improving mapreduce performance in heterogeneous environments." in *Osdi*, vol. 8, no. 4, 2008, p. 7.

[20] M. Kim, J.-y. Sohn, and J. Moon, "Coded matrix multiplication on a group-based model," *arXiv preprint arXiv:1901.05162*, 2019.

[21] W. Halbawi, N. Azizan, F. Salehi, and B. Hassibi, "Improving distributed gradient descent using reed-solomon codes," in *2018 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2018, pp. 2027–2031.

[22] W. Halbawi, Z. Liu, and B. Hassibi, "Balanced reed-solomon codes for all parameters," in *Information Theory Workshop (ITW), 2016 IEEE*. IEEE, 2016, pp. 409–413.