Hierarchical Coding for Distributed Computing

Hyegyeong Park, Kangwook Lee, Jy-yong Sohn, Changho Suh and Jaekyun Moon
School of Electrical Engineering, Korea Advanced Institute of Science and Technology (KAIST)
Email: {parkh, kw1jjang, jysohn1108, chsuh}@kaist.ac.kr, jmoon@kaist.edu

***Abstract*—Coding for distributed computing supports low-latency computation by relieving the burden of straggling workers. While most existing works assume a simple master-worker model, we consider a hierarchical computational structure consisting of groups of workers, motivated by the need to reflect the architectures of real-world distributed computing systems. In this work, we propose a *hierarchical coding scheme* for this model, as well as analyze its decoding cost and expected computation time. Specifically, we first provide upper and lower bounds on the expected computing time of the proposed scheme. We also show that our scheme enables efficient parallel decoding, thus reducing decoding costs by orders of magnitude over non-hierarchical schemes. When considering both decoding cost and computing time, the proposed hierarchical coding is shown to outperform existing schemes in many practical scenarios.**

I. Introduction

Enabling large-scale computations for big data analytics, distributed computing systems have received significant attention in recent years [1]. The distributed computing system divides a computational task to a number of subtasks, each of which is allocated to a different worker. This helps reduce computing time by exploiting parallel computing options and thus enables handling of large-scale computing tasks.

In a distributed computing system, the "stragglers", which refers to the computing nodes that slow down in some random fashion due to a variety of factors, may increase the total runtime of the computing system. To address this problem, the notion of *coded computation* is introduced in [2] where an (n, k) maximum distance separable (MDS) code is employed to speed up distributed matrix multiplications. The authors show that for linear computing tasks, one can design n distributed computing tasks such that *any* k out of n tasks suffice to complete the assigned task. Since then, coded computation has been applied to a wide variety of task scenarios such as matrix-matrix multiplication [3], [4], distributed gradient computation [5]–[8], convolution [9], Fourier transform [10] and matrix sparsification [11], [12].

While the idea of coded computation has been studied in various settings, existing works have not taken into account the underlying *hierarchical* nature of practical distributed systems [13]–[15]. In modern distributed computing systems, each group of workers is collocated in the same rack, which contains a Top of Rack (ToR) switch, and cross-rack communication is available only via these ToR switches. Surveys on real cloud computing systems show that cross-rack communication through the ToR switches is highly unstable due to the limited bandwidth, whereas intra-rack communication is faster and more reliable [14], [15]. A natural question is whether one can devise a coded computation scheme that exploits such hierarchical structure.

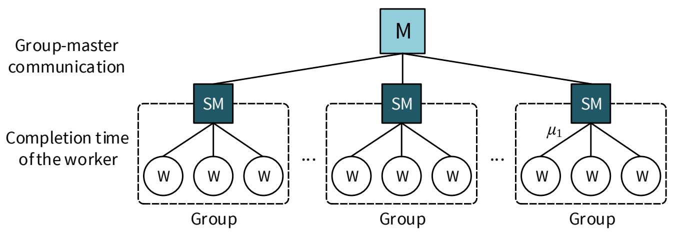


Fig. 1. Illustration of the hierarchical computing system

A. Contribution

In this work, we first model a distributed computing system with a tree-like hierarchical structure illustrated in Fig. 1, which is inspired by the practical computing systems in [13]–[15]. The workers (denoted by "W") are divided into groups, each of which has a submaster (denoted by "SM"). Each submaster sends the computational result of its group to the master (denoted by "M"). The suggested model can be viewed as a generalization of the existing non-hierarchical coded computation.

In this framework, we propose a hierarchical coding scheme which employs an $(n_1^{(i)}, k_1^{(i)})$ MDS code within group i and another (n_2, k_2) outer MDS code across the groups as depicted in Fig. 2. We also develop a parallel decoding algorithm which exploits the concatenated code structure and allows low complexity.

Moreover, we analyze the latency performance of our proposed solution. It turns out that the latency performance of our scheme cannot be analyzed via simple order statistics as in other existing schemes. Here we resort to find lower and upper bounds on the average latency performance: Our upper bound relies on concentration inequalities, and our lower bound is obtained via constructing and analyzing an auxiliary Markov chain (to be detailed later).

B. Related Work

Previous works on coded computation have rarely considered the inherent hierarchical structure of most real-world systems. Whereas a very recent work [16] deals with the multi-rack computing system reflecting imbalance between intra- and cross-rack communications, it is based on the settings of the coded MapReduce architecture which do not include general linear computation tasks that we focus on in this work. Another distinction is that the analysis of [16] includes only

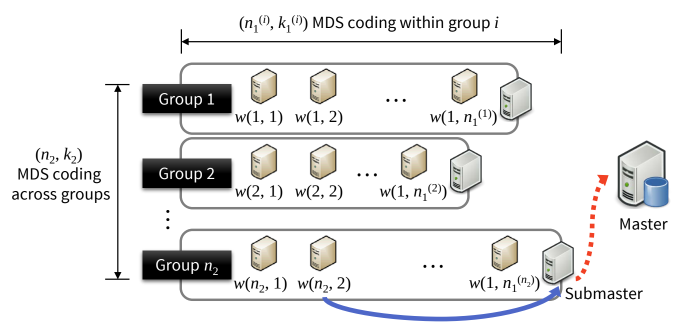


Fig. 2. Illustration of the proposed coding scheme applied to the hierarchical computing system. An $(n_1^{(i)}, k_1^{(i)})$ MDS code is employed within group i, and an (n_2, k_2) MDS code is applied across the groups. $w(i, j)$ denotes worker j in group i.

the cross-rack redundancy whereas our analysis considers both intra- and cross-group coding.

C. Notations

We use boldface uppercase letters for matrices and boldface lowercase letters for vectors. The transpose of a matrix $\mathbf{A}$ is denoted by $\mathbf{A}^T$. For a matrix $\mathbf{A}$ satisfying $\mathbf{A}^T = [\mathbf{A}_1^T \ \mathbf{A}_2^T]$, we write $\mathbf{A} = [\mathbf{A}_1; \mathbf{A}_2]$. For a positive integer n, the set $\{1, 2, \ldots, n\}$ is denoted by $[n]$. The j^{th} worker in group i is represented by $w(i, j)$ for $i \in [n_2]$ and $j \in [n_1^{(i)}]$. The symbol $\lfloor r \rfloor$ indicates the largest integer less than or equal to a real number r.

II. Hierarchical Coded Computation

A. Proposed Coding Scheme

Consider a matrix-vector multiplication task, i.e., computing $\mathbf{Ax}$ for a matrix $\mathbf{A} \in \mathbb{R}^{m \times d}$ and a vector $\mathbf{x} \in \mathbb{R}^{d \times 1}$. The input matrix $\mathbf{A}$ is split into k_2 submatrices as $\mathbf{A} = [\mathbf{A}_1; \mathbf{A}_2; \ldots; \mathbf{A}_{k_2}]$, where $\mathbf{A}_i \in \mathbb{R}^{\frac{m}{k_2} \times d}$ for $i \in [k_2]$. Here we assume that m is divisible by k_2 for simplicity. Then, we apply an (n_2, k_2) MDS code to set $\{\mathbf{A}_i\}_{i \in [k_2]}$ in obtaining $\{\widetilde{\mathbf{A}}_i\}_{i \in [n_2]}$. Then, each coded matrix $\widetilde{\mathbf{A}}_i$ is further divided into $k_1^{(i)}$ submatrices as $\widetilde{\mathbf{A}}_i = [\widetilde{\mathbf{A}}_{i,1}; \widetilde{\mathbf{A}}_{i,2}; \ldots; \widetilde{\mathbf{A}}_{i,k_1^{(i)}}]$ where $\widetilde{\mathbf{A}}_{i,j} \in \mathbb{R}^{\frac{m}{k_1^{(i)} k_2} \times d}$ for $j \in [k_1^{(i)}]$ and m divisible by $k_1^{(i)} k_2$. Afterwards, for each $i \in [n_2]$, we apply an $(n_1^{(i)}, k_1^{(i)})$ MDS code to set $\{\widetilde{\mathbf{A}}_{i,j}\}_{j \in [k_1^{(i)}]}$ to obtain $\{\widehat{\mathbf{A}}_{i,j}\}_{j \in [n_1^{(i)}]}$. Then, for each $i \in [n_2]$ and $j \in [n_1^{(i)}]$, worker $w(i, j)$ computes $\widehat{\mathbf{A}}_{i,j}\mathbf{x}$. Fig. 2 illustrates the proposed coding scheme for the hierarchical computing system with a different number of workers in each group. In the case of $n_1^{(i)} = n_1$ and $k_1^{(i)} = k_1$ for all $i \in [n_2]$, we will refer this coding scheme as $(n_1, k_1) \times (n_2, k_2)$ coded computation.

We present our code in Fig. 3 via a toy example. In this example, $(n_1, k_1) = (n_2, k_2) = (3, 2)$. That is, the input matrix $\mathbf{A} = [\mathbf{A}_1; \mathbf{A}_2]$ is encoded via $(n_2, k_2) = (3, 2)$ MDS code, yielding $[\widetilde{\mathbf{A}}_1; \widetilde{\mathbf{A}}_2; \widetilde{\mathbf{A}}_1 + \widetilde{\mathbf{A}}_2]$. Afterwards, the matrix $\widetilde{\mathbf{A}}_i = [\widetilde{\mathbf{A}}_{i,1}; \widetilde{\mathbf{A}}_{i,2}]$ is encoded via an $(n_1, k_1) = (3, 2)$ MDS code, producing $[\widehat{\mathbf{A}}_{i,1}; \widehat{\mathbf{A}}_{i,2}; \widehat{\mathbf{A}}_{i,1} + \widehat{\mathbf{A}}_{i,2}]$. For notational simplicity, we define $\widetilde{\mathbf{A}}_3 = \widetilde{\mathbf{A}}_1 + \widetilde{\mathbf{A}}_2$ and $\widehat{\mathbf{A}}_{i,3} = \widehat{\mathbf{A}}_{i,1} + \widehat{\mathbf{A}}_{i,2}$.

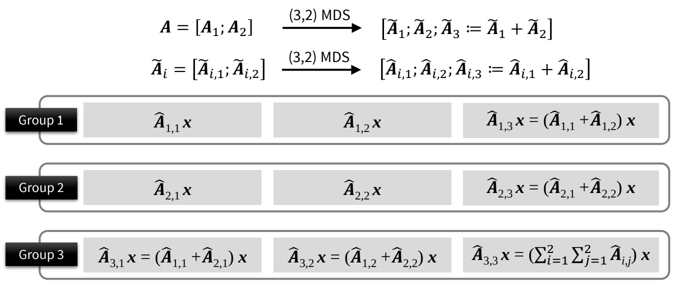

$\boldsymbol{A} = [\boldsymbol{A}_1; \boldsymbol{A}_2] \xrightarrow{(3,2)\text{ MDS}} [\widetilde{\boldsymbol{A}}_1; \widetilde{\boldsymbol{A}}_2; \widetilde{\boldsymbol{A}}_3 := \widetilde{\boldsymbol{A}}_1 + \widetilde{\boldsymbol{A}}_2]$
$\widetilde{\boldsymbol{A}}_i = [\widetilde{\boldsymbol{A}}_{i,1}; \widetilde{\boldsymbol{A}}_{i,2}] \xrightarrow{(3,2)\text{ MDS}} [\widehat{\boldsymbol{A}}_{i,1}; \widehat{\boldsymbol{A}}_{i,2}; \widehat{\boldsymbol{A}}_{i,3} := \widehat{\boldsymbol{A}}_{i,1} + \widehat{\boldsymbol{A}}_{i,2}]$
Group 1
$\widehat{\boldsymbol{A}}_{1,1}\boldsymbol{x}$
$\widehat{\boldsymbol{A}}_{1,2}\boldsymbol{x}$
$\widehat{\boldsymbol{A}}_{1,3}\boldsymbol{x} = (\widehat{\boldsymbol{A}}_{1,1} + \widehat{\boldsymbol{A}}_{1,2})\,\boldsymbol{x}$
Group 2
$\widehat{\boldsymbol{A}}_{2,1}\boldsymbol{x}$
$\widehat{\boldsymbol{A}}_{2,2}\boldsymbol{x}$
$\widehat{\boldsymbol{A}}_{2,3}\boldsymbol{x} = (\widehat{\boldsymbol{A}}_{2,1} + \widehat{\boldsymbol{A}}_{2,2})\,\boldsymbol{x}$
Group 3
$\widehat{\boldsymbol{A}}_{3,1}\boldsymbol{x} = (\widehat{\boldsymbol{A}}_{1,1} + \widehat{\boldsymbol{A}}_{2,1})\,\boldsymbol{x}$
$\widehat{\boldsymbol{A}}_{3,2}\boldsymbol{x} = (\widehat{\boldsymbol{A}}_{1,2} + \widehat{\boldsymbol{A}}_{2,2})\,\boldsymbol{x}$
$\widehat{\boldsymbol{A}}_{3,3}\boldsymbol{x} = (\sum_{i=1}^{2}\sum_{j=1}^{2}\widehat{\boldsymbol{A}}_{i,j})\,\boldsymbol{x}$

Fig. 3. Allocation of the computational task to workers in a $(3, 2) \times (3, 2)$ coded computation

For $i, j \in \{1, 2, 3\}$, worker $w(i, j)$ computes $\widehat{\mathbf{A}}_{i,j}\mathbf{x}$. Note that group i is assigned a subtask with respect to $\widetilde{\mathbf{A}}_i$.

We now describe the decoding algorithm for our proposed coding scheme. When a worker completes its task, it sends the result to its submaster. With the aid of the (n_1, k_1) MDS code, submaster i (in group i) can compute $\widetilde{\mathbf{A}}_i\mathbf{x}$ as soon as the task results from any k_1 workers within group i are collected. Once $\widetilde{\mathbf{A}}_i\mathbf{x}$ is computed, it is sent to the master. The master can obtain $\mathbf{Ax}$ by retrieving $\widetilde{\mathbf{A}}_i\mathbf{x}$ from any k_2 submasters. For each worker, we define *completion time* as the sum of the runtime of the worker and the time required for delivering its computation result to the submaster. For each group, we further define *intra-group latency* as the time for completing its assigned subtask. The total computation time is defined as the time from when the workers start to run until the master completes computing $\mathbf{Ax}$. The proposed computation framework can be applied to practical multi-rack systems where the input data $\mathbf{A}$ is coded and distributed into n_2 racks; the i^{th} rack contains $\widetilde{\mathbf{A}}_i$. For instance, in the Facebook's warehouse cluster, data is encoded with a $(14, 10)$ MDS code, and then the 14 encoded chunks are stored across different racks [17]. Once $\mathbf{x}$ is given from the master, the i^{th} rack can compute $\widetilde{\mathbf{A}}_i\mathbf{x}$ using the coded data $\widetilde{\mathbf{A}}_i$ that it contains.

B. Application: Matrix-Matrix Multiplications

Our scheme can be also applied to matrix-matrix multiplications. More specifically, consider computing $\mathbf{A}^T\mathbf{B}$ for given matrices $\mathbf{A}$ and $\mathbf{B} = [\mathbf{b}_1 \, \mathbf{b}_2 \, \cdots \, \mathbf{b}_{k_2}]$. After applying an (n_2, k_2) MDS code to $\mathbf{B}$, we have $\check{\mathbf{B}} = [\check{\mathbf{b}}_1 \, \check{\mathbf{b}}_2 \, \cdots \, \check{\mathbf{b}}_{n_2}]$. Moreover, group i divides $\mathbf{A}$ into $k_1^{(i)}$ equal-sized submatrices as $\mathbf{A} = [\mathbf{A}_{i,1} \, \mathbf{A}_{i,2} \, \cdots \, \mathbf{A}_{i,k_1^{(i)}}]$, and we apply an $(n_1^{(i)}, k_1^{(i)})$ MDS code, resulting in $\check{\mathbf{A}}_i = [\check{\mathbf{A}}_{i,1} \, \check{\mathbf{A}}_{i,2} \, \cdots \, \check{\mathbf{A}}_{i,n_1^{(i)}}]$. The computation $\check{\mathbf{A}}_{i,j}^T\check{\mathbf{b}}_i$ is assigned to worker $w(i, j)$. Using an $(n_1^{(i)}, k_1^{(i)})$ MDS code, submaster i can compute $\mathbf{A}^T\check{\mathbf{b}}_i$ when any $k_1^{(i)}$ workers within its group delivered their computation results. The master can calculate $\mathbf{A}^T\mathbf{B}$ by gathering $\mathbf{A}^T\check{\mathbf{b}}_i$ results from any k_2 submasters, using the (n_2, k_2) MDS code. Under the homogeneous setting of $n_1^{(i)} = n_1$ and $k_1^{(i)} = k_1$ for all $i \in [n_2]$, the encoding algorithm of the proposed scheme reduces to that of the product coded scheme [3]. However, the suggested scheme with the homogeneous setting is shown to reduce the decoding cost compared to the product coded

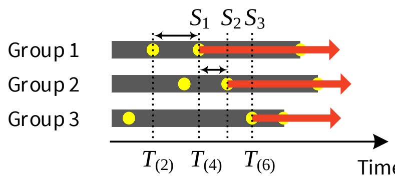


Fig. 4. Illustration of obtaining $\mathcal{L}$ in a $(3,2)\times(3,2)$ coded computation scheme under the hierarchical computing structure: a detailed analysis is in Sec. IV.

III. Latency Analysis

We start by providing some preliminaries for the order statistics. For n random variables, the k^{th} order statistic is defined by the k^{th} smallest one of n. From the known results from the order statistics [18], the expected value of the k^{th} order statistics out of n $(n>k)$ exponential random variables with rate μ is $(H_n - H_{n-k})/\mu$, where $H_k = \sum_{l=1}^{k}\frac{1}{l} \simeq \log k + \gamma$ as k grows for a fixed constant γ. This leads to $(H_n - H_{n-k})/\mu \simeq \frac{1}{\mu}\log\frac{n}{n-k}$. For $n=k$, the expected latency is given by $H_n/\mu \simeq (\log n)/\mu$. Further, define $H_0 := 0$ for ease of exposition.

Consider the hierarchical computing system[1] of Fig. 2. Assume that for $i\in[n_2]$ and $j\in[n_1]$, the completion time $T_{i,j}$ of worker $w(i,j)$ is exponentially distributed with rate μ_1 (i.e., $\Pr[T_{i,j}\le t] = 1-e^{-\mu_1 t}$). Further, the communication time $T_i^{(c)}$ from the i^{th} group to the master is also exponentially distributed with rate μ_2 (i.e., $\Pr[T_i^{(c)}\le t] = 1-e^{-\mu_2 t}$). Here, we assume that all latencies are independent with one another. Given the assumptions, the total computation time of the $(n_1,k_1)\times(n_2,k_2)$ coded computation is written as

$$T = {k_2}^{\text{th}}\min_{i\in[n_2]}\left(T_i^{(c)} + S_i\right) \tag{1}$$

where

$$S_i = {k_1}^{\text{th}}\min_{j\in[n_1]} T_{i,j} \tag{2}$$

denotes the time to wait for the k_1 fastest workers in group i. The group index i is relabeled such that $S_1\le S_2\le\cdots\le S_{n_2}$. In other words, the fastest group that finishes its assigned subtask is relabeled as the 1st group, while the slowest group is relabeled as the ${n_2}^{\text{th}}$ group. Here we provide upper and lower bounds on $\mathbb{E}[T]$.

A. Lower Bound

Let $T_{(m)}$ be the m^{th} smallest element of $\{T_{i,j}\}_{i\in[n_2],j\in[n_1]}$. Then, $T_{(1)}\le T_{(2)}\le\cdots\le T_{(n_1 n_2)}$ holds. Using this notation, we derive a lower bound on $\mathbb{E}[T]$, formally stated below.

Theorem 1: The expected total computation time of the $(n_1,k_1)\times(n_2,k_2)$ coded computation is lower bounded as

$$\mathbb{E}[T] \ge \mathbb{E}\left[{k_2}^{th}\min_{i\in[n_2]}\left(T_i^{(c)} + T_{(ik_1)}\right)\right] := \mathcal{L}\,. \tag{3}$$

[1]For simplicity of analysis, we only consider the homogeneous setting of $n_1^{(i)} = n_1$ and $k_1^{(i)} = k_1$ for all $i\in[n_2]$.

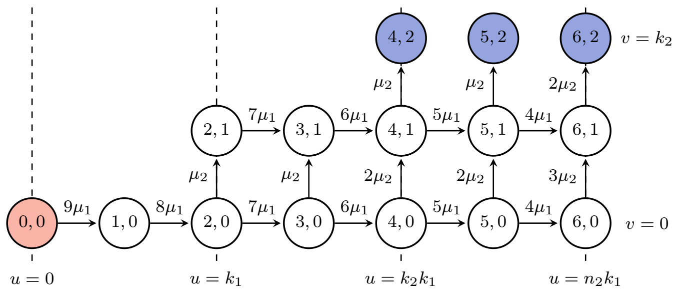


Fig. 5. State transition diagram producing the lower bound $\mathcal{L}$ on the expected latency in a $(3,2)\times(3,2)$ coded computation. Each state is labeled with (u,v), where u is the number of completed workers and v is the number of groups that have sent their computation results.

Proof: Consider a realization of $\{T_{i,j}\}_{i\in[n_2],j\in[n_1]}$ and $\{T_i^{(c)}\}_{i\in[n_2]}$. Recall that a group finishes its assigned subtask if k_1 workers within the group complete their tasks. Hence, it is impossible for the i^{th} group to finish its work if the total number of completed workers in the system is less than ik_1. In other words, it must hold that

$$T_{(ik_1)} \le S_i \text{ for all } i\in[n_2]\,. \tag{4}$$

Thus, the total computation time in (1) should be:

$$T = {k_2}^{\text{th}}\min_{i\in[n_2]}(T_i^{(c)} + S_i) \ge {k_2}^{\text{th}}\min_{i\in[n_2]}(T_i^{(c)} + T_{(ik_1)})\,.$$

Averaging over all possible realizations, we complete the proof. ■

To further illustrate the proof, we provide a schematic example in Fig. 4. Consider a $(3,2)\times(3,2)$ coded computation. The yellow circles denote the completion times of the workers. After $k_1=2$ workers in a group finish their computations, the group-master communication, shown as the red arrows, starts from each group. As can be seen, $T_{(2)}\le S_1$, $T_{(4)}\le S_2$ and $T_{(6)}\le S_3$, which concur with (4). The following lemma shows that $\mathcal{L}$ can be computed by analyzing the hitting time of an auxiliary Markov chain.

Lemma 1: Let $\mathbb{C}$ be the continuous-time Markov chain defined over the state space $(u,v)\in\{0,1,\ldots,n_2k_1\}\times\{0,1,\ldots,k_2\}$. The state transition rates of $\mathbb{C}$ are defined as follows:

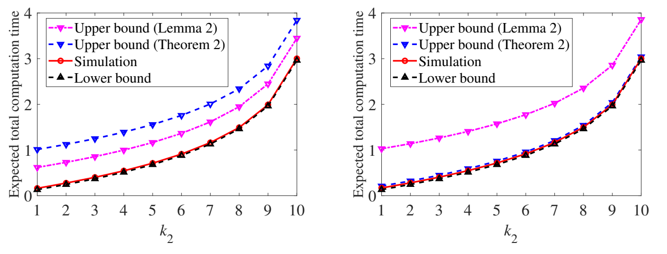
- From state (u,v) to state $(u+1,v)$ at rate $(n_1n_2-i)\mu_1$, if $vk_1\le u< n_2k_1$,
- From state (u,v) to state $(u,v+1)$ at rate $(\lfloor u/k_1\rfloor - v)\,\mu_2$, if $0\le v<\min\{\lfloor u/k_1\rfloor, k_2\}$.

Then, the expected hitting time of $\mathbb{C}$ from state $(0,0)$ to the set of states $\{(u,k_2)\}_{u=k_2k_1}^{n_2k_1}$ is equal to $\mathcal{L}$.

Proof: See Appendix A for the proof. ■

Markov chain $\mathbb{C}$ defined in Lemma 1 consists of the states (u,v), where u represents the number of completed workers and v indicates the number of groups which have delivered their computation results to the master.

For an illustrative example, the state transition diagram for a $(3,2)\times(3,2)$ coded computation yielding a lower bound is shown in Fig. 5. The overall computation is terminated when



(a) $n_1 = 10$, $k_1 = 5$, $n_2 = 10$ (b) $n_1 = 600$, $k_1 = 300$, $n_2 = 10$

Fig. 6. The expected total computation time of the $(n_1, k_1) \times (n_2, k_2)$ coded computing with its bounds for varying k_2

the $k_2 = 2$ groups finish conveying their computational results to the master, i.e., when the Markov chain visits the states with $v = 2$ for the first time. We see that u increases by one when a worker completes its computation, and v increases by one when master receives the computation result from a group. The rightward transition (to increase u) rate is determined by the product of μ_1 and the number of remaining workers. The upward transition (to increase v) rate is the product of μ_2 and the number of groups that have not delivered their computation results to the master. The proposed lower bound $\mathcal{L}$ can be easily computed from the first-step analysis [19] of the Markov chain produced by Lemma 1.

B. Upper Bound

We here provide two upper bounds on the expected total computation time. The first bound in the following lemma is applicable for all values of n_1 and k_1.

Lemma 2: The expected total computation time of the $(n_1, k_1) \times (n_2, k_2)$ *coded computation is upper bounded as* $\mathbb{E}[T] \leq H_{n_1 n_2}/\mu_1 + (H_{n_2} - H_{n_2 - k_2})/\mu_2$.

Proof: See Appendix B for the proof. ■

We now establish another upper bound using the following two steps. First we find an upper bound on the maximum intra-group latency among n_2 groups. Afterwards, adding this value to the expected latency of the group-master communication yields an upper bound on the expected total computation time. For given n_1 and k_1, we use $\delta_1 > 0$ which satisfies $n_1 = (1 + \delta_1)k_1$. We now present the asymptotic upper bound as follows.

Theorem 2: For a fixed constant $\delta_1 > 0$*, the expected total computation time of the* $(n_1, k_1) \times (n_2, k_2)$ *coded computing system is upper bounded as* $\mathbb{E}[T] \leq [\log(1 + \delta_1)/\delta_1]/\mu_1 + (H_{n_2} - H_{n_2 - k_2})/\mu_2 + o(1)$ *in the limit of* k_1.

Proof: See Appendix C for the proof. ■

C. Evaluation of Bounds

Fig. 6 shows the behavior of the expected total computation time and its upper/lower bounds with varying k_2. Here we consider two upper bounds proposed in Lemma 2 and Theorem 2. To see the impact of k_1, the values of k_1 are fixed to 5 and 300 in Figs. 6a and 6b, respectively. The other code parameters are set to $n_1 = (1 + \delta_1)k_1, n_2 = 10$ for both figures, where δ_1 is fixed to 1. The rates of the completion time of the worker and group-master communication are set to $\mu_1 = 10$ and $\mu_2 = 1$. For a relatively small values of k_1, the upper bound in Lemma 2 is a tighter upper bound than the upper bound in Theorem 2. As can be seen in Fig. 6b, the asymptotic upper bound in Theorem 2 becomes tighter as k_1 grows, which concurs with Theorem 2. We also have numerically confirmed that the proposed lower bound is tight.

IV. Decoding Complexity

In this section, we compare decoding complexity of our hierarchical coding with the replication and non-hierarchical coding schemes including the $(n_1, k_1) \times (n_2, k_2)$ product code [3] and the (n, k) polynomial code [4]. For fair comparison, we set $n = n_1 n_2$ and $k = k_1 k_2$. We further assume that the decoding complexity of the (n, k) MDS code is $\mathcal{O}(k^\beta)$ for some $\beta > 1$.[2] In our framework, the n_2 intra-group codes can be decoded in parallel, followed by decoding of the cross-group code using the k_2 fastest results. Thus, the overall decoding procedure consists of 1) parallel decoding of (n_1, k_1) intra-group MDS codes and 2) decoding of the (n_2, k_2) cross-group MDS code, resulting in the total decoding cost of $\mathcal{O}(k_1^\beta + k_1 k_2^\beta)$. Similarly, one can show that the decoding cost of polynomial codes is $\mathcal{O}(k^\beta)$, and that of product code is $\mathcal{O}(k_1 k_2^\beta + k_2 k_1^\beta)$. We note that the hierarchical code can have a substantial improvement, sometimes by an order of magnitude, in decoding complexity, compared to the product code. For instance, if $\beta = 2$ and $k_1 = k_2^2$, the decoding cost of hierarchical code becomes $\mathcal{O}(k_2^4)$ while that of the product code is $\mathcal{O}(k_2^5)$; if $k_1 = k_2^{1.5}$, the decoding costs are $\mathcal{O}(k_2^{3.5})$ and $\mathcal{O}(k_2^4)$, respectively. In general, if $k_1 = k_2^p$, one can show that the relative gain of the hierarchical codes in decoding cost monotonically increases as p increases, providing a guideline for efficient code designs. Table I summarizes the computing times and decoding costs of various coding schemes.

We now compare the expected total execution time defined as $T_{\text{exec}} := T_{\text{comp}} + \alpha T_{\text{dec}}$, where T_{comp} is the computing time, T_{dec} is the decoding cost, and $\alpha \geq 0$ is the relative weight of the decoding cost. We note that α is a system-specific parameter that depends on 1) the relative CPU speed of the master compared to the workers and 2) dimension of the input data. Shown in Fig. 7 are the expected total execution times for parameters of $(n_1, k_1) = (800, 400)$, $(n_2, k_2) = (40, 20), (\mu_1, \mu_2) = (10, 1)$ and $\beta = 2$.

We first observe that with all tested practical values of μ_1 and μ_2, the hierarchical code strictly outperforms the product code for all values of α. Further, we observe that the optimal choice of coding scheme depends on the value of α as follows:

- (moderate α) when both T_{comp} and T_{dec} have to be minimized, the hierarchical code achieves the lowest T_{exec} by striking a balance between them;
- (low α) when T_{dec} is negligible, the polynomial code achieves the lowest T_{exec}; and
- (high α) when T_{dec} dominates T_{exec}, the replication code is the best.

[2]Note that this is the case for most practical decoding algorithms [20], [21]. Decoding with $\beta = 1$ requires a large field size [4].

TABLE I
COMPARISONS OF VARIOUS CODING SCHEMES

Coding scheme	Computing time (T_{comp})	Decoding cost (T_{dec})
Replication	$kH_k/(n\mu_2)$	0
Hierarchical code	$\mathbb{E}[T]$	$\mathcal{O}(k_1 k_2^\beta + k_1^\beta)$
Product code [3]	$\frac{1}{\mu_2}\log\left(\frac{\sqrt{n/k}+\sqrt[4]{n/k}}{\sqrt{n/k}-1}\right)$	$\mathcal{O}(k_1 k_2^\beta + k_2 k_1^\beta)$
Polynomial code [4]	$(H_n - H_{n-k})/\mu_2$	$\mathcal{O}(k_1^\beta k_2^\beta)$

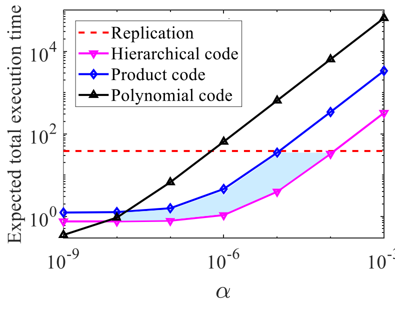


Fig. 7. $\mathbb{E}[T_{\text{exec}}]$ of various coding schemes for parameters of $(n_1, k_1) = (800, 400)$, $(n_2, k_2) = (40, 20)$, $(\mu_1, \mu_2) = (10, 1)$ and $\beta = 2$

Note that the shaded area in Fig. 7 represents the additional achievable $(\alpha, \mathbb{E}[T_{\text{exec}}])$ region thanks to introducing the hierarchical code.

APPENDIX A
PROOF OF LEMMA 1

Note that the lower bound $\mathcal{L}$ in Theorem 1 can be illustrated as in Fig. 8. The lower bound depends on two types of variables: $\{T_{(m)}\}_{m=1}^{n_2 k_1}$, the set of $n_2 k_1$ smallest realizations of $n_2 n_1$ exponentially distributed random variables with rate μ_1 and $\{T_c^{(l)}\}_{l=1}^{n_2}$, the set of n_2 exponentially distributed random variables with rate μ_2. Consider arbitrary realizations of $\{T_{(m)}\}$ and $\{T_c^{(l)}\}$. For a given time t, define

$$u := \max_{m \in [n_2 k_1]} \{t \geq T_{(m)}\}, \tag{5}$$

$$v := \left| \{ l \in [n_2] : t \geq T_c^{(l)} + T_{(l k_1)} \} \right|. \tag{6}$$

Thus, each time slot t can be assigned to a state (u, v) for $u \in \{0, 1, \cdots, n_2 k_1\}$ and $v \in \{0, 1, \cdots, n_2\}$. From the definition of $\mathcal{L}$ in (3), the lower bound corresponds to the expected time t to achieve $v = k_2$. Thus, we consider the state space of $(u, v) \in \{0, 1, \cdots, n_2 k_1\} \times \{0, 1, \cdots, k_2\}$, and find the expected time to arrive at states (u, v) with $v = k_2$ from state $(0, 0)$.

We now examine the state transition rates. From the definitions of $T_{(m)}$ and (5), the transition from state (u, v) to state $(u + 1, v)$ occurs with rate $(n_1 n_2 - u)\mu_1$, since there are $n_1 n_2 - u$ remaining $\{T_{(m)}\}_{m=u+1}^{n_1 n_2}$ such that $t < T_{(m)}$ holds. Moreover, for a given time t and the corresponding state (u, v), we have $\{T_{(l k_1)}\}_{l=1}^{\lfloor u/k_1 \rfloor}$ which satisfies $t \geq T_{(l k_1)}$, and v in (6) is expressed as

$$v = \left| \{ l \in \{1, 2, \cdots, \lfloor u/k_1 \rfloor\} : t \geq T_c^{(l)} + T_{(l k_1)} \} \right| \tag{7}$$

since $T_c^{(l)}$ is a random variable with nonnegative values. Thus, out of $\lfloor u/k_1 \rfloor$ activated (i.e., $t \geq T_{(l k_1)}$) random

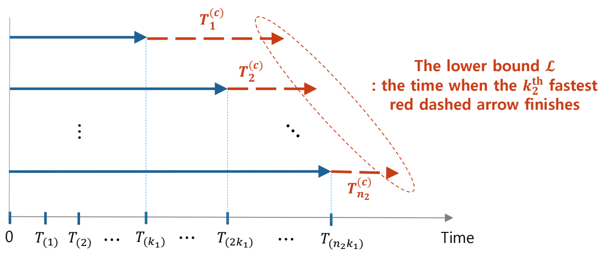

$T_1^{(c)}$
$T_2^{(c)}$
$T_{n_2}^{(c)}$
The lower bound $\mathcal{L}$
: the time when the k_2^{th} fastest
red dashed arrow finishes
0
$T_{(1)}$
$T_{(2)}$
$T_{(k_1)}$
$T_{(2k_1)}$
$T_{(n_2 k_1)}$
Time

Fig. 8. Illustration of the lower bound $\mathcal{L}$

variables $\{T_{(l k_1)}\}_{l=1}^{\lfloor u/k_1 \rfloor}$, only v random variables satisfy $t \geq T_c^{(l)} + T_{(l k_1)}$. Therefore, the transition from state (u, v) to state $(u, v + 1)$ occurs with rate $(\lfloor u/k_1 \rfloor - v)\mu_2$, for $v \in \{0, 1, \cdots, \min\{\lfloor u/k_1 \rfloor, k_2\} - 1\}$. Fig. 9 shows the consequent state transition diagram. This Markov chain is identical to $\mathbb{C}$, which completes the proof.

APPENDIX B
PROOF OF LEMMA 2

$H_{n_1 n_2}/\mu_1$ is the maximum intra-group latency, which comes from waiting for all $n_1 n_2$ workers. Assuming that every group starts the group-master communication at time $H_{n_1 n_2}/\mu_1$, the expected total computation time can be obtained by summing up the group-master communication time to $H_{n_1 n_2}/\mu_1$. The group-master communication time is calculated from the time that the k_2th fastest group finishes communication to the master, which is given by $(H_{n_2} - H_{n_2 - k_2})/\mu_2$. This completes the proof.

APPENDIX C
PROOF OF THEOREM 2

A part of the proof generalizes the idea of [3], which analyzes the latency of the product code. First, we focus on the intra-group latency of each group. The expected latency of the k_1th fastest worker out of n_1 is given by $(H_{n_1} - H_{n_1 - k_1})/\mu_1$, where the latency of a worker assumes an exponential distribution with rate μ_1. Noticing that the expected completion time of a worker $(H_{n_1} - H_{n_1 - k_1})/\mu_1$ is rewritten as $\frac{1}{\mu_1}\log\frac{1+\delta_1}{\delta_1}$ for a fixed constant $\delta_1 > 0$ and a sufficiently large n_1, we define

$$t_0 := \frac{1}{\mu_1}\log\frac{1+\delta_1}{\delta_1} + \alpha\sqrt{\frac{\log k_1}{k_1}}$$

for some constant $\alpha > 0$.

Consider group i_0 with n_1 workers. Then, for worker $w(i_0, j)$, assume a Bernoulli random variable $X_{i_0, j}$ which takes 0 when worker $w(i_0, j)$ has completed its computation by time t_0, and takes 1 otherwise. Then, probability p_0 that $X_{i_0, j}$ takes 1 is:

$$\begin{aligned} p_0 := \Pr[T_{i_0, j} > t_0] = e^{-\mu_1 t_0} &= \frac{\delta_1}{1+\delta_1} e^{-\mu_1 \alpha \sqrt{\frac{\log k_1}{k_1}}} \\ &\simeq \frac{\delta_1}{1+\delta_1}\left(1 - \mu_1 \alpha \sqrt{\frac{\log k_1}{k_1}}\right), \end{aligned} \tag{8}$$

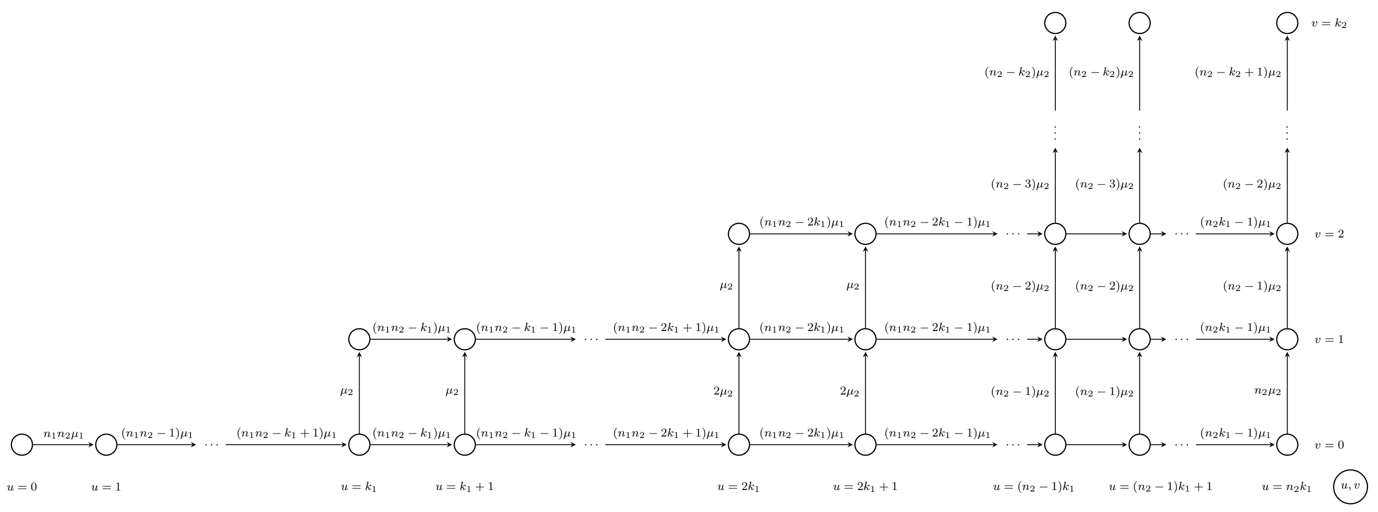


Fig. 9. State transition diagram for the $(n_1, k_1) \times (n_2, k_2)$ coded computation producing a lower bound. Any state is denoted by (u, v), where u describes the number of completed workers and v is the number of groups that sent their computation results.

where (8) follows because $\mu_1 \alpha \sqrt{\frac{\log k_1}{k_1}}$ is quite small with a sufficiently large k_1. Out of n_1 workers in group i_0, the set of workers not completed by time t_0 is represented as

$$S_{t_0} = \{j \in [n_1] : T_{i_0,j} > t_0\} = \{j \in [n_1] : X_{i_0,j} = 1\}$$

with $|S_{t_0}|$ representing the number of workers not completed by time t_0, where $|\cdot|$ denotes the cardinality of a set.

Since $X_{i_0,j}$ is a Bernoulli random variable with parameter p_0, the expected number of workers not completed in group i_0 is calculated as $n_1 p_0 = \delta_1(k_1 - \mu_1 \alpha \sqrt{k_1 \log k_1})$ for a given $n_1 = (1 + \delta_1)k_1$. Recall that a group finishes its assigned subtask when k_1 out of n_1 workers in the group completed their works. At time t_0, we thus denote a case where the number of stragglers in group i is greater than $\delta_1 k_1 = n_1 - k_1$ by an error event E_i for $i \in [n_2]$. For group i_0, we wish to find an upper bound on the probability that E_{i_0} occurs, which is equivalent to the probability that group i_0 has not finished its assigned subtask by time t_0. We establish such a bound using Hoeffding's inequality [22] to bound the deviation of $|S_{t_0}|$ from the mean:

$$\Pr[|S_{t_0}| - \delta_1(k_1 - \mu_1 \alpha \sqrt{k_1 \log k_1}) \geq t] \leq e^{-\frac{2t^2}{(1+\delta_1)k_1}} .$$

By setting $t = \delta_1 \mu_1 \alpha \sqrt{k_1 \log k_1}$, we obtain

$$\Pr\left[|S_{t_0}| \geq \delta_1 k_1\right] \leq e^{-\frac{2\delta_1^2 \mu_1^2 \alpha^2}{1+\delta_1} \log k_1} = k_1^{-\frac{2\delta_1^2 \mu_1^2 \alpha^2}{1+\delta_1}} .$$

Combining all n_2 groups, the upper bound on the probability that n_2 groups not finished their assigned subtasks by time t_0 is obtained by the union bound. Let T_I be the time when all n_2 groups finish their assigned subtasks. Hence, we have

$$\begin{aligned} \Pr[T_I > t_0] &= \Pr[E_1 \cup E_2 \cup \cdots \cup E_{n_2}] && (9) \\ &\leq \sum_{i=1}^{n_2} \Pr[E_i] = n_2 k_1^{-\frac{2\delta_1^2 \mu_1^2 \alpha^2}{1+\delta_1}} = o(k_1^{-1}) , && (10) \end{aligned}$$

where the last equality holds since α can be made arbitrarily large. Then the expected intra-group latency $\mathbb{E}[T_I]$ satisfies

$$\begin{aligned} \mathbb{E}[T_I] &\leq \Pr[T_I \leq t_0] t_0 + \Pr[T_I > t_0] \left(\frac{H_{n_1 n_2}}{\mu_1} + t_0 \right) && (11) \\ &= \left(1 - o(k_1^{-1})\right) t_0 + o(k_1^{-1}) \left(\frac{H_{n_1 n_2}}{\mu_1} + t_0 \right) && (12) \\ &= \frac{1}{\mu_1} \log \frac{1 + \delta_1}{\delta_1} + o(1) , && (13) \end{aligned}$$

where (11) is due to the fact that t_0 is the worst case latency for all events satisfying $T_I \leq t_0$, and $H_{n_1 n_2}/\mu_1 + t_0$ is an upper bound on $\mathbb{E}[T_I | T_I > t_0]$.

From (13), we conclude that all the n_2 groups embark on the group-master communication before time $\frac{1}{\mu_1} \log \frac{1+\delta_1}{\delta_1}$, as k_1 grows large. Hence, adding the latency of the group-master communication $(H_{n_2} - H_{n_2 - k_2})/\mu_2$ to (13) gives an upper bound on the expected total computation time. This completes the proof of the case where $n_2 > k_2$. When $n_2 = k_2$, the group-master communication time is represented by H_{n_2}/μ_2. Thus, adding this value to (13) completes the proof, using $H_0 = 0$.

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 *Proc. NIPS*, 2012, pp. 1223–1231.

[2] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Speeding up distributed machine learning using codes," *IEEE Trans. Inf. Theory*, vol. PP, no. 99, pp. 1–1, 2017.

[3] K. Lee, C. Suh, and K. Ramchandran, "High-dimensional coded matrix multiplication," in *Proc. IEEE ISIT*, June 2017, pp. 2418–2422.

[4] Q. Yu, M. Maddah-Ali, and S. Avestimehr, "Polynomial codes: An optimal design for high-dimensional coded matrix multiplication," in *Proc. NIPS*, 2017, pp. 4406–4416.

[5] R. Tandon, Q. Lei, A. G. Dimakis, and N. Karampatziakis, "Gradient coding: Avoiding stragglers in distributed learning," in *Proc. ICML*, 2017, pp. 3368–3376.

[6] W. Halbawi, N. Azizan-Ruhi, F. Salehi, and B. Hassibi, "Improving distributed gradient descent using Reed-Solomon codes," *arXiv:1706.05436*, 2017.

[7] N. Raviv, I. Tamo, R. Tandon, and A. G. Dimakis, "Gradient coding from cyclic MDS codes and expander graphs," *arXiv:1707.03858*, 2017.
[8] Z. Charles, D. Papailiopoulos, and J. Ellenberg, "Approximate gradient coding via sparse random graphs," *arXiv:1711.06771*, 2017.
[9] S. Dutta, V. Cadambe, and P. Grover, "Coded convolution for parallel and distributed computing within a deadline," in *Proc. IEEE ISIT*, June 2017, pp. 2403–2407.
[10] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Coded Fourier transform," *Proc. Allerton Conf.*, 2017.
[11] S. Dutta, V. Cadambe, and P. Grover, "Short-Dot: Computing large linear transforms distributedly using coded short dot products," in *Proc. NIPS*, 2016, pp. 2100–2108.
[12] G. Suh, K. Lee, and C. Suh, "Matrix sparsification for coded matrix multiplication," in *Proc. Allerton Conf.*, 2017.
[13] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, 2008.
[14] F. Ahmad, S. T. Chakradhar, A. Raghunathan, and T. Vijaykumar, "ShuffleWatcher: Shuffle-aware scheduling in multi-tenant MapReduce clusters." in *Proc. USENIX ATC*, 2014, pp. 1–12.
[15] 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.
[16] S. Gupta and V. Lalitha, "Locality-aware hybrid coded MapReduce for server-rack architecture," *arXiv:1709.01440*, 2017.
[17] K. V. Rashmi, N. B. Shah, D. Gu, H. Kuang, D. Borthakur, and K. Ramchandran, "A solution to the network challenges of data recovery in erasure-coded distributed storage systems: A study on the Facebook warehouse cluster." in *Proc. USENIX HotStorage*, 2013.
[18] H. A. David and H. N. Nagaraja, *Order Statistics*. Wiley, New York, 2003.
[19] P. Brémaud, *Markov chains: Gibbs fields, Monte Carlo simulation, and queues*. Springer Science & Business Media, 2013, vol. 31.
[20] W. Halbawi, Z. Liu, and B. Hassibi, "Balanced Reed-Solomon codes," in *Proc. IEEE ISIT*, July 2016, pp. 935–939.
[21] ——, "Balanced Reed-Solomon codes for all parameters," in *Proc. IEEE ITW*, Sept. 2016, pp. 409–413.
[22] W. Hoeffding, "Probability inequalities for sums of bounded random variables," *J. Am. Stat. Assoc.*, vol. 58, no. 301, pp. 13–30, 1963.