Coded Distributed Computing: Performance Limits and Code Designs

Mohammad Vahid Jamali, Mahdi Soleymani, and Hessam Mahdavifar
Department of Electrical Engineering and Computer Scienc, University of Michigan, Ann Arbor, MI 48109, USA
E-mails: {mvjamali, mahdy, hessam}@umich.edu

***Abstract*—We consider the problem of coded distributed computing where a large linear computational job, such as a matrix multiplication, is divided into k smaller tasks, encoded using an (n, k) linear code, and performed over n distributed nodes. The goal is to reduce the average execution time of the computational job. We provide a connection between the problem of characterizing the average execution time of a coded distributed computing system and the problem of analyzing the error probability of codes of length n used over erasure channels. Accordingly, we present closed-form expressions for the execution time using binary random linear codes and the best execution time any linear-coded distributed computing system can achieve. It is also shown that there exist *good* binary linear codes that attain, asymptotically, the best performance any linear code, not necessarily binary, can achieve. We also investigate the performance of coded distributed computing systems using polar and Reed-Muller (RM) codes that can benefit from low-complexity decoding, and superior performance, respectively, as well as explicit constructions. The proposed framework in this paper can enable efficient designs of distributed computing systems given the rich literature in the channel coding theory.**

I. Introduction

There has been increasing interest in recent years toward applying ideas from coding theory to improve the performance of various computation, communication, and networking applications. For example, ideas from repetition coding has been applied to several setups in computer networks, e.g., by running a request over multiple servers and waiting for the first completion of the request by discarding the rest of the request duplicates [1]–[3]. Another direction is to investigate the application of coding theory in cloud networks and distributing computing systems [4], [5]. A rule of thumb is that when the computational job consists of linear operations, coding techniques can be applied to improve the run-time performance of the system under consideration.

Distributed computing refers to the problem of performing a large computational job over many, say n, nodes with limited processing capabilities. A coded computing scheme aims to divide the job to $k < n$ tasks and then to introduce $n-k$ redundant tasks using an (n, k) code, in order to alleviate the effect of slower nodes, also referred to as *stragglers*. In such a setup, it is assumed that each node is assigned one task and hence, the total number of *encoded tasks* is n equal to the number of nodes.

Recently, there has been extensive research activities to leverage coding schemes in order to boost the performance of distributed computing systems [2], [5]–[14]. For example, [5] has applied coding theory to combat the deteriorating effects of stragglers in matrix multiplication and data shuffling. The authors in [7] considered coded distributed computing in heterogeneous clusters consisting of servers with different computational capabilities.

Most of the work in the literature focus on the application of maximum distance separable (MDS) codes. However, encoding and decoding of MDS codes over real numbers, especially when the number of servers is large, e.g., more than 100, face several barriers, such as numerical stability, and decoding complexity. In particular, decoding of MDS codes is not robust against unavoidable rounding errors when used over real numbers [15]. Employing large finite fields, e.g., coded matrix multiplication using *polynomial codes* in [16], can be an alternative approach. However, applying large finite fields imposes further numerical barriers due to quantization when used over real-valued data.

As we will show in Section III, MDS codes are *theoretically* optimal in terms of minimizing the average execution time of any linear-coded distributed computing system. However, as discussed above, their application comes with some practical impediments, either when used over real-valued inputs or large finite fields, in most of distributed computing applications comprised of large number of local nodes. A sub-optimal yet practically interesting approach is to apply binary linear codes, consisting of 0's and 1's, and then perform the computation over real values. In this case, there is no need for the quantization as a zero in the (i, j)-th element of the generator matrix of the binary linear code means that the i-th task is not included in the j-th encoded task sent to the j-th node while a one means it is included. To this end, in this paper, we consider (n, k) binary linear codes where all computations are performed over real-valued data inputs. A related work to this model is the very recent work in [17] where binary polar codes are applied for distributed matrix multiplication. The authors in [17] justify the application of binary codes over real-valued data and provide a decoding algorithm using polar decoder.

In this work, we connect the problem of characterizing the average execution time of any coded distributed computing system to the error probability of the underlying coding scheme over n uses of erasure channels (see Lemma 1). Using this connection, we characterize the performance limits of distributed computing systems such as the average execution time that any linear code can achieve (see Theorem 2), the average job completion time using binary random linear codes (see Corollary 4), and the best achievable average execution time of any linear code (see Corollary 5) that can, provably, be attained using MDS codes requiring operations over large finite fields. Moreover, we study the gap between the average execution time of binary random linear codes and the optimal performance (see Theorem 7) showing the normalized gap approaches zero as $n \to \infty$ (see Corollary 8). This implies that there exist binary linear codes that attain, asymptotically, the best performance any linear code, not necessarily binary, can achieve. We further study the performance of coded distributed computing systems using polar and Reed-Muller (RM) codes that can benefit from low-complexity decoding and superior performance, respectively.

II. System Model

We consider a distributed computing system consisting of n local nodes with the same computational capabilities. The run time T_i of each local node i is modeled using a shifted-exponential random variable (RV), mainly adopted in the literature [5], [7], [18]. Then, when the computational job is equally divided to k tasks, the cumulative distribution function

This work was supported by the National Science Foundation under the grant CCF–1763348.

(CDF) of T_i is given by

$$\Pr(T_i \leqslant t) = 1 - \exp\left(-\mu(kt-1)\right), \quad \forall\, t \geqslant 1/k, \tag{1}$$

where μ is the exponential rate of each local node, also called the straggling parameter. Using (1) one can observe that the probability of the task assigned to the i-th server not being completed (equivalent to erasure) until time $t \geqslant 1/k$ is

$$\epsilon(t) \triangleq \Pr(T_i > t) = \exp\left(-\mu(kt-1)\right), \tag{2}$$

and is one for $t < 1/k$. Therefore, given any time t, the problem of computing k parts of the computational job over n servers can be interpreted as the traditional problem of transmitting k symbols, using an (n,k) code, over n independent-and-identically-distributed (i.i.d.) erasure channels. Note that the form of the CDF in (1) suggests that $t_0 \triangleq 1/k$ is the (normalized) deterministic time required for each server to process its assigned $1/k$ portion of the total job (all tasks are erased before t_0), while any time elapsed after t_0 refers to the stochastic time as a result of servers' statistical behavior (tasks are not completed with probability $\epsilon(t)$ for $t \geqslant t_0$).

Given a certain code and a corresponding decoder over erasure channels, a *decodable* set of tasks refers to a pattern of unerased symbols resulting in a successful decoding with probability 1. Then, $P_e(\epsilon, n)$ is defined as the probability of decoding failure over an erasure channel with erasure probability ϵ. For instance, $P_e(\epsilon, 1) = \epsilon$ for a $(1,1)$ code. Note that the reason to keep n in the notation is to specify that the number of servers, when the code is used in distributed computation, is also n. Finally, the total job completion time T is defined as the time at which a decodable set of tasks/outputs is obtained from the servers.

III. Fundamental Limits

The following Lemma connects the average execution time of any linear-coded distributed computing system to the error probability of the underlying coding scheme over n uses of an erasure channel.

Lemma 1. *The average execution time of a linear-coded distributed computing system using a given (n,k) code can be characterized as*

$$T_{\text{avg}} \triangleq \mathbb{E}[T] = \int_0^\infty P_e(\epsilon(\tau), n) d\tau \tag{3}$$

$$= \frac{1}{k} + \frac{1}{\mu k} \int_0^1 \frac{P_e(\epsilon, n)}{\epsilon} d\epsilon, \tag{4}$$

where $\epsilon(\tau)$ is defined in (2)*.*

Proof: It is well-known that the expected value of any RV T is related to its CDF $F_T(\tau)$ as $\mathbb{E}[T] = \int_0^\infty (1 - F_T(\tau)) d\tau$. Note that $1 - F_T(\tau) = \Pr(T > \tau)$ is the probability of the event that the job is not completed until some time τ. Therefore, using the system model in Section II, we can interpret $\Pr(T > \tau)$ as the probability of decoding failure $P_e(\epsilon(\tau), n)$ of the code when used over n i.i.d. erasure channels with the erasure probability $\epsilon(\tau)$. This completes the proof of (3). Now given that for the shifted-exponential distribution $d\epsilon(\tau)/d\tau = -\mu k \epsilon(\tau)$, and that $P_e(\epsilon(\tau), n) = 1$ for all $\tau \leqslant 1/k$, we have (4) by the change of variables. ∎

Remark 1. Note that (3) holds given any model for the distribution of the run time of the servers, while (4) is obtained under shifted-exponential distribution, with servers having a same straggling parameter μ, and can be extended to other distributions in a similar approach.

Theorem 2. *The average execution time of any linear-coded distributed computing system can be expressed as*

$$T_{\text{avg}} = \frac{1}{k}\left[1 + \sum_{i=n-k+1}^{n} \frac{1}{i\mu}\right] + \frac{1}{\mu k} \sum_{i=1}^{n-k} \frac{1}{i} p(i,k), \tag{5}$$

where $p(i,k)$ is the average conditional probability of decoding failure, for an underlying decoder, given i encoded symbols are erased at random.

Proof: Using the law of total probability and the definition of $p(i,k)$ we have

$$P_e(\epsilon, n) = \sum_{i=1}^{n} \binom{n}{i} \epsilon^i (1-\epsilon)^{n-i} p(i,k). \tag{6}$$

Accordingly, characterizing T_{avg} requires computing integrals of the form $f_i \triangleq \int_0^1 \epsilon^{i-1}(1-\epsilon)^{n-i} d\epsilon$ for $i = 1, 2, ..., n$. Using part-by-part integration one can find the recursive relation $f_{i+1} = \frac{i}{n-i} f_i$ which results in $1/f_i = i\binom{n}{i}$. Note that $p(i,k) = 1$ for $i > n-k$, since one cannot extract the k parts of the original job from less than k encoded symbols. Then plugging (6) into (4) leads to (5). ∎

Next, we characterize the average execution time using a random ensemble of binary linear codes with full-rank generator matrices. This random ensemble, denoted by $\mathcal{R}(n,k)$, is obtained by picking entries of the $k \times n$ generator matrix independently and uniformly at random followed by removing those matrices not having a full row rank from the ensemble.

Remark 2. Note that (6) together with the integral form in (4) suggest that a coded computing system should always encode with a full-rank generator matrix, otherwise, the average execution time does not converge. This is the reason behind picking the particular ensemble described above. Note that this is in contrast with the conventional block coding, where we can get an arbitrarily small average probability of error over a random ensemble of all $k \times n$ binary generator matrices.

Lemma 3. *The probability that the generator matrix of a code picked from $\mathcal{R}(n,k)$ does not remain full row rank after erasing i columns uniformly at random, denoted by $p_f(i,k)$, can be expressed as*

$$p_f(i,k) = 1 - \frac{\prod_{j=1}^{k}\left(1 - 2^{j-1-n+i}\right)}{\prod_{l=1}^{k}\left(1 - 2^{l-1-n}\right)}. \tag{7}$$

Proof: Define $l(m,k)$ as the probability of k binary uniform random vectors $\mathbf{v}_i \in \mathbb{F}_2^m$ being linearly independent. It is well-known that

$$l(m,k) = \prod_{i=1}^{k}\left(1 - 2^{i-1-m}\right). \tag{8}$$

Let $\tilde{\mathbf{G}}$ denote the $k \times (n-i)$ matrix after removing i columns of the $k \times n$ generator matrix $\mathbf{G}$ uniformly at random. Then

$$p_f(i,k) = \Pr\Big(\{\text{rank}(\tilde{\mathbf{G}}) \neq k\} | \{\text{rank}(\mathbf{G}) = k\}\Big) \tag{9}$$

$$= 1 - \frac{\Pr(\{\text{rank}(\tilde{\mathbf{G}}) = k\})}{\Pr\left(\{\text{rank}(\mathbf{G}) = k\}\right)} \tag{10}$$

$$= 1 - \frac{\prod_{j=1}^{k}\left(1 - 2^{j-1-n+i}\right)}{\prod_{l=1}^{k}\left(1 - 2^{l-1-n}\right)}, \tag{11}$$

where (9) is by the definition of $p_f(i,k)$, (10) is by noting that $\Pr(\{\text{rank}(\mathbf{G}) = k\} | \{\text{rank}(\tilde{\mathbf{G}}) = k\}) = 1$, and (11) is by (8). ∎

Corollary 4. *The average execution time using random linear codes from the ensemble $\mathcal{R}(n,k)$ under maximum a posteriori (MAP) decoding is given by* (5) *while replacing $p(i,k)$ in* (5) *by $p_f(i,k)$, characterized in Lemma 3.*

Proof: The proof is by noting that the optimal MAP decoder fails to recover the k input symbols given $n-i$ unerased encoded symbols if and only if the corresponding $k \times (n-i)$ sub-matrix of the generator matrix of the code is

not full row rank which occurs with probability $p_f(i,k)$. ∎

Remark 3. Theorem 2 implies that the average execution time using linear codes consists of two terms. The first term is independent of the performance of the underlying coding scheme and is fixed given k, n, and μ. However, the second term is determined by the error performance of the coding scheme, i.e., $p(i,k)$ for $i = 1, 2, ..., n-k$, and hence, can be minimized by properly designing the coding scheme.

The following corollary of Theorem 2 demonstrates that MDS codes, if they exist,[1] are optimal in the sense that they minimize the average execution time by eliminating the second term of the right hand side in (5). However, for a large number of servers n, the field size needs to be also large, e.g., $q > n$ for Reed-Solomon (RS) codes.

Corollary 5 (Optimality of MDS Codes)**.** *For given n,k, and underlying field size q, an (n,k) MDS code, if exists, achieves the minimum average execution time that can be attained by any (n,k) linear code.*

Proof: MDS codes have the minimum distance of $d_{\min}^{\text{MDS}} = n-k+1$ and can recover up to $d_{\min}^{\text{MDS}} - 1 = n-k$ erasures leading to $p(i,k) = 0$ for $i = 1, 2, ..., n-k$. Therefore, the second term of (5) becomes zero for MDS codes and they achieve the following minimum average execution time that can be attained by any (n,k) linear code:

$$T_{\text{avg}}^{\text{MDS}} = \frac{1}{k} + \frac{1}{\mu k} \sum_{i=n-k+1}^{n} \frac{1}{i}. \tag{12}$$

∎

Using Theorem 2 and Remark 3, and given that the generator matrix of any (n,k) linear code with minimum distance $d_{\min}$ remains full rank after removing up to any $d_{\min} - 1$ columns, we have the following proposition for the *optimality criterion* in terms of minimizing the average execution time.

Proposition 6 (Optimality Criterion)**.** *An (n,k) linear code that minimizes $\sum_{i=d_{\min}}^{n-k} p(i,k)/i$ also minimizes the average execution time of a coded distributed computing system.*

Although MDS codes meet the aforementioned optimality criterion over large field sizes, to the best of our knowledge, the optimal linear codes, given the field size q and in particular for $q = 2$, per Proposition 6 are not known and have not been studied before, which calls for future studies.

In the following theorem we characterize the gap between the execution time of binary random linear codes and the optimal execution time. Then Corollary 8 proves that binary random linear codes *asymptotically* achieve the normalized optimal execution time, thereby demonstrating the existence of *good* binary codes for distributed computation over real-valued data. The reason we compare the normalized nT_{avg}'s instead of T_{avg}'s is that, using (5), T_{avg} has a factor of $1/k$ and hence, $\lim_{n\to\infty} T_{\text{avg}} = 0$ for a fixed rate[2] $R \triangleq k/n > 0$.

Theorem 7 (Gap of Binary Random Linear Codes to the Optimal Performance)**.** *Let $T_{\text{avg}}^{\text{BRC}}$ denote the average execution time of a coded distributed computing system using binary random linear codes. Then, for any given k, n, we have*

$$\frac{1}{3\mu R(1-R)n} < |nT_{\text{avg}}^{\text{MDS}} - nT_{\text{avg}}^{\text{BRC}}| < \frac{1}{\mu R} \times \left[\frac{v(n)}{n-k-v(n)+1} + nR2^{-v(n)} \ln\left(n-k-v(n)\right) \right], \tag{13}$$

where R is the rate and $v(n)$ is an arbitrary function of n with $0 \leqslant v(n) \leqslant n-k$.

Proof: Using Corollary 4 and Corollary 5, we have

$$\mathcal{S} \triangleq \mu R |nT_{\text{avg}}^{\text{MDS}} - nT_{\text{avg}}^{\text{BRC}}| = \sum_{i=1}^{n-k} \frac{1}{i} p_f(i,k). \tag{14}$$

The lower bound in (13) is by noting that $\mathcal{S} > p_f(n-k,k)/(n-k)$, where $p_f(n-k,k)$ can be expressed as

$$p_f(n-k,k) = 1 - \frac{1-2^{-k}}{1-2^{-n}} \cdot \frac{1-2^{-k+1}}{1-2^{-n+1}} \cdot \ldots \cdot \frac{1-2^{-1}}{1-2^{k-n-1}}. \tag{15}$$

Note that $p_f(n-k,k) = 0$ for $n = k$. For $n > k$, since $1 - 2^{-k+j} > 1 - 2^{-n+j}$ for $j = 0, 1, ..., k-2$, we have

$$p_f(n-k,k) > 1 - \frac{1-2^{-1}}{1-2^{k-n-1}} > 1 - \frac{1-2^{-1}}{1-2^{-1-1}} = \frac{1}{3}. \tag{16}$$

Therefore, $\mathcal{S} > \frac{1}{3(1-R)n}$.

To prove the upper bound, the summation in (14) is split as $\mathcal{S} = \mathcal{S}_1 + \mathcal{S}_2$ where

$$\mathcal{S}_1 \triangleq \sum_{i=n-k-v(n)+1}^{n-k} \frac{1}{i} p_f(i,k) < \frac{v(n)}{n-k-v(n)+1}, \tag{17}$$

$$\mathcal{S}_2 \triangleq \sum_{i=1}^{n-k-v(n)} \frac{1}{i} p_f(i,k). \tag{18}$$

To upper-bound $\mathcal{S}_2$, we first note that $p_f(i,k)$, defined in (7), is a monotonically increasing function of i. Then,

$$\mathcal{S}_2 \leqslant p_f(n-k-v(n),k) \sum_{i=1}^{n-k-v(n)} \frac{1}{i} \tag{19}$$

$$< p_f(n-k-v(n),k) \ln\left(n-k-v(n)\right). \tag{20}$$

We can further upper-bound $p_f(n-k-v(n),k)$ as

$$p_f(n-k-v(n),k) < 1 - \prod_{j=1}^{k} (1 - 2^{j-1-k-v(n)}) \tag{21}$$

$$< 1 - \left[1 - 2^{-v(n)}\right]^k \tag{22}$$

$$\leqslant nR2^{-v(n)}, \tag{23}$$

where (21) is by (7) together with $\prod_{l=1}^{k} \left(1 - 2^{l-1-n}\right) \leqslant 1$, (22) follows by noting that $\prod_{j=1}^{k}(1 - 2^{j-1-k-v(n)}) = \prod_{j'=1}^{k}(1 - 2^{-j'-v(n)}) > [1 - 2^{-v(n)}]^k$, and (23) follows by Bernoulli's inequality $(1-x)^k \geqslant 1 - kx$ for any $0 < x < 1$ and then inserting $k = nR$. ∎

Corollary 8 (Asymptotic Optimality of Binary Random Linear Codes)**.** *The normalized average execution time $nT_{\text{avg}}^{\text{BRC}}$ approaches $nT_{\text{avg}}^{\text{MDS}}$ as n grows large. More precisely, for a given rate R, there exist constants $c_1, c_2 > 0$ such that for sufficiently large n, i.e., $k = nR$, we have*

$$c_1 \frac{1}{n} \leqslant |nT_{\text{avg}}^{\text{MDS}} - nT_{\text{avg}}^{\text{BRC}}| \leqslant c_2 \frac{\log_2 n}{n}. \tag{24}$$

Proof: The lower bound holds with $c_1 = 1/3\mu R(1-R)$ according to the left hand side of (13). Observe that with the choice of $v(n) = 2\log_2 n$ both terms in the right hand side of (13) become $O(\frac{\log_2 n}{n})$. Note that $n-k = n(1-R) \geqslant 2\log_2 n$, for sufficiently large n. Hence, the upper bound of (24) also holds with a proper choice of c_2. ∎

Remark 4. Using (12) and a similar approach to [5], one can show that the asymptotically-optimal encoding rate R^* for an MDS-coded distributed computing system is the solution to

$$(1-R^*)\ln(1-R^*) = \mu(1-R^*) - R^*. \tag{25}$$

Corollary 8 implies that for distributed computation using

[1] It is in general an open problem whether given n, k, and q, there exists an (n,k) MDS code over $\mathbb{F}_q$ [19, Ch. 11.2].

[2] More precisely, the coding rate over field size q is equal to $k\log_2 q/n$ but with slight abuse of terminology we have dropped the factor of $\log_2 q$ since this factor is not relevant for coded distributed computing.

binary random linear codes, the gap of $nT_{\text{avg}}^{\text{BRC}}$ to $nT_{\text{avg}}^{\text{MDS}}$ converges to zero as n grows large. Accordingly, the optimal encoding rate also approaches R^*, described in (25).

IV. Practical Codes and Simulation Results

In this section, simulation results for the expected-time performance of various coding schemes over distributed computing systems are presented. In particular, their gap to the optimal performance are shown and also, their performance gains are compared with the uncoded computation.

A. Polar-Coded Distributed Computation

Binary polar codes are capacity-achieving linear codes with explicit constructions and low-complexity encoding and decoding [20]. Also, the low-complexity $O(n \log n)$ encoding and decoding of polar codes can be adapted to work over real-valued data when dealing with erasures as in coded computation systems, as also noted in [17]. Next, we briefly explain the encoding and decoding procedure of real-valued data using binary polar codes and delineate how we can obtain the average execution times using Lemma 1.

1) Encoding Procedure: Arıkan's $n \times n$ polarization matrix $\mathbf{G}_n = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}^{\otimes r}$ is considered, where $r = \log_2 n$ and $\mathbf{A}^{\otimes r}$ denotes the r-th Kronecker power of $\mathbf{A}$. Next, a design parameter ϵ_d is picked, as specified later in Section IV-C. Then the polarization transform $\mathbf{G}_n$ is applied to a binary erasure channel with erasure probability ϵ_d, BEC(ϵ_d). The erasure probabilities of polarized bit-channels, denoted by $\{Z_i\}_{i=1}^n$, are sorted and the k rows of $\mathbf{G}_n$ corresponding to the indices of the k smallest Z_i's are picked to construct the $k \times n$ generator matrix $\mathbf{G}$. The encoding procedure using the resulting $k \times n$ generator matrix $\mathbf{G}$, which also applies to any (n, k) binary linear code operating over real-valued data, is as follows. First, the computational job is divided into k smaller tasks. Then the j-th encoded task which will be sent to the j-th node, for $j = 1, 2, \ldots, n$, is the sum of all tasks i's for which the (i, j)-th element of $\mathbf{G}$ is 1.

2) Decoding Procedure: The recursive structure of polar codes can be applied for low-complexity detection/decoding of real-valued data using parallel processing for more speedups [21]. It is well-known that in the case of successive cancellation (SC) decoding over BECs, the probability of decoding failure of polar codes is $P_e^{\text{SC}}(\epsilon, n) = 1 - \prod_{i \in \mathcal{A}}(1 - Z_i)$, where $\mathcal{A}$ denotes the set of indices of the selected rows.

Remark 5. Since polar SC decoder is sub-optimal in terms of successful decoding performance, one can think of optimal maximum-likelihood (ML) decoder to attain a lower failure probability at the cost of higher complexities. Consequently, investigating the possibility of attaining close-to-ML performance, e.g., using SC list decoding of polar codes [22], over real-valued data is an interesting problem deserving future studies when taking into account all time-consuming components of a coded distributed computing system.

3) Performance Characterization: Given the decoding method adopted we can find the average execution time using Lemma 1. In particular, when SC decoding is adopted, T_{avg} can be obtained by numerically evaluating the integral of (4) involving $P_e^{\text{SC}}(\epsilon, n)$. Moreover, for the ML decoding, we first estimate the error probability $P_e^{\text{ML}}(\epsilon, n)$ using Monte-Carlo (MC) simulations and then apply (4).

B. RM-Coded Distributed Computation

RM codes are closely related to polar codes, where for an (n, k) RM code the generator matrix $\mathbf{G}$ is constructed by choosing the k rows of $\mathbf{G}_n$ (defined in Section IV-A1) having the largest Hamming weights. It is recently shown that RM codes are capacity achieving over BECs [23], though

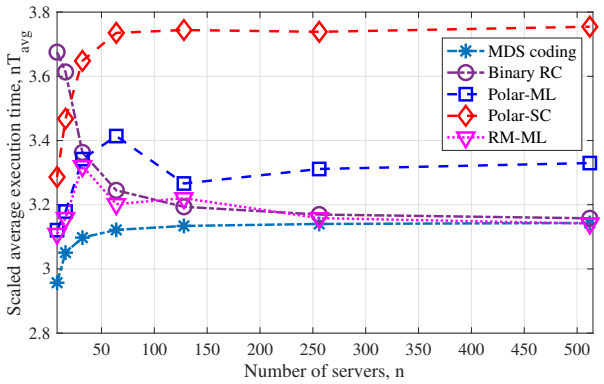

Figure 1. Scaled average execution time of a homogeneous distributed computing system with $\mu = 1$ using various coding schemes for finite number of servers $n = 8$, 16, 32, 64, 128, 256, and 512.

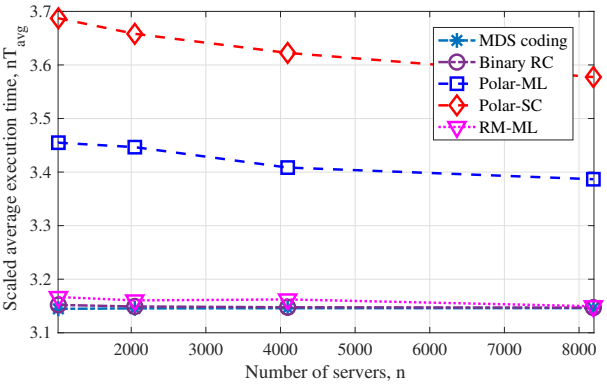

Figure 2. Scaled average execution time of a homogeneous distributed computing system with $\mu = 1$ using various coding schemes for asymptotically large number of servers $n = 1024$, 2048, 4096, and 8192.

under bit-MAP decoding, and numerical results suggest that they actually achieve the capacity with *almost* optimal scaling [24]. There is still a considerable interest in constructing low-complexity decoding algorithms for RM codes attaining such performances. In this paper, we apply the MC-based simulation to estimate $P_e^{\text{ML}}(\epsilon, n)$ for RM codes with the optimal ML decoder, and then evaluate their execution time, numerically, using (4). The inspiration behind considering RM codes in this paper is that they are believed to have the *almost* optimal scaling which, we conjecture, is sufficient for asymptotic optimality, similar to random linear codes in Corollary 8, for coded distributed computing. The simulation results, provided next, support this conjecture.

C. Simulation Results

Numerical results for the performance of the coded distributed computing systems utilizing MDS codes, binary random linear codes, polar codes, and RM codes, are presented in Table I and are compared with the uncoded scenario over small block-lengths. We assume $\mu = 1$ for all numerical results in this section. For MDS and random linear codes, T_{avg} is calculated using (12) and Corollary 4, respectively, and for polar and RM codes, it is numerically evaluated using (4) as discussed in Sections IV-A and IV-B. Then k^* is obtained by minimizing T_{avg} for all possible values of k. We designed the polar code with $\epsilon_d = 0.1$, which is observed to be good enough for this range of block-lengths but one can also attain slightly better performance for polar codes by optimizing over ϵ_d specifically for each n. Characterizing the best ϵ_d as a function of block-length n is left for the future work. In Table I, G_{cod} is defined as the percentage of the gain in T_{avg} compared to the uncoded scenario and g_{opt} is defined as the gap of T_{avg} for the underlying coding scheme to that of MDS codes, in percentage. Intuitively, G_{cod} for a coding

Table I
AVERAGE EXECUTION TIME AND OPTIMAL k^* VALUES FOR DIFFERENT CODING SCHEMES AS WELL AS THEIR GAP $g_{\rm opt}$ TO THE OPTIMAL PERFORMANCE AND THEIR PERFORMANCE IMPROVEMENT GAIN $G_{\rm cod}$ COMPARED TO THE UNCODED COMPUTING.

n	Uncoded	MDS coding	Binary random coding	Polar coding with SC	Polar coding with ML	RM coding with ML
	$(T_{\rm avg}, g_{\rm opt})$	$(T_{\rm avg}, k^*, G_{\rm cod})$	$(T_{\rm avg}, k^*, g_{\rm opt}, G_{\rm cod})$	$(T_{\rm avg}, k^*, g_{\rm opt}, G_{\rm cod})$	$(T_{\rm avg}, k^*, g_{\rm opt}, G_{\rm cod})$	$(T_{\rm avg}, k^*, g_{\rm opt}, G_{\rm cod})$
8	$(0.4647, 25\%)$	$(0.370, 6, 20\%)$	$(0.460, 7, 25\%, 1.1\%)$	$(0.412, 7, 11\%, 12\%)$	$(0.40, 7, 5.5\%, 16\%)$	$(0.389, 7, 5.1\%, 16\%)$
16	$(0.2738, 44\%)$	$(0.191, 11, 31\%)$	$(0.226, 11, 18\%, 18\%)$	$(0.217, 11, 14\%, 21\%)$	$(0.199, 11, 4.2\%, 28\%)$	$(0.198, 11, 3.6\%, 28\%)$
32	$(0.1581, 63\%)$	$(0.0968, 22, 39\%)$	$(0.105, 21, 8.6\%, 34\%)$	$(0.114, 24, 18\%, 28\%)$	$(0.105, 26, 7.9\%, 34\%)$	$(0.104, 26, 7.2\%, 34\%)$
64	$(0.0897, 84\%)$	$(0.0488, 44, 46\%)$	$(0.051, 43, 3.9\%, 44\%)$	$(0.0584, 44, 20\%, 35\%)$	$(0.0533, 46, 9.4\%, 41\%)$	$(0.050, 42, 2.6\%, 44\%)$
128	$(0.0503, 105\%)$	$(0.0245, 88, 51\%)$	$(0.025, 87, 1.9\%, 50\%)$	$(0.0293, 88, 19\%, 42\%)$	$(0.0255, 91, 4.2\%, 50\%)$	$(0.0252, 97, 2.8\%, 50\%)$
256	$(0.0278, 127\%)$	$(0.0123, 175, 56\%)$	$(0.0124, 174, 0.9\%, 56\%)$	$(0.0146, 182, 19\%, 48\%)$	$(0.0129, 186, 5.5\%, 54\%)$	$(0.0123, 166, 0.6\%, 56\%)$
512	$(0.0153, 149\%)$	$(0.0061, 350, 60\%)$	$(0.0062, 349, 0.5\%, 60\%)$	$(0.0073, 388, 19\%, 52\%)$	$(0.0065, 393, 5.9\%, 57\%)$	$(0.0061, 353, 0.1\%, 60\%)$

scheme determines *how much gain* this scheme attains and g_{opt} indicates *how close* this scheme is to the optimal solution. Observe that polar codes with the low-complexity SC decoder achieve large enough $G_{\rm cod}$'s, close to the optimal values of $G_{\rm cod}$, e.g., 52% for $n = 512$ versus 60% for the MDS code. Closer performance to the optimal $T_{\rm avg}$ can be obtained by decoding polar codes with ML decoder, e.g., $g_{\rm opt} = 5.5\%$ for $n = 256$. Figure 1 shows that random linear codes have weak performance in the beginning but they quickly approach the optimal $T_{\rm avg}$ so that they have small gaps to the optimal values, e.g., $g_{\rm opt} = 0.5\%$ for $n = 512$. Also, observe that RM codes always outperform polar codes since, perhaps, they have better distance distribution leading to better $p(i, k)$'s defined in Theorem 2.

In the case of $\mu = 1$, by numerically solving (25), we have for the asymptotically-optimal encoding rate $R^* = 0.6822$. Motivated by this fact, in Figure 2, the rate of all discussed underlying coding schemes is fixed to R^* and $nT_{\rm avg}$ is plotted for moderately large block-lengths, i.e., $T_{\rm avg}$ is not optimized over rates for the results demonstrated in this plot. Additionally, the polar code is designed with $\epsilon_d = 1 - R^* = 0.3178$, which makes the code to be capacity-achieving for an erasure channel with capacity equal to R^*. Note that there is still a gap between polar codes with ML decoder and MDS codes. We believe this is due to the fact that binary polar codes with the 2×2 polarization kernel do not have an optimal scaling exponent [25]. Furthermore, Figure 2 suggests that RM codes attain the optimal performance, and also do so relatively fast, supporting our conjecture in Section IV-B.

V. CONCLUSION

In this paper, we presented a coding-theoretic approach toward coded distributed computing systems by connecting the problem of characterizing their average execution time to the traditional problem of finding the error probability of a coding scheme over erasure channels. Using this connection, we provided results on the performance of coded distributed computing systems, such as their best performance bounds and asymptotic results using binary random linear codes. We further analyzed the performance of polar and RM codes in the context of distributed computing systems. We conjecture that achieving the capacity of BECs with optimal scaling exponent is a sufficient condition for binary codes to be asymptotically optimal, in the sense defined in Theorem 7. We have shown this for binary random linear codes which are well-known to have optimal scaling exponent, even with sparse generator matrices [26], and numerically verified this for RM codes by observing that they attain close to optimal performance using a moderate number of servers. It is also interesting to see whether having an optimal scaling exponent is also a necessary condition for codes to be asymptotically optimal, e.g., whether binary polar codes with the 2×2 polarization kernel are asymptotically optimal or not.

REFERENCES

[1] G. Ananthanarayanan, A. Ghodsi, S. Shenker, and I. Stoica, "Effective straggler mitigation: Attack of the clones," in *Presented as part of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*, 2013, pp. 185–198.

[2] A. Vulimiri, P. B. Godfrey, R. Mittal, J. Sherry, S. Ratnasamy, and S. Shenker, "Low latency via redundancy," in *Proceedings of the ninth ACM conference on Emerging networking experiments and technologies*. ACM, 2013, pp. 283–294.

[3] K. Gardner, S. Zbarsky, S. Doroudi, M. Harchol-Balter, and E. Hyytia, "Reducing latency via redundant requests: Exact analysis," *ACM SIGMETRICS Performance Evaluation Review*, vol. 43, no. 1, pp. 347–360, 2015.

[4] E. Jonas, Q. Pu, S. Venkataraman, I. Stoica, and B. Recht, "Occupy the cloud: Distributed computing for the 99%," in *Proceedings of the 2017 Symposium on Cloud Computing*. ACM, 2017, pp. 445–451.

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

[6] S. Li, M. A. Maddah-Ali, and A. S. Avestimehr, "A unified coding framework for distributed computing with straggling servers," *arXiv preprint arXiv:1609.01690*, 2016.

[7] A. Reisizadeh, S. Prakash, R. Pedarsani, and A. S. Avestimehr, "Coded computation over heterogeneous clusters," *arXiv preprint arXiv:1701.05973*, 2017.

[8] S. Li, M. A. Maddah-Ali, and A. S. Avestimehr, "Coded distributed computing: Straggling servers and multistage dataflows," in *54th Annual Allerton Conference*. IEEE, 2016, pp. 164–171.

[9] ——, "Coding for distributed fog computing," *IEEE Commun. Mag.*, vol. 55, no. 4, pp. 34–40, 2017.

[10] Y. Yang, P. Grover, and S. Kar, "Computing linear transformations with unreliable components," *IEEE Trans. Inf. Theory*, vol. 63, no. 6, pp. 3729–3756, 2017.

[11] K. Lee, C. Suh, and K. Ramchandran, "High-dimensional coded matrix multiplication," in *IEEE Int. Symp. Inf. Theory (ISIT)*. IEEE, 2017, pp. 2418–2422.

[12] S. Dutta, V. Cadambe, and P. Grover, "Short-dot: Computing large linear transforms distributedly using coded short dot products," in *Adv. in Neural Info. Proc. Systems (NIPS)*, 2016, pp. 2100–2108.

[13] S. Wang, J. Liu, and N. Shroff, "Coded sparse matrix multiplication," *arXiv preprint arXiv:1802.03430*, 2018.

[14] A. Mallick, M. Chaudhari, and G. Joshi, "Rateless codes for near-perfect load balancing in distributed matrix-vector multiplication," *arXiv preprint arXiv:1804.10331*, 2018.

[15] N. J. Higham, *Accuracy and stability of numerical algorithms*. Siam, 2002, vol. 80.

[16] Q. Yu, M. Maddah-Ali, and S. Avestimehr, "Polynomial codes: an optimal design for high-dimensional coded matrix multiplication," in *Adv. in Neural Info. Proc. Systems (NIPS)*, 2017, pp. 4403–4413.

[17] B. Bartan and M. Pilanci, "Polar coded distributed matrix multiplication," *arXiv preprint arXiv:1901.06811*, 2019.

[18] G. Liang and U. C. Kozat, "TOFEC: achieving optimal throughput-delay trade-off of cloud storage using erasure codes," in *IEEE Conf. Computer Commun. (INFOCOM)*. IEEE, 2014, pp. 826–834.

[19] F. J. MacWilliams and N. J. A. Sloane, *The theory of error-correcting codes*. New York: North-Holland, 1977.

[20] E. Arikan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Trans. Inf. Theory*, vol. 55, no. 7, pp. 3051–3073, 2009.

[21] M. V. Jamali and H. Mahdavifar, "A low-complexity recursive approach toward code-domain NOMA for massive communications," in *Proc. IEEE Global Commun. Conf. (GLOBECOM), Dec. 2018*, pp. 1–6.

[22] I. Tal and A. Vardy, "List decoding of polar codes," *IEEE Trans. Inf. Theory*, vol. 61, no. 5, pp. 2213–2226, 2015.

[23] S. Kudekar, S. Kumar, M. Mondelli, H. D. Pfister, E. Şaşoğlu, and R. L. Urbanke, "Reed-Muller codes achieve capacity on erasure channels," *IEEE Trans. Inf. Theory*, vol. 63, no. 7, pp. 4298–4316, 2017.

[24] H. Hassani, S. Kudekar, O. Ordentlich, Y. Polyanskiy, and R. Urbanke, "Almost optimal scaling of Reed-Muller codes on BEC and BSC channels," in *IEEE Int. Symp. Inf. Theory*. IEEE, 2018, pp. 311–315.

[25] S. H. Hassani, K. Alishahi, and R. L. Urbanke, "Finite-length scaling for polar codes," *IEEE Trans. Inf. Theory*, vol. 60, no. 10, pp. 5875–5898, 2014.

[26] H. Mahdavifar, "Scaling exponent of sparse random linear codes over binary erasure channels," in *IEEE Int. Symp. Inf. Theory (ISIT)*. IEEE, 2017, pp. 689–693.