Communication-Efficient Gradient Coding for Straggler Mitigation in Distributed Learning

Swanand Kadhe, O. Ozan Koyluoglu, and Kannan Ramchandran
Dept. of Electrical Engineering and Computer Sciences
University of California, Berkeley
Emails: {swanand.kadhe,ozankoyluoglu,kannanr}@berkeley.edu

Abstract—Distributed implementations of gradient-based methods, wherein a server distributes gradient computations across worker machines, need to overcome two limitations: delays caused by slow running machines called *stragglers*, and communication overheads. Recently, Ye and Abbe [ICML 2018] proposed a coding-theoretic paradigm to characterize a fundamental trade-off between computation load per worker, communication overhead per worker, and straggler tolerance. However, proposed coding schemes suffer from heavy decoding complexity and poor numerical stability. In this paper, we develop a communication-efficient gradient coding framework to overcome these drawbacks. Our proposed framework enables using any linear code to design the encoding and decoding functions. When a particular code is used in this framework, its block-length determines the computation load, dimension determines the communication overhead, and minimum distance determines the straggler tolerance. The flexibility of choosing a code allows us to gracefully trade-off the straggler threshold and communication overhead for smaller decoding complexity and higher numerical stability. Further, we show that using a maximum distance separable (MDS) code generated by a random Gaussian matrix in our framework yields a gradient code that is optimal with respect to the trade-off and, in addition, satisfies stronger guarantees on numerical stability as compared to the previously proposed schemes. Finally, we evaluate our proposed framework on Amazon EC2 and demonstrate that it reduces the average iteration time by 16% as compared to prior gradient coding schemes.

I. INTRODUCTION

The scale of training datasets and model parameters in real-world machine learning applications is continuously growing. Therefore, it has become crucial to implement learning algorithms in a distributed fashion. A commonly used distributed learning framework is data parallelism, in which large-scale datasets are distributed over multiple *worker machines* for parallel processing in order to speed up computation.

Gradient descent based algorithms form an important class of learning algorithms, and are popular in practice. In a typical distributed set up using a gradient descent based algorithm, each worker machine computes *partial gradients* using their local data batches, and sends them to a parameter server which aggregates them to update the model parameters. Any distributed implementation of gradient descent (in a synchronous

setting) needs to address the following two challenges: delays caused by stragglers and communication overheads.

The latency performance of every iteration of a synchronous distributed gradient descent algorithm is determined by *stragglers* – workers that are slowed down due to unpredictable factors such as network latency, hardware failures, etc. [1], [2]. Using coding-theoretic ideas to mitigate stragglers has gained significant research attention, see, e.g., [3]–[6] for distributed computing, and [7]–[18] for distributed learning.

The gains due to parallelization are also bottlenecked in practice by heavy communication overheads between workers and the parameter server. This is especially the case for modern deep learning applications using models with millions of parameters (e.g., ResNet [19]). One approach to reduce communication overheads is to compress and quantize gradients, see, e.g., [20]–[23]. Coding theoretic ideas have also been proposed to trade-off computation for saving communication, see, e.g., [24]–[27].

Recently, Ye and Abbe [28], [29] proposed a gradient coding paradigm to mitigate stragglers in distributed gradient aggregation and simultaneously reduce the communication overhead for workers. We refer to this paradigm as communication-efficient gradient coding. The setup consists of n worker machines and a parameter server. Training samples are partitioned into k parts, and every worker is assigned l of the k parts. Every worker first computes the partial gradients each of length d on its assigned samples, then encodes them into a vector of length d/m, and returns the result to the server. The data placement and encoding should ensure that the server can recover the sum of gradients even if any s workers straggle. Note that l specifies the computation load per worker, m specifies the communication saving, and s specifies the straggler tolerance.

The communication-efficient gradient coding paradigm characterizes a three dimensional trade-off between l, m, and s. In particular, Ye and Abbe [28], [29] showed that any coding scheme with communication saving m and straggler tolerance s must incur the computation load of $l \geq k(s + m)/n$. This implies that the higher the straggler tolerance and/or communication saving, the larger is the computation load per worker. They also presented two schemes that achieve the smallest computation load l for a given communication saving m and straggler tolerance s—one based on recursive polynomials and another based on random Gaussian matrices.

The main drawback of the schemes in [28], [29] is that, to recover the gradient sum, the server needs to invert a matrix of size $n-s$. This results in a decoding complexity of $\mathcal{O}\left(n^3\right)$. In addition, the proposed schemes demonstrate poor numerical stability. Specifically, the authors observe that the numerical stability of the recursive polynomials based scheme quickly deteriorates as n becomes larger than 20, and the scheme becomes unstable for $n > 25$. The Gaussian-random-matrix based scheme is observed to be numerically stable for $n \leq 30$, however, its numerical stability performance for larger number of machines remains unclear. The goal of this paper is to design schemes that overcome these drawbacks.

Our Contributions: We present a communication-efficient gradient coding framework which enables gracefully trading off the straggler threshold and communication saving for higher numerical stability and smaller decoding complexity. Our proposed framework uses the *fractional repetition* scheme to redundantly assign training data across workers, similar to [7], wherein the workers are partitioned into several groups and all the workers in a given group are assigned to a subset of samples. We demonstrate that the fractional repetition placement enables one to use any linear code to design the encoding and decoding functions We refer to this framework as **Comm**unication-Efficient **F**ractional **R**epetition-based **G**radient **C**oding *(CommFR-GC)*.

This flexibility of CommFR-GC to use any linear code enables us to choose a suitable code to reduce decoding complexity and achieve strong numerical stability guarantees. As a case study, we demonstrate that using a low-density parity check (LDPC) code in CommFR-FC yields a gradient code with decoding complexity linear in the number of workers. Next, we show that using a maximum distance separable (MDS) code generated by a random Gaussian matrix in CommFR-GC yields an optimal gradient code with stronger guarantees on numerical stability as compared to the schemes in [29]. Specifically, we consider the straggler threshold under a given numerical stability requirement as in [29], and prove that the CommFR-GC with an MDS code achieves higher straggler thresholds for a wide range of parameters. Finally, we evaluate our proposed framework on Amazon EC2 and demonstrate that it reduces the average iteration time by 16% as compared to prior gradient coding schemes.

Related Work: In [27], the authors presented a joint design of data allocation, communication strategy, and gradient coding that mitigates stragglers as well as reduces bandwidth congestion at the parameter server. Our focus on the other hand is on reducing communication overhead at workers along with straggler mitigation. In [13], the authors proposed a straggler-robust gradient descent scheme for linear models that uses LDPC codes to encode the second-moment of the data. In contrast, our proposed coding framework, similar to [29], can be used for any distributed learning problem. In [30]–[32], the authors develop numerically stable coding schemes for polynomial computations and distributed matrix multiplication. Here, we focus on designing numerically stable gradient coding schemes.

II. PROBLEM SETUP

A. Distributed Training

The process of learning the parameters $\mathbf{w} \in \mathbb{R}^d$ of a model given a dataset $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^M$ of M samples, where $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$, can be cast as the *empirical risk minimization* (ERM) problem: $\min_{\mathbf{w}} \frac{1}{M} \sum_{i=1}^M \ell\left(\mathbf{x}_i, y_i; \mathbf{w}\right)$, where $\ell\left(\mathbf{x}_i, y_i; \mathbf{w}\right)$ is a loss function that measures the accuracy of the prediction made by $\mathbf{w}$ on $(\mathbf{x}_i, y_i)$.

In distributed settings, a popular method to approximately solve the ERM is mini-batch stochastic gradient descent (SGD). In every iteration of mini-batch SGD, a (possibly random) subset S_t of B samples is chosen and the model is updated as $\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \frac{\alpha_t}{B} \sum_{i \in S_t} \nabla \ell\left(\mathbf{x}_i, y_i; \mathbf{w}^{(t)}\right)$, where α_t is the learning rate at iteration t. In the remainder of the paper, we focus our attention to a given iteration t, and fix a batch of B samples. We omit the explicit dependence on the iteration t hereafter, since our focus is on a given iteration.

B. Gradient Coding

In this section, we describe the communication-efficient gradient coding setup from [28], [29]. Consider a distributed master-worker setting consisting of n worker machines $W_1, W_2, \ldots, W_n$, and a parameter server. The mini-batch of samples is partitioned into k subsets of equal size, denoted as $D_1, D_2, \ldots, D_k$. Define the gradient vector of the partial data D_i, called *partial gradient*, as $g_i := \sum_{\mathbf{x}_j, \mathbf{y}_j \in D_i} \nabla \ell\left(\mathbf{x}_j, y_j; \mathbf{w}\right)$.

A communication-efficient gradient code is parameterized by three (non-negative) integers: its per worker computation load l, per worker communication saving m, and straggler tolerance s. In particular, given n and k, a gradient code is said to achieve a triple (l, m, s), satisfying $1 \leq l \leq k$, $0 \leq s \leq n - 1$, and $m \geq 1$, if there exist

1) a *placement scheme* that assigns l datasets to each worker;
2) an *encoding scheme* that allows every worker W_i, $1 \leq i \leq n$, to encode its l partial gradients $\{g_{i_1}, g_{i_2}, \ldots, g_{i_l}\}$, total of a dl dimensional vector, to a $\lceil d/m \rceil$-dimensional vector; and
3) a *decoding scheme* that allows the parameter server to decode the sum of gradients $\sum_{i=1}^k g_i$ from any subset of $n - s$ workers.

As in [28], we restrict our attention to linear coding schemes for the sake of low complexity.

It is shown in [28] that, given n and k, a triple (l, m, s) is achievable if and only if

$$\frac{l}{k} \geq \frac{s + m}{n}. \tag{1}$$

This essentially shows that the fractional computation load l/k at each worker must increase with the straggler tolerance s and the communication savings m. Further, one can observe that for $m = 1$, the result reduces to the case of gradient coding without any communication saving considered in [7]–[9].

Coding Schemes in [29]: The schemes proposed in [29] use cyclic placement, which assumes $k = n$,

and assigns to the i-th worker the following l datasets $\{D_i, D_{(i+1) \mod n}, \ldots, D_{(i+l-1) \mod n}\}$. Encoding uses two matrices $B \in \mathbb{R}^{mn \times (n-s)}$ and $V \in \mathbb{R}^{(n-s) \times n}$. The matrices are chosen such that any $(n-s) \times (n-s)$ sub-matrix of V is non-singular, and a specific set of the rows of B is orthogonal to V. (See Appendix A for details.) To recover the gradient sum, the server effectively needs to invert an $(n-s) \times (n-s)$ matrix V_T, where $T \subseteq [n]$ is the set of non-straggling workers and and V_T is the $(n-s) \times |T|$ sub-matrix of V corresponding to the columns indexed by T. The authors present two coding schemes—when V is a Vandermonde matrix, and when V is a random Gaussian matrix.

The main drawback of the schemes in [29] is that, in the decoding phase, the server needs to invert an $(n-s) \times (n-s)$ matrix. This not only increases the decoding complexity, but, more importantly, also results in poor numerical stability. In the next section, we present a novel framework to construct gradient coding schemes which can gracefully trade off their straggler threshold and communication saving for higher numerical stability and/or smaller decoding complexity.

III. COMMUNICATION-EFFICIENT FRACTIONAL REPETITION BASED GRADIENT CODING (COMMFR-GC)

We present a coding framework that utilizes any arbitrary linear code (over $\mathbb{R}$) to design encoding and decoding schemes. We begin with a brief review of linear codes over $\mathbb{R}$ (see, e.g., [33]). A linear code $\mathcal{C}$ of block-length N and dimension K is a K-dimensional subspace of $\mathbb{R}^N$. The minimum distance δ of $\mathcal{C}$ is $\min\{d_H(x,y) : x, y \in \mathcal{C}, x \neq y\}$, where d_H denotes the Hamming distance $d_H(x,y) = |\{1 \leq i \leq N \mid x_i \neq y_i\}|$. Such a code is denoted as an $[N, K, \delta]$ code. The well-known upper bound on the minimum distance, called Singleton bound, is given as $\delta \leq N - K + 1$, and codes which attain this bound with equality are referred to as Maximum Distance Separable (MDS) codes.

An $[N, K]$ linear code $\mathcal{C}$ can be described using a generator matrix $G \in \mathbb{R}^{K \times N}$ of rank K, whose rows form a basis of $\mathcal{C}$. Alternatively, it is possible to describe a linear code $\mathcal{C}$ using a parity-check matrix $H \in \mathbb{R}^{(N-K) \times N}$ of rank $N - K$ such that $\mathcal{C}$ is the null space of H. A generator matrix leads to a *systematic* encoding, if for each $\mathbf{x} \in \mathbb{R}^K$, the codeword $\mathbf{c} = \mathbf{x}G$ contains $\mathbf{x}$ on some k coordinates.

A. CommFR-GC Framework

Consider the parameters n, k, N, and K such that $N \mid n$ and $n \mid kN$. Let $l = kN/n$. Further, let $p = n/N = k/l$. We propose a framework based on fractional repetition placement scheme, considered in the first gradient coding work [7]. This placement enables us to use of any $[N, K, \delta]$ code to construct a communication-efficient gradient code. We refer to the framework as Communication-efficient Fractional Repetition Gradient Coding (CommFR-GC). The CommFR-GC coding framework consists of three components: placement, encoding, and decoding described in the following.

1. Fractional Repetition Placement Scheme: The k datasets $\{D_1, \cdots, D_k\}$ are assigned to n workers as follows.

1) Partition n workers into p groups each of size N, denoted as $L_1, \ldots, L_p$.
2) Each of the N workers in group L_i stores the following l datasets $D^{(i)} = \{D_{(i-1)l+1}, D_{(i-1)l+2}, \cdots, D_{il}\}$.

2. Encoding Scheme (Worker Side): Let $G \in \mathbb{R}^{K \times N}$ be a generator matrix of an $[N, K, \delta]$ code $\mathcal{C}$.

1) Each worker first computes the sum of partial gradients for all datasets assigned to it. In particular, for $1 \leq i \leq p$, every worker in group L_i computes

$$g^{(i)} = \sum_{u \in D^{(i)}} g_u. \qquad (2)$$

2) Each worker in group i, $1 \leq i \leq p$, appends $g^{(i)}$ with $(K\lceil d/K \rceil - d)$ zeros.
3) For $1 \leq i \leq p$, $1 \leq j \leq N$, the j-th worker in group L_i computes

$$\tilde{g}_j^{(i)} = g_{mat}^{(i)} G_j, \qquad (3)$$

where G_j is the j-th column of G, and $g_{mat}^{(i)}$ is a $\lceil d/K \rceil \times K$ matrix obtained by arranging $g^{(i)}$ as

$$g_{mat}^{(i)} = \begin{bmatrix} g^{(i)}(1) & \cdots & g^{(i)}((K-1)\lceil d/K \rceil + 1) \\ g^{(i)}(2) & \cdots & g^{(i)}((K-1)\lceil d/K \rceil + 2) \\ \vdots & \ddots & \vdots \\ g^{(i)}(\lceil d/K \rceil) & \cdots & g^{(i)}(K\lceil d/K \rceil) \end{bmatrix}. $$
$$(4)$$

4) For $1 \leq i \leq p$, the j-th worker in group L_i sends $\tilde{g}_j^{(i)}$ (of length $\lceil d/K \rceil$) to the parameter server.

3. Decoding Scheme (Server Side): The parameter server waits for the first t ($\geq m$) workers from each group, and decodes the gradient sum as follows. (We characterize the exact value of t in Theorem 1.)

1) For a group i, let $i_1, i_2, \cdots, i_t$ be the first t workers to finish. From these t workers, the master receives $\begin{bmatrix} \tilde{g}_{i_1}^{(i)} & \tilde{g}_{i_2}^{(i)} & \cdots & \tilde{g}_{i_t}^{(i)} \end{bmatrix}$.
2) For each group $1 \leq i \leq p$, the master decodes $g_{mat}^{(i)}$ by solving

$$\begin{bmatrix} \tilde{g}_{i_1}^{(i)} & \tilde{g}_{i_2}^{(i)} & \cdots & \tilde{g}_{i_t}^{(i)} \end{bmatrix} = g_{mat}^{(i)} G^{(i)}, \qquad (5)$$

where $G^{(i)}$ is the $K \times t$ sub-matrix of G consisting of the columns $i_1, i_2, \cdots, i_t$.
3) For each group $1 \leq i \leq p$, the master obtains $g^{(i)}$ by rearranging the entries of $g_{mat}^{(i)}$ as a vector and removing the last $(K\lceil d/K \rceil - d)$ zeros (see (4)).
4) From $g^{(1)}, g^{(2)}, \cdots, g^{(p)}$, the master computes the gradient sum as $g = \sum_{i=1}^{p} g^{(i)}$. It is straightforward to verify that $g = \sum_{j=1}^{k} g_j$ from (2).

Example: Consider $n = 8$ workers, $k = 4$ datasets, and gradient-length $d = 4$. We use an $[N = 4, K = 2, \delta = 3]$ MDS code (over $\mathbb{R}$) with the following generator matrix:

$$G = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 2 \end{bmatrix}. \qquad (6)$$
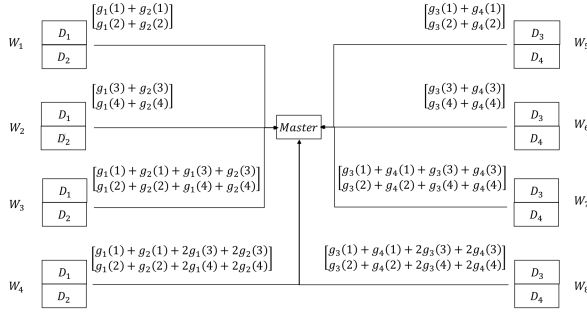
Note that $l = kN/n = 2$.

Fig. 1. Example of CommFR-GC for $n = 8$ workers, $k = 4$ datasets, and gradient-length $d = 4$. We use an $[N = 4, K = 2, \delta = 3]$ MDS code with the generator matrix in (6). This yields a gradient code with computation load $l = 2$, communication saving $m = 2$, and straggler threshold $s = 2$.

Fractional repetition placement scheme: First four workers form group 1 and store D_1, D_2, whereas the last four workers form group 2 and store D_3, D_4 (see Fig. 1).

Encoding scheme: Each worker in group 1 (resp. group 2) computes $g^{(1)} = g_1 + g_2$ (resp. $g^{(2)} = g_3 + g_4$). For $1 \le j \le 4$, the j-th worker in group i computes,

$$\tilde{g}_j^{(i)} = \begin{bmatrix} g^{(i)}(1) & g^{(i)}(3) \\ g^{(i)}(2) & g^{(i)}(4) \end{bmatrix} G_j,$$

where G_j is the j-th column of G (see Fig. 1).

Decoding: Since any two columns of G are independent, we can see that, even if any two workers straggle in each group L_1 and L_2, the master can recover $g^{(1)} = g_1 + g_2$ from non-stragglers in L_1 and $g^{(2)} = g_3 + g_4$ from non-stragglers in L_2. Thus, in the worst case, the scheme can tolerate any two stragglers.

B. Trade-Off Characterization

Here, we characterize the trade-off between the computation load, communication saving, and straggler threshold for the CommFR-GC framework as a function of the underlying code.

Theorem 1. *Given n, k, N, and K such that $N \mid n$ and $n \mid kN$, a CommFR-GC using an $[N, K, \delta]$ code achieves the triple $(l = kN/n, m = K, s = \delta - 1)$.*

Proof: By construction, each worker computes $l = kN/n$ partial gradients, and communicates a vector of length $\lceil d/K \rceil$. To see that the straggler threshold is $\delta - 1$, note that, for a linear code with block-length N and minimum distance δ, it is always possible to solve (5) for any $t \ge N - \delta + 1$. Thus, every group can tolerate up to $\delta - 1$ stragglers, which results in the straggler threshold of $s = \delta - 1$. ∎

Corollary 1. *Given n, k, N, and K such that $N \mid n$ and $n \mid kN$, a CommFR-GC scheme using an $[N, K, N - K + 1]$ MDS code is optimal with respect to (1).*

Remark 1. *CommFR-GC with an $[N, 1, N]$ repetition code reduces to the fractional repetition gradient code in [7].*

IV. ANALYSIS

We analyze how the choice of the code in CommFR-GC impacts the decoding complexity and numerical stability.

A. Decoding Complexity

First, we characterize the decoding complexity of an optimal CommFR-GC scheme that uses a systematic MDS code.

Proposition 1. *Using a systematic $[m+s, m, s+1]$ MDS code in the CommFR-GC framework yields an optimal gradient code with decoding complexity $\mathcal{O}\left(\frac{(\min\{m,s\})^3 n}{\max\{m,s\}} \right)$.*

Proof: The optimality with respect to (1) is immediate. The decoding complexity follows from the fact that to decode the code one needs to invert a matrix of size $t \times t$, where $t = \min\{m, s\}$, for each group, and the number of groups is $n/(m+s)$. ∎

Remark 2. *We compare the complexity with the schemes in [29]. We consider the regime $s = \mathcal{O}(n)$, as the number of stragglers in practical systems is typically a small percentage of the number of workers (see, e.g., [7], [34], [35]). When $m = \mathcal{O}(n)$, CommFR-GC with an MDS code has $\mathcal{O}(n^3)$ decoding complexity, which is same as that of the schemes in [29]. On the other hand, CommFR-GC with an MDS code achieves smaller decoding complexity when m is sub-linear in n. In other words, when $m = \mathcal{O}(n^\alpha)$ for some $\alpha < 1$, then the decoding complexity is $\mathcal{O}(n^{3\alpha})$.*

Next, we demonstrate that is possible to achieve linear decoding complexity (i.e., $\mathcal{O}(n)$) by choosing a suitable code. In particular, we consider the well-known class of computationally efficient codes, called low-density parity check (LDPC) codes (see, e.g., [36]). Note that our focus here is on the regime $s = m = \mathcal{O}(n)$ with sufficiently large n. This regime is of interest for *serverless systems* [37], which can invoke several thousands of workers and suffer from significantly large number of stragglers [34], [35].

An $[N, K]$ LDPC code is defined using a sparse binary parity-check matrix H of size $(N - K) \times N$, which is typically chosen randomly from an ensemble. LDPC codes admit an iterative decoder, called peeling decoder, that can recover a random subset of erasures with complexity $\mathcal{O}(N)$.

As a case study, we consider a specific LDPC ensemble $\mathcal{C}(d_c, d_v)$ defined by the random binary parity-check matrix with each row having d_c ones and each column having d_v ones. Let $p*$ be the largest $\epsilon \in (0, 1)$ such that $\epsilon(1 - (1 - x)^{d_c - 1})^{d_v - 1} < x$ for all $x \in (0, 1]$. Note that $p*$ is referred to as the threshold associated with (d_c, d_v). Let each coordinate of a codeword $\mathbf{c} \in \mathcal{C}$ be independently erased with some probability p. It can be shown that for sufficiently large n, for any $p \le p*$, the peeling decoder recovers the codeword with high probability [36, Chapter 3]. Utilizing this result, we state the following proposition.

Proposition 2. *Consider the case $m = s = O(n)$, and a CommFR-GC using an $[m + s, m]$ LDPC code chosen from the $\mathcal{C}(d_c, d_v)$ ensemble. When each worker straggles*

independently with probability $p \leq p*$, the server can recover the gradient sum with high probability for sufficiently large n, with decoding complexity $\mathcal{O}(n)$.

Remark 3. *As an example, let $n = 30000$ and $m = s = 5000$. Again, we note that this regime is applicable in serverless systems that usually invoke tens of thousands of workers [34], [35], [37]. We consider a CommFR-GC with a $[10000, 5000]$ LDPC code chosen from the ensemble $\mathcal{C}(3, 6)$. For this case, it is possible to show that $p* = 0.4294$ [36, Chapter 3]. Using a peeling decoder will allow the server to recover, with high probability, the gradient sum from a random set of 4249 stragglers in each group with decoding complexity $\mathcal{O}(10000)$. On the other hand, using a $[10000, 5000]$ MDS code in CommFR-GC allows the server to recover from any set of 5000 stragglers in each group with decoding complexity $\mathcal{O}(10000^3)$. This demonstrates how CommFR-GC allows one to back off from the optimal straggler threshold to achieve smaller decoding complexity. In fact, using an LDPC code from a capacity-achieving ensemble will yield a scheme with an optimal straggler threshold, but only for applications in which stragglers are random.*

B. Numerical Stability

In this section, we demonstrate that CommFR-GC using a Gaussian random matrix as a generator matrix achieves significantly better numerical stability compared to the Gaussian matrix based scheme in [29]. To quantify the numerical stability, we consider the straggler threshold achievable under numerical stability parameter κ as follows.

Definition 1. *[29] Given n and k, a coding scheme designed to achieve a triple (l, m, s) is said to have straggler tolerance s_κ under numerical stability constraint κ, if the scheme can tolerate any s_κ stragglers such that the condition number of any matrix involved in the decoding is upper bounded by κ.*

To aid our analysis, we define a function $f_{s,m,\kappa}(t)$ for given m, s, $m \leq t \leq m + s$, and κ as follows:

$$f_{s,m,\kappa}(t) = \frac{1}{\sqrt{2\pi}} \binom{m+s}{t} \left(\frac{Ct}{\kappa(t-m+1)} \right)^{t-m+1}, \quad (7)$$

where $C \leq 6.414$ is a universal positive constant. Now, it is straightforward to verify that, for any

$$\kappa > \max \left\{ \left(\frac{1}{\epsilon\sqrt{2\pi}} \right)^{1/(s+1)} \left(\frac{C(m+s)}{s+1} \right), \frac{Cs}{2} \right\} \quad (8)$$

the function $f_{s,m,\kappa}(t)$ is monotonically decreasing in t, and $f_{s,m,\kappa}(m+s) \leq \epsilon$. Therefore, for any κ satisfying (8), there exists an integer $m \leq t \leq m+s$ such that $f_{s,m,\kappa}(t) \leq \epsilon$. This allows us to analyze the straggler threshold of the proposed scheme under the numerical stability constraint κ in terms of $f_{s,m,\kappa}(\cdot)$ as follows.

Theorem 2. *Consider the CommFR-GC scheme using an $[m+s, m]$ code defined by a generator matrix whose the elements are chosen as i.i.d. standard normal. For any $0 < \epsilon < 1$, and any κ satisfying (8), it holds, with probability at least $1 - \epsilon$,*

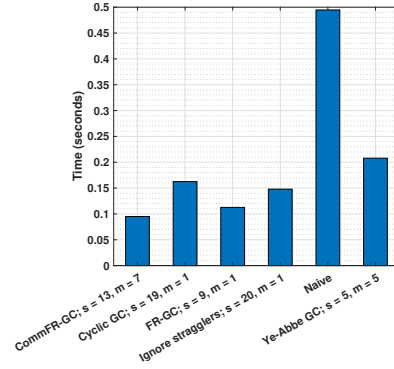| n	s	m	s_κ^{YA}	s_κ
60	3	2	0	2
60	8	2	2	6
60	13	2	6	11
60	3	12	0	1
60	8	12	2	4
60	13	12	2	4
1000	40	10	8	32
1000	90	10	29	78
1000	190	10	85	172
1000	40	210	8	16
1000	90	210	29	48
1000	190	210	85	121

Fig. 2. Average time per iteration for $n = 60$ `t2.micro` worker instances on EC2, with gradient-length $d = 241915$ for various coding schemes.
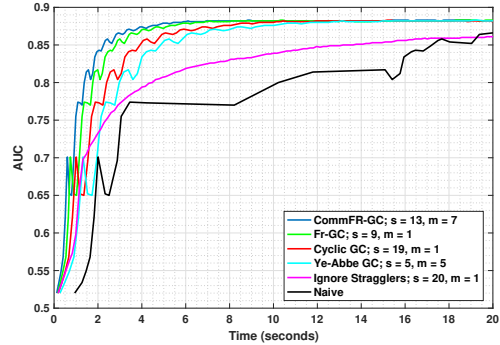

Fig. 3. AUC vs no. of iterations for $n = 60$ `t2.micro` worker instances on EC2, with gradient-length $d = 241915$ for various coding schemes.

that $s_\kappa \geq s + m - t^*$, where t^* is the smallest integer between m and $s + m$ such that $f_{m+s,m,\kappa}(t^*) \leq \epsilon$.

Proof: See Appendix B. ∎

Comparison with the Ye-Abbe scheme [29]: It is not hard to verify that the straggler threshold with numerical stability constraint for the Ye-Abbe scheme using an $(n-s) \times n$ Gaussian random matrix can be directly expressed by substituting $m + s$ by n and m by $n - s$ in Theorem 2. In Table I, we compare the straggler threshold s_κ for CommFR-GC against

the straggler threshold for the scheme in [29], denoted as s_κ^{YA}. We take $n = 60$ as a representative for small n regime, and $n = 1000$ for large n regime. We choose s to be around 5%, 10%, and 20% of n. For m, we consider two regimes, $s \geq m$ and $s \leq m$. We observe that CommFR-GC yields much larger straggler threshold under the required numerical stability. We have carried out simulations for a wide range of values of n, s, m, κ, and ϵ, and observed that CommFR-GC consistently outperforms the Ye-Abbe scheme [29].

V. Experiments on Amazon EC2

In this section, we evaluate CommFR-GC on Amazon EC2, and compare its performance with the other schemes in the literature. Specifically, we compare CommFR-GC using a systematic MDS code against: (1) the naïve scheme, where the data is partitioned among all workers, and the server waits for all workers; (2) the ignoring Stragglers approach, where the data is partitioned among all workers, and the server waits for the first $n-s$ workers; and (3) the gradient coding schemes in [7], [29].

We train a logistic regression model on the Amazon Employee Access dataset from Kaggle, and we compare the average running time and generalization AUC (area under the curve) on a validation set. We used 26,210 training samples, and a model dimension of 241,915 (after one-hot encoding with interaction terms), and adopted Nesterov's Accelerated Gradient (NAG) descent with a constant learning rate, chosen using cross-validation.

We used Python with `mpi4py` package (similar to [7], [9], [29]), `t2.micro` instances on Amazon EC2 as workers, and a single `c3.8xlarge` instance as the server. The results for $n = 60$ workers are shown in Figures 2 and 3. We observe that CommFR-GC achieves the smaller mean time per iteration by 16.02% than other codes. Note that, the curve corresponding to CommFR-GC is always on the left side of the curves corresponding to the other schemes, which means that CommFR-CC achieves the target generalization error faster than the other schemes.

References

[1] T. Hoefler, T. Schneider, and A. Lumsdaine, "Characterizing the influence of system noise on large-scale applications by simulation," in *Proc. of the ACM/IEEE Int. Conf. for High Perf. Comp., Networking, Storage and Analysis*, 2010, pp. 1–11.

[2] J. Dean and L. A. Barroso, "The tail at scale," *Commun. ACM*, vol. 56, no. 2, pp. 74–80, Feb. 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, March 2018.

[4] 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*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds., 2016, pp. 2100–2108.

[5] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Straggler mitigation in distributed matrix multiplication: Fundamental limits and optimal coding," *CoRR*, vol. abs/1801.07487, 2018.

[6] M. F. Aktas, P. Peng, and E. Soljanin, "Effective straggler mitigation: Which clones should attack and when?" *SIGMETRICS Perform. Eval. Rev.*, vol. 45, no. 2, pp. 12–14, Oct 2017.

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

[8] W. Halbawi, N. A. Ruhi, F. Salehi, and B. Hassibi, "Improving distributed gradient descent using reed-solomon codes," *CoRR*, vol. abs/1706.05436, 2017.

[9] N. Raviv, R. Tandon, A. G. Dimakis, and I. Tamo, "Gradient coding from cyclic MDS codes and expander graphs," in *35th International Conference on Machine Learning*, Jul 2018.

[10] S. Li, S. M. M. Kalan, A. S. Avestimehr, and M. Soltanolkotabi, "Near-optimal straggler mitigation for distributed gradient methods," *CoRR*, vol. abs/1710.09990, 2017.

[11] Z. Charles, D. Papailiopoulos, and J. Ellenberg, "Approximate gradient coding via sparse random graphs," *CoRR*, vol. abs/1711.06771, 2017.

[12] Z. Charles and D. Papailiopoulos, "Gradient coding using the stochastic block model," in *2018 IEEE International Symposium on Information Theory (ISIT)*, June 2018, pp. 1998–2002.

[13] R. K. Maity, A. Singh Rawa, and A. Mazumdar, "Robust gradient descent via moment encoding and ldpc codes," in *2019 IEEE International Symposium on Information Theory (ISIT)*, July 2019, pp. 2734–2738.

[14] C. Karakus, Y. Sun, S. Diggavi, and W. Yin, "Redundancy techniques for straggler mitigation in distributed optimization and learning," *Journal of Machine Learning Research*, vol. 20, no. 72, pp. 1–47, 2019.

[15] S. Li, S. M. M. Kalan, Q. Yu, M. Soltanolkotabi, and A. S. Avestimehr, "Polynomially coded regression: Optimal straggler mitigation via data encoding," *CoRR*, vol. abs/1805.09934, 2018. [Online]. Available: http://arxiv.org/abs/1805.09934

[16] R. Bitar, M. Wootters, and S. E. Rouayheb, "Stochastic gradient coding for straggler mitigation in distributed learning," *CoRR*, vol. abs/1905.05383, 2019. [Online]. Available: http://arxiv.org/abs/1905.05383

[17] S. Wang, J. Liu, and N. Shroff, "Fundamental limits of approximate gradient coding," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 3, no. 3, Dec. 2019.

[18] S. Kadhe, O. O. Koyluoglu, and K. Ramchandran, "Gradient coding based on block designs for mitigating adversarial stragglers," in *2019 IEEE International Symposium on Information Theory (ISIT)*, July 2019, pp. 2813–2817.

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 770–778.

[20] W. Wen, C. Xu, F. Yan, C. Wu, Y. Wang, Y. Chen, and H. Li, "Terngrad: Ternary gradients to reduce communication in distributed deep learning," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., 2017, pp. 1509–1519.

[21] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "QSGD: Communication-efficient SGD via gradient quantization and encoding," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., 2017, pp. 1709–1720.

[22] D. Alistarh, T. Hoefler, M. Johansson, N. Konstantinov, S. Khirirat, and C. Renggli, "The convergence of sparsified gradient methods," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., 2018, pp. 5973–5983.

[23] M. Yu, Z. Lin, K. Narra, S. Li, Y. Li, N. S. Kim, A. Schwing, M. Annavaram, and S. Avestimehr, "Gradiveq: Vector quantization for bandwidth-efficient gradient aggregation in distributed cnn training," in *Advances in Neural Information Processing Systems*, 2018, pp. 5123–5133.

[24] M. Kiamari, C. Wang, and A. S. Avestimehr, "On heterogeneous coded distributed computing," in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, Dec 2017, pp. 1–7.

[25] S. Li, M. A. Maddah-Ali, Q. Yu, and A. S. Avestimehr, "A fundamental tradeoff between computation and communication in distributed computing," *IEEE Transactions on Information Theory*, vol. 64, no. 1, pp. 109–128, Jan 2018.

[26] S. Li, M. A. Maddah-Ali, and A. S. Avestimehr, "Compressed coded distributed computing," in *2018 IEEE International Symposium on Information Theory (ISIT)*, June 2018, pp. 2032–2036.

[27] A. Reisizadeh, S. Prakash, R. Pedarsani, and A. S. Avestimehr, "Codedreduce: A fast and robust framework for gradient aggregation

in distributed learning," *CoRR*, vol. abs/1902.01981, 2019. [Online]. Available: https://arxiv.org/abs/1902.01981

[28] M. Ye and E. Abbe, "Communication-computation efficient gradient coding," in *35th International Conference on Machine Learning*, Jul 2018.

[29] ——, "Communication-computation efficient gradient coding," *CoRR*, vol. abs/1802.03475, 2018. [Online]. Available: http://arxiv.org/abs/1802.03475

[30] M. Fahim and V. R. Cadambe, "Numerically stable polynomially coded computing," in *2019 IEEE International Symposium on Information Theory (ISIT)*, July 2019, pp. 3017–3021.

[31] A. M. Subramaniam, A. Heidarzadeh, and K. R. Narayanan, "Random khatri-rao-product codes for numerically-stable distributed matrix multiplication," in *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, Sep. 2019, pp. 253–259.

[32] H. Jeong, Y. Yang, and P. Grover, "Systematic matrix multiplication codes," in *2019 IEEE International Symposium on Information Theory (ISIT)*, July 2019, pp. 1–5.

[33] T. Marshall, "Coding of real-number sequences for error correction: A digital signal processing problem," *IEEE Journal on Selected Areas in Communications*, vol. 2, no. 2, pp. 381–392, March 1984.

[34] V. Gupta, S. Wang, T. A. Courtade, and K. Ramchandran, "Oversketch: Approximate matrix multiplication for the cloud," in *IEEE International Conference on Big Data, Big Data 2018, Seattle, WA, USA, December 10-13, 2018*, 2018, pp. 298–304.

[35] V. Gupta, S. Kadhe, T. A. Courtade, M. Mahoney, and K. Ramchandran, "Oversketched newton: Fast convex optimization for serverless systems," in *ICML Workshop on Coding Theory for Large-scale Machine Learning*, June 2019.

[36] T. Richardson and R. Urbanke, *Modern coding theory*. Cambridge University Press, 2008.

[37] 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*, ser. SoCC '17, 2017, pp. 445–451.

[38] Z. Chen and J. Dongarra, "Condition numbers of gaussian random matrices," *SIAM Journal on Matrix Analysis and Applications*, vol. 27, no. 3, pp. 603–620, 2005.

APPENDIX A
CODING SCHEMES OF [29]

1. Placement: The schemes proposed in [29] use cyclic placement, which assumes $k = n$, and assigns to the i-th worker the following l datasets $\{D_i, D_{(i+1) \mod n}, \ldots, D_{(i+l-1) \mod n}\}$.

2. Encoding: The encoding matrix is computed as $G = BV$, where B is an $mn \times (n-s)$ matrix B and V is an $(n-s) \times n$ matrix. The matrix V must have the property that any of its $(n-s) \times (n-s)$ sub-matrix is non-singular. Further, the matrix B must be such that a specific set of its rows are orthogonal to V (see [29] for details). The i-th worker arranges its partial gradients in a $\lceil d/m \rceil \times mn$ matrix g_{mat}, and computes $g_{mat}G_i$, where G_i is the i-th column of G. The orthogonality property ensures that the i-column of G has a support corresponding to the partial gradients assigned to the i-th worker.

3. Decoding: To recover the gradient sum, the server effectively needs to invert an $(n-s) \times (n-s)$ matrix V_T, where $T \subseteq [n]$ is the set of non-straggling workers and and V_T is a $(n-s) \times |T|$ submatrix of a $(n-s) \times n$ matrix V corresponding to columns indexed by T.

The authors present two coding schemes—when V is a Vandermonde matrix, and when V is a random Gaussian matrix.

APPENDIX B
PROOF OF THEOREM 2

The proof leverages the following result on the condition number of a random Gaussian matrix from [38, Theorem 4.5].

Lemma 1. *([38]) For any $u \geq 2$, $v \geq 2$, and $x \geq |v - u| + 1$, the condition number of $u \times v$ matrix M with i.i.d. standard normal elements satisfies*

$$Pr\left(\frac{\text{cond}(M)}{v/(|v-u|+1)} > x\right) \leq \sqrt{2\pi}\left(\frac{C}{x}\right)^{|v-u|+1}, \quad (9)$$

where $\text{cond}(M)$ denotes the condition number of M, and $C \leq 6.414$ is a universal positive constant independent of u, v, and κ.

For a subset $T \subseteq [s+m]$, let G_T denote the submatrix of G consisting of columns of G indexed by T. Suppose the server waits for the first t workers from each group, where $m \leq t \leq s + m$. First, note that if

$$\max_{T \subseteq [s+m], |T|=t} \text{cond}(G_T) \leq \kappa, \quad (10)$$

then $s_\kappa \geq s + m - t$ Next, we show that (10) is satisfied with probability at least $1 - \epsilon$, if $f_{s,m,\kappa}(\epsilon) \leq \epsilon$. Towards this, consider the probability that the maximum condition number over submatrices is greater than κ as follows:

$$\Pr\left(\max_{T \subseteq [s+m], |T|=t} \text{cond}(G_T) > \kappa\right) \quad (11)$$

$$\overset{(a)}{\leq} \binom{s+m}{t} \Pr(\text{cond}(G_T) > \kappa) \quad (12)$$

$$\overset{(b)}{<} \binom{s+m}{t} \frac{1}{\sqrt{2\pi}} \left(\frac{6.414t}{\kappa(t-m+1)}\right)^{t-m+1} \quad (13)$$

$$\overset{(c)}{=} f_{m+s,m,\kappa}(t), \quad (14)$$

$$\overset{(d)}{\leq} \epsilon, \quad \forall t \geq t^* \quad (15)$$

where (a) follows from the union bound, (b) follows from (9),[1] (c) follows from (7), and (d) follows from the definition of t^*. Therefore, if $f_{m+s,m,\kappa}(t) \leq \epsilon$, (10) is satisfied with probability at least $1 - \epsilon$.

---

[1] Since the condition on κ in (8) ensures that $\kappa > t - m + 1$ for any $C \geq 3$, the conditions of Theorem 1 hold.