Robust Distributed Bayesian Learning with Stragglers via Consensus Monte Carlo

Hari Hara Suthan Chittoor,   Osvaldo Simeone

KCLIP Lab, Department of Engineering, Kings College London, UK.

Email: {hari.hara, osvaldo.simeone}@kcl.ac.uk

Abstract—This paper studies distributed Bayesian learning in a setting encompassing a central server and multiple workers by focusing on the problem of mitigating the impact of stragglers. The standard one-shot, or embarrassingly parallel, Bayesian learning protocol known as consensus Monte Carlo (CMC) is generalized by proposing two straggler-resilient solutions based on grouping and coding. Two main challenges in designing straggler-resilient algorithms for CMC are the need to estimate the statistics of the workers' outputs across multiple shots, and the joint non-linear post-processing of the outputs of the workers carried out at the server. This is in stark contrast to other distributed settings like gradient coding, which only require the per-shot sum of the workers' outputs. The proposed methods, referred to as Group-based CMC (G-CMC) and Coded CMC (C-CMC), leverage redundant computing at the workers in order to enable the estimation of global posterior samples at the server based on partial outputs from the workers. Simulation results show that C-CMC may outperform G-CMC for a small number of workers, while G-CMC is generally preferable for a larger number of workers.

Index Terms—Distributed Bayesian learning, stragglers, Consensus Monte Carlo, grouping, coded computing

I. INTRODUCTION

One of the main problems in distributed computing systems [1]–[4] is the presence of stragglers – i.e., working machines whose random computing time is much larger than other machines [5]. The effect of stragglers may be mitigated by leveraging redundant storage and computing at the workers, whereby each worker is allocated, and computes over, multiple data shards. State-of-the-art techniques leverage *grouping*, whereby groups of workers are assigned the same shards and compute the same output, and/or *coding*, whereby computed outputs are coded at the workers and jointly decoded at the server [6]–[8].

Existing work on grouping and coded distributed computing for machine learning applications focuses on frequentist learning. In *frequentist learning*, the goal is to identify a single model parameter vector that approximately minimizes the training loss, e.g., via gradient descent [6]–[8]. Frequentist learning is limited in its ability to quantify uncertainty, incorporate prior knowledge, guide active learning, and enable continual learning. *Bayesian learning* provides a principled approach to address all these limitations, at the cost of an increase in computational complexity [9]–[13].

Scalable implementations of Bayesian learning are based on either *variational inference (VI)* – replacing integration
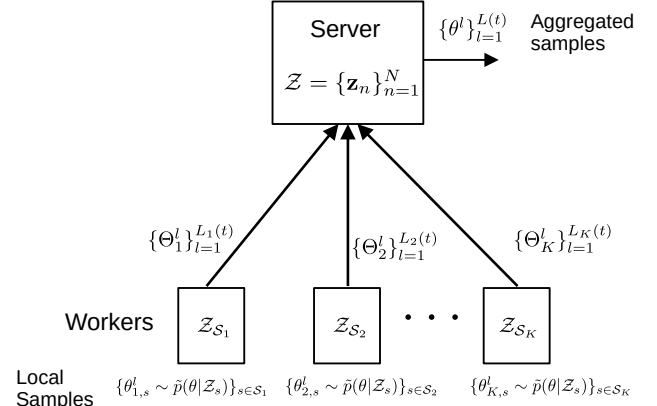
Fig. 1. Distributed Bayesian learning via Consensus Monte Carlo (CMC).

with optimization over an approximate posterior distributions – or *Monte Carlo (MC) sampling* – replacing integration with sampling from the posterior distribution [11]. VI-based protocols for distributed Bayesian learning follow the same general principles of standard distributed frequentist learning (e.g., [14] and references therein). Distributed MC sampling protocols are either one-shot, i.e., embarrassingly parallel [15]–[17]; or else based on iterative gradient-based methods [18], [19] .

In this paper, we focus on the standard one-shot protocol known as *Consensus Monte Carlo (CMC)* [15]. CMC aims at obtaining samples from the global posterior distribution based on local sampling at the workers and aggregation at the server (see Fig. 1). CMC assumes that all workers respond to the server by delivering their local samples before a global sample can be produced by the server. This paper considers, for the first time, the problem of stragglers for CMC.

Two extensions of CMC are proposed that obtain resilience to stragglers based on grouping and coding. The first protocol, referred to as *Group-based CMC (G-CMC)*, requires the partition of workers into groups, with each group being responsible for the computation of local samples for a given subset of shards. In contrast to grouping methods proposed for frequentist learning and distributed computing [7], [20], a novel feature of G-CMC scheme is that all the computed samples can be eventually utilized, even if produced by straggling workers. The second protocol, *Coded-CMC (C-CMC)* applies an erasure correcting code to the produced samples, in a manner similar to gradient coding [6]. Unlike gradient coding, C-CMC requires the design of a novel pre-processing step of the local samples in order to enable CMC-based aggregation at the server.

II. SYSTEM MODEL

This paper considers the problem of drawing samples from a posterior distribution on a large training data set to implement Bayesian learning via Monte Carlo (MC) sampling. Let $\mathcal{Z} = \{\mathbf{z}_n\}_{n=1}^N$ represents the training data and $\theta \in \mathbb{R}^d$ represents the model parameter vector. The global posterior distribution is given as

$$\text{(Global Posterior)} \qquad p(\theta|\mathcal{Z}) \propto p(\theta)p(\mathcal{Z}|\theta), \qquad (1)$$

where $p(\theta)$ is the prior distribution and $p(\mathcal{Z}|\theta)$ is the likelihood. We assume that the data points are conditionally independent and identically distributed (i.i.d.), given the model parameter vector θ, i.e., $p(\mathcal{Z}|\theta) = \prod_{n=1}^N p(\mathbf{z}_n|\theta)$. The goal is to draw L samples $\{\theta^l\}_{l=1}^L$ from the global posterior $p(\theta|\mathcal{Z})$.

We adopt a data center computing platform that consists of a server and K workers, as shown in Fig. 1. The training data $\mathcal{Z}$ is partitioned into K disjoint shards $\mathcal{Z} = \{\mathcal{Z}_s\}_{s=1}^K$, each of size N/K where K assumed to be an integer divisor of N, and allocated to the workers by following a data allocation scheme. We allow for a redundant shard allocation, so that each shard is allocated to r workers, where $r \in [K] \triangleq \{1, ..., K\}$ is referred to as the redundancy parameter. Each worker k has a set $\mathcal{S}_k \subseteq [K]$ of $|\mathcal{S}_k| = r$ shards, which are denoted as $\mathcal{Z}_{\mathcal{S}_k} = \{\mathcal{Z}_s\}_{s \in \mathcal{S}_k}$, with per worker storage capacity Nr/K. Following CMC, we assume that the sampling from each subposterior,

$$\text{(Subposterior)} \quad \tilde{p}(\theta|\mathcal{Z}_s) \propto p(\theta)^{1/K} p(\mathcal{Z}_s|\theta), \qquad (2)$$

is tractable, where $p(\mathcal{Z}_s|\theta) = \prod_{\mathbf{z} \in \mathcal{Z}_s} p(\mathbf{z}|\theta)$ represents the local likelihood function for the s-th shard. In (2), the prior is underweighted as $p(\theta)^{1/K}$ in order to preserve the total prior, so that the global posterior (1) can be expressed as the product of subposteriors $p(\theta|\mathcal{Z}) \propto \prod_{s=1}^K \tilde{p}(\theta|\mathcal{Z}_s)$.

Each worker k computes in parallel r samples $\theta_{k,s}^l \sim \tilde{p}(\theta|\mathcal{Z}_s)$ for all the r allocated shards $\mathcal{Z}_{s: \; s \in \mathcal{S}_k}$, where index $l \in \{1, 2, \cdots\}$ runs over the generated samples. We refer to the collection of such samples as $\Theta_k^l = \{\theta_{k,s}^l\}_{s \in \mathcal{S}_k}$. The produced samples may be processed at the worker, and the outcome of this calculation is sent to the server. The server uses this information to produce global samples θ^l, that are approximately distributed according to the global posterior distribution (1).

We assume that the wall-clock time ΔT_k^l required to compute any l-th batch Θ_k^l of r local samples from the subposteriors (2) of the shards allocated to each worker k is random with mean ηr, for some $\eta > 0$. The computing times $\{\Delta T_k^l\}_{k \in [K]}$ are i.i.d. across the workers and across index l [2], [6], [7]. An example distribution of the computing time is Pareto with scale-shape parameters $(\eta r(\beta-1)/\beta, \beta)$, which gives mean ηr, with $\eta > 0, \beta > 1$ as constants [7].

For any continuous time t, define as $L(t)$ the number of global samples produced at the server based on the information received so far from the workers. Following the prior works [16], [21], we evaluate the error of a CMC algorithm by fixing a test function $f(\cdot)$, and comparing the empirical average obtained with the produced global samples, $\{\theta^l\}_{l=1}^{L(t)}$, available at time t with the corresponding ensemble average $\text{E}_{p(\theta|\mathcal{Z})}[f(\theta)]$ with respect to the true posterior distribution $p(\theta|\mathcal{Z})$ in (1). This can

be written as

$$\text{err}(t) = \frac{\left| \frac{1}{L(t)} \sum_{l=1}^{L(t)} f(\theta^l) - \text{E}_{p(\theta|\mathcal{Z})}[f(\theta)] \right|}{\text{E}_{p(\theta|\mathcal{Z})}[f(\theta)]}. \qquad (3)$$
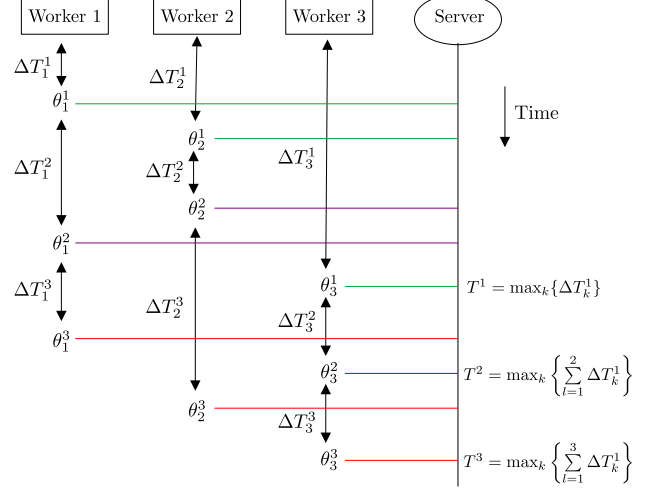

Fig. 2. An illustration of computing times at $K = 3$ workers and at the server. As an example, at time instant T^2, the server has access to $L_1(T^2) = 3, L_2(T^2) = 2, L_3(T^2) = 2$ local samples from the respective workers, and hence it can aggregate such sets of local samples, producing $L(T^2) = 2$ global samples.

III. CMC WITH STRAGGLERS

In this section, we describe the standard CMC protocol in the context of the system under study with random computing times. The purpose of this novel formulation is to study the effect of stragglers in the CMC protocol [15]. Following [15], we focus here on the standard case with no computing redundancy, i.e., $r = 1$, and we let each data shard, $\mathcal{Z}_s$, be allocated only to worker $k = s$. Throughout this section, we accordingly simplify the notation by writing θ_k^l for the sample $\theta_{k,k}^l$ generated at worker k for the k-th shard $\mathcal{Z}_k$. Note also that we have $\Theta_k^l = \{\theta_{k,k}^l\} = \{\theta_k^l\}$.

Each worker k communicates a generated sample θ_k^l to the server as soon as it is produced, where index $l \in \{1, 2, \cdots\}$ runs over the samples. Given the model described in Section II, we denote as $L_k(t)$ the number of samples $\{\theta_k^l\}_{l=1}^{L_k(t)}$ received by the server up to time t from worker k, which is given by

$$L_k(t) = \max\left\{ l : \sum_{l'=1}^{l} \Delta T_k^{l'} \leq t \right\}. \qquad (4)$$

As soon as all the K local samples $\{\theta_k^l\}_{k=1}^K$ are received, a global sample θ^l can be computed at the server by aggregating the corresponding local samples $\{\theta_k^l\}_{k=1}^K$. Therefore, at time t, the number of global samples aggregated at the server is $L(t) = \min_{k \in [K]} L_k(t)$. An illustration of the computing times of each worker and the server are shown in Fig 2, with $K = 3$ workers. The computing time of global samples at the server is limited by the computing time of worker 3.

CMC makes the working assumption that the local samples θ_k^l are Gaussian $\mathcal{N}(\mu_k, C_k)$ with mean μ_k and covariance C_k. Under this condition, which is practically and approximately

valid only as $N/K \to \infty$, the optimal aggregation function is [15]

$$\theta^l = \sum_{k=1}^{K} W_k \theta_k^l, \qquad (5)$$

with weight matrices

$$W_k = \left(\sum_{k=1}^{K} C_k^{-1} \right)^{-1} C_k^{-1}. \qquad (6)$$

The weight matrices in (6) are not directly computable, since the parameters $\{\mu_k, C_k\}_{k=1}^{K}$ are unknown. However, at time t, the server can estimate the mean $\mu_k(t)$ and covariance $C_k(t)$ of the subposterior $\tilde{p}(\theta|\mathcal{Z}_k)$ by using the $L_k(t)$ samples $\{\theta_k^l\}_{l=1}^{L_k(t)}$ received from the worker k up to time t as

$$\hat{\mu}_k(t) = \frac{1}{L_k(t)} \sum_{l=1}^{L_k(t)} \theta_k^l, \text{ and} \qquad (7a)$$

$$\hat{C}_k(t) = \sigma^2 I + \frac{1}{L_k(t)} \sum_{l=1}^{L_k(t)} \left(\theta_k^l - \hat{\mu}_k(t) \right) \left(\theta_k^l - \hat{\mu}_k(t) \right)^T, \quad (7b)$$

respectively, where σ^2 is a regularization parameter. Using these estimates CMC approximates the weight matrices in (6) and obtain an estimate of the global sample θ^l, for each $l \in [L(t)]$, at time t, using (5), as

$$\theta^l(t) = \sum_{k=1}^{K} \left(\sum_{k=1}^{K} (\hat{C}_k(t))^{-1} \right)^{-1} (\hat{C}_k(t))^{-1} \theta_k^l. \qquad (8)$$

At any time T^l at which the sever has received a new set of local samples $\{\theta_k^l\}_{k\in[K]}$, the server computes all global samples $\{\theta^{l'}(T^l)\}_{l'=1}^{l}$ using (8), with the updated estimates of covariance matrices, $\{\hat{C}_k(T^l)\}_{k\in[K]}$, calculated using (7b). CMC is summarized in Algorithm 1.

Algorithm 1 CMC with straggling workers

1: **Input:** Number of workers K, data shards $\{\mathcal{Z}_k\}_{k=1}^{K}$
2: **Data allocation:**
3: **for** each $k \in [K]$ **do**
4:     Allocate shard $\mathcal{Z}_k$ to worker k
5: **end for**
6: **At worker k:** set $l = 1$
7: **repeat**
8:         compute l-th sample $\theta_k^l \sim \tilde{p}(\theta|\mathcal{Z}_k)$
9:         when completed, i.e., at time $\sum_{l'=1}^{l} \Delta T_k^{l'}$, send sample θ_k^l to the server
10: **end repeat**
11: **At the server:** set $l = 1$
12: **repeat**
13:     when receiving all l-th local samples $\{\theta_k^l\}_{k\in[K]}$, i.e., at time $T^l = \max_{k\in[K]}(\sum_{l'=1}^{l} \Delta T_k^{l'})$, compute the covariance matrices $\{\hat{C}_k(T^l)\}_{k\in[K]}$ using (7),
14:     for each $l' \in [l]$ compute $\theta^{l'}(T^l)$ using (8) with covariance matrices $\{\hat{C}_k(T^l)\}_{k\in[K]}$
15: **end repeat**

IV. GROUP-BASED CMC (G-CMC)

In this section, we propose a protocol named *G-CMC* that aims at leveraging the redundancy in data allocation to mitigate the effect of stragglers on the performance of CMC. The approach clusters all the workers into groups, similar to [7], [20], [22], [23], and allocates the same set of r shards to all the workers in a group. In this way, in order to generate the l-th global sample θ^l, the server must only wait to receive one batch of r local samples from the fastest worker in each group.

To elaborate, G-CMC partitions the set of K shards, $\{\mathcal{Z}_s\}_{s\in[K]}$, into G disjoint groups $\{\mathsf{Z}_g\}_{g\in[G]}$ each having r shards, and the set of workers, $[K]$, into G disjoint groups $\{\mathsf{K}_g\}_{g\in[G]}$ each having r workers. Accordingly, we have $K = Gr$. For any $g \in [G]$, the group of shards Z_g is allocated exclusively to all the workers in group K_g, i.e., $\mathcal{Z}_{\mathcal{S}_k} = \mathsf{Z}_g$ for all $k \in \mathsf{K}_g$. Therefore, each data shard is available at r workers, and each worker has access to exactly r shards. For $r = 1$, we get $K = G$, i.e., each group has exactly one user, and G-CMC is equivalent to CMC (see Section III).

The main idea underlying G-CMC is to treat each group as a "super-worker", and apply the CMC protocol presented in Section III across the G "super-workers". The l-th batch of r local samples received from the g-th group K_g is $\Theta_{\mathsf{K}_g}^l = \{\theta_{\mathsf{K}_g,s}^l \sim \tilde{p}(\theta|\mathcal{Z}_s)\}_{s:\ \mathcal{Z}_s\in\mathsf{Z}_g}$. We denote as $L_{\mathsf{K}_g}(t) = \sum_{k\in\mathsf{K}_g} L_k(t)$, for $g \in [G]$, the number of batches of samples received from the group K_g up to time t, where $L_k(t)$ is the number (4) of batches of samples generated by the user $k \in \mathsf{K}_g$ up to time t. The $L_{\mathsf{K}_g}(t)$ batches of samples $\{\Theta_{\mathsf{K}_g}^l\}_{l\in[L_{\mathsf{K}_g}(t)]}$ are received at the server from the group K_g at time instants $\{T_{\mathsf{K}_g}^l\}_{l\in[L_{\mathsf{K}_g}(t)]}$ respectively, where $T_{\mathsf{K}_g}^l$ is the l-th order statistic, i.e., the l-th smallest value, of the variables $\{\{\sum_{l'=1}^{i} \Delta T_k^{l'}\}_{i\in[L_k(t)]}\}_{k\in\mathsf{K}_g}$.

As soon as all the l-th batches of samples, $\{\Theta_{\mathsf{K}_g}^l\}_{g\in[G]}$, with each batch having r local samples are received at the server from all the G groups $\{\mathsf{K}_g\}_{g\in[G]}$, a global sample θ^l is computed at the server by aggregating the corresponding $K = Gr$ local samples $\bigcup_{g\in[G]} \Theta_{\mathsf{K}_g}^l$, and hence G-CMC is resilient to $G(r-1)$ stragglers, as long as K is a multiple of r (see also Section VI). Therefore, at time t, the number of global samples aggregated at the server is $L(t) = \min_{g\in[G]} L_{\mathsf{K}_g}(t)$. To this end, the server estimates the mean $\mu_s(t)$ and covariance $C_s(t)$ of the subposterior $\tilde{p}(\theta|\mathcal{Z}_s)$, for $\mathcal{Z}_s \in \mathsf{Z}_g$, by using the $L_{\mathsf{K}_g}(t)$ samples $\{\theta_{\mathsf{K}_g,s}^l \in \Theta_{\mathsf{K}_g}^l\}_{l\in[L_{\mathsf{K}_g}(t)]}$ received from the group K_g up to time t as

$$\hat{\mu}_s(t) = \frac{1}{L_{\mathsf{K}_g}(t)} \sum_{l=1}^{L_{\mathsf{K}_g}(t)} \theta_{\mathsf{K}_g,s}^l, \text{ and} \qquad (9a)$$

$$\hat{C}_s(t) = \sigma^2 I + \frac{1}{L_{\mathsf{K}_g}(t)} \sum_{l=1}^{L_{\mathsf{K}_g}(t)} (\theta_{\mathsf{K}_g,s}^l - \hat{\mu}_s(t))(\theta_{\mathsf{K}_g,s}^l - \hat{\mu}_s(t))^T, \qquad (9b)$$

respectively, where σ^2 is a regularization parameter. Using these estimates, the weight matrices in (6) are approximated to obtain the global sample θ^l, for each $l \in [L(t)]$, at time t, using (5), as

$$\theta^l(t) = \sum_{s=1}^{K} \left(\sum_{s=1}^{K} (\hat{C}_s(t))^{-1} \right)^{-1} (\hat{C}_s(t))^{-1} \theta_{\mathsf{K}_g,s}^l. \qquad (10)$$

At any time T^l at which the sever has received a new set of batches of samples $\{\Theta^l_{\mathsf{K}_g}\}_{g\in[G]}$, the server computes all global samples $\{\theta^{l'}(T^l)\}^l_{l'=1}$ using (10) with the updated estimates of covariance matrices, $\{\hat{C}_s(T^l)\}_{s\in[K]}$, calculated using (9b). G-CMC is summarized in Algorithm 2.

Algorithm 2 Group-based CMC (G-CMC)

1: **Input:** Partition the set of workers $[K]$ into G groups $\{\mathsf{K}_g\}_{g\in[G]}$ each having r workers, Partition the set of shards $\{\mathcal{Z}_s\}^K_{s=1}$ into G groups $\{\mathsf{Z}_g\}_{g\in[G]}$ each having r shards, where $r = \frac{K}{G}$ is the redundancy parameter
2: **Data allocation:**
3: **for** each $g \in [G]$ **do**
4:     allocate all r shards in the group Z_g to all r workers in group K_g, i.e., $\mathcal{Z}_{\mathcal{S}_k} = \mathsf{Z}_g$ for all $k \in \mathsf{K}_g$
5: **end for**
6: **At the group of workers** K_g: set $l = 1$
7: **repeat**
8:     l-th batch of r local samples $\Theta^l_{\mathsf{K}_g} = \{\theta^l_{\mathsf{K}_g,s} \sim \tilde{p}(\theta|\mathcal{Z}_s)\}_{\mathcal{Z}_s\in\mathsf{Z}_g}$, is computed at any of the worker in the group K_g
9:     when completed, i.e., at time T^l_g, send $\Theta^l_{\mathsf{K}_g}$ to the server, where T^l_g denote the l-th order statistic of the variables $\{\{\sum^l_{l'=1}\Delta T^{l'}_k\}_{l\in[L_k(t)]}\}_{k\in\mathsf{K}_g}$
10: **end repeat**
11: **At the server:** set $l = 1$
12: **repeat**
13:     when receiving all the l-th batches of samples $\{\Theta^l_{\mathsf{K}_g}\}_{g\in[G]}$, at time $T^l = \max_{g\in[G]} T^l_g$, compute the covariance matrices $\{\hat{C}_s(T^l)\}_{s\in[K]}$ using (9)
14:     for each $l' \in [l]$ compute $\theta^{l'}(T^l)$ using (8) with covariance matrices $\{\hat{C}_s(T^l)\}_{s\in[K]}$
15: **end repeat**

V. Coded CMC (C-CMC)

In this section, we introduce C-CMC. To start, we fix a $K \times K$ encoding matrix B and a $F \times K$ decoding matrix A that define a *gradient coding* scheme [6] robust to $r-1$ stragglers, with $F = \binom{K}{K-r+1}$. Accordingly, matrices A and B satisfy the equality $AB = 1$, where 1 is the $F \times K$ all-1 matrix. The shards are allocated to the workers according to the non-zero entries of the encoding matrix B, i.e., a shard $\mathcal{Z}_s$ is allocated to the worker k if the (k,s)-th entry of B is not equal to zero. The row weight and column weight of B are all equal to r, accounting for the facts that each shard is available to r workers and that each worker has access to exactly r shards.

We assume that workers share common randomness, i.e., common random seeds, so that two workers k and k' assigned the same shard $\mathcal{Z}_s$ produce the same l-th sample $\theta^l_{k,s} = \theta^l_{k',s}$. Accordingly, we henceforth write $\theta^l_{k,s} = \theta^l_{k',s} = \theta^l_s$. Note that common randomness is a requirement for C-CMC and not for G-CMC, which assumes the independence of the samples $\{\theta^l_{k,s}\}_{k:\ s\in\mathcal{S}_k}$ produced by all workers that are assigned the same shard $\mathcal{Z}_s$ (see Section IV).

Worker k estimates the mean $\mu_{k,s}(t)$ and covariance $C_{k,s}(t)$ of the subposterior $\tilde{p}(\theta|\mathcal{Z}_s)$, with $s \in \mathcal{S}_k$, by using $L_k(t)$ batches of samples computed by it up to time t as

$$\hat{\mu}_{k,s}(t) = \frac{1}{L_k(t)} \sum^{L_k(t)}_{l'=1} \theta^{l'}_s, \text{ and} \tag{11a}$$

$$\hat{C}_{k,s}(t) = \sigma^2 I + \frac{1}{L_k(t)} \sum^{L_k(t)}_{l'=1} (\theta^{l'}_s - \hat{\mu}_{k,s}(t))(\theta^{l'}_s - \hat{\mu}_{k,s}(t))^T \tag{11b}$$

respectively, where σ^2 is a regularization parameter. This is in contrast to CMC and G-CMC, where the mean vector and covariance matrix of the subposterior are estimated at the server.

At time $t = T^l_k = \sum^l_{l'=1} \Delta T^{l'}_k$, worker k computes the l-th batch of samples $\Theta^l_k = \{\theta^l_s\}_{s\in[\mathcal{S}_k]}$. Then, it updates the covariance matrices $\hat{C}_{k,s}(t)$ using (11) for all $s \in \mathcal{S}_k$. Given the assumption of common random seeds described above, the estimates $\hat{C}_{k,s}(t)$ and $\hat{C}_{k',s}(t')$ evaluated at any two workers k and k', with $s \in \mathcal{S}_k$, $s \in \mathcal{S}_{k'}$ and $L_k(t) = L_{k'}(t')$, are equal.

Let b_k be the $1 \times K$ row vector of the encoding matrix B corresponding to the worker k. Let $\Theta^l_k = \{\theta^l_{s_i}\}_{i\in[r]}$, with $s_i \in \mathcal{S}_k$ for each $i \in [r]$, be the l-th batch of r samples computed at the worker k. Each sample $\theta^l_{s_i}$ is pre-processed as $(\hat{C}_{k,s_i}(T^l_k))^{-1}\theta^l_{s_i}$ and the resulting processed samples are encoded using the encoding vector b_k as

$$\tilde{\theta}^l_k = [(\hat{C}_{k,s_1}(T^l_k))^{-1}\theta^l_{s_1} \cdots (\hat{C}_{k,s_r}(T^l_k))^{-1}\theta^l_{s_r}](\tilde{b}_k)^T, \tag{12}$$

with $\tilde{b}_k$ being the $1 \times r$ row vector given as $[b_k(s_1) \quad b_k(s_2) \quad \cdots \quad b_k(s_r)]$, where $b_k(s_i)$ is the s_i-th element of b_k.

The server waits until it receives transmissions corresponding to l-th samples from at least $K - r + 1$ workers, and proceeds to decoding to finally compute the l-th global sample θ^l. This occurs at time T^l equal to the $(K-r+1)$-th order statistic, i.e., the $(K-r+1)$-th smallest value, of the variables $\{T^l_k\}_{k\in[K]}$, and hence C-CMC is resilient to $r-1$ stragglers. Let the set of $K - r + 1$ non-stragglers for the l-th sample be indexed by an integer $j \in [\binom{K}{K-r+1}]$ and a_j be the corresponding $1 \times K$ row vector of the decoding matrix A. Let $\mathcal{K}^l_j = \{k^l_1, k^l_2, \cdots, k^l_{K-r+1}\} \subseteq [K]$ be the corresponding subset of $K - r + 1$ non-stragglers. The server decodes the sum of the processed l-th samples, by using the transmissions from the workers in $\mathcal{K}^l_j$, as

$$[\tilde{\theta}^l_{k^l_1} \quad \tilde{\theta}^l_{k^l_2} \quad \cdots \quad \tilde{\theta}^l_{k^l_{K-r+1}}](\tilde{a}_j)^T = \sum^K_{s=1}(\hat{C}^l_s)^{-1}\theta^l_s = \phi^l. \tag{13}$$

with $\tilde{a}_j$ being the $1 \times (K - r + 1)$ row vector given by $[a_j(k^l_1) \ a_j(k^l_2) \ \cdots \ a_j(k^l_{K-r+1})]$, where $a_j(k^l_i)$ is the k^l_i-th element of the $1 \times K$ row vector a_j for any $i \in [K-r+1]$. The matrix $\hat{C}^l_s$ is the empirical covariance matrix of the subposterior $\tilde{p}(\theta|\mathcal{Z}_s)$ computed using the first l samples, computed at any worker in $\mathcal{K}^l_j$, and is equal to $\hat{C}^l_{k^l_i,s}(T^l_{k^l_i})$ for any $i \in [K-r+1]$.

In order to compute the l-th global sample θ^l in (8) the server has to pre-multiply the decoded sample ϕ^l in (13) with the matrix $(\sum^K_{s=1}(\hat{C}^l_s)^{-1})^{-1}$. We propose that the server estimates the matrix $\sum^K_{s=1}(\hat{C}^l_s)^{-1}$ as the empirical covariance of the l decoded samples $\phi^{l'}$ for $l' \in [l]$, i.e., as the matrix

$$\hat{D}^l = \frac{1}{l}\sum_{l'=1}^{l}(\phi^{l'}-\overline{\phi})(\phi^{l'}-\overline{\phi})^T \approx \sum_{s=1}^{K}C_s^{-1} \qquad (14)$$

with $\overline{\phi} = \frac{1}{l}\sum_{l'=1}^{l}\phi^{l'}$. The approximate equality in (14) can be seen to be exact when $l \to \infty$. Accordingly the final global sample is $\theta^l = (\sigma^2 I + \hat{D}^l)^{-1}\phi^l$, where σ^2 is a regularization parameter. The overall algorithm is summarized in Algorithm 3. The rationale for the proposed estimate (14) is provided in the Appendix.

Algorithm 3 Coded CMC (C-CMC)

1: **Input:** Number of workers K, Maximum number of stragglers $r-1$, Data shards $\{\mathcal{Z}_s\}_{s=1}^{K}$
2: Consider two matrices A, B such that it forms a GC scheme robust to $r-1$ stragglers [6]
3: **Data allocation:**
4: **for** each $k, s \in [K]$ **do**
5:     **if** $B(k,s) \neq 0$ **then**
6:         Allocate shard $\mathcal{Z}_s$ to worker k
7:     **end if**
8: **end for**
9: **At worker k:** set $l = 1$
10: **repeat**
11:     computes the l-th batch of samples $\Theta_k^l = \{\theta_s^l\}_{s \in [\mathcal{S}_k]}$
12:     when completed, i.e., at time $T_k^l = \sum\limits_{l'=1}^{l}\Delta T_k^{l'}$, worker k estimates the covariance matrices $\{\hat{C}_s^l\}_{s \in \mathcal{S}_k}$ of the sub-posteriors $\{\tilde{p}(\theta|\mathcal{Z}_s)\}_{s \in \mathcal{S}_k}$ using the l batches of samples $\{\Theta_k^{l'}\}_{l' \in [l]}$ computed at the worker k.
13:     worker k transmits $\tilde{\theta}_k^l$ using (12)
14: **end repeat**
15: **At the server:** set $l = 1$
16: **repeat**
17:     using the transmissions from $K-r+1$ non-stragglers, compute $\sum_{s=1}^{K}(\hat{C}_s^l)^{-1}\theta_s^l$ using (13) at time T^l being equal to $(K-r+1)$-th order statistic of the variables $\{T_k^l\}_{k \in [K]}$
18:     Estimate $\sum_{s=1}^{K}(\hat{C}_s^l)^{-1}$ using (14) and compute global sample θ^l using (8).
19: **end repeat**

VI. EXPERIMENTS

In this section, we evaluate the performance of the considered CMC schemes in the presence of stragglers. We present two experiments on distributed computing systems with $K = 5$ and $K = 40$ workers[1]. In both experiments, we assume that the subposterior $\tilde{p}(\theta|\mathcal{Z}_s)$ for each shard $\mathcal{Z}_s$ where $s \in [K]$ to be a 5-dimensional multivariate Gaussian distribution $\mathcal{N}(0, C_s)$ with a symmetric Toeplitz covariance matrix C_s using first column $[1 \ \rho_s \ \rho_s^2 \ \rho_s^3 \ \rho_s^4]^T$ with $\rho_s = (s-1)/K$ [21]. As in [16], [21], we calculate the error in (3) for multiple test functions, with each test function $f_{i,j}(\theta) = \theta[i]\theta[j]$ being an element in the outer

---

[1]We have carried out all experiments on a laptop with i7 processor and 16 GB RAM by using MATLAB. Code is available at <https://github.com/kclip/Straggler-resilient-CMC>.
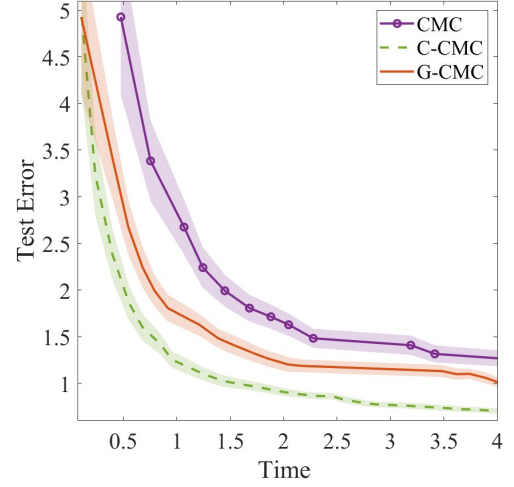

Fig. 3. Average test error versus time with $K = 5$ workers and $r = 2$. C-CMC is resilient to 1 straggler. As K/r is not an integer, we apply G-CMC on a group of 4 workers, with the last group having one worker. As a result, G-CMC is not resilient to stragglers, from the last group.
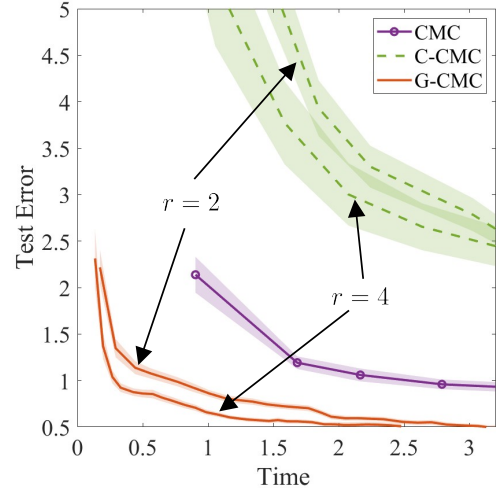

Fig. 4. Average test error versus time with $K = 40$ workers. For $r = 2$, C-CMC and G-CMC are resilient to 1 and 20 stragglers respectively. For $r = 4$, C-CMC and G-CMC are resilient to 3 and 30 stragglers respectively.

product matrix $\theta\theta^T$ for $i, j \in [d]$, and average them to obtain the final error. We consider Pareto distribution with $\eta = 0.1$ and shape parameter $\beta = 1.2$, which corresponds to scale parameter $r/60$ and mean $0.1r$ (see Section II), for the computing time ΔT_k^l at the workers (see, e.g., [7]).

In Fig. 3 and Fig. 4, we plot the test error averaged over 50 random realizations of computing times, as a function of time, at $K = 5$ and $K = 40$ workers respectively, following a Pareto distribution with $\eta = 0.1$ and shape parameter $\beta = 1.2$, which corresponds to scale parameter $r/60$, for the Pareto distribution with mean $0.1r$. The curve represents the average error, and shaded region represents the error bars corresponding to 0.15 times the standards deviation of the error across the realizations. We plot the curves for CMC, C-CMC and G-CMC. For $K = 5$, as K/r is not an integer, G-CMC is applied to three groups with one of the groups having a single worker. If

the straggler is from the group containing a single worker, this delays the computation of global sample at the server, which in turn degrades the performance of G-CMC for smaller K. Note that C-CMC can be directly applied as there is no such restriction on r in C-CMC. From Fig. 3 and Fig. 4, we can conclude that G-CMC is more efficient for straggler mitigation for large values of K, where as C-CMC is efficient for smaller values of K.

VII. Conclusions

In this paper, we have considered, for the first time, the problem of stragglers in Consensus Monte Carlo (CMC) for distributed Bayesian learning. We studied the effect of stragglers in the standard CMC protocol, and proposed two schemes, Group-based CMC (G-CMC) and Coded CMC (C-CMC) that effectively leverage the redundancy in data allocation.

Appendix

In this Appendix, we detail the approximation in (14). To this end, we write the empirical covariance of the l decoded samples $\phi^{l'}$, for $l' \in [l]$, as

$$
\begin{aligned}
\hat{D}^l &= \frac{1}{l} \sum_{l'=1}^{l} \left(\phi^{l'} (\phi^{l'})^T \right) - \overline{\phi}(\overline{\phi})^T \\
&= \frac{1}{l} \sum_{l'=1}^{l} \left[\left(\sum_{s=1}^{K} (\hat{C}_s^{l'})^{-1} \theta_s^{l'} \right) \left(\sum_{m=1}^{K} (\theta_m^{l'})^T (\hat{C}_m^{l'})^{-1} \right) \right] \\
&\quad - \left(\frac{1}{l} \sum_{l'=1}^{l} \sum_{s=1}^{K} (\hat{C}_s^{l'})^{-1} \theta_s^{l'} \right) \left(\frac{1}{l} \sum_{l'=1}^{l} \sum_{m=1}^{K} (\theta_m^{l'})^T (\hat{C}_m^{l'})^{-1} \right) \\
&\stackrel{(a)}{\approx} \frac{1}{l} \sum_{l'=1}^{l} \left[\left(\sum_{s=1}^{K} C_s^{-1} \theta_s^{l'} \right) \left(\sum_{m=1}^{K} (\theta_m^{l'})^T C_m^{-1} \right) \right] \\
&\quad - \left(\frac{1}{l} \sum_{l'=1}^{l} \sum_{s=1}^{K} C_s^{-1} \theta_s^{l'} \right) \left(\frac{1}{l} \sum_{l'=1}^{l} \sum_{m=1}^{K} (\theta_m^{l'})^T C_m^{-1} \right)
\end{aligned}
$$

where, the approximation in (a) is justified by the fact that, as $l \to \infty$, the covariance matrix $\hat{C}_s^l$ converges to the ground-truth covariance matrix C_s. Letting $\mu_s^l = \frac{1}{l} \sum_{l'=1}^{l} \theta_s^{l'}$ be the mean of first l samples of $\tilde{p}(\theta|\mathcal{Z}_s)$, we can write $\hat{D}^l$ as

$$
\begin{aligned}
\hat{D}^l &\approx \frac{1}{l} \sum_{l'=1}^{l} \left[\left(\sum_{s=1}^{K} C_s^{-1} \theta_s^{l'} \right) \left(\sum_{m=1}^{K} (\theta_m^{l'})^T C_m^{-1} \right) \right] \\
&\quad - \frac{1}{l} \sum_{l'=1}^{l} \left[\left(\sum_{s=1}^{K} C_s^{-1} \mu_s^l \right) \left(\sum_{m=1}^{K} (\mu_m^l)^T C_m^{-1} \right) \right] \\
&= \frac{1}{l} \sum_{l'=1}^{l} \sum_{s=1}^{K} \left[C_s^{-1} \theta_s^{l'} (\theta_s^{l'})^T C_s^{-1} - C_s^{-1} \mu_s^l (\mu_s^l)^T C_s^{-1} \right] \\
&\quad + \frac{1}{l} \sum_{l'=1}^{l} \sum_{\substack{s,m=1 \\ s \neq m}}^{K} \left[C_s^{-1} \theta_s^{l'} (\theta_m^{l'})^T C_m^{-1} - C_s^{-1} \mu_s^l (\mu_m^l)^T C_m^{-1} \right] \\
&\stackrel{(b)}{\approx} \sum_{s=1}^{K} C_s^{-1} \mathrm{E}[\theta_s^{l'} (\theta_s^{l'})^T - \mu_s^l (\mu_s^l)^T] C_s^{-1} \\
&\quad + \sum_{\substack{s,m=1 \\ s \neq m}}^{K} C_s^{-1} \mathrm{E}[\theta_s^{l'} (\theta_m^{l'})^T - \mu_s^l (\mu_m^l)^T] C_m^{-1}
\end{aligned}
$$

$$
= \sum_{s=1}^{K} C_s^{-1} C_s C_s^{-1} + \sum_{\substack{s,m=1 \\ s \neq m}}^{K} C_s^{-1}(0) C_m^{-1} = \sum_{s=1}^{K} C_s^{-1},
$$

where the approximation (b) is exact as $l \to \infty$.

References

[1] I. Stoica *et al.*, "A Berkeley view of systems challenges for AI," *CoRR*, 2017. [Online]. Available: http://arxiv.org/abs/1712.05855

[2] M. Li, D. G. Andersen, J. W. Park, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita, and B.-Y. Su, "Scaling distributed machine learning with the parameter server," *Proc USENIX Conference on Operating Systems Design and Implementation*, pp. 583–598, Oct 2014.

[3] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, Jan 2008.

[4] M. Abadi *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *CoRR*, 2016. [Online]. Available: http://arxiv.org/abs/1603.04467

[5] J. Dean and L. A. Barroso, "The tail at scale," *Commun of the ACM*, vol. 56, pp. 74–80, Feb 2013.

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

[7] J. Zhang and O. Simeone, "LAGC: Lazily aggregated gradient coding for straggler-tolerant and communication-efficient distributed learning," *IEEE Trans on Neural Networks and Learning Systems*, vol. 32, no. 3, pp. 962–974, Mar 2021.

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

[9] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," *Proc Int Conf on Machine Learning*, pp. 1321–1330, Aug 2017.

[10] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," *Proc Int Conf on Neural Information Processing Systems*, pp. 6405–6416, Dec 2017.

[11] E. Angelino, M. Johnson, and R. Adams, *Patterns of Scalable Bayesian Inference*. Now Publishers, 2016.

[12] D. Barber, *Bayesian Reasoning and Machine Learning*. Cambridge University Press, 2012.

[13] O. Simeone, *A Brief Introduction to Machine Learning for Engineers*. Now Foundations and Trends, 2018.

[14] T. D. Bui, C. V. Nguyen, S. Swaroop, and R. E. Turner, "Partitioned variational inference: A unified framework encompassing federated and continual learning," *arXiv preprint arXiv:1811.11206*, 2018.

[15] S. L. Scott, A. W. Blocker, F. V. Bonassi, H. A. Chipman, E. I. George, and R. E. McCulloch, "Bayes and big data: The Consensus Monte Carlo algorithm," *Int Jour of Management Science and Engineering Management*, vol. 11, pp. 78–88, 2016.

[16] M. Rabinovich, E. Angelino, and M. I. Jordan, "Variational consensus Monte Carlo," *Proc Advances in Neural Information Processing Systems*, vol. 28, 2015.

[17] R. Kassab and O. Simeone, "Federated generalized bayesian learning via distributed stein variational gradient descent," *IEEE Transactions on Signal Processing*, vol. 70, pp. 2180–2192, 2022.

[18] S. Ahn, B. Shahbaba, and M. Welling, "Distributed stochastic gradient MCMC," *Proc Int Conf on Machine Learning*, vol. 32, no. 2, pp. 1044–1052, Jun 2014.

[19] D. Liu and O. Simeone, "Wireless federated langevin monte carlo: Repurposing channel noise for bayesian sampling and privacy," 2021. [Online]. Available: https://arxiv.org/abs/2108.07644

[20] E. Ozfatura, D. Gündüz, and S. Ulukus, "Gradient coding with clustering and multi-message communication," in *2019 IEEE Data Science Workshop (DSW)*, 2019, pp. 42–46.

[21] D. Liu and O. Simeone, "Channel-driven monte carlo sampling for bayesian distributed learning in wireless data centers," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 2, pp. 562–577, 2022.

[22] K. Shanmugam, M. Ji, A. M. Tulino, J. Llorca, and A. G. Dimakis., "Finite-length analysis of caching-aided coded multicasting," *IEEE Tran on Inf Theory*, vol. 62, no. 10, pp. 5524–5537, Oct 2016.

[23] E. Lampiris and P. Elia, "Adding transmitters dramatically boosts coded-caching gains for finite file sizes," *IEEE Jour on Selected Areas in Communications*, vol. 36, no. 6, pp. 1176–1188, Jun 2018.