Coded Computation of Multiple Functions

Wilton Kim, Stanislav Kruglik, and Han Mao Kiah

School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore

Emails: wilt0002, stanislav.kruglik, hmkiah@ntu.edu.sg

***Abstract*—We consider the problem of evaluating arbitrary multivariate polynomials over several massive datasets in a distributed computing system with a single master node and multiple worker nodes. We focus on the general case when each multivariate polynomial is evaluated over its dataset and propose a generalization of the Lagrange Coded Computing framework (Yu et al. 2019) to provide robustness against stragglers who do not respond in time, adversarial workers who respond with wrong computation and information-theoretic security of dataset against colluding workers. Our scheme introduces a small computation overhead which results in a reduction in download cost and also offers comparable resistance to stragglers over existing solutions.**

I. Introduction

Due to the enormous size of current datasets, computational operations must be carried out in a distributed manner by outsourcing the workload to several servers which operate in parallel [1]. However, we may face several problems such as slow-responding servers (*stragglers*) (see [2], [3]) and extra communication costs. There is also a possibility that some servers are *adverserial* (which respond with wrong computations) or *colluding* (which communicate with other servers to obtain some information on the datasets) (see [4]–[8]).

Coded distributed computation is a newly emerging research area that outsources computation to external working nodes in encoded form to mitigate possible adversarial behavior. Such behaviour may include providing wrong results, colluding, straggling, or their combination. Polynomial codes were proposed in [9] to compute high-dimensional distributed matrix multiplication which tolerates stragglers. In [10], the authors proposed a scheme that combines polynomial codes with Ben-Or, Goldwasser and Wigderson (BGW) scheme [11] to keep the datasets private. It was further improved in [12] by proposing *Lagrange Coded Computing* (LCC) which has resiliency against stragglers and adversaries and provides security against colluding workers. In [13], the authors considered the general distributed computing framework, where each server performs as both Master Node and Worker Node, and wants to obtain the computation of several functions $\psi_1, \dots, \psi_\ell$ on the given dataset X. However, in this paper, we consider the distributed computing setup that comprises one Master Node that wants to obtain the result of computation on his own data and several Worker Nodes that assist the Master Node (as in [12], [14]). We also assume that Worker Nodes cannot communicate with each other. For a detailed survey of distributed computing, readers can refer to [15], [16].

Our work is closely related to LCC [12], which evaluates a *single* multivariate polynomial ψ on some datasets in a distributed manner. In [14], a generalization of LCC (GLCC) which splits the datasets into several parts to construct the computational task for the Worker Nodes was proposed. It gives rise to trade-offs between communication and computation costs and the required number of workers. The Master Node has the flexibility to decide on how to split the datasets to optimize the performance. In both [12] and [14], the system consists of one Master Node with many Worker Nodes, such that, given the dataset $\boldsymbol{X} = (X_1, \dots, X_M)$, the Master Node wants to obtain evaluations $\psi(X_1), \dots, \psi(X_M)$ of multivariate polynomial ψ. For instance, given X_1, X_2, X_3, the Master Node wants to obtain X_1^2, X_2^2, X_3^{21}.

In this paper, the Master Node wants to evaluate different functions on different elements from the same dataset $\boldsymbol{X}$. For example, in the above-mentioned setup the Master Node wants to obtain $X_1^{10}, X_2^{10}, X_3^2$. We discuss briefly some naive ways to solve this problem by modifying existing approaches:

- *Scheme 0*: The Master Node constructs a new polynomial ψ, so that it can apply LCC in a single round. For instance, to obtain $X_1^{10}, X_2^{10}, X_3^2$, the Master Node constructs $\psi(u, v) = u^{10} + v^2$ and sees the computations as $\psi(\tilde{\boldsymbol{X}}_1), \psi(\tilde{\boldsymbol{X}}_2), \psi(\tilde{\boldsymbol{X}}_3)$, where $\tilde{\boldsymbol{X}}_1 = (X_1, 0), \tilde{\boldsymbol{X}}_2 = (X_2, 0), \tilde{\boldsymbol{X}}_3 = (0, X_3)$ and 0 is the zero matrix of the same dimension as X_i.
- *Scheme I*: The Master Node splits the Worker Nodes into L groups, $G_1, \dots, G_L$ and applies LCC in each group such that from the group G_i, the Master Node obtains all computations of ψ_i. For instance, to obtain $X_1^{10}, X_2^{10}, X_3^2$, the Master Node splits the Worker Nodes into two groups G_1 and G_2 and applies LCC separately on G_1 and G_2 such that from G_1, the Master Node obtains X_1^{10}, X_2^{10} and from G_2, the Master Node obtains X_3^2.
- *Scheme II*: The Master Node can apply L rounds of LCC, such that in the i-th round, it performs all computations on datasets related with ψ_i. For instance, to obtain $X_1^{10}, X_2^{10}, X_3^2$, the Master Node applies LCC in two rounds.

In this paper, we propose a new scheme (defined as Scheme III) that computes all computations in one round, by modifying the task given to the Worker Nodes. The scheme requires the Worker Nodes to perform a little bit more computation and has a slightly worse tolerance to stragglers, but the download cost is significantly lower. We will elaborate it more in Section IV. The key components of our scheme are partitioning the computations into L different groups, and introducing the function h for all servers to compute so that its values in pre-selected points, up to field multiplier, give us all required computations. We employ the similar technique as in [14] to construct the function h.

II. Problem Formulation

For any positive integer n, we denote the set $\{1, 2 \dots, n\}$ as $[n]$ and we use $\mathbb{F}$ to denote a sufficiently large finite field. Let us consider the scenario where the Master Node (MN) has M elements of information data $\boldsymbol{X} = (X_1, \dots, X_M)$, $X_i \in \mathbb{U}$, where $\mathbb{U}$ is a vector space over $\mathbb{F}$. For instance, if the information data is of a square-matrix form, then $\mathbb{U} = \mathbb{F}^{n \times n}$. Suppose that the Master Node wants to obtain the following $\ell = \sum_{i=1}^{L} \ell_i$ many computations,

$$\left\{\psi_1\left(\boldsymbol{X}_1^{(1)}\right), \dots, \psi_1\left(\boldsymbol{X}_{\ell_1}^{(1)}\right), \dots, \psi_L\left(\boldsymbol{X}_1^{(L)}\right), \dots, \psi_L\left(\boldsymbol{X}_{\ell_L}^{(L)}\right)\right\}, \quad (1)$$

[1]We note that in general case X_1, X_2, X_3 are elements of vector space over large enough finite field $\mathbb{F}$.

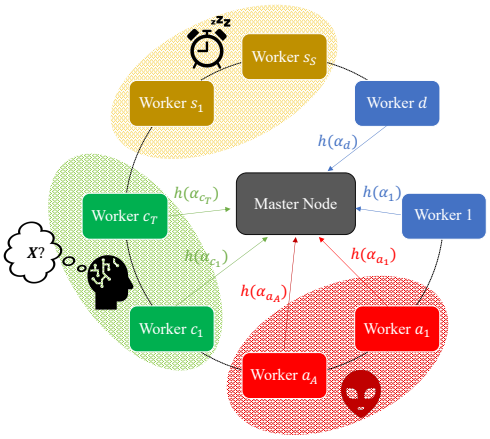

Fig. 1: A distributed computing system with one Master Node and d Worker Nodes. To recover computation results, the Master Node waits for the first K Worker Nodes to respond, which might include A adversary responses. Up to T workers may collude to obtain some information on $\boldsymbol{X}$.

where $\boldsymbol{X}_i^{(j)}$ contains s_j specified elements of $\boldsymbol{X}$, for all $i \in [\ell_j]$, $j \in [L]$ and $\psi_j : \mathbb{U}^{s_j} \to \mathbb{V}$ is a multivariate polynomial of total degree D_j, where $\mathbb{V}$ is a vector space over $\mathbb{F}$. In other words, the Master Node is interested in ℓ computations involving L many polynomials, $\psi_1, \ldots, \psi_L$, where each polynomial ψ_j takes s_j many elements of $\mathbb{U}$ as input, and produces an element of $\mathbb{V}$ as output. For instance, for the case of $\mathbb{U} = \mathbb{V} = \mathbb{F}^{n\times n}$, the Master Node has $M = 2$ elements of information data $\boldsymbol{X} = (X_1, X_2)$, and wants to compute $\psi_1(X_1) = X_1^2$, $\psi_1(X_2) = X_2^2$ and $\psi_2(X_1, X_2) = X_1X_2$. In this case, $s_1 = 1$, $s_2 = 2$, $L = 2$, $\ell_1 = 2$, $\ell_2 = 1$ and $\ell = 3$. To obtain them, the Master Node outsources the workload to $d \geq \ell$ Worker Nodes by sending encoded data $\boldsymbol{Y}_i$ to the i-th Worker Node. After receiving $\boldsymbol{Y}_i$, the i-th Worker Node performs some computations on them and sends their results back to the Master Node. From the Master Node's point of view, it is the value of a polynomial h at the i-th Worker Node's assigned evaluation point α_i. We consider the case when there are S stragglers, A adversaries, and T colluding nodes (see Figure 1). Therefore, we aim to design a scheme that satisfies these two constraints.

- *T-Secure*: Any T colluding nodes are not able to obtain any extra information about the dataset. That is,

$$I(\boldsymbol{X}; \boldsymbol{Y}_{t_1}, \ldots, \boldsymbol{Y}_{t_T}) = 0, \text{ for all } \{t_1, \ldots, t_T\} \subseteq [d]. \quad (2)$$

- *Correctness*: For some $K \leq d$, the scheme can correctly recover the required computation from the fastest K responses, even with the existence of A adversarial responses among them. In other words,

$$H(\boldsymbol{X}|\tilde{\boldsymbol{Y}}_{k_1}, \ldots, \tilde{\boldsymbol{Y}}_{k_K}) = 0, \text{ for all } \{k_1, \ldots, k_K\} \subseteq [d], \quad (3)$$

where $\tilde{\boldsymbol{Y}}_i$ is the result of worker i computations.

We consider the following performance metrics.

1) Stragglers Resistance (SR): The number of stragglers the scheme can tolerate.
2) Upload Cost (UC): The number of elements in $\mathbb{U}$ the Master Node needs to send.
3) Download Cost (DC): The number of elements in $\mathbb{V}$ the Master Node needs to download.
4) Computation in Master Node (MN): The number of $\cdot_{\mathbb{U}}$ and $\times$ multiplications the Master Node needs to perform to distribute the computation to each Worker Node, where $\cdot_{\mathbb{U}}$ is the multiplication of a field element in $\mathbb{F}$ with an element in $\mathbb{U}$ and $\times$ is the multiplication of two elements in $\mathbb{U}$.[2]We assume that all necessary multiplications of field element $\mathbb{F}$ are pre-computed.
5) Computation in Worker Node (WN): The number of multiplications $\cdot_{\mathbb{V}}$ and $\times$ that the Worker Node needs to perform to complete the task given by the Master Node, where $\cdot_{\mathbb{V}}$ is the multiplication of a field element in $\mathbb{F}$ with an element in $\mathbb{V}$ and $\times$ is the multiplication of two elements in $\mathbb{U}$.[2]We assume that all necessary multiplications of field elements $\mathbb{F}$ are pre-computed.

III. Main Result and Discussion

Theorem 1 (Scheme III)**.** *Consider ℓ computations involving L polynomials $\psi_1, \ldots, \psi_L$ such that each ψ_j takes s_j many inputs and there are ℓ_j computations associated with ψ_j and $\sum_{j\in[L]} \ell_j = \ell$, over a distributed computing system scheme with d Worker Nodes that tolerates A-adversary nodes and is secure against T-colluding nodes, as long as $d \geq \max_j\{D_j(\ell_j + T - 1) + \ell - \ell_j\} + 2A + 1$, the following performance metrics are achievable in Scheme III (defined in Section IV-D).*

- *SR:* $d - (\max_j\{D_j(\ell_j + T - 1) + \ell - \ell_j\} + 2A + 1)$.
- *UC:* $d\sum_{j\in[L]} s_j$.
- *DC:* $\max_j\{D_j(\ell_j + T - 1) + \ell - \ell_j\} + 2A + 1$.
- *Computation in MN:* $\sum_{j\in[L]}(\ell_j + T)s_j$ *multiplications of $\cdot$'s.*
- *Computation in WN: L multiplications of $\cdot$'s and the total number of multiplications $\times$'s in all ψ_j to compute.*

We briefly discuss the scheme mentioned in Theorem 1 formally proven in Section IV-D. The Master Node partitions the ℓ computations into L groups, where the j-th part is formed from all computations involving ψ_j. For each of j-th partition, the Master Node constructs a sharing polynomial of degree $\ell_j + T - 1$, utilizing all involved dataset and T random elements. The Master Node encodes the dataset using these L polynomials and distributes it to the Worker Nodes. Each Worker Node applies the polynomial ψ_j to the j-th encoded data and multiplies it with some field element for all $j \in [L]$, then take the sum of all of them. The task can be seen as evaluating the value of polynomial h of degree $\max_j\{D_j(\ell_j + T - 1) + \ell - \ell_j\}$ at a given field element. In other words, the computations performed by the Worker Nodes form a codeword of a Reed-Solomon Code of length d and dimension $\max_j\{D_j(\ell_j + T - 1) + \ell - \ell_j\} + 1$. We denote the Reed-Solomon code of length n and dimension k as $\mathrm{RS}(n, k)$. We note that all required computations are values of polynomial h at some pre-selected points. So, with at least $K = \max_j\{D_j(\ell_j + T - 1) + \ell - \ell_j\} + 2A + 1$ responses from the Worker Nodes with at most A adversaries, the polynomial h can be recovered using $O(K(\log K)^2 \log\log K)$ arithmetic operations by any Reed-Solomon decoder (see [17] for more details).

A. Example of Scheme III

Consider the task where the Master Node has the dataset $\boldsymbol{X} = (X_1, X_2, X_3, X_4)$, where $X_i \in \mathbb{F}^{n\times n}, \mathbb{F} = GF(2^8)$ and wants to obtain

$$X_1^{10}, \quad X_2^{10}, \quad X_3^7 \quad \text{and} \quad X_4^2,$$

[2]To simplify the notation, we will write $\cdot$ for both $\cdot_{\mathbb{U}}$ and $\cdot_{\mathbb{V}}$.

by employing $d > 4$ Worker Nodes ensuring security against one colluding node and tolerating one adversary node. The Master Node wants to find $\ell = 4$ values of evaluations for $L = 3$ functions $\psi_1(X_1) = X_1^{10}, \psi_1(X_2) = X_2^{10}, \psi_2(X_3) = X_3^7$ and $\psi_3(X_4) = X_4^2$. By considering a set of distinct evaluation points $\boldsymbol{\beta} = \left\{\beta_1^{(D,1)}, \beta_2^{(D,1)}, \beta_1^{(D,2)}, \beta_1^{(D,3)}\right\} \cup \left\{\beta_1^{(R)}\right\}$, the Master Node constructs $L = 3$ many *sharing* polynomials so that

$$\begin{cases} f_1\left(\beta_1^{(D,1)}\right) = X_1, f_1\left(\beta_2^{(D,1)}\right) = X_2, f_1\left(\beta_1^{(R)}\right) = Z_1, \\ f_2\left(\beta_1^{(D,2)}\right) = X_3, f_2\left(\beta_1^{(R)}\right) = Z_2, \\ f_3\left(\beta_1^{(D,3)}\right) = X_4, f_3\left(\beta_1^{(R)}\right) = Z_3, \end{cases} \tag{4}$$

where $Z_1, Z_2, Z_3 \in \mathbb{F}^{n\times n}$ are independent random elements uniformly distributed over the same alphabets as X_i's. The polynomials f_1, f_2, f_3 can be obtained by Lagrange Interpolation, that is,

$$\begin{aligned} f_1(x) = {} & X_1 \frac{\left(x-\beta_2^{(D,1)}\right)\left(x-\beta_1^{(R)}\right)}{\left(\beta_1^{(D,1)}-\beta_2^{(D,1)}\right)\left(\beta_1^{(D,1)}-\beta_1^{(R)}\right)} \\ & + X_2 \frac{\left(x-\beta_1^{(D,1)}\right)\left(x-\beta_1^{(R)}\right)}{\left(\beta_2^{(D,1)}-\beta_1^{(D,1)}\right)\left(\beta_2^{(D,1)}-\beta_1^{(R)}\right)} \\ & + Z_1 \frac{\left(x-\beta_1^{(D,1)}\right)\left(x-\beta_2^{(D,1)}\right)}{\left(\beta_1^{(R)}-\beta_1^{(D,1)}\right)\left(\beta_1^{(R)}-\beta_2^{(D,2)}\right)}, \end{aligned} \tag{5}$$

$$f_2(x) = X_3 \frac{x-\beta_1^{(R)}}{\beta_1^{(D,2)}-\beta_1^{(R)}} + Z_2 \frac{x-\beta_1^{(D,2)}}{\beta_1^{(R)}-\beta_1^{(D,2)}}, \text{ and} \tag{6}$$

$$f_3(x) = X_4 \frac{x-\beta_1^{(R)}}{\beta_1^{(D,3)}-\beta_1^{(R)}} + Z_3 \frac{x-\beta_1^{(D,3)}}{\beta_1^{(R)}-\beta_1^{(D,3)}}. \tag{7}$$

We can easily see that f_1 is a polynomial of degree 2 and both f_2 and f_3 are polynomials of degree 1. The Master Node also considers another set of evaluation points $\boldsymbol{\alpha} = \{\alpha_1, \ldots, \alpha_d\}$, $\boldsymbol{\alpha} \cap \boldsymbol{\beta} = \emptyset$ and assigns each Worker Node with a unique evaluation point from $\boldsymbol{\alpha}$. The Master Node sends $(f_1(\alpha_i), f_2(\alpha_i), f_3(\alpha_i))$ to the i-th Worker Node and asks it to compute

$$\begin{aligned} & h(\alpha_i) = \\ & \psi_1(f_1(\alpha_i))\left(\alpha_i-\beta_1^{(D,2)}\right)\left(\alpha_i-\beta_1^{(D,3)}\right) \\ & + \psi_2(f_2(\alpha_i))\left(\alpha_i-\beta_1^{(D,1)}\right)\left(\alpha_i-\beta_2^{(D,1)}\right)\left(\alpha_i-\beta_1^{(D,3)}\right) \\ & + \psi_3(f_3(\alpha_i))\left(\alpha_i-\beta_1^{(D,1)}\right)\left(\alpha_i-\beta_2^{(D,1)}\right)\left(\alpha_i-\beta_1^{(D,2)}\right). \end{aligned} \tag{8}$$

As a result, the Master Node expects to obtain

$$(h(\alpha_1), \ldots, h(\alpha_d)), \tag{9}$$

which forms a codeword of $\mathrm{RS}(d, 23)$. Hence, with any 25 responses, the Master Node can recover h, in presence of one adversary response. We can easily check that

$$\begin{aligned} & h\left(\beta_1^{(D,1)}\right) = \gamma_1^{(1)} X_1^{10},\ h\left(\beta_2^{(D,1)}\right) = \gamma_2^{(1)} X_2^{10}, \\ & h\left(\beta_1^{(D,2)}\right) = \gamma_1^{(2)} X_3^7,\ \ h\left(\beta_1^{(D,3)}\right) = \gamma_1^{(3)} X_4^2, \end{aligned} \tag{10}$$

for some $\gamma_1^{(1)}, \gamma_2^{(1)}, \gamma_1^{(2)}, \gamma_1^{(3)} \in \mathbb{F}$ that can be pre-computed.

IV. General Description of Schemes

We first consider a naive scheme to solve the problem mentioned in Section II, where the Master Node rewrites the ℓ computations so that they become computations of a single function (Scheme 0). Then, we consider another naive scheme where the Master Node splits the Worker Nodes into L many groups where each group performs different computation (Scheme I). Then, we consider the scheme where the Master Node performs LCC in L rounds (Scheme II). After it, we describe the proposed scheme where the Master Node performs all computations in one round (Scheme III). We note that Schemes 0, I, and II are minor modifications of existing schemes from [12]. Nevertheless, we explicity state their performance in Theorems 2, 3, and 4. We defer their corresponding proofs to the extended version [18] due to space limitations.

Remark 1. Due to space limitations, we only consider schemes based on the LCC framework [12]. Nevertheless, it is also possible to apply GLCC on top of our schemes to obtain a more flexible performance in terms of trade-off among computation and communication costs.

A. Scheme 0 (First Naive Approach)

The Master Node forms the multivariate polynomial,

$$\Psi(\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_L) = \sum_{i\in[L]} \psi_i(\boldsymbol{u}_i), \tag{11}$$

where $\boldsymbol{u}_j$ consists of s_j many elements of $\mathbb{U}$. For example, the computation $\psi_1\left(\boldsymbol{X}_1^{(1)}\right)$ can be expressed as $\Psi\left(\boldsymbol{X}_1^{(1)}, \boldsymbol{0}, \ldots, \boldsymbol{0}\right)$. Hence, the Master Node can reformulate the problem into obtaining the computations

$$\left\{\Psi\left(\tilde{\boldsymbol{X}}_1^{(1)}\right), \ldots, \Psi\left(\tilde{\boldsymbol{X}}_{\ell_1}^{(1)}\right), \ldots, \Psi\left(\tilde{\boldsymbol{X}}_1^{(L)}\right), \ldots, \Psi\left(\tilde{\boldsymbol{X}}_{\ell_L}^{(L)}\right)\right\}, \tag{12}$$

where $\tilde{\boldsymbol{X}}_i^{(j)}$ is formed so that its j-th component is $\boldsymbol{X}_i^{(j)}$ and other components are zero. Therefore, the Master Node can apply LCC to obtain (12). The values of performance metrics for such a scheme are formulated in the theorem below.

Theorem 2 (Scheme 0)**.** *For $d \geq \max_j\{D_j\}(\ell+T-1)+2A+1$ Worker Nodes from whose A are adversary and T are colluding, Scheme 0 achieves the following performance metrics.*

- *SR:* $d - (\max_j\{D_j\}(\ell + T - 1) + 2A + 1)$.
- *UC:* $d\sum_{j\in[L]} s_j$.
- *DC:* $\max_j\{D_j\}(\ell + T - 1) + 2A + 1$.
- *Computation in MN:* $(\ell+T)\sum_{j\in[L]} s_j$ *multiplications of $\cdot$'s.*
- *Computation in WN: no multiplications of $\cdot$'s and the number of multiplications $\times$'s in Ψ to compute.*

For the other schemes, the Master Node constructs L many *sharing* polynomials, by considering a set of distinct evaluation points $\boldsymbol{\beta} = \left\{\beta_i^{(D,j)} \in \mathbb{F} : i \in [\ell_j], j \in [L]\right\} \cup \left\{\beta_i^{(R)} : i \in [T]\right\}$, such that, for all $j \in [L]$, $f_j : \mathbb{F} \to \mathbb{U}^{s_j}$ and

$$\begin{aligned} f_j\left(\beta_1^{(D,j)}\right) &= \boldsymbol{X}_1^{(j)}, \ldots, f_j\left(\beta_{\ell_j}^{(D,j)}\right) = \boldsymbol{X}_{\ell_j}^{(j)}, \\ f_j\left(\beta_1^{(R)}\right) \ &= \boldsymbol{Z}_1^{(j)}, \ldots, f_j\left(\beta_T^{(R)}\right) \ = \boldsymbol{Z}_T^{(j)}, \end{aligned} \tag{13}$$

where $\boldsymbol{Z}_i^{(j)}$ are independent random elements. The Master Node also assigns each Worker Node with a unique evaluation point from set $\boldsymbol{\alpha} = \{\alpha_i : i \in [d]\} \subseteq \mathbb{F}, \boldsymbol{\alpha} \cap \boldsymbol{\beta} = \emptyset$. We can easily verify that recovering $\psi_j(f_j(x))$ for all $j \in [L]$ gives us all required computations.

B. Scheme I (Second Naive Approach)

The Master Node splits the Worker Nodes into L many groups, $G_1, \ldots, G_L$, where G_j contains d_j many Worker Nodes and $\sum_{j\in[L]} d_j = d$. Let $\alpha_{j_i} = \alpha_{i+\sum_{k<j} d_k}$. For each group G_j, the Master Node sends $f_j(\alpha_{j_i})$ to the i-th Worker Node in it and asks to compute $\psi_j\left(f_j\left(\alpha_{j_i}\right)\right)$. By doing this, the Master Node expects to obtain from G_j the following values

$$\left(\psi_j\left(f_j\left(\alpha_{j_1}\right)\right), \ldots, \psi_j(f_j(\alpha_{j_{d_j}}))\right), \tag{14}$$

where $\psi_j(f_j(x))$ is a polynomial of x of degree $D_j\left(\ell_j + T - 1\right)$. It is clear that elements of (14) forms a codeword of $\mathrm{RS}\left(d_j, D_j(\ell_j + T - 1) + 1\right)$. Hence, to accomplish the necessary computations, the Master Node needs at least $D_j(\ell_j + T - 1) + 2A + 1$ nodes to respond.

Theorem 3 (Scheme I)**.** *For d Worker Nodes from whose A are adversary and T are colluding, satisfying $d_j \geq D_j(\ell_j + T - 1) + 2A + 1$ for all $j \in [L]$, $\sum_{j\in[L]} d_j = d$, Scheme I achieves the following performance metrics.*

- *SR:* $\max_{d_1,\ldots,d_L}\{\min_j\{d_j - (D_j(\ell_j + T - 1) + 2A + 1)\}\}$.
- *UC:* $\sum_{j\in[L]} d_j s_j$.
- *DC:* $\sum_{j\in[L]} \left(D_j(\ell_j + T - 1) + 2A + 1\right)$.
- *Computation in MN: $(\ell_j + T)s_j$ multiplications of $\cdot$'s for group G_j.*
- *Computation in WN: no multiplications of $\cdot$'s and the number of multiplications in ψ_j to compute for the Worker Nodes in group G_j.*

C. Scheme II (Lagrange Coded Computing in L rounds)

The Master Node sends $(f_1(\alpha_i), \ldots, f_L(\alpha_i))$ to the i-th Worker Node and requests to compute the values

$$\left(\psi_1(f_1(\alpha_i)), \psi_2(f_2(\alpha_i)), \ldots, \psi_L(f_L(\alpha_i))\right)^T. \tag{15}$$

As a result, the computations of all d involved Worker Nodes can be represented as

$$\left(\begin{pmatrix}\psi_1(f_1(\alpha_1))\\ \psi_2(f_2(\alpha_1))\\ \vdots \\ \psi_L(f_L(\alpha_1))\end{pmatrix}, \begin{pmatrix}\psi_1(f_1(\alpha_2))\\ \psi_2(f_2(\alpha_2))\\ \vdots \\ \psi_L(f_L(\alpha_2))\end{pmatrix}, \ldots, \begin{pmatrix}\psi_1(f_1(\alpha_d))\\ \psi_2(f_2(\alpha_d))\\ \vdots \\ \psi_L(f_L(\alpha_d))\end{pmatrix}\right), \tag{16}$$

where $\psi_j(f_j(x))$ is a polynomial of x of degree $D_j(\ell_j + T - 1)$. The j-th component of the Worker Nodes computations forms a codeword of $\mathrm{RS}(d, D_j(\ell_j + T - 1) + 1)$. There are two ways to recover the required computations, the Worker Node can send its computations all at once or the Worker Node can send its computation one by one. For both cases, the Master Node requires $\max_j\{D_j(\ell_j + T - 1) + 2A + 1\}$ Worker Nodes to respond in order to recover $\psi_1(f_1(x)), \ldots, \psi_L(f_L(x))$. But clearly, the first approach will have a higher Download Cost (which is $L\max_j\{D_j(\ell_j + T - 1) + 2A + 1\}$) in comparison to the second approach for which the Download Cost is $\sum_{j\in[L]}(D_j(\ell_j + T - 1) + 2A + 1)$. In the next theorem, we provide the performance metrics for such a scheme by the second approach.

Theorem 4 (Scheme II)**.** *For $d \geq \max_j\{D_j(\ell_j + T - 1) + 2A + 1\}$ Worker Nodes from whose A are adversary and T are colluding, Scheme II achieves the following performance metrics.*

- *SR:* $d - \max_j\{D_j(\ell_j + T - 1) + 2A + 1\}$.
- *UC:* $d\sum_{j\in[L]} s_j$.
- *DC:* $\sum_{j\in[L]}(D_j(\ell_j + T - 1) + 2A + 1)$.
- *Computation in MN:* $\sum_{j\in[L]}(\ell_j + T)s_j$ *multiplications of $\cdot$'s.*
- *Computation in WN: no multiplications of $\cdot$'s and the total number of multiplications $\times$'s in all ψ_j to compute.*

D. Scheme III (Lagrange Coded Computing in One Round)

The Master Node sends $(f_1(\alpha_i), \ldots, f_L(\alpha_i))$ to the i-th Worker Nodes and asks the i-th Worker Node to compute

$$h(\alpha_i) = \sum_{j\in[L]} \psi_j(f_j(\alpha_i)) \prod_{\beta\in\boldsymbol{\beta}\setminus\boldsymbol{\beta}_j} (\alpha_i - \beta), \tag{17}$$

where $\boldsymbol{\beta}_j = \left\{\beta_i^{(D,j)} \in \mathbb{F} : i \in [\ell_j]\right\}$ for all $j \in [L]$. It can be verified that

$$h\left(\beta_i^{(D,j)}\right) = \gamma_i^{(j)}\psi_j\left(\boldsymbol{X}_i^{(j)}\right), \tag{18}$$

for some $\gamma_i^{(j)} \in \mathbb{F}$, for all $i \in [\ell_j], j \in [L]$. The Master Node expects to obtain the following values

$$\left(h(\alpha_1), h(\alpha_2), \ldots, h(\alpha_d)\right), \tag{19}$$

where $h(x)$ is a polynomial of x of degree $\max_j\{D_j(\ell_j + T - 1) + \ell - \ell_j\}$. This forms a codeword of $\mathrm{RS}(d, \max_j\{D_j(\ell_j + T - 1) + \ell - \ell_j\} + 1)$. Thus, with $\max_j\{D_j(\ell_j + T - 1) + \ell - \ell_j\} + 2A + 1$ responses, the Master Node is able to recover h, even with the existence of A adversary responses. Then by Equation (18), the Master Node computes $h\left(\beta_i^{(D,j)}\right)$ for all $i \in [\ell_j]$ and $j \in [L]$ and recovers all required computations. Let us formally prove Theorem 1.

Proof. Since (19) forms a codeword of $\mathrm{RS}(d, \max_j\{D_j(\ell_j + T - 1) + \ell - \ell_j\} + 1)$, with $\max_j\{D_j(\ell_j + T - 1) + \ell - \ell_j\} + 2A + 1$ responses from those up to A are adversary, the Master Node can recover h by any Reed-Solomon code decoder [17].

Stragglers Resistance. It is clear that the scheme can tolerate $d - (\max_j\{D_j(\ell_j + T - 1) + \ell - \ell_j\} + 2A + 1)$ stragglers.

Upload and Download Cost. The Master Node sends $f_1\left(\alpha_i\right), \ldots, f_L\left(\alpha_i\right)$ to the i-th Worker Node while each $f_j\left(\alpha_i\right)$ contains s_j elements of $\mathbb{U}$. So, the Upload Cost is $d\sum_{j\in[L]} s_j$ elements of $\mathbb{U}$. To do the recovery, the Master Node needs to download $\max_j\{D_j(\ell_j + T - 1) + \ell - \ell_j\} + 2A + 1$ elements of $\mathbb{V}$.

Computation in MN. The Master Node only needs to compute the values of sharing polynomials at some evaluation point. To compute one value of f_j, the Master Node performs $(\ell_j + T)s_j$ many $\cdot$ multiplications (assuming that all multiplications of field elements are pre-computed). Hence, there are $\sum_{j\in[L]}(\ell_j + T)s_j$ multiplication of $\cdot$'s for each node.

Computation in WN. The Worker Nodes apply $\psi_1, \ldots, \psi_L$ to the received shared information. So, the number of $\times$ performed by the Worker Nodes is equal to the number of $\times$ in all ψ_j. In addition, the Worker Nodes also need to do L many $\cdot$ multiplications to compute necessary value of h.

Security. The notion of security is equivalent to zero-mutual information between the dataset $\boldsymbol{X}$ and T values of sharing polynomials. The proof is technical and employs the classical idea of mixing information symbols with random ones from Wiretap Channel II [19]. Due to space limitations, we moved the whole proof to the extended version [18]. □

Method	SR	UC	DC	Computation in MN		Computation in WN	
				$\cdot$	$\times$	$\cdot$	$\times$
Scheme 0	$d-43$	$3d$	43	15	0	0	16
Scheme I	$\left\lfloor \frac{d-38}{3} \right\rfloor$	d	38	3 for G_1 2 for G_2 2 for G_3	0	0	9 in G_1 6 in G_2 1 in G_3
Scheme II	$d-23$	$3d$	38	7	0	0	16
Scheme III	$d-25$	$3d$	25	7	0	3	16

Table 1: Summary of the performance metrics of each schemes to obtain $X_1^{10}, X_2^{10}, X_3^7, X_4^2$ as in Example III-A.

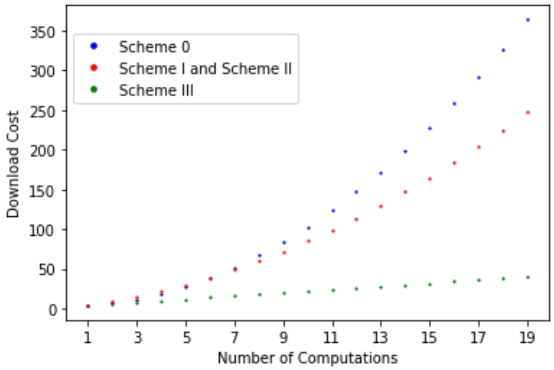

Fig. 2: Comparison of download cost for different schemes

V. Numerical Analysis

The performance metrics for the running example in Section III-A are presented in Table 1. We can see from Table 1 that, in terms of Straggler Resistance, Scheme II outperforms all other schemes as it only requires 24 Worker Nodes to tolerate 1 straggler. Scheme III performs slightly worse in Straggler Resistance as it requires 2 more Worker Nodes to achieve the same performance. Scheme III also requires 3 more $\cdot$ multiplications by Worker Nodes. However, it has a significantly lower Download Cost than other schemes. For further comparison, we consider a more general set-up when the Master Node wants to obtain ℓ computations involving $\psi_1, \ldots, \psi_\ell$ with $\deg(\psi_j) = j$ for all $j \in [\ell]$ which ensures security against one colluding node and tolerates one adversary node, and plot the graph of Download Cost for each scheme against the total number of computations ℓ (see Figure 2). We do note that for the case of $\ell = 1$, Download Cost for all schemes coincides, while for higher values of ℓ, Scheme III performs better.

VI. Conclusion

We considered the problem of efficiently evaluating several arbitrary multivariate polynomials in a distributed computing system. We proposed a new scheme based on the Lagrange Coded Computing framework and compared its efficiency against certain modifications of pre-existing schemes. While our proposed scheme Scheme III (see Theorem 1) has slightly worse straggler resistance in comparison to Scheme II, we observe that Scheme III provides a significantly lower download cost in comparison to Scheme 0, I, and II.

VII. Acknowledgements.

This research / project is supported by the National Research Foundation, Singapore under its Strategic Capability Research Centres Funding Initiative, and Singapore Ministry of Education Academic Research Fund Tier 2 Grants MOE2019-T2-2-083 and MOE-T2EP20121-0007. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore.

Authors would like to thank anonymous reviewers for their useful comments and suggestions.

References

[1] M. Abadi, P. Barham, J. Chen, *et al.*, "Tensorflow: A system for large-scale machine learning," in *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2016, pp. 265–283.

[2] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Speeding up distributed machine learning using codes," in *2016 IEEE International Symposium on Information Theory (ISIT)*, pp. 1143–1147.

[3] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Straggler mitigation in distributed matrix multiplication: Fundamental limits and optimal coding," in *2018 IEEE International Symposium on Information Theory (ISIT)*, pp. 2022–2026.

[4] H. Sun and S. A. Jafar, "The capacity of private computation," in *2018 IEEE International Conference on Communications (ICC)*, pp. 1–6.

[5] M. Soleymani and H. Mahdavifar, "Distributed multi-user secret sharing," in *2018 IEEE International Symposium on Information Theory (ISIT)*, pp. 1141–1145.

[6] M. Soleymani, H. Mahdavifar, and A. S. Avestimehr, "Privacy-preserving distributed learning in the analog domain," [Online]. Available: https://arxiv.org/abs/2007.08803.

[7] M. Soleymani, R. E. Ali, H. Mahdavifar, and A. S. Avestimehr, "List-decodable coded computing: Breaking the adversarial toleration barrier," *IEEE Journal on Selected Areas in Information Theory*, vol. 2, no. 3, pp. 867–878, 2021.

[8] Q. Yu and A. S. Avestimehr, "Coded computing for resilient, secure, and privacy-preserving distributed matrix multiplication," *IEEE Transactions on Communications*, vol. 69, no. 1, pp. 59–72, 2021.

[9] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Polynomial codes: An optimal design for high-dimensional coded matrix multiplication," in *Proceedings of the 31st Annual Conference on Neural Information Processing Systems (NIPS)*, 2017, pp. 4406–4416.

[10] H. A. Nodehi and M. A. Maddah-Ali, "Limited-sharing multi-party computation for massive matrix operations," in *2018 IEEE International Symposium on Information Theory (ISIT)*, pp. 1231–1235.

[11] M. Ben-Or, S. Goldwasser, and A. Wigderson, "Completeness theorems for non-cryptographic fault-tolerant distributed computation," in *Proceedings of Annual ACM Symposium on Theory of Computing (STOC)*, 1988, pp. 1–10.

[12] Q. Yu, S. Li, N. Raviv, S. M. M. Kalan, M. Soltanolkotabi, and S. A. Avestimehr, "Lagrange coded computing: Optimal design for resiliency, security, and privacy," in *Proceedings of 22-nd International Conference on Artificial Intelligence and Statistics (ICAIS)*, vol. 89, PMLR, 2019, pp. 1215–1225.

[13] S. Li, M. A. Maddah-Ali, and A. S. Avestimehr, "Fundamental tradeoff between computation and communication in distributed computing," in *2016 IEEE International Symposium on Information Theory (ISIT)*, pp. 1814–1818.

[14] J. Zhu and S. Li, "Generalized lagrange coded computing: A flexible computation-communication tradeoff," in *2022 IEEE International Symposium on Information Theory (ISIT)*, pp. 832–837.

[15] S. Li and S. Avestimehr, "Coded computing: Mitigating fundamental bottlenecks in large-scale distributed computing and machine learning," *Foundations and Trends® in Communications and Information Theory*, vol. 17, pp. 1–148, 2020.

[16] J. S. Ng, W. Y. B. Lim, N. C. Luong, *et al.*, "A comprehensive survey on coded distributed computing: Fundamentals, challenges, and networking applications," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 3, pp. 1800–1837, 2021.

[17] S. Lin and D. J. Costello, *Error control coding: fundamentals and applications*. Pearson/Prentice Hall, 2004.

[18] W. Kim, S. Kruglik, and H. M. Kiah, "Coded computation of multiple functions," [Online]. Available: https://arxiv.org/abs/2212.07091.

[19] L. H. Ozarow and A. D. Wyner, "Wire-tap channel II," *AT&T Bell Laboratories Technical Journal*, vol. 63, no. 10, pp. 2135–2157, 1984.