An Integrated Method to Deal with Partial Stragglers and Sparse Matrices in Distributed Computations

Anindya Bijoy Das and Aditya Ramamoorthy
Department of Electrical and Computer Engineering, Iowa State University, Ames, IA 50011 USA
{abd149,adityar}@iastate.edu

2022 IEEE International Symposium on Information Theory (ISIT) | 978-1-6654-2159-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/ISIT50566.2022.9834346

***Abstract*—The speed of distributed matrix computations over large clusters is often dominated by the stragglers (slow or failed worker nodes). Several techniques based on coding theory have been introduced to mitigate the straggler issue where every worker node is assigned smaller task(s) of multiplying encoded submatrices of the original matrices. However, many of these methods consider the stragglers as erasures, i.e., they discard the potentially useful partial computations done by the slower workers. Moreover, the "input" matrices can be sparse in many scenarios. In this case encoding schemes that combine a large number of input submatrices can adversely affect the worker computation time.**

In this work, we proposed an integrated approach which addresses both of the issues mentioned above. We allow limited amount of encoding for the submatrices of both $\mathbf{A}$ and $\mathbf{B}$; this helps us to preserve the sparsity of the encoded matrices, so that the worker computation can be fast. Our approach provides a trade-off between straggler resilience and worker computation speed, while utilizing partial computations at the workers. Crucially, at one operating point we can ensure that the failure resilience of the system is optimal. Comprehensive numerical analysis done in Amazon Web Services (AWS) cluster confirms the superiority of our approach when compared with previous methods.

I. Introduction

Distributed matrix computation is often used repeatedly in several large scale machine learning problems. Typically, the job is subdivided into smaller tasks and assigned to multiple worker nodes within the cluster. However, these cluster often suffer from the issue of stragglers (slow or failed workers). Recently, several coding theory techniques [1]–[12] have been proposed to mitigate the effect of stragglers for matrix-vector and matrix-matrix multiplications (see [13] for a tutorial overview). For instance, consider a matrix $\mathbf{A} \in \mathbb{R}^{t\times r}$ and a vector $\mathbf{x} \in \mathbb{R}^t$; the approach in [1] proposes to compute $\mathbf{A}^T\mathbf{x}$ by partitioning the matrix $\mathbf{A}$ into two block-columns as $\mathbf{A} = [\mathbf{A}_0 \mid \mathbf{A}_1]$, and assigning the job of computing $\mathbf{A}_0^T\mathbf{x}$, $\mathbf{A}_1^T\mathbf{x}$ and $(\mathbf{A}_0 + \mathbf{A}_1)^T\mathbf{x}$, respectively, to three different workers. In this way, $\mathbf{A}^T\mathbf{x}$ can be recovered if *any* two out of three workers return their results; in other words, the system is resilient to one straggler. In general, if we have n worker nodes, we define the recovery threshold as the minimum number of workers (τ) that need to finish their jobs such that the result $\mathbf{A}^T\mathbf{x}$ (for matrix-vector multiplication), or $\mathbf{A}^T\mathbf{B}$ (for matrix-matrix multiplication; where $\mathbf{B} \in \mathbb{R}^{t\times w}$) can be recovered from any subset of $\tau \leq n$ worker nodes.

This work was supported in part by the National Science Foundation (NSF) under grant CCF-1910840 and grant CCF-2115200.

Although the recovery threshold is a very important metric considered in coded computation literature, there are certain other important issues that also need to be addressed. For example, in many of the machine learning problems or optimization problems, the corresponding matrices $\mathbf{A}$ and/or $\mathbf{B}$ can be sparse. If we have a linear combination of m submatrices of $\mathbf{A}$, then the density of non-zero entries in the encoded matrices can be up to m-times higher than the density of $\mathbf{A}$ depending on the corresponding sparsity pattern. This can result in a significant increase in the worker node computation time [14], [15]. Thus, developing a scheme that combines relatively few submatrices while continuing to have a good recovery threshold is important. Moreover, the idea of recovery threshold implicitly assumes that no partial computation is received from the remaining $n - \tau$ workers. Thus many of the prior works (see [14]–[22] for some exceptions) treat stragglers as erasures. But a slower worker may not be a useless worker and efficient utilization of the partial computations done by the slower workers could enhance the overall job execution speed.

In this paper, we have proposed a distributed matrix-matrix multiplication approach which can utilize partial computations obtained from the slower workers and deal with the sparse 'input' matrices too. In this approach, most of the assigned $\mathbf{A}$ submatrices are uncoded which can preserve the sparsity of original matrix $\mathbf{A}$ and enhances the worker computation speed. Moreover, in comparison to [15] and [19], we reduce the weight of coding for the encoded submatrices of both $\mathbf{A}$ and $\mathbf{B}$ which further makes the computation faster. Our approach also enjoys the optimal recovery threshold (see [3]) for worker node storage capacities of the form $1/k_A$ and $1/k_B$, respectively. Owing to space limitations, most of the proofs appear in [23].

II. Overview of the Proposed Scheme

We assume that the system has n workers each of which can store $\gamma_A = \frac{1}{k_A}$ and $\gamma_B = \frac{1}{k_B}$ fractions of matrices $\mathbf{A}$ and $\mathbf{B}$, respectively. We partition matrix $\mathbf{A}$ into $\Delta_A = \text{LCM}(n, k_A)$ submatrices (block-columns) as $\mathbf{A}_0, \mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_{\Delta_A-1}$ and matrix $\mathbf{B}$ into $\Delta_B = k_B$ submatrices as $\mathbf{B}_0, \mathbf{B}_1, \mathbf{B}_2, \ldots, \mathbf{B}_{\Delta_B-1}$ and set $\Delta = \Delta_A\Delta_B$. We denote the number of assigned submatrices from $\mathbf{A}$ and $\mathbf{B}$ to any worker as ℓ_A and ℓ_B respectively, so $\ell_A = \frac{\Delta_A}{k_A}$ and $\ell_B = \frac{\Delta_B}{k_B} = 1$. Any worker will compute all pairwise block-products, thus the worker will be responsible for computing $\ell = \ell_A\ell_B = \ell_A$ block-products.

We say that any submatrix $\mathbf{A}_i$, for $i = 0, 1, \ldots, \Delta_A - 1$, appears within a worker node as an uncoded block if $\mathbf{A}_i$ is

Algorithm 1: Proposed scheme for distributed matrix-matrix multiplication

Input : Matrices $\mathbf{A}$ and $\mathbf{B}$, n-number of worker nodes, s-number of stragglers, storage fraction $\gamma_A = \frac{1}{k_A}$ and $\gamma_B = \frac{1}{k_B}$; $s \leq s_m = n - k_A k_B$.

1 Set $x = s_m - s$ and $y = \lfloor \frac{k_A x}{s_m} \rfloor$;
2 Set $\Delta_A = \text{LCM}(n, k_A)$ and $\Delta_B = k_B$ and Partition $\mathbf{A}$ and $\mathbf{B}$ into Δ_A and Δ_B block-columns, respectively;
3 Set $\Delta = \Delta_A \Delta_B$, $p = \frac{\Delta}{n}$ and $\ell = \frac{\Delta_A}{k_A}$;
4 Number of coded submatrices of $\mathbf{A}$ in each worker node, $\ell_c = \ell - p$;
5 Set $\omega = 1 + \lceil \frac{s_m}{k_B} \rceil$ and $\zeta = 1 + k_B - \left\lceil \frac{k_B}{\omega} \right\rceil$;
6 Define $\mathcal{C}_i = \left\{ \mathbf{A}_i,\ \mathbf{A}_{\ell+i},\ \ldots,\ \mathbf{A}_{(k_A-1)\ell+i} \right\}$, and $\lambda_i = 0$, for $i = 0, 1, \ldots, \ell - 1$;
7 **for** $i \leftarrow 0$ **to** $n - 1$ **do**
8 $u \leftarrow i \times \frac{\Delta_A}{n}$;
9 Define $T = \{u, u+1, \ldots, u+p-1\}$ (modulo Δ_A);
10 Assign all $\mathbf{A}_m$'s sequentially from top to bottom to worker node i, where $m \in T$;
11 **for** $j \leftarrow 0$ **to** $\ell_c - 1$ **do**
12 $v \leftarrow u + p + j$ (mod ℓ);
13 Denote $\mathbf{Y} \in \mathcal{C}_v$ as the set of the element submatrices at locations (modulo k_A) $\lambda_v, \lambda_v + 1, \lambda_v + 2, \ldots, \lambda_v + k_A - y - 1$ of $\mathcal{C}_v$;
14 Assign a random linear combination of $\mathbf{A}_q$'s where $\mathbf{A}_q \in \mathbf{Y}$;
15 $\lambda_v \leftarrow \lambda_v + k_A - y$ (modulo k_A);
16 **end**
17 Define $V = \{i, i+1, \ldots, i+\zeta-1\}$ (modulo Δ_B);
18 Assign a random linear combination of $\mathbf{B}_q$'s where $\mathbf{B}_q \in \mathbf{V}$;
19 **end**

Output : $\langle n, \gamma_A, \gamma_B \rangle$-scheme for distributed matrix-matrix multiplication.

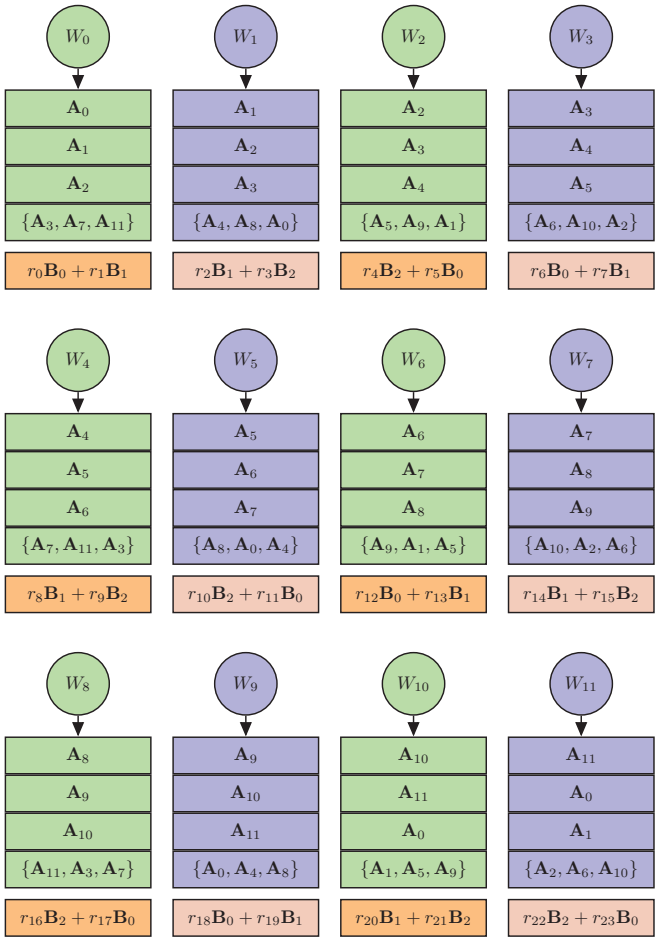

Fig. 1: Distributed matrix multiplication over $n = 12$ worker nodes with $\gamma_A = \gamma_B = \frac{1}{3}$; so $\Delta_A = 12$ and $\Delta_B = 3$. Any $\{\mathbf{A}_i, \mathbf{A}_j, \mathbf{A}_k\}$ means a random linear combination of $\mathbf{A}_i, \mathbf{A}_j$ and $\mathbf{A}_k$. Coefficients r_i's are chosen i.i.d. at random from a continuous distribution.

assigned to that worker as an uncoded submatrix. Similarly, $\mathbf{A}_i$ is said to appear within a worker node in a coded block if a random linear combination of some submatrices including $\mathbf{A}_i$ is assigned to that worker. In each worker node there are locations numbered $0, 1, \ldots, \ell - 1$ where 0 indicates the top location and $\ell - 1$ the bottom location. For this system, if the central node can decode $\mathbf{A}^T\mathbf{B}$ from *any* Q block products (respecting the top-to-bottom computation order), we say that the scheme has the corresponding Q/Δ value. A smaller Q/Δ value of a system indicates that the system can utilize the partial computations of the slower workers more efficiently than a system with higher Q/Δ value.

Our proposed scheme is specified formally in Algorithm 1 and incorporates several ideas that allow us to guarantee the straggler resilience and the Q/Δ value for the scheme. In the discussion below, we provide a top-level overview by appealing to Fig. 1 which shows an example of our scheme with $n = 12$ workers, $\gamma_A = \gamma_B = 1/3$ so that $\Delta_A = 12$ and $\Delta_B = 3$.

Weight of the linear combination of $\mathbf{A}$ *and* $\mathbf{B}$ *submatrices:* Note that $s_m = n - k_A k_B$ is the maximum number of stragglers that the scheme can be resilient to, whereas we want resilience to $s \leq s_m$ stragglers. Line 1 in Alg. 1 sets the parameter $x = s_m - s$. Thus, x measures the relaxation of the straggler resilience that we are able to tolerate. This allows us to reduce the weight of the linear combination of the $\mathbf{A}$ submatrices. In particular, let $y = \lfloor \frac{k_A x}{s_m} \rfloor$. Then our algorithm combines at most $k_A - y$ submatrices of $\mathbf{A}$.

The encoded submatrices of $\mathbf{B}$ are obtained by combining ζ submatrices of $\{\mathbf{B}_0, \mathbf{B}_1, \ldots, \mathbf{B}_{\Delta_B - 1}\}$. Line 5 specifies the assignment of ζ; it can be observed that $\zeta \leq \Delta_B = k_B$.

Assignment of encoded submatrices of $\mathbf{A}$*:* We further divide the set $\{\mathbf{A}_0, \mathbf{A}_1, \ldots, \mathbf{A}_{\Delta_A - 1}\}$ into ℓ disjoint classes $\mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_{\ell-1}$, i.e.,

$$\mathcal{C}_m = \left\{ \mathbf{A}_m,\ \mathbf{A}_{\ell+m},\ \mathbf{A}_{2\ell+m},\ \ldots,\ \mathbf{A}_{(k_A-1)\ell+m} \right\}. \quad (1)$$

This implies that $|\mathcal{C}_m| = k_A$, for $m = 0, 1, \ldots, \ell - 1$, and submatrix $\mathbf{A}_i$ belongs to $\mathcal{C}_{i \,(\text{mod}\, \ell)}$.

The worker nodes are assigned submatrices from each class $\mathcal{C}_m, 0 \leq m \leq \ell-1$ in a block-cyclic fashion; the block shift is specified by Δ_A/n (line 8). In each worker node, the first $p = \Delta/n$ assignments are uncoded, i.e., they correspond to a specific element of the corresponding class. The remaining $\ell_c = \ell - p$ assignments are coded. Each coded assignment corresponds to random linear combination of an appropriate $(k_A - y)$-sized subset of the corresponding class. This is discussed in line 8 – 16 in Alg. 1.

As each location of every worker node is populated by a submatrix from a class $\mathcal{C}_m$ where $0 \leq m \leq \ell-1$, we will occasionally say that the class $\mathcal{C}_m$ appears at a certain location (between 0 to $\ell-1$) at a certain worker node. To ensure that each submatrix of $\mathcal{C}_m$ participates in "almost" the same number of coded assignments, we use a counter λ_i to keep track of the linear combination that will be formed from the corresponding class $\mathcal{C}_i, 0 \leq i \leq \ell-1$ (lines 6, 13 – 15 in Alg. 1).

Example 1. In Fig. 1, we set $x = 0$ so that $y = 0$ and the classes are specified by

$$\mathcal{C}_0 = \{\mathbf{A}_0, \mathbf{A}_4, \mathbf{A}_8\}, \qquad \mathcal{C}_1 = \{\mathbf{A}_1, \mathbf{A}_5, \mathbf{A}_9\},$$
$$\mathcal{C}_2 = \{\mathbf{A}_2, \mathbf{A}_6, \mathbf{A}_{10}\}, \quad \text{and} \quad \mathcal{C}_3 = \{\mathbf{A}_3, \mathbf{A}_7, \mathbf{A}_{11}\}.$$

In this specific case the value of the block shift equals $\Delta_A/n = 1$. It can be observed that the assignment in worker W_0 follows the pattern $\mathcal{C}_0, \mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3$ from top to bottom. The assignments from $\mathcal{C}_0, \mathcal{C}_1, \mathcal{C}_2$ are uncoded, whereas the last one is a random linear combination of all the submatrices in $\mathcal{C}_3$. We also note that there are $p = \Delta/n = 3$ uncoded $\mathbf{A}$ submatrices and one coded $\mathbf{A}$ submatrix in each worker node.

Assignment of encoded submatrices of $\mathbf{B}$*:* For worker W_i, consider the set $V = \{i, i+1, \ldots, i+\zeta-1\} \pmod{\Delta_B}$. A random linear combination of $\mathbf{B}_k$ for $k \in V$ is assigned to worker W_i. We note here that $\zeta \leq k_B$ and can in fact be as small as $\lceil k_B/2 \rceil$ depending upon the values of k_B and s_m.
Order of jobs: Each worker node computes the product of its assigned submatrices in the top to bottom order.

Example 2. In Fig. 1, $s_m = 3$ and $k_B = 3$, so that $\omega = 2$ which implies that $\zeta = 2$. Thus, for instance for worker W_8, the set $V = \{8, 9\}$ and it is assigned a random linear combination of $\mathbf{B}_2$ and $\mathbf{B}_0$. The order of the jobs in W_0 (for example) will be $\mathbf{A}_0^T(r_0\mathbf{B}_0 + r_1\mathbf{B}_1), \mathbf{A}_1^T(r_0\mathbf{B}_0 + r_1\mathbf{B}_1), \mathbf{A}_2^T(r_0\mathbf{B}_0 + r_1\mathbf{B}_1)$ and finally $(r'_3\mathbf{A}_3 + r'_7\mathbf{A}_7 + r'_{11}\mathbf{A}_{11})^T(r_0\mathbf{B}_0 + r_1\mathbf{B}_1)$, where r'_3, r'_7 and r'_{11} represent the random coefficients for the coded $\mathbf{A}$ submatrix in worker W_0.

III. Characteristics of the Proposed Method

In this section, we discuss the coding methods for both matrices $\mathbf{A}$ and $\mathbf{B}$; and state the corresponding lemmas and theorems which describe the properties of our proposed scheme.

A. Coding for Matrix $\mathbf{A}$

Let $\mathbf{U}_i$ denote the subset of worker nodes where $\mathbf{A}_i$ appears in an uncoded block, for $i = 0, 1, \ldots, \Delta_A - 1$. Likewise, $\mathbf{V}_i$ denotes the subset of worker nodes where $\mathbf{A}_i$ appears in a coded block. Our first claim states that the number of coded appearances of any two submatrices in a class can differ by at most *one*. The detailed proof is given in [23].

Claim 1. If the jobs are assigned to the workers according to Alg. 1, for any $\mathbf{A}_i, \mathbf{A}_j \in \mathcal{C}_m$,

$$\Big||\mathbf{V}_i| - |\mathbf{V}_j|\Big| \leq 1.$$

We now present a lemma which outlines the key properties of the structure of encoding submatrices of $\mathbf{A}$. It includes the details on how a given submatrix $\mathbf{A}_i$ and the different classes appear at different locations over all the worker nodes. The detailed proof of the lemma is given in [23].

Lemma 1. Assume that the jobs are assigned to the workers according to Alg. 1, and consider any submatrix $\mathbf{A}_i$, for $i = 0, 1, 2, \ldots, \Delta_A - 1$. Then (i) $|\mathbf{U}_i| = k_B$, (ii) $|\mathbf{V}_i| \geq s$ and $\mathbf{U}_i \cap \mathbf{V}_i = \emptyset$, and (iii) a given class $\mathcal{C}_m$, where $0 \leq m \leq \ell-1$, appears at all different locations $0, 1, \ldots, \ell-1$ within the worker nodes of any worker group $\mathcal{G}_\lambda$, where $0 \leq \lambda \leq c-1$.

The following corollary states that the submatrices in $\mathcal{C}_m$ are assigned to $k_A k_B$ distinct workers as uncoded blocks and to the remaining $s_m = n - k_A k_B$ workers as coded blocks. The proof appears in Appendix of [23].

Corollary 1. If $\mathcal{C}_m = \{\mathbf{A}_m, \mathbf{A}_{\ell+m}, \ldots, \mathbf{A}_{(k_A-1)\ell+m}\}$, then

$$(i) \quad \left|\left(\bigcup_{i:\mathbf{A}_i \in \mathcal{C}_m} \mathbf{U}_i\right)\right| = k_A k_B\,, \quad \left|\left(\bigcup_{i:\mathbf{A}_i \in \mathcal{C}_m} \mathbf{V}_i\right)\right| = s_m\,; \text{ and}$$
$$(ii) \quad \left(\bigcup_{i:\mathbf{A}_i \in \mathcal{C}_m} \mathbf{U}_i\right) \cap \left(\bigcup_{i:\mathbf{A}_i \in \mathcal{C}_m} \mathbf{V}_i\right) = \emptyset\,.$$

B. Coding for Matrix $\mathbf{B}$

To discuss the coding for matrix $\mathbf{B}$, first we consider a $k_B \times n$ matrix, where each column has $\zeta \leq k_B$ non-zero entries which are chosen i.i.d. from a continuous distribution. Moreover, the indices of non-zero entries are consecutive and shifted in a cyclic fashion, reduced modulo k_B. For example, if we have a system with $n = 12$ workers with $k_A = 2$ and $k_B = 5$, then $\zeta = 3$ and the corresponding coding matrix for $\mathbf{B}$, denoted as $\mathbf{R}^B_{k_B,n}$, can be written as

$$\mathbf{R}^B_{k_B,n} = \begin{bmatrix} * & 0 & 0 & * & * & * & 0 & 0 & * & * & * & 0 \\ * & * & 0 & 0 & * & * & * & 0 & 0 & * & * & * \\ * & * & * & 0 & 0 & * & * & * & 0 & 0 & * & * \\ 0 & * & * & * & 0 & 0 & * & * & * & 0 & 0 & * \\ 0 & 0 & * & * & * & 0 & 0 & * & * & * & 0 & 0 \end{bmatrix}. \quad (2)$$

Here $*$ indicates the non-zero entries. The entries at indices $i, i+1, \ldots, i+\zeta-1$ (reduced modulo k_B) are non-zero (chosen i.i.d. from a continuous distribution) within column i of $\mathbf{R}^B_{k_B,n}$ and the other entries are set to zero. The non-zero coefficients are used to specify the random linear combination of the submatrices of $\mathbf{B}$ assigned to worker W_i.

Definition 1. A type i submatrix, for $i = 0, 1, 2 \ldots, k_B - 1$, is a random linear combination of the submatrices, $\mathbf{B}_i, \mathbf{B}_{i+1}, \ldots, \mathbf{B}_{i+\zeta-1}$ (indices reduced modulo k_B). Thus we

can say worker node W_j is assigned a type $j \pmod{k_B}$ submatrix (line 18 of Alg. 1).

Consider the case of $x = 0$ and any $\mathbf{A}_i$, $i = 0, 1, 2, \ldots, \Delta_A - 1$. From Lemma 1, we know $|\mathbf{U}_i| = k_B$ and $|\mathbf{V}_i| = s_m$ (since $x = 0, s = s_m$). Thus $\mathbf{A}_i$ appears at $\sigma = k_B + s_m$ worker nodes. We now provide a claim about the types (*cf.* Def. 1) of the coded submatrices of $\mathbf{B}$ in those σ worker nodes; whose proof is given in [23].

Claim 2. Consider the construction in Alg. 1 with $x = 0$ and let k be the minimum index of the worker node where $\mathbf{A}_i$ appears (uncoded or coded) and consider the worker nodes in $\mathbf{U}_i \cup \mathbf{V}_i$. The assigned submatrices of $\mathbf{B}$ for those worker nodes are, respectively, from types $k, k+1, k+2, \ldots, k+\sigma-1$ (reduced modulo k_B), which are σ consecutive types.

First we provide a lemma which states the property; and the proof is given in details in [23].

Lemma 2. Consider any $\mathbf{A}_i$, $i = 0, 1, 2, \ldots, \Delta_A - 1$, for the case of $x = 0$. Construct a $k_B \times \sigma$ matrix $\mathbf{R}_i$ where the columns of $\mathbf{R}_i$ correspond to the coefficients for coded submatrices of the worker nodes in $\mathbf{U}_i \cup \mathbf{V}_i$; $\sigma = k_B + s_m$. If $\zeta > k_B - \left\lceil \frac{k_B}{\omega} \right\rceil$, any $k_B \times k_B$ submatrix of $\mathbf{R}_i$ is full rank, where $\omega = 1 + \lceil \frac{s_m}{k_B} \rceil$.

For any class $\mathcal{C}_m$, the encoded submatrices of $\mathbf{A}$ within different worker nodes can be specified in terms of a $k_A \times n$ "generator" matrix. Similarly the encoded submatrices of $\mathbf{B}$ within different worker nodes can be specified in terms of a $k_B \times n$ "generator" matrix, as shown by an example in (2). We use this formalism in the discussion below, where we provide the theorem for straggler resilience of our proposed scheme; whose proof is detailed in [23].

Theorem 1. Alg. 1 proposes a distributed matrix-matrix multiplication scheme which is resilient to $s = s_m - x$ stragglers, where $s_m = n - k_A k_B$.

Now we present the result of our work on utilizing the partial computations. It provides the calculation of the value of Q for our scheme for different system parameters. The detailed proof of this theorem is given in Appendix of [23].

Theorem 2. Alg. 1 proposes a distributed matrix-matrix multiplication scheme which provides Q such that $Q_{lb} \leq Q \leq Q_{ub}$. Here the bounds are given by

$$Q_{lb} = \frac{n(\ell-1)}{2} + c\sum_{i=0}^{c_1^0-1}(\ell - i) + c_2^0(\ell - c_1^0) + \left\lceil \frac{s_m y}{k_A} \right\rceil + 1$$

$$\text{and } Q_{ub} = \frac{n(\ell-1)}{2} + c\sum_{i=0}^{c_1^x-1}(\ell - i) + c_2^x(\ell - c_1^x) + 1\,;$$

where $c = \frac{n}{\ell}$, $c_1^x = \left\lfloor \frac{k_A k_B + x - 1}{c} \right\rfloor$, $c_2^x = k_A k_B + x - 1 - c c_1^x$ and $y = \left\lfloor \frac{k_A x}{s_m} \right\rfloor$.

When $x = 0$, then $\tau = k_A k_B$, $c_1^x = c_1^0$ and $c_2^x = c_2^0$, hence $Q_{lb} = Q_{ub} = Q$.

Example 3. We consider an example with $n = 8$ and $\gamma_A = \frac{1}{3}, \gamma_B = \frac{1}{2}$. We partition $\mathbf{A}$ into $\Delta_A = \text{LCM}(n, k_A) = 24$ submatrices and $\mathbf{B}$ into $\Delta_B = k_B = 2$ submatrices. For $x = 0$, the recovery threshold is 6, and $Q_{lb} = Q_{ub} = Q = 59$. However, it should be noted that the central node requires Q block-products to recover $\mathbf{A}^T\mathbf{B}$ in the worst case scenario. In a random scenario, the central node may be able to recover the result from a significantly smaller number of block-products.

Moreover, for $x = 1$, the recovery threshold is 7 and $Q_{lb} = 60 \leq Q = 61 \leq Q_{ub} = 62$ where $Q_{ub} - Q_{lb} = 2$. While Q increases with the increase of x, the worker computation will be faster for sparse "input" matrices.

C. Dealing with Sparse Input Matrices

We now discuss the performance of different schemes when the input matrices are sparse. Consider that $\mathbf{A} \in \mathbb{R}^{t \times r}$ and $\mathbf{B} \in \mathbb{R}^{t \times w}$ are two sparse random matrices, where the entries are chosen independently to be non-zero with probability η. Thus, when we obtain a coded submatrix as the linear combination of k_A submatrices of $\mathbf{A}$, the probability of any entry to be non-zero is approximately $k_A\eta$; we assume η is very small. Similarly, the probability of any entry in a coded submatrix of $\mathbf{B}$ to be non-zero is approximately $k_B\eta$, if it is obtained by a linear combination of k_B submatrices. Now for the dense coded approaches [3], [7], [8], every worker node stores $1/k_A$ and $1/k_B$ fractions of matrices $\mathbf{A}$ and $\mathbf{B}$, and thus the computational complexity of every worker node is approximately $O\left((\eta k_A \eta k_B \times t) \times \frac{r}{k_A}\frac{w}{k_B}\right) = O\left(\eta^2 \times rwt\right)$.

In our proposed approach with $x = 0$, the coded submatrix for $\mathbf{B}$ is obtained by a random linear combination of ζ uncoded submatrices. Thus, the computational complexity to compute the block product between an uncoded $\mathbf{A}$ and coded $\mathbf{B}$ submatrix is $O\left((\eta \times \eta\zeta \times t)\frac{r}{\Delta_A}\frac{w}{k_B}\right) = O\left(\eta^2 \times rwt \times \frac{\zeta}{\Delta_A \Delta_B}\right)$; and to compute the block product between a coded $\mathbf{A}$ and coded $\mathbf{B}$ submatrix is $O\left((\eta k_A \times \eta\zeta \times t)\frac{r}{\Delta_A}\frac{w}{k_B}\right) = O\left(\eta^2 \times rwt \times \frac{\zeta k_A}{\Delta_A \Delta_B}\right)$. Since the workers need to compute p uncoded-coded and $\ell - p$ coded-coded block products, the total computation cost is approximately $O\left(\eta^2 \times rwt \times \left(\frac{\zeta}{n} + \frac{\zeta s_m}{n k_B}\right)\right)$. Thus, the computational complexity of every worker node of our approach is around $O\left(\frac{\zeta}{n}\left(1 + \frac{s_m}{k_B}\right)\right)$ times smaller than that of the dense coded approaches. Thus our approach is much more suited to sparse input matrices than the dense coded approaches in [3], [7], [8].

Remark 1. Our scheme is also applicable for distributed matrix-vector multiplication. In that case, the usual assumption is that each worker can store the whole vector $\mathbf{x}$, and we can prove similar theorems by substituting $\gamma_B = 1$ (or $k_B = 1$).

IV. Numerical Experiments and Comparisons

In this section, we compare the performance of our approach with different competing methods in terms of various metrics. It should be noted that [3], [7], [8], [15], [19] and our proposed method (for $x = 0$) have the same recovery threshold, communication load and worker computational load when

TABLE I: Comparison of worker computation time (in seconds) for matrix-matrix multiplication for $n = 24, \gamma_A = \frac{1}{4}$ and $\gamma_B = \frac{1}{5}$ (*for [10], we assume $\gamma_A = \frac{2}{5}$ and $\gamma_B = \frac{1}{4}$) when randomly chosen 95%, 98% and 99% entries of both of matrices $\mathbf{A}$ and $\mathbf{B}$ are zero.

METHODS	s	WORKER COMPUTATION TIME (S)		
		$\mu = 99\%$	$\mu = 98\%$	$\mu = 95\%$
POLY CODE [3]	4	1.23	3.10	8.21
ORTHO-POLY [7]	4	1.25	3.13	8.14
RKRP CODE [8]	4	1.21	3.09	8.10
CONV. CODE* [10]	4	1.92	5.07	10.72
SCS OPT. SCH. [15]	4	0.91	1.89	4.67
METHOD IN [19]	4	0.76	1.45	4.71
PROP. SCH. ($x = 0$)	4	0.54	0.97	3.68
PROP. SCH. ($x = 2$)	2	0.45	0.81	3.21

the "input" matrices are dense. However in case of sparse matrices, our proposed method has some significant advantages. Exhaustive numerical experiments done in `AWS` (Amazon Web Services) cluster support our claims. In order to carry out the experiments, a `t2.2xlarge` machine is used as the central node and `t2.small` machines are used as the worker nodes.

Worker Computation Time: We consider a distributed matrix multiplication system with $n = 24$ workers with $\gamma_A = \frac{1}{4}$ and $\gamma_B = \frac{1}{5}$. The input matrices $\mathbf{A}$ and $\mathbf{B}$, of sizes 12000×15000 and 12000×13500, are assumed to be sparse. We assume three different cases where sparsity (μ) of the input matrices are 95%, 98% and 99%, respectively, which indicates that randomly chosen 95%, 98% and 99% entries of both of matrices $\mathbf{A}$ and $\mathbf{B}$ are zero. Table I shows the corresponding comparison of the different methods for the worker computation time for this example. It can easily be verified from the table that the workers take significantly less time to compute the submatrix products for our proposed approach than the other methods [3], [7], [8], [10]. This is because in the other methods the coded submatrices are linear combinations of all $k_A = 4$ submatrices from $\mathbf{A}$ (or $k_B = 5$ submatrices from $\mathbf{B}$).

The works most closely related to our approach are our prior works in [15] (SCS optimal scheme, see Section V in [15]) and [19]. All these approaches partition $\mathbf{A}$ and $\mathbf{B}$ into $\Delta_A = \text{LCM}(n, k_A)$ and $\Delta_B = k_B$ submatrices, respectively. Moreover, all of them assign some uncoded submatrices of $\mathbf{A}$ and then some coded submatrices of $\mathbf{A}$; and assign a coded submatrix of $\mathbf{B}$ to each of the worker nodes.

However, there are some crucial differences. [15] requires the weight of the encoding of the $\mathbf{A}$ submatrices to be $\Delta_A - p$ which is much higher than $k_A - y$. Furthermore [15] and [19] do not allow for a trade-off between the number of stragglers and the weight of the coded $\mathbf{A}$ submatrices; this is a salient feature of our approach. Moreover, for the coding of $\mathbf{B}$, the schemes in [15] and [19] assign linear combinations of k_B submatrices, whereas in our proposed approach we assign linear combinations of ζ submatrices where ζ can be significantly smaller than k_B. We emphasize that the our proposed approach continues to enjoy the optimal straggler resilience when $x = 0$. However, we point out that we lose a small amount in the

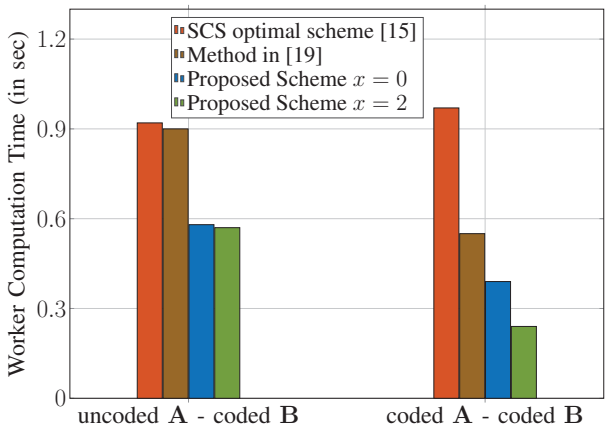

Fig. 2: The comparison of worker computation time for $\mu = 98\%$ sparse matrices. We show the time required for multiplying p uncoded $\mathbf{A}$ submatrices with the coded $\mathbf{B}$ submatrix and the time required for multiplying $(\ell - p)$ coded $\mathbf{A}$ submatrices with the coded $\mathbf{B}$ submatrix.

TABLE II: Comparison of utilization of partial stragglers and numerical stability among different approaches

METHODS	Q/Δ	κ_{worst}
POLY CODE [3]	N/A	2.40×10^{10}
ORTHO-POLY [7]	N/A	1.96×10^{6}
RKRP CODE [8]	N/A	2.83×10^{5}
CONV CODE* [10]	N/A	2.65×10^{4}
SCS OPT. SCH. [15]	$124/120$	4.93×10^{6}
METHOD IN [19]	$139/120$	2.94×10^{6}
PROP. SCH. ($x = 0$)	$139/120$	2.37×10^{6}
PROP. SCH. ($x = 2$)	$\frac{141}{120} \leq \frac{Q}{\Delta} \leq \frac{142}{120}$	2.25×10^{4}

Q/Δ metric, with respect to SCS optimal scheme in [15], but we match the value of [19] for $x = 0$.

Value of Q/Δ: Many of the available approaches in coded matrix computations literature [3], [7], [8], [10] cannot leverage the slow workers, because they assign exactly one job to each of the worker nodes. On the other hand, the proposed approach assigns multiple jobs to each of the worker nodes which allows the opportunity to leverage partial stragglers.

Table II shows the comparison among different methods in terms of Q/Δ for the same example of $n = 24$ worker nodes. We can see that our approach has a slightly higher Q/Δ than the approach in [15] and the value of Q/Δ can increase for the choice of $x > 0$. However our proposed approach has a significant gain over [15] in terms of worker computation speed as shown in Table I. It should be noted that the approaches in [3], [7], [8] can be extended to utilizing the partial stragglers, but that can lead to numerically instability of the systems [15].

Numerical Stability: For the same system we find the worst case condition number (κ_{worst}) of the decoding matrices over all different choices of s stragglers for different methods and present them in Table II. As expected, the polynomial code approach [3] has a very high κ_{worst}. The works in [7], [8], [10] have significantly smaller κ_{worst}; however they cannot leverage the partial computations of the slower worker nodes. Our proposed methods, [15] and [19] can utilize the partial stragglers and provide similar κ_{worst} values compared to [6].

REFERENCES

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

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

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

[4] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Straggler mitigation in distributed matrix multiplication: Fundamental limits and optimal coding," *IEEE Trans. on Info. Th.*, vol. 66, no. 3, pp. 1920–1933, 2020.

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

[6] A. B. Das and A. Ramamoorthy, "Distributed matrix-vector multiplication: A convolutional coding approach," in *Proc. of IEEE Intl. Symp. on Info. Th.*, 2019, pp. 3022–3026.

[7] M. Fahim and V. R. Cadambe, "Numerically stable polynomially coded computing," in *Proc. of IEEE Intl. Symp. on Info. Th.*, 2019, pp. 3017–3021.

[8] A. M. Subramaniam, A. Heidarzadeh, and K. R. Narayanan, "Random Khatri-Rao-product codes for numerically-stable distributed matrix multiplication," in *Proc. of Annual Conf. on Comm., Control, and Computing (Allerton)*, Sep. 2019, pp. 253–259.

[9] A. Ramamoorthy and L. Tang, "Numerically stable coded matrix computations via circulant and rotation matrix embeddings," *IEEE Trans. on Info. Th.*, vol. 68, no. 4, pp. 2684–2703, 2022.

[10] A. B. Das, A. Ramamoorthy, and N. Vaswani, "Efficient and robust distributed matrix computations via convolutional coding," *IEEE Trans. on Info. Th.*, vol. 67, no. 9, pp. 6266–6282, 2021.

[11] A. K. Pradhan, A. Heidarzadeh, and K. R. Narayanan, "Factored LT and factored raptor codes for large-scale distributed matrix multiplication," *IEEE J. Select. Areas Info. Th.*, vol. 2, no. 3, pp. 893–906, 2021.

[12] L. Tang, K. Konstantinidis, and A. Ramamoorthy, "Erasure coding for distributed matrix multiplication for matrices with bounded entries," *IEEE Comm. Letters*, vol. 23, no. 1, pp. 8–11, 2019.

[13] A. Ramamoorthy, A. B. Das, and L. Tang, "Straggler-resistant distributed matrix computation via coding theory: Removing a bottleneck in large-scale data processing," *IEEE Sig. Proc. Mag.*, vol. 37, no. 3, pp. 136–145, 2020.

[14] S. Wang, J. Liu, and N. Shroff, "Coded sparse matrix multiplication," in *Proc. of Intl. Conf. on Machine Learning (ICML)*, 2018, pp. 5152—5160.

[15] A. B. Das and A. Ramamoorthy, "Coded sparse matrix computation schemes that leverage partial stragglers," *IEEE Trans. on Info. Th.*, 2022 (to appear), [Online] Available: https://arxiv.org/abs/2012.06065.

[16] S. Kiani, N. Ferdinand, and S. C. Draper, "Exploitation of stragglers in coded computation," in *Proc. of IEEE Intl. Symp. on Info. Th.*, 2018, pp. 1988–1992.

[17] A. Mallick, M. Chaudhari, U. Sheth, G. Palanikumar, and G. Joshi, "Rateless codes for near-perfect load balancing in distributed matrix-vector multiplication," *Proceedings of the ACM on Meas. and Analysis of Comp. Syst.*, vol. 3, no. 3, pp. 1–40, 2019.

[18] S. Kianidehkordi, N. Ferdinand, and S. C. Draper, "Hierarchical coded matrix multiplication," *IEEE Trans. on Info. Th.*, vol. 67, no. 2, pp. 726–754, 2021.

[19] A. B. Das and A. Ramamoorthy, "A unified treatment of partial stragglers and sparse matrices in coded matrix computation," in *Proc. of IEEE Info. Th. Workshop*, 2021, pp. 1–6.

[20] E. Ozfatura, S. Ulukus, and D. Gündüz, "Distributed gradient descent with coded partial gradient computations," in *Proc. of IEEE Intl. Conf. on Acoustics, Speech and Sig. Proc. (ICASSP)*, 2019, pp. 3492–3496.

[21] A. Ramamoorthy, L. Tang, and P. O. Vontobel, "Universally decodable matrices for distributed matrix-vector multiplication," in *Proc. of IEEE Intl. Symp. on Info. Th.*, 2019, pp. 1777–1781.

[22] B. Hasırcıoğlu, J. Gómez-Vilardebó, and D. Gündüz, "Bivariate hermitian polynomial coding for efficient distributed matrix multiplication," in *Proc. of IEEE Glob. Comm. Conf. (GLOBECOM)*, 2020, pp. 1–6.

[23] A. B. Das and A. Ramamoorthy, "A unified treatment of partial stragglers and sparse matrices in coded matrix computation," preprint, 2021, [Online] Available: https://arxiv.org/abs/2109.12070.