Distributed Matrix Computations With Low-Weight Encodings

Anindya Bijoy Das, *Member, IEEE*, Aditya Ramamoorthy, *Senior Member, IEEE*, David J. Love, *Fellow, IEEE*, and Christopher G. Brinton, *Senior Member, IEEE*

***Abstract*—Straggler nodes are well-known bottlenecks of distributed matrix computations which induce reductions in computation/communication speeds. A common strategy for mitigating such stragglers is to incorporate Reed-Solomon based MDS (maximum distance separable) codes into the framework; this can achieve resilience against an optimal number of stragglers. However, these codes assign dense linear combinations of submatrices to the worker nodes. When the input matrices are sparse, these approaches increase the number of non-zero entries in the encoded matrices, which in turn adversely affects the worker computation time. In this work, we develop a distributed matrix computation approach where the assigned encoded submatrices are random linear combinations of a small number of submatrices. In addition to being well suited for sparse input matrices, our approach continues to have the optimal straggler resilience in a certain range of problem parameters. Moreover, compared to recent sparse matrix computation approaches, the search for a "good" set of random coefficients to promote numerical stability in our method is much more computationally efficient. We show that our approach can efficiently utilize partial computations done by slower worker nodes in a heterogeneous system which can enhance the overall computation speed. Numerical experiments conducted through Amazon Web Services (AWS) demonstrate up to 30% reduction in per worker node computation time and 100× faster encoding compared to the available methods.**

***Index Terms*—Distributed computing, MDS codes, stragglers, condition number, sparsity.**

I. Introduction

MATRIX operations are the fundamental building blocks of data-intensive algorithms (e.g., machine learning modeling) executed on contemporary computing platforms. The ever-increasing volumes of data generated by end users translates to high dimensional matrices for storage and processing, underscoring the potential benefits of distributed computation. The idea behind these schemes is to break down the whole matrix computation into smaller tasks and distribute them across multiple worker nodes. In these systems, it is well known that the overall job execution time can be dominated by slower (or failed) worker nodes, which are referred to as stragglers.

Recently, a number of coding theory techniques [1], [2], [3], [4], [5], [6], [7], [8], [9] have been proposed to mitigate the effect of stragglers for distributed matrix multiplications. For example, consider the computation of $\mathbf{A}^T\mathbf{x}$ (where $\mathbf{A} \in \mathbb{R}^{t\times r}$ and $\mathbf{x} \in \mathbb{R}^t$) across three nodes. A popular approach [1] would partition $\mathbf{A}$ as $\mathbf{A} = [\mathbf{A}_0 \mid \mathbf{A}_1]$, and assign nodes W_0, W_1 and W_2 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. While each worker must carry half of the overall computational load, we can recover $\mathbf{A}^T\mathbf{x}$ as soon as *any* two out of three workers return their results. In other words, the system is resilient to one straggler. The recovery threshold, i.e., the minimum number of worker nodes (τ) that need to finish their respective jobs such that the result $\mathbf{A}^T\mathbf{x}$ can be recovered from any subset of τ worker nodes, has emerged as an important optimization metric.

Similar to the matrix-vector case, coding theory techniques have been developed for distributed matrix-matrix multiplication, i.e., to compute $\mathbf{A}^T\mathbf{B}$, with the same goal: to minimize the recovery threshold [4], [10]. Under the assumption of a homogeneous system where each worker can store $1/k_A$ and $1/k_B$ fraction of matrices $\mathbf{A}$ and $\mathbf{B}$, respectively, and each node is assigned a $1/k_{AB}$ fraction ($k_{AB} = k_A k_B$) of the overall load of computing $\mathbf{A}^T\mathbf{B}$, the achievable recovery threshold is lower bounded by $k_A k_B$ [4]. Note that, with the assumption $k_A = k_B$, the approach in [10] achieves a recovery threshold of $2k_A - 1$ whereas the method in [4] provides a threshold k_A^2. However, for given matrices $\mathbf{A} \in \mathbb{R}^{t\times r}$ and $\mathbf{B} \in \mathbb{R}^{t\times w}$, the per worker node computational complexity of the approach in [10] is $\mathcal{O}(\frac{rwt}{k_A})$, which is around k_A times higher than the corresponding computational complexity of every node for the approach in [4], which is $\mathcal{O}(\frac{rwt}{k_A^2})$.

Several works based on maximum distance separable (MDS) codes [2], [4], [11], [12] have met this optimal recovery threshold. However, they have other limitations in practical distributed computing systems. First, real-world data matrices are often sparsely populated (e.g., see examples in [13]), leading to structures that can be exploited for computational

Manuscript received 29 January 2023; revised 23 May 2023 and 16 July 2023; accepted 21 August 2023. Date of publication 30 August 2023; date of current version 14 September 2023. This work was supported in part by the National Science Foundation (NSF) under Grant CNS-2146171, Grant CCF-1910840, and Grant CCF-2115200; and in part by the Office of Naval Research (ONR) under Grant N000142212305 and Grant N000142112472. The material in this work has appeared in part at the 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Rhodes Island, Greece, June 2023 [DOI: 10.1109/ICASSP49357.2023.10095450], and the 2023 IEEE International Symposium on Information Theory (ISIT), Taipei, Taiwan, June 2023 [DOI: 10.1109/ISIT54713.2023.10206445]. *(Corresponding author: Anindya Bijoy Das.)*

Anindya Bijoy Das, David J. Love, and Christopher G. Brinton are with the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47906 USA (e-mail: das207@purdue.edu; djlove@purdue.edu; cgb@purdue.edu).

Aditya Ramamoorthy is with the Department of Electrical and Computer Engineering, Iowa State University, Ames, IA 50011 USA (e-mail: adityar@iastate.edu).

Digital Object Identifier 10.1109/JSAIT.2023.3308768

efficiency gains. However, MDS code-based techniques are not built to preserve data sparsity which can significantly increase the overall job execution time. In addition, most of the available methods are not focused on the case of heterogeneous systems, where different worker nodes are rated with different storage capacities and speeds (e.g., if the computation is being distributed across wireless edge devices).

Motivated by these limitations, in this work, we develop a novel approach for distributed matrix-vector and matrix-matrix multiplication which explicitly accounts for sparsity in the input matrices. Our proposed approach assigns coded submatrices as random linear combinations of a very small number of uncoded submatrices to preserve the inherent sparsity up to a certain level. Moreover, unlike the straggler optimal schemes in [5] and [8], our approach involves a much less computationally burdensome process to find a "good" set of random coefficients for numerical stability of the system. In addition, our approach addresses the case where the nodes are heterogeneous in nature, having different computation and communication speeds.

The paper is organized as follows. In Section II, we discuss the problem formulation, related literature background and summarize our contributions. Then, in Sections III and IV, we present the details of our distributed matrix-vector and matrix-matrix multiplication schemes, respectively, with results on straggler resilience, extension to heterogeneous systems, and utilization of partial computations. Next, Section V discusses different properties of our proposed schemes in terms of worker computation delay, communication delay, numerical stability and the required time to find a "good" set of coefficients. In Section VI, we present numerical experiments comparing the performance of our proposed method with other recent approaches. Finally, Section VII concludes the paper with a discussion of possible future directions.

II. Problem Formulation, Background and Summary of Contributions

A. Problem Formulation

We consider a distributed system comprised of a central node and a set of worker nodes aiming to compute $\mathbf{A}^T\mathbf{x}$ for matrix-vector multiplication or $\mathbf{A}^T\mathbf{B}$ for matrix-matrix multiplication, for given matrices $\mathbf{A} \in \mathbb{R}^{t\times r}$, $\mathbf{B} \in \mathbb{R}^{t\times w}$ and vector $\mathbf{x} \in \mathbb{R}^t$. In the homogeneous setting, we assume a system of n worker nodes rated with the same computation and communication speeds, for local data processing and transmitting/receiving processed data, respectively. In particular, each worker can store the equivalent of $\gamma_A = \frac{1}{k_A}$ fraction of $\mathbf{A}$ and the whole vector $\mathbf{x}$ (or $\gamma_B = \frac{1}{k_B}$ fraction of $\mathbf{B}$) for matrix-vector multiplication (or matrix-matrix multiplication). In the heterogeneous setting, by contrast, workers are rated with different storage capacity and speeds. Stragglers arise in practice from speed variations or failures experienced by the nodes at particular times [2].

In our approach in the homogeneous setting, we first partition matrices $\mathbf{A}$ and $\mathbf{B}$ into k_A and k_B disjoint block-columns, respectively, as $\mathbf{A} = \begin{bmatrix}\mathbf{A}_0 & \mathbf{A}_1 & \cdots \mathbf{A}_{k_A-1}\end{bmatrix}$ and $\mathbf{B} = \begin{bmatrix}\mathbf{B}_0 & \mathbf{B}_1 & \cdots & \mathbf{B}_{k_B-1}\end{bmatrix}$, such that $\mathbf{A}_i \in \mathbb{R}^{t\times r/k_A}$ and $\mathbf{B}_j \in \mathbb{R}^{t\times w/k_B}$, for $0 \le i \le k_A - 1$ and $0 \le j \le k_B - 1$. Next, we will assign a random linear combination of some block-columns of $\mathbf{A}$ and the vector $\mathbf{x}$ (or another random linear combination of some block-columns of $\mathbf{B}$) to each worker node for matrix-vector multiplication (for matrix-matrix multiplication). As discussed in Section I, assigning dense linear combinations can destroy the inherent sparsity of the matrices. Instead, we aim to assign linear combinations of a lesser number of submatrices. To quantify this, we define the "weight" of the encoded submatrices as follows.

Definition 1: We define the weights of the encoding process ω_A and ω_B for matrices A and B, respectively, as the number of submatrices that are linearly combined to obtain each encoded submatrix. We assume uniform weights across the worker nodes, i.e., the combination received by each worker is formed from the same number of submatrices.

Thus, in this work, we consider the problem of minimizing recovery threshold for both matrix-vector and matrix-matrix multiplication in the homogeneous system while maintaining low ω_A and ω_B for the assigned submatrices. We will extend the resulting approach for the heterogeneous setting as well by assigning tasks proportional to worker capabilities. Note that we outline the important notations in Appendix A (Table V).

B. Background and Literature Review

Several coded computation schemes have been proposed for matrix multiplication [1], [2], [3], [4], [5], [6], [7], [8], [9], [11], [12], [14], [15], [16], [17], [18], [19] in recent years. We give a comparative summary between these schemes in terms of properties they support in Table I; for a more detailed overview, we refer the reader to [20]. Here we begin with an illustration of the polynomial code approach [4].

Consider a homogeneous system with $n = 5$ worker nodes where each worker can store $\gamma_A = \gamma_B = 1/2$ fraction of both matrices $\mathbf{A}$ and $\mathbf{B}$. We partition matrices $\mathbf{A}$ and $\mathbf{B}$ into $k_A = k_B = 2$ block-columns each, as $\mathbf{A}_0$, $\mathbf{A}_1$ and $\mathbf{B}_0, \mathbf{B}_1$, respectively. Next we define two matrix polynomials as $\mathbf{A}(z) = \mathbf{A}_0 + \mathbf{A}_1 z$ and $\mathbf{B}(z) = \mathbf{B}_0 + \mathbf{B}_1 z^2$, so that $\mathbf{A}^T(z)\,\mathbf{B}(z) = \mathbf{A}_0^T\mathbf{B}_0 + \mathbf{A}_1^T\mathbf{B}_0 z + \mathbf{A}_0^T\mathbf{B}_1 z^2 + \mathbf{A}_1^T\mathbf{B}_1 z^3$. The central node evaluates $\mathbf{A}(z)$ and $\mathbf{B}(z)$ at $n = 5$ distinct real numbers and transmits the corresponding matrices to worker node W_i, for $i = 0, 1, 2, \ldots, n-1$. Now each node computes its respective assigned matrix-matrix block-product and returns the result to the central node. Since $\mathbf{A}^T(z)\,\mathbf{B}(z)$ is a degree-3 polynomial, once the central node receives results from the fastest $\tau = 4$ worker nodes, it can decode all the coefficients in $\mathbf{A}^T(z)\,\mathbf{B}(z)$, hence, $\mathbf{A}^T\mathbf{B}$. Thus, the recovery threshold is $\tau = 4$ and the system is resilient to $s = 1$ straggler.

Unlike the schemes in [18], [24] which are sub-optimal in terms of straggler resilience, the polynomial code approach is among the first to provide the optimal recovery threshold. However, recent works on matrix computations have identified metrics beyond recovery threshold that also need to be considered. Here we discuss the importance of factoring them into our methodology.

Sparsity of matrices $\mathbf{A}$ *and* $\mathbf{B}$*:* Sparsity is quite prevalent in real-world datasets with applications in optimization,

TABLE I
COMPARISON AMONG EXISTING WORKS ON CODED MATRIX-COMPUTATIONS (THE APPROACH IN [10] INVOLVES A HIGHER COMPUTATIONAL COMPLEXITY)

CODES	MAT-MAT MULT?	OPTIMAL THRESH.?	NUMERICAL STABILITY?	SPARSELY CODED?	HETERO. SYSTEM?
REPETITION CODES	✓	✗	✓	✓	✓
RATELESS CODES [18]	✗	✗	✓	✗	✓
GEOMETRIC CONV. CODES [19]	✗	✓	✓	✗	✓
PROD. CODES [21], FACT. CODES [9]	✓	✗	✓	✗	✗
POLYNOMIAL CODES [4]	✓	✓	✗	✗	✓
MATDOT CODES [10]	✓	✓	✗	✗	✓
BIVARIATE POLY. CODE [22]	✓	✓	✗	✗	✓
ORTHOPOLY [11], RKRP CODE [12], CONV. CODE [2], CIR. ROT. MAT. [23]	✓	✓	✓	✗	✓
SPARSE PRIVATE APPROACH [14]	✗	✗	✓	✓	✓
β-LEVEL CODING [5]	✓	✗	✓	✓	✗
SCS OPTIMAL SCHEME [5]	✓	✓	✓	✓	✗
CLASS-BASED SCHEME [8]	✓	✓	✓	✓	✗
Proposed Scheme	✓	✓	✓	✓	✓

deep learning, power systems, electromagnetism etc. (see [13] for such examples). In other words, there are many practical problems where the corresponding matrices to be operated on are sparse, which can be exploited to significantly reduce matrix computation time [24]. Consider two column vectors of length m, denoted by $\mathbf{a}$ and $\mathbf{y}$, where $\mathbf{a}$ has around ψm non-zero entries ($0 < \psi << 1$). It takes approximately $2\psi m$ floating point operations (FLOPs) to compute $\mathbf{a}^T\mathbf{y}$, whereas it could take around $2m$ FLOPs if $\mathbf{a}$ was dense.

As shown in Table I, many existing coding approaches do not preserve sparsity. For example, in the polynomial code approach [4] or its variants [11], the encoded submatrices of $\mathbf{A}$ and $\mathbf{B}$ are obtained by linearly combining k_A and k_B submatrices. Thus, the number of non-zero entries can increase by up to k_A and k_B times, respectively, compared with the original matrices $\mathbf{A}$ and $\mathbf{B}$, which would lead to a significantly higher computation time. This underscores the importance of developing schemes that minimize the number of uncoded submatrices that are combined.

Note that there are several methods available in the literature [5], [8], [14], [24], [25] which demonstrate some advantages in sparse matrix computations. However, the approach in [14] is not developed for matrix-matrix multiplication, and the approach in [25] has different assumptions than ours: the central node is also responsible for some computations. In addition, the approach in [24] and β-level coding scheme in [5] do not meet the exact optimal recovery threshold, which require more nodes to finish their respective tasks. While the approach in [8] and the SCS-optimal scheme in [5] meet the exact optimal recovery threshold, they partition the matrices into large number of block-columns, hence, need a large amount of time to find a "good" set of coefficients to obtain the linear combinations (details are given in Sections V-B and VI). Besides, some of the assigned submatrices in those approaches are densely coded, thus, an improved coding approach could further optimize the computational speed over those methods.

Numerical stability of the system: Another important issue is the numerical stability of the system. Since the encoding and decoding methods in coded computation operate over the real field, the decoding of the unknowns from a system of equations can be quite inaccurate if the corresponding system matrix is ill-conditioned. There can be a blow-up of round-off errors in the decoded result owing to the high condition numbers of the corresponding decoding matrices. For example, the polynomial code approach in [4] incorporates Vandermonde matrices into the encoding process which are known to be ill-conditioned. Literature aiming to address this issue [2], [11], [12] has emphasized that the worst case condition number (κ_{worst}) over all different choices of stragglers should be treated as an important metric for minimization.

However, many of the numerically stable methods are based on random codes [2], [5], [8], [12] which require significant time to find a "good" set of random coefficients that make the system numerically stable. The idea is to first generate a set of random coefficients and find the κ_{worst} over all choices of stragglers. Next, repeat this step several times (say, 20) and retain the set of random coefficients which gives minimum κ_{worst}. The latency incurred by this process increases with the number of worker nodes and can delay the encoding process.

Heterogeneous system and partial computations done by the stragglers: Another important issue arises in distributed computing with heterogeneous worker nodes (different memory, speed and bandwidth) where algorithms based on homogeneous assumptions may lead to sub-optimal performance [26]. In this work, we address this issue by assigning each worker a processing load according to its memory and speed. Some of the approaches [4], [11], [12] that were originally developed for a homogeneous system could similarly be extended to the heterogeneous setting.

Moreover, the performance of distributed computing often depends on how well the associated scheme is able to utilize (rather than discard) partial computations done by slower workers [27]. Specifically, efficient utilization of the partial computations done by slower nodes may enhance the overall speed. To address this, in our scheme in the heterogeneous setting, we assign *multiple* smaller tasks to some workers, with

the size of each task dictated by the limitations of the least powerful worker. The workers then compute their respective tasks and return the results sequentially. With this division, define Q to be the minimum number of block products that must be returned to the central node for the guarantee of decoding the intended result ($\mathbf{A}^T\mathbf{x}$ or $\mathbf{A}^T\mathbf{B}$) successfully even in the worst case, i.e., from *any* Q block products returned across all the worker nodes (respecting the computation order in each node) [5]. We associate the scheme with the metric Q/Δ, where Δ is the number of submatrix products to be recovered in the intended product ($\Delta = k_A$ or k_Ak_B for matrix-vector or matrix-matrix case, respectively). Q/Δ is always lower bounded by 1, and a system with a small Q/Δ can utilize the partial computations of the slower workers efficiently [5], [8].

C. Summary of Contributions

The contributions of this work can be summarized as follows.

- We develop novel straggler-resilient approaches for distributed matrix-vector and matrix-matrix multiplication. For a system with n homogeneous worker nodes, each of which can store $1/k_A$ and $1/k_B$ fractions of matrices $\mathbf{A}$ and $\mathbf{B}$, respectively, our developed approach for distributed matrix-matrix multiplication can be resilient to *any* $s = n - k_Ak_B$ stragglers (where $s \leq \min(k_A, k_B)$). Thus, our approach is straggler optimal, since it meets the lower bounds on straggler resilience as given in [4]. In addition, with the assumption that each node stores the whole vector $\mathbf{x}$, our approach for the matrix-vector case is also straggler-optimal.
- While our approaches are applicable to any types of matrices, it is specifically suited to sparse "input" matrices. A very limited number of uncoded submatrices are linearly combined in our encoding, so that the inherent sparsity of the input matrices can be preserved up to certain level. For example, in a system with $n = 12$, $k_A = 10$ and $s = 2$ for distributed matrix-vector multiplication, the traditional dense codes [2], [4], [11], [12] require linear combinations of $k_A = 10$ submatrices, whereas our scheme combines only $s + 1 = 3$ uncoded submatrices (see Example 1). Thus, our approach will be significantly impactful in the scenario where $s+1 < k_A$ (as mentioned in Remark 1).
- We also show that our proposed approaches are numerically stable. It has been verified by comparing other approaches in terms of the worst case condition number over different choices of stragglers. Moreover, our scheme involves a significantly less computationally expensive step compared to [5], [8] to find a "good" set of random coefficients for numerically stability, in other words, the encoding time can be significantly reduced in our schemes.
- In addition, we extend our algorithms to the heterogeneous case where the worker nodes are heterogeneous in nature having different computation and communication speeds and storage capacities. Furthermore, our approaches meet the lower bound of Q/Δ value which indicates that these can efficiently utilize the partial computations done by the slower worker nodes in a heterogeneous system, thus can enhance the overall job execution speed significantly.
- Finally, we conduct exhaustive numerical experiments on Amazon Web Services (AWS) cluster using large-sized sparse matrices, and present comparisons that demonstrate the advantages of our schemes in terms of worker node computation time, communication time, numerical stability and coefficient determination time. The results show that our proposed approach can have up to 30% reduction in computational complexity per worker node and can be 100× faster for determining a "good" set of coefficients for numerical stability.

III. Proposed Approach for Matrix-Vector Multiplication

In this section, we detail our approach for straggler resilient distributed matrix-vector multiplication in case of both homogeneous and heterogeneous worker nodes.

A. Homogeneous System

First we discuss our coded matrix-vector multiplication approach in the homogeneous system with resilience to up to $s = n - k_A$ stragglers. The overall procedure is given in Alg. 1. We partition matrix $\mathbf{A}$ into k_A disjoint block columns as $\mathbf{A}_0, \mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_{k_A-1}$, and assign a random linear combination of ω_A (weight) submatrices of $\mathbf{A}$ to every worker node. Formally, we set $\omega_A = \min(s+1, k_A)$, and assign a linear combination of $\mathbf{A}_i, \mathbf{A}_{i+1}, \mathbf{A}_{i+2}, \ldots, \mathbf{A}_{i+\omega_A-1}$ (indices modulo k_A) to worker node W_i, for $i = 0, 1, 2, \ldots, n-1$, where the linear coefficients are chosen i.i.d. (independent and identically distributed) from a continuous distribution. Note that every worker node W_i has access to the vector $\mathbf{x}$. Once the fastest $\tau = k_A$ worker nodes return their computation results, the central node decodes $\mathbf{A}^T\mathbf{x}$. The following theorem establishes the resiliency of Alg. 1 to straggler nodes.

Theorem 1: Assume that a system has n worker nodes each of which can store $1/k_A$ fraction of matrix $\mathbf{A}$ and the whole vector $\mathbf{x}$ for the distributed matrix-vector multiplication $\mathbf{A}^T\mathbf{x}$. If we assign the jobs according to Alg. 1, we achieve resilience to any $s = n - k_A$ stragglers.

Proof:

Case 1 (First consider the case when $s < k_A$): Since we have partitioned the matrix $\mathbf{A}$ into k_A disjoint block-columns, to recover $\mathbf{A}^T\mathbf{x}$, we need to decode all k_A vector unknowns, $\mathbf{A}_0^T\mathbf{x}, \mathbf{A}_1^T\mathbf{x}, \mathbf{A}_2^T\mathbf{x}, \ldots, \mathbf{A}_{k_A-1}^T\mathbf{x}$. We denote the set of these k_A unknowns as $\mathcal{B}$. Now we choose an arbitrary set of k_A worker nodes. Each of these worker nodes corresponds to an equation in terms of ω_A of those k_A unknowns. We denote the set of k_A equations as $\mathcal{C}$, thus, $|\mathcal{B}| = |\mathcal{C}| = k_A$.

Now we consider a bipartite graph $\mathcal{G} = \mathcal{C} \cup \mathcal{B}$, where any vertex (equation) in $\mathcal{C}$ is connected to some vertices (unknowns) in $\mathcal{B}$ which participate in the corresponding equation. Our goal is to show the existence of a perfect matching among the vertices of $\mathcal{C}$ and $\mathcal{B}$. We argue this according to

Algorithm 1: Proposed Scheme for Distributed Matrix-Vector Multiplication

Input : Matrix $\mathbf{A}$, vector $\mathbf{x}$, n-number of worker nodes, s-number of stragglers, storage fraction $\gamma_A = \frac{1}{k_A}$; $s \leq n - k_A$.

Partition $\mathbf{A}$ into k_A block-columns as $\mathbf{A} = [\mathbf{A}_0\ \mathbf{A}_1 \cdots \mathbf{A}_{k_A-1}]$;

Create a $n \times k_A$ random matrix $\mathbf{R}$ with entries $r_{i,j}$, $0 \leq i \leq n-1$ and $0 \leq j \leq k_A - 1$;

Set weight $\omega_A = min(s+1, k_A)$;

for $i \leftarrow 0$ **to** $n-1$ **do**

  Define $T = \{i, i+1, \ldots, i+\omega_A - 1\}$ (modulo k_A);

  The central node creates a random linear combination of $\mathbf{A}_q$'s where $q \in T$, thus $\tilde{\mathbf{A}}_i = \sum_{q \in T} r_{i,q}\mathbf{A}_q$. Then, it assigns encoded submatrix $\tilde{\mathbf{A}}_i$ to worker node W_i;

  Worker node W_i computes $\tilde{\mathbf{A}}_i^T\mathbf{x}$;

end

Output: The central node recovers $\mathbf{A}^T\mathbf{x}$ from the results of the fastest k_A worker nodes.

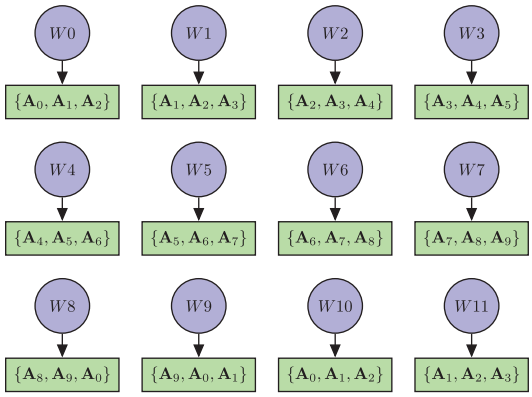

Fig. 1. Submatrix allocation for $n = 12$ workers and $s = 2$ stragglers, with $\gamma_A = \frac{1}{10}$ according to Alg. 1. The weight of every submatrix is $\omega_A = \min(s+1, k_A) = 3$. Any assignment $\{\mathbf{A}_i, \mathbf{A}_j, \mathbf{A}_k\}$ indicates a random linear combination of the corresponding submatrices where the coefficients are chosen i.i.d. at random from a continuous distribution.

Hall's marriage theorem [28] for which we need to show that for any $\bar{\mathcal{C}} \subseteq \mathcal{C}$, the cardinality of the neighbourhood of $\bar{\mathcal{C}}$, denoted as $\mathcal{N}(\bar{\mathcal{C}}) \subseteq \mathcal{B}$, is at least as large as $|\bar{\mathcal{C}}|$. Thus, for $|\bar{\mathcal{C}}| = m \leq k_A$, we need to show that $|\mathcal{N}(\bar{\mathcal{C}})| \geq m$.

Case 1a (First we consider the case that $m \leq 2s$):, thus, we assume that m is equal to either $2p$ or $2p-1$, where $1 \leq p \leq s$. Here we recall that the number of unknowns participating in any equation is $\omega_A = \min(s+1, k_A)$, and the participating unknowns are shifted in a cyclic manner among the equations. If we choose any δ worker nodes out of the first k_A worker nodes $(W_0, W_1, W_2, \ldots, W_{k_A-1})$, according to the proof of cyclic scheme in [5, Appendix C], the minimum number of total participating unknowns is $\min(\omega_A+\delta-1, k_A)$. In other words, the first equation (among those δ equations) consists of ω_A unknowns, and then any additional equation includes at least one additional unknown until the number of total participating unknowns is k_A.

Now, according to Alg. 1, the same unknowns participate in two different equations corresponding to two different worker nodes, W_j and W_{k_A+j}, where $j = 0, 1, \ldots, s-1$. Thus for any $|\bar{\mathcal{C}}| = m = 2p, 2p-1 \leq 2s$, we have $|\mathcal{N}(\bar{\mathcal{C}})| \geq \min(\omega_A + \lceil m/2 \rceil - 1, k_A) = \min(\omega_A + p - 1, k_A)$. Now, since $\omega_A = \min(s+1, k_A)$, we can say that $|\mathcal{N}(\bar{\mathcal{C}})| \geq \min(s+p, k_A) \geq m$.

Case 1b (Now we consider the remaining case where $m = 2s+q$*:)*, $1 \leq q \leq k_A - 2s$. We need to find the minimum number of unknowns which participate in any set of m equations. As we have discussed before, the same unknowns participate in two different equations corresponding to two different worker nodes, W_j and W_{k_A+j}, where $j = 0, 1, \ldots, s-1$. Thus, the additional q equations will correspond to at least q additional unknowns. Therefore, $|\mathcal{N}(\bar{\mathcal{C}})| \geq \min(\omega_A + \lceil 2s/2 \rceil + q - 1, k_A)$. Now, since $\omega_A = \min(s+1, k_A)$, we have, $|\mathcal{N}(\bar{\mathcal{C}})| \geq \min(2s+q, k_A) = m$.

Thus, in this case when $s < k_A$, for any $\mathcal{C}$, we have shown that $|\mathcal{N}(\bar{\mathcal{C}})| \geq |\bar{\mathcal{C}}|$. So, there exists a perfect matching among the vertices of $\mathcal{C}$ and $\mathcal{B}$ according to Hall's marriage theorem. Now we consider the largest matching where the vertex $c_i \in \mathcal{C}$ is matched to the vertex $b_j \in \mathcal{B}$, which indicates that b_j participates in the equation corresponding to c_i. Let us consider a $k_A \times k_A$ system matrix where row i corresponds to the equation associated to c_i where b_j participates. Let us replace row i of the system matrix by $\mathbf{e}_j$ where $\mathbf{e}_j$ is a unit row-vector of length k_A with the j-th entry being 1, and 0 otherwise. Thus we have a $k_A \times k_A$ matrix where each row has only one non-zero entry which is 1. Since we have a perfect matching, this $k_A \times k_A$ matrix will have only one non-zero entry in every column. This is a permutation of the identity matrix, and, thus, is full rank. Since the matrix is full rank for a choice of definite values, according to Schwartz-Zippel lemma [29], the matrix continues to be full rank for random choices of non-zero entries. Thus, the central node can recover all k_A unknowns from any set of k_A worker nodes.

Case 2 (Next consider the case when $s \geq k_A$*):* In this case, $\omega_A = \min(s+1, k_A) = k_A$, thus all the worker nodes are assigned linear combinations of k_A submatrices. Since the entries are chosen randomly from a continuous distribution, we can say that any $k_A \times k_A$ submatrix of the $n \times k_A$ system matrix is full rank. Thus we can recover all k_A unknowns from the returned results of any k_A out of n worker nodes. ■

Remark 1: While our proposed method is applicable for any values of s and k_A, it is particularly impactful if $s < k_A - 1$ (thus, $\omega_A < k_A$), i.e., the percentage of the number of stragglers is less than 50% of the total number of nodes (in the common practical cases, this percentage can be even less than 10% [2], [30]). In these cases, the encoded submatrices have less weights than the dense coded approaches [4], [11], [12] which could reduce the expected communication delay and the average worker node computation delay for sparse "input" matrices.

Example 1: Consider a system with $n = 12$ worker nodes each of which can store $1/10$ fraction of matrix $\mathbf{A}$. We partition matrix $\mathbf{A}$ into $k_A = 10$ disjoint block-columns, $\mathbf{A}_0, \mathbf{A}_1, \ldots, \mathbf{A}_9$. According to Alg. 1, we set the weight $\omega_A = \min(s+1, k_A) = \min(n - k_A + 1, k_A) = 3$, and assign random linear combinations of submatrices $\mathbf{A}_i, \mathbf{A}_{i+1}, \mathbf{A}_{i+2}$ (indices modulo 10) to worker node W_i, for $i = 0, 1, \ldots, 11$, as shown in Fig. 1. Thus, according to

Theorem 1, the system has a recovery threshold $\tau = k_A = 10$, and it is resilient to any $s = 2$ stragglers.

B. Extension to Heterogeneous System

In this section, we extend our approach in Alg. 1 to a heterogeneous system of $\bar{n}$ worker nodes where the nodes may have different computation speeds and communication speeds. We assume that true knowledge about the storage and speeds of the worker nodes are available prior to the assignment of the jobs. We also assume that we have λ different types of nodes in the system, with worker node type $0, 1, \ldots, \lambda - 1$. First, without loss of generality (w.l.o.g.), we sort the worker nodes in a non-ascending order in terms of the worker node types. Next, suppose that α is the number of the assigned columns and β is the number of processed columns per unit time in the "weakest" type node. In this scenario, we assume that a worker node W_i of type j_i receives $c_{j_i}\alpha$ coded columns of data matrix $\mathbf{A}$ and has a computation speed $c_{j_i}\beta$, where $c_{j_i} \geq 1$ is an integer. Thus, a higher c_{j_i} indicates a "stronger" type node W_i which has a c_{j_i} times higher memory and can process at a c_{j_i} times higher computation speed than the "weakest" type node. Since we sort the nodes in a non-ascending order in terms of the worker node types, we have $j_0 \geq j_1 \geq j_2 \geq \cdots \geq j_{\bar{n}-1} = 0$, hence $c_{j_0} \geq c_{j_1} \geq c_{j_2} \geq \cdots \geq c_{j_{\bar{n}-1}} = 1$. Note that $\lambda = 1$ and all $c_{j_i} = 1$ lead us to the homogeneous system in Section III-A where $0 \leq i \leq n-1$ and $j_i = 0$.

Assume that the "weakest" type worker node requires μ units of time to process α columns of $\mathbf{A}$. Thus, any node W_i of type j_i can process $c_{j_i}\alpha$ columns in time μ. In this scenario, from the computation and storage perspective, worker node W_i (of type j_i) can be considered as a combination of $c_{j_i} \geq 1$ worker nodes of the "weakest" type. Thus, $\bar{n}$ worker nodes in the heterogeneous system can be thought as homogeneous system of $n = \sum_{i=0}^{\bar{n}-1} c_{j_i}$ worker nodes of the "weakest" type. In other words, the worker node W_k in the heterogeneous system ($0 \leq k \leq \bar{n}-1$) can be thought as a combination of worker nodes $\bar{W}_m, \bar{W}_{m+1}, \ldots, \bar{W}_{m+c_{kj}-1}$ in a homogeneous setting, where $m = \sum_{i=0}^{k-1} c_{j_i}$ and W_k is of type j_i worker node. Now, for any worker node index $\bar{k}_A$ (such that $0 \leq \bar{k}_A \leq \bar{n}-1$), we define $k_A = \sum_{i=0}^{\bar{k}_A-1} c_{j_i}$ and $s = \sum_{i=\bar{k}_A}^{\bar{n}-1} c_{j_i}$, so, $n = \sum_{i=0}^{\bar{n}-1} c_{j_i} = k_A + s$. Thus, a heterogeneous system of $\bar{n}$ worker nodes can be thought as a homogeneous system of $n = k_A + s$ nodes, for any $\bar{k}_A$ ($0 \leq \bar{k}_A \leq \bar{n}-1$). We state the following corollary (proof is in [31]) of Theorem 1 for heterogeneous system.

Corollary 1: Consider a heterogeneous system of $\bar{n}$ nodes of different types and assume any $\bar{k}_A$ (where $0 \leq \bar{k}_A \leq \bar{n}-1$). Now, if the jobs are assigned to the modified homogeneous system of $n = k_A + s$ "weakest" type worker nodes according to Alg. 1, the system (a) will be resilient to s such nodes and (b) will provide a Q/Δ value as 1.

As we have discussed in Section III-A, under the assumption that each worker node has been assigned $1/k_A$ fraction of the whole job, our proposed approach for a homogeneous system can be resilient to any s stragglers out of $n = k_A + s$ worker nodes. Now, the heterogeneous system of $\bar{n}$ worker nodes is resilient to $s = \sum_{i=\bar{k}_A}^{n-1} c_{j_i}$ block-column processing, where each node is assigned $1/k_A$ fraction of the whole job. Varying the indexing of the nodes depending on different node types or changing the value of $\bar{k}_A$, one can decrease (or increase) the per worker load of the job, which can increase (or decrease) the value of $\bar{s}$. The number of actual stragglers that the system is resilient to can vary depending on the node types.

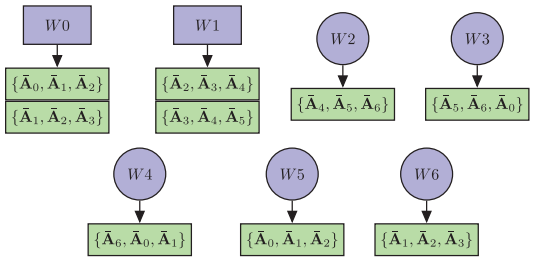

Fig. 2. A heterogeneous system where $\bar{n} = 7$ and $\bar{k}_A = 5$, thus $n = 9$ and $k_A = 7$. Each of W_0 and W_1 is assigned twice the load of each of $W_2, W_3, \ldots, W_6$. This system is resilient to any $s = 2$ block-column processing, i.e., it is resilient to any two type 0 nodes (e.g., W_3 and W_6) or any one type 1 node (e.g., W_1).

Example 2: Consider the example in Fig. 2 consisting of $\bar{n} = 7$ worker nodes. Let us assume, $c_{j_i} = 2$ when $i = 0, 1$ and $c_{j_i} = 1$ when $2 \leq i \leq 6$, thus, $n = \sum_{i=0}^{n-1} c_{j_i} = 9$. Now assume that $\bar{k}_A = 5$, thus $k_A = \sum_{i=0}^{\bar{k}_A-1} c_{j_i} = 7$, and $s = \sum_{i=\bar{k}_A}^{\bar{n}-1} c_{j_i} = 2$. So, each weakest device is assigned $1/7$-th fraction of the whole job. This scheme is resilient to any $s = 2$ block-column processing, in other words, it is resilient to any *two* type 0 nodes or any *one* type 1 node.

Moreover, for the heterogeneous setting, our proposed approach provides a Q/Δ value to be 1, which indicates that the central node can recover all $\Delta = 7$ unknowns from *any* $Q = 7$ block-products. In this example, W_0 and W_1 are assigned multiple jobs and our proposed approach can efficiently utilize their partial computations if any of them is slower than their rated speed. For instance, assume that W_6 is a failure, and W_0 is slower and able to compute one (out of two) of the submatrix products while each of $W_1, W_2, \ldots, W_5$ completes its respective assigned job. Then, the central node can recover all the submatrix products from the successful worker nodes with the help of the partial computations done by W_0.

Remark 2: It is well-known in the cloud computation that the low-cost machines (which are the "weaker" ones) are the most probable ones to straggle [7]. In that case, the number of stragglers that our proposed heterogeneous matrix computation scheme can be resilient to will be higher.

IV. Proposed Approach for Matrix-Matrix Multiplication

A. Homogeneous System

First, we discuss our distributed matrix-matrix multiplication approach for the homogeneous system with resilience to $s = n - k_A k_B$ stragglers where $s \leq \max(k_A, k_B)$. Without loss of generality (w.l.o.g.), we can assume that $k_A \geq k_B$, thus, $s \leq k_A$. Each node stores the equivalent of $1/k_A$ and $1/k_B$ fractions (of

Algorithm 2: 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 $\gamma_A = \frac{1}{k_B}$; $s \leq n - k_A k_B \leq \max(k_A, k_B)$.

Partition $\mathbf{A}$ and $\mathbf{B}$ into k_A and k_B block-columns, respectively;

Create a $n \times k_A$ random matrix $\mathbf{R}_A$ with entries $r^A_{i,j}$, $0 \leq i \leq n-1$ and $0 \leq j \leq k_A - 1$;

Create a $n \times k_B$ random matrix $\mathbf{R}_B$ with entries $r^B_{i,j}$, $0 \leq i \leq n-1$ and $0 \leq j \leq k_B - 1$;

Set weights ω_A and ω_B (where $\omega_A \geq \omega_B$) in such a way that $\omega_A \omega_B > s$, $1 < \omega_A < k_A$ and $1 < \omega_B < k_B$;

for $i \leftarrow 0$ **to** $n-1$ **do**

  Define $T = \{i, i+1, \ldots, i+\omega_A - 1\}$ (modulo k_A);

  Create a random linear combination of $\mathbf{A}_q$'s where $q \in T$, thus $\tilde{\mathbf{A}}_i = \sum_{q \in T} r^A_{i,q} \mathbf{A}_q$;

  Set $j = \lfloor i/k_A \rfloor$, and define $S = \{j, j+1, \ldots, j+\omega_B - 1\}$ (modulo k_B);

  Create a random linear combination of $\mathbf{B}_q$'s where $q \in S$, thus $\tilde{\mathbf{B}}_i = \sum_{q \in S} r^B_{i,q} \mathbf{B}_q$;

  The central node assigns encoded submatrices $\tilde{\mathbf{A}}_i$ and $\tilde{\mathbf{B}}_i$ to worker node W_i;

  Worker node W_i computes $\tilde{\mathbf{A}}_i^T \tilde{\mathbf{B}}_i$;

end

Output: The central node recovers $\mathbf{A}^T\mathbf{B}$ from the fastest $k_A k_B$ worker nodes.

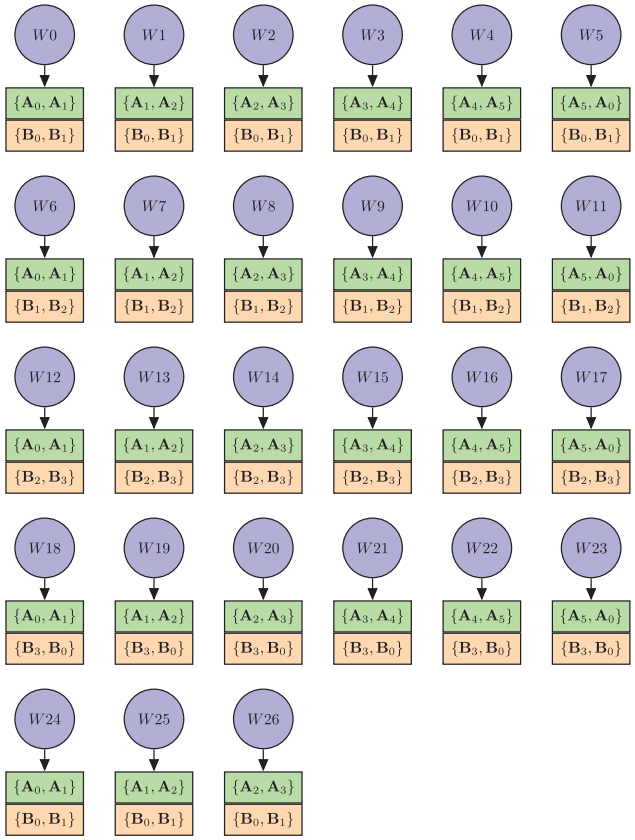

Fig. 3. Submatrix allocation according to Alg. 2 when $n = 27$ and $s = 3$, with $\gamma_A = \frac{1}{6}$ and $\gamma_B = \frac{1}{4}$. The weights of the submatrices are $\omega_A = \omega_B = 2$. Any assignment $\{\mathbf{A}_i, \mathbf{A}_j\}$ or $\{\mathbf{B}_i, \mathbf{B}_j\}$ indicates a random linear combination of the corresponding submatrices where the coefficients are chosen i.i.d. at random from a continuous distribution.

block-columns) of matrices $\mathbf{A}$ and $\mathbf{B}$, respectively. Thus, if a node multiplies its respective assignments from $\mathbf{A}$ and $\mathbf{B}$, it completes $1/k_{AB}$ fraction of overall job of computing $\mathbf{A}^T\mathbf{B}$, where $k_{AB} = k_A k_B$. The overall procedure is given in Alg. 2.

In our approach, we partition matrices $\mathbf{A}$ and $\mathbf{B}$ into k_A and k_B disjoint block columns, respectively, as $\mathbf{A}_0, \mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_{k_A-1}$ and $\mathbf{B}_0, \mathbf{B}_1, \mathbf{B}_2, \ldots, \mathbf{B}_{k_B-1}$, respectively. Next, we set ω_A and ω_B in such a way so that $\omega_A \omega_B > s$; and assign a random linear combination of ω_A (weight) submatrices of $\mathbf{A}$ and another random linear combination of ω_B (weight) submatrices of $\mathbf{B}$ to every worker node where $1 < \omega_A < k_A$ and $1 < \omega_B < k_B$. It should be noted that for a given storage fraction $\gamma_A = 1/k_A$ (or $\gamma_B = 1/k_B$) of matrix $\mathbf{A}$ (or $\mathbf{B}$) for each of the worker nodes, the case $\omega_A = 1$ (or $\omega_B = 1$) leads to an approach which provides suboptimal performance in terms of number of stragglers that the system is resilient to [5].

Formally, we assign a random linear combination of $\mathbf{A}_i, \mathbf{A}_{i+1}, \ldots, \mathbf{A}_{i+\omega_A-1}$ (indices of $\mathbf{A}$ are reduced modulo k_A) to worker node W_i, $0 \leq i \leq n-1$. Thus, we can say that the participating submatrices of $\mathbf{A}$ are shifted in a cyclic manner over all n worker nodes. Next we set $j = \lfloor i/k_A \rfloor$, and assign $\mathbf{B}_j, \mathbf{B}_{j+1}, \ldots, \mathbf{B}_{j+\omega_B-1}$ (indices of $\mathbf{B}$ are reduced modulo k_B) to worker node W_i. Once the fastest $\tau = k_A k_B$ worker nodes finish and return their computation results, the central node can recover all the unknowns in the form of $\mathbf{A}_u^T\mathbf{B}_v$, where $0 \leq u \leq k_A - 1$ and $0 \leq v \leq k_B - 1$.

Example 3: Consider the example in Fig. 3 where $n = 27$, $\gamma_A = 1/6$ and $\gamma_B = 1/4$. So, we partition $\mathbf{A}$ and $\mathbf{B}$ into $k_A = 6$ and $k_B = 4$ block-columns, respectively. In each node, we assign one coded submatrix from $\mathbf{A}$ and one from $\mathbf{B}$ which are linear combinations of $\omega_A = \omega_B = 2$ uncoded submatrices with coefficients chosen i.i.d. at random from a continuous distribution. It can be verified that this scheme is resilient to $s = n - k_A k_B = 3$ stragglers. In what follows, we will use this example several times to describe different structures and properties of our scheme.

1) Structure of the Job Assignment: To describe the structure of the proposed scheme, first we partition the worker nodes into k_A disjoint classes, denoted by $\mathcal{M}_i$'s, where any $\mathcal{M}_i$ consists of all the worker nodes W_j's if $j \equiv i (\text{mod } k_A)$. In other words, $\mathcal{M}_i = \{W_i, W_{k_A+i}, W_{2k_A+i}, \ldots\}$, for $i = 0, 1, 2, \ldots, k_A - 1$. Since $n = k_A k_B + s \leq k_A(k_B + 1)$ (as $s \leq k_A$), we can say $|\mathcal{M}_i|$ is either k_B or $k_B + 1$. Moreover, according to our proposed scheme, the participating submatrices of $\mathbf{A}$ are the same over all the worker nodes in any $\mathcal{M}_i$. For instance, in Example 3, we have $\mathcal{M}_0 = \{W_0, W_6, W_{12}, W_{18}, W_{24}\}$, and random linear combinations of $\mathbf{A}_0$ and $\mathbf{A}_1$ are assigned to all the corresponding worker nodes. At this point, we define a set, $\mathcal{D}_i^A = \{\mathbf{A}_i, \mathbf{A}_{i+1}, \ldots, \mathbf{A}_{i+\omega_A-1}\}$, which consists of the participating submatrices of $\mathbf{A}$ corresponding to worker node set $\mathcal{M}_i$, where the indices are reduced modulo k_A. Now we state the

following claim which gives a lower bound for the cardinality of the union of any arbitrary number of $\mathcal{D}_i^A$'s.

Claim 1: Consider any q sets $\mathcal{D}_i^A$'s, $q \leq k_A - \omega_A + 1$, denoted w.l.o.g., $\bar{\mathcal{D}}_j^A$, $0 \leq j \leq q-1$ arbitrarily. Then $\left|\bigcup_{j=0}^{q-1} \bar{\mathcal{D}}_j^A\right| \geq \omega_A + q - 1$.

Proof: From the definition of $\mathcal{D}_i^A$ above, we can say that the participating uncoded submatrices of $\mathbf{A}$ are shifted in a cyclic fashion within $\mathcal{D}_0^A, \mathcal{D}_1^A, \ldots, \mathcal{D}_{k_A-1}^A$. Thus, according to the proof of cyclic scheme in [5, Appendix C], the minimum number of total constituent submatrices of $\mathbf{A}$ within any L of $\mathcal{D}_i^A$'s is $\min(\omega_A + L - 1, k_A)$. Now consider any q arbitrary $\bar{\mathcal{D}}_j^A$'s, $0 \leq j \leq q-1$. Thus, $\left|\bigcup_{j=0}^{q-1} \bar{\mathcal{D}}_j^A\right| \geq \min(\omega_A + q - 1, k_A) = \omega_A + q - 1$, since $q \leq k_A - \omega_A + 1$. ∎

Example 4: Consider Example 3 in Fig. 3 where $k_A = 6$ and $\omega_A = 2$. Now, choose any arbitrary $q \leq 5$ sets of $\mathcal{D}_i^A$'s. For example, we choose $q = 3$ of $\mathcal{D}_i^A$'s, such as $\mathcal{D}_0^A$, $\mathcal{D}_1^A$ and $\mathcal{D}_3^A$ (w.l.o.g. we can denote them as $\bar{\mathcal{D}}_0^A, \bar{\mathcal{D}}_1^A$ and $\bar{\mathcal{D}}_2^A$). Then, according to Claim 1 we have, $\left|\mathcal{D}_0^A \cup \mathcal{D}_1^A \cup \mathcal{D}_3^A\right| = \left|\bar{\mathcal{D}}_0^A \cup \bar{\mathcal{D}}_1^A \cup \bar{\mathcal{D}}_2^A\right| \geq \omega_A + 2 = 4$. Thus, the total number of constituent submatrices of $\mathbf{A}$ within $\mathcal{M}_0$, $\mathcal{M}_1$ and $\mathcal{M}_3$ is lower bounded by 4. It can be verified from Fig. 3 that the exact number is 5, where $\mathcal{D}_0^A = \{\mathbf{A}_0, \mathbf{A}_1\}$, $\mathcal{D}_1^A = \{\mathbf{A}_1, \mathbf{A}_2\}$ and $\mathcal{D}_3^A = \{\mathbf{A}_3, \mathbf{A}_4\}$.

Now, in our approach, according to Alg. 2, the participating submatrices of $\mathbf{B}$ are shifted in a cyclic fashion over the worker nodes of any $\mathcal{M}_i$. For instance, in Example 3, the participating submatrices, $\mathbf{B}_0, \mathbf{B}_1, \mathbf{B}_2$ and $\mathbf{B}_3$, are shifted in a cyclic fashion within the worker nodes of $\mathcal{M}_0$, i.e., W_0, W_6, W_{12}, W_{18} and W_{24}. Next, in the following claim, we find the minimum number of participating unknowns (in the form of $\mathbf{A}_u^T\mathbf{B}_v$) within any δ worker nodes from any $\mathcal{M}_q$. Now we state the following claim with the proof in [31].

Claim 2: Consider $\mathcal{M}_q$, $0 \leq q \leq k_A - 1$. Denote the minimum of total number of participating unknowns (in the form of $\mathbf{A}_i^T\mathbf{B}_j$) within any δ worker nodes from $\mathcal{M}_q$ by ρ. Then $\rho = \omega_A \times \omega_B$ if $\delta = 1$. Otherwise, if $2 \leq \delta \leq k_B + 1$, then

$$\rho = \begin{cases} \omega_A \times \min(\omega_B + \delta - 1, k_B) & \text{if } \left|\mathcal{M}_q\right| = k_B; \\ \omega_A \times \min(\omega_B + \delta - 2, k_B) & \text{if } \left|\mathcal{M}_q\right| = k_B + 1. \end{cases}$$

2) Rearrangement of $\mathcal{M}_i$'s: Before stating the necessary theorem and corresponding claims, we discuss a preprocessing step that rearranges the $\mathcal{M}_i$'s. Choose any arbitrary m worker nodes ($m \leq k_A k_B$), and assume that δ_i worker nodes have been chosen from $\mathcal{M}_i$, for $0 \leq i \leq k_A - 1$, so that $\sum_{i=0}^{k_A-1} \delta_i = m$. Now, we rearrange the $\mathcal{M}_i$'s in the following process.

(i) We rearrange the δ_i's in a decreasing sequence so that $\tilde{\delta}_0 \geq \tilde{\delta}_1 \geq \tilde{\delta}_2 \geq \cdots \geq \tilde{\delta}_{k_A-1}$ and rename the corresponding $\mathcal{M}_i$'s as $\tilde{\mathcal{M}}_i$'s so that $\tilde{\delta}_i$ nodes have been chosen from $\tilde{\mathcal{M}}_i$.

(ii) If multiple δ_i's are equal, we place the $\mathcal{M}_i$'s first which have smaller cardinality. In other words, if $\delta_i = \delta_j$ for any $i, j \leq k_A - 1$, where $|\mathcal{M}_i| = k_B$ and $\left|\mathcal{M}_j\right| = k_B + 1$, then we place $\mathcal{M}_i$ first (i.e., rename as $\tilde{\mathcal{M}}_k$ for some k) and then we place $\mathcal{M}_j$ (rename as $\tilde{\mathcal{M}}_{k+1}$).

Now, we denote ρ_0 as the minimum of total number of participating unknowns (in the form of $\mathbf{A}_i^T\mathbf{B}_j$) within the $\tilde{\delta}_0$ worker nodes of $\tilde{\mathcal{M}}_0$. Thus, according to Claim 2, if $\tilde{\delta}_0 = 1$, $\rho_0 = \omega_A \times \omega_B$. Otherwise, if $2 \leq \tilde{\delta}_0 \leq k_B + 1$, then

$$\rho_0 = \begin{cases} \omega_A \times \min\left(\omega_B + \tilde{\delta}_0 - 1, k_B\right) & \text{if } \left|\mathcal{M}_q\right| = k_B; \\ \omega_A \times \min\left(\omega_B + \tilde{\delta}_0 - 2, k_B\right) & \text{if } \left|\mathcal{M}_q\right| = k_B + 1. \end{cases} \tag{1}$$

After that, we move to $\tilde{\mathcal{M}}_1, \tilde{\mathcal{M}}_2, \ldots, \tilde{\mathcal{M}}_{k_A-\omega_A}$, sequentially, to find the number of additional participating unknowns within the corresponding $\tilde{\delta}_i$ worker nodes of $\tilde{\mathcal{M}}_i$, where $1 \leq i \leq k_A - 1$. We denote ρ_i as the minimum number of such additional participating unknowns in $\tilde{\mathcal{M}}_i$.

Here, according to Claim 1, $\left|\bigcup_{j=0}^{0} \bar{\mathcal{D}}_j^A\right| \geq \omega_A$ and $\left|\bigcup_{j=0}^{1} \bar{\mathcal{D}}_j^A\right| \geq \omega_A + 1$. Thus, there will be at least one additional participating submatrix of $\mathbf{A}$ in $\tilde{\mathcal{M}}_0 \cup \tilde{\mathcal{M}}_1$ in comparison to $\tilde{\mathcal{M}}_0$, and the property will continue to hold until we consider the set $\tilde{\mathcal{M}}_0 \cup \tilde{\mathcal{M}}_1 \cup \cdots \cup \tilde{\mathcal{M}}_0 \cup \tilde{\mathcal{M}}_{k_A-\omega_A}$. Now, since the submatrices of $\mathbf{B}$ (which will be multiplied by the additional submatrix of $\mathbf{A}$) are shifted in a cyclic fashion within any $\tilde{\mathcal{M}}_i$, if $\tilde{\delta}_i = 1$, then $\rho_i = \omega_B$; otherwise, if $2 \leq \tilde{\delta}_i \leq k_B + 1$, then

$$\rho_i = \begin{cases} \min\left(\omega_B + \tilde{\delta}_i - 1, k_B\right) & \text{if } \left|\tilde{\mathcal{M}}_i\right| = k_B; \\ \min\left(\omega_B + \tilde{\delta}_i - 2, k_B\right) & \text{if } \left|\tilde{\mathcal{M}}_i\right| = k_B + 1; \end{cases} \tag{2}$$

for $1 \leq i \leq k_A - \omega_A$. Note that, ρ_i has a trivial lower bound, *zero*, when $k_A - \omega_A + 1 \leq i \leq k_A - 1$.

Now we state the following corollary which is a special case of Lemma 1 stated later in this section. The lemma (hence, the corollary) provides a lower bound on the minimum number of participating unknowns (in the form of $\mathbf{A}_u^T\mathbf{B}_v$) in the equations from any arbitrary m nodes. Here, we assume that $k_A \geq k_B$. Note that, if $k_B > k_A$, we can compute $\mathbf{A}^T\mathbf{B}$ as $(\mathbf{B}^T\mathbf{A})^T$ without any additional computational cost. Thus, we can assume $k_A \geq k_B$ without loss of generality.

Corollary 2 (Corollary of Upcoming Lemma 1): For any arbitrary $k_A \geq 3$ and $k_B \geq 3$ (where $k_A \geq k_B$ without loss of generality), let us assign the jobs to $n = k_A k_B + s$ (where $s \leq 3$) worker nodes according to Alg. 2 using $\omega_A = \omega_B = 2$. Then the total number of participating unknowns (in the form of $\mathbf{A}_i^T\mathbf{B}_j$) within any m nodes ($m \leq k_A k_B$) is at least m.

Proof: First, we choose any arbitrary $m \leq k_A k_B$ worker nodes out of all n nodes. As discussed in the preprocessing step above, we choose $\tilde{\delta}_i$ worker nodes from the set $\tilde{\mathcal{M}}_i$, thus, $\sum_{i=0}^{k_A-1} \tilde{\delta}_i = m$. Moreover, the minimum number of participating unknowns in $\tilde{\mathcal{M}}_0$ is denoted by ρ_0 as given by (1). The minimum number of additional participating unknowns from $\tilde{\mathcal{M}}_i$ is denoted by ρ_i and given by (2) when $1 \leq i \leq k_A - \omega_A$ or trivially lower bounded by *zero* for other values of i when $k_A - \omega_A + 1 \leq i \leq k_A - 1$. Thus, the minimum number of participating unknowns in these m worker nodes is at least $\sum_{i=0}^{k_A-1} \rho_i = \sum_{i=0}^{k_A-\omega_A} \rho_i + \sum_{i=k_A-\omega_A+1}^{k_A-1} \rho_i \geq \sum_{i=0}^{k_A-\omega_A} \rho_i$. In this corollary, $\omega_A = 2$. Thus, to prove the corollary, it is sufficient to prove that

$$\sum_{i=0}^{k_A-2} \rho_i \geq \sum_{i=0}^{k_A-1} \tilde{\delta}_i. \tag{3}$$

Now we carry out the following exhaustive case analysis and we show that for every case, (3) is true. Here, according

to the pre-processing step (i) for the rearrangement of $\mathcal{M}_i$'s in Section IV-A2, we have $\tilde{\delta}_0 \geq \tilde{\delta}_1 \geq \tilde{\delta}_2 \geq \cdots \geq \tilde{\delta}_{k_A-1}$. Since $n = k_Ak_B + s$ and $s \leq 3 \leq k_A$, any $\tilde{\delta}_i$ can have a value at most $k_B + 1$. The cases are based on different values of $\tilde{\delta}_0$: in case 1, we assume $1 \leq \tilde{\delta}_0 \leq k_B$ and in case 2, we assume that $\tilde{\delta}_0 = k_B + 1$. Moreover, since $s \leq 3$ and $m \leq k_Ak_B$, there can be at most three $\tilde{\delta}_i$'s which can have value $k_B + 1$. We discuss those in Cases 2a, 2b and 2c, respectively.

Case 1 ($1 \leq \tilde{\delta}_0 \leq k_B$): If, $\tilde{\delta}_0 = 1$ or $1 < \tilde{\delta}_0 \leq k_B$, since $\omega_A = \omega_B = 2$, according to (1), we have, $\rho_0 \geq 2\tilde{\delta}_0 \geq \tilde{\delta}_0 + \tilde{\delta}_{k_A-1}$, as $\tilde{\delta}_{k_A-1} \leq \tilde{\delta}_0$. Now, according to (2), for $1 \leq i \leq k_A-2$, we have $\rho_i \geq \tilde{\delta}_i$. Thus, $\sum_{i=0}^{k_A-2} \rho_i = \rho_0 + \sum_{i=1}^{k_A-2} \rho_i \geq \tilde{\delta}_0 + \tilde{\delta}_{k_A-1} + \sum_{i=1}^{k_A-2} \tilde{\delta}_i = \sum_{i=0}^{k_A-1} \tilde{\delta}_i$; hence, (3) is true.

Case 2 ($\tilde{\delta}_0 = k_B + 1$): In this case, since $\omega_A = \omega_B = 2$, we have $\rho_0 = 2k_B$ according to (1). Now, for simplicity we consider the following three sub-cases.

Case 2a ($\tilde{\delta}_0 = k_B + 1$ and $\tilde{\delta}_1 \leq k_B$): Since $\sum_{i=0}^{k_A-1} \tilde{\delta}_i = m \leq k_Ak_B$ and $\tilde{\delta}_0 = k_B + 1$, $\tilde{\delta}_{k_A-1}$ can be at most $k_B - 1$. Thus $\rho_0 = 2k_B \geq \tilde{\delta}_0 + \tilde{\delta}_{k_A-1}$. Thus similar to Case 1, we can say that (3) is true.

Case 2b ($\tilde{\delta}_0 = \tilde{\delta}_1 = k_B + 1$ and $\tilde{\delta}_2 \leq k_B$): In this case, $\rho_0 + \rho_1 = 3k_B$. Now, $m = \sum_{i=0}^{k_A-1} \tilde{\delta}_i \leq k_Ak_B$ which indicates that $\sum_{i=2}^{k_A-1} \tilde{\delta}_i \leq (k_A - 2)k_B - 2$ and we know that $\tilde{\delta}_i$'s are arranged in a non-increasing order. Thus, it can happen that $\tilde{\delta}_{k_A-1} \leq k_B - 2$ or $\tilde{\delta}_{k_A-1} = \tilde{\delta}_{k_A-2} = k_B - 1$. If $\tilde{\delta}_{k_A-1} \leq k_B - 2$, we have $\tilde{\delta}_0 + \tilde{\delta}_1 + \tilde{\delta}_{k_A-1} = k_B + 1 + k_B + 1 + k_B - 2 = \rho_0 + \rho_1$. Hence, (3) is true since $\sum_{i=0}^{k_A-2} \rho_i \geq \sum_{i=0}^{k_A-2} \tilde{\delta}_i$, as we know $\rho_i \geq \tilde{\delta}_i$ for $i = 2, 3, \ldots, k_A - 2$ from (2).

The remaining case is, $\tilde{\delta}_{k_A-1} = \tilde{\delta}_{k_A-2} = k_B - 1$. Now, since the corollary aims at resilience to at most *three* stragglers, there can be at most *three* $\tilde{\mathcal{M}}_i$'s which have cardinality $k_B + 1$. But, $\tilde{\delta}_0 = \tilde{\delta}_1 = k_B + 1$, and thus, there can be at most *one* more $\tilde{\mathcal{M}}_i$ left with cardinality $k_B + 1$. So, either both of $\tilde{\mathcal{M}}_{k_A-2}$ and $\tilde{\mathcal{M}}_{k_A-1}$ will have cardinality k_B or one of them will have cardinality $k_B + 1$. However, according to our rearrangement procedure (ii) in Section IV-A2, if two $\tilde{\delta}_i$'s are equal, we place the $\tilde{\mathcal{M}}_i$ which has smaller cardinality, first. Thus, in both cases, $\tilde{\mathcal{M}}_{k_A-2}$ must have cardinality k_B, hence $\rho_{k_A-2} = k_B$ according to (2). Thus,

$$\rho_0 + \rho_1 + \rho_{k_A-2} = 4k_B = \tilde{\delta}_0 + \tilde{\delta}_1 + \tilde{\delta}_{k_A-2} + \tilde{\delta}_{k_A-1},$$

hence we are done similar to Case 1, since for $2 \leq i \leq k_A - 3$, we have $\rho_i \geq \tilde{\delta}_i$.

Case 2c ($\tilde{\delta}_0 = \tilde{\delta}_1 = \tilde{\delta}_2 = k_B + 1$): In this case, $\rho_0 + \rho_1 + \rho_2 = 4k_B$. Note that $m \leq k_Ak_B$, thus $\tilde{\delta}_{k_A-1}$ can be at most $k_B - 1$. Consider the scenario, when $\tilde{\delta}_{k_A-1} \leq k_B - 3$. In this scenario,

$$\sum_{i=0}^{2} \tilde{\delta}_i + \tilde{\delta}_{k_A-1} \leq 3(k_B + 1) + k_B - 3 = 4k_B = \sum_{i=0}^{2} \rho_i,$$

hence we are done similar to Case 1, since for $3 \leq i \leq k_A - 2$, we have $\rho_i \geq \tilde{\delta}_i$. If $\tilde{\delta}_{k_A-1} = k_B - 2$, then $k_B - 2 \leq \tilde{\delta}_{k_A-2} \leq k_B - 1$, in that case $\rho_{k_A-2} \geq 1 + \tilde{\delta}_{k_A-2}$ according to (2) since $|\tilde{\mathcal{M}}_{k_A-2}| = k_B$. Thus, $\tilde{\delta}_0 + \tilde{\delta}_1 + \tilde{\delta}_2 + \tilde{\delta}_{k_A-2} + \tilde{\delta}_{k_A-1} \leq 3(k_B + 1) + \rho_{k_A-2} - 1 + k_B - 2 = 4k_B + \rho_{k_A-2}$. So, $\tilde{\delta}_0 + \tilde{\delta}_1 + \tilde{\delta}_2 + \tilde{\delta}_{k_A-2} + \tilde{\delta}_{k_A-1} \leq \rho_0 + \rho_1 + \rho_2 + \rho_{k_A-2}$. Hence, we are done. Finally, if $\tilde{\delta}_{k_A-1} = k_B - 1$, since $\sum_{i=0}^{k_A-1} \tilde{\delta}_i \leq k_Ak_B$ and $\tilde{\delta}_i$'s are arranged in a non-increasing order, we must have $\tilde{\delta}_{k_A-2} = \tilde{\delta}_{k_A-3} = k_B - 1$. Thus, $\rho_0 + \rho_1 + \rho_2 + \rho_{k_A-3} + \rho_{k_A-2} = 4k_B + 2k_B = 6k_B$, and

$$\tilde{\delta}_0 + \tilde{\delta}_1 + \tilde{\delta}_2 + \tilde{\delta}_{k_A-3} + \tilde{\delta}_{k_A-2} + \tilde{\delta}_{k_A-1} = 3(k_B + 1) + 3(k_B - 1) = 6k_B,$$

hence we are done, since $\rho_i \geq \tilde{\delta}_i$, for $3 \leq i \leq k_A - 4$. ■

Lemma 1: For any arbitrary $k_A \geq 3$ and $k_B \geq 3$ (where $k_A \geq k_B$ without loss of generality), if we assign the jobs to $n = k_Ak_B + s$ worker nodes (where $s \leq k_A$) according to Alg. 2, then the minimum of total number of participating unknowns within any m worker nodes ($m \leq k_Ak_B$) will be lower bounded by m.

Proof: The proof of this lemma appears in Appendix B. ■

Theorem 2: Assume that a system has n worker nodes each of which can store the equivalent of $1/k_A$ fraction of matrix $\mathbf{A}$ and $1/k_B$ fraction of matrix $\mathbf{B}$ (without loss of generality $k_A \geq k_B$) for distributed matrix-matrix multiplication $\mathbf{A}^T\mathbf{B}$. If we assign the jobs according to Alg. 2, we achieve resilience to *any* $s = n - k_Ak_B$ stragglers where $s \leq k_A$.

Proof: This theorem is proved in a similar manner as Theorem 1 with the help of Lemma 1 and Schwartz-Zippel lemma [29]. The proof is detailed in [31]. ■

B. Extension to Heterogeneous System

Similar to the matrix-vector case in Section III-B, we extend our approach in Alg. 2 to heterogeneous system where the worker nodes may have different computation and communication speeds. We have all the same assumptions as we had in the matrix-vector case in Section III-B. We have λ different types of devices in the system, with worker node type $0, 1, \ldots, \lambda-1$. Any worker node W_i (for $0 \leq i \leq n-1$) receives $c_{j_i}\alpha_A$ columns of matrix $\mathbf{A}$ and $c_{j_i}\alpha_B$ columns of matrix $\mathbf{B}$ where any worker node of the weakest type receives α_A and α_B columns, respectively, and $c_{j_i} \geq 1$ is a positive integer. Moreover any worker node W_i of node type j_i has a computation speed $c_{j_i}\beta$, where β is the computation speed for the worker node of the weakest type.

As we have discussed for the matrix-vector case in Section III-B, from the computation and storage perspective, W_i can be considered as a collection of $c_{j_i} \geq 1$ worker nodes of the "weakest" type. Thus, $\bar{n}$ worker nodes in the heterogeneous system can be thought as homogeneous system of $n = \sum_{i=0}^{\bar{n}-1} c_{j_i}$ worker nodes of the "weakest" type. Now, for any worker node index $\bar{k}_{AB}$ (such that $0 \leq \bar{k}_{AB} \leq \bar{n} - 1$), we define $k_{AB} = \sum_{i=0}^{\bar{k}_{AB}-1} c_{j_i}$ and $s = \sum_{i=\bar{k}_{AB}}^{\bar{n}-1} c_{j_i}$, so, $n = \sum_{i=0}^{\bar{n}-1} c_{j_i} = k_{AB} + s$. Thus, a heterogeneous system of $\bar{n}$ worker nodes can be thought as a homogeneous system $n = k_{AB} + s$ nodes, for any $\bar{k}_{AB}$ ($0 \leq \bar{k}_{AB} \leq \bar{n} - 1$). Now we state the following corollary (of Theorem 2) for heterogeneous system with the proof in [31].

Corollary 3: Consider a heterogeneous system of $\bar{n}$ nodes of different types for distributed matrix-matrix multiplication and assume any $\bar{k}_{AB}$ (where $0 \leq \bar{k}_{AB} \leq \bar{n} - 1$). Now, if the jobs are assigned to the modified homogeneous system of $n = k_{AB} + s$ "weakest" type worker nodes according to Alg. 2 where

$k_{AB} = k_A k_B$, (a) the system will be resilient to s such nodes and (b) will provide a Q/Δ value as 1.

V. Properties of Our Proposed Schemes

A. Computational Complexity for a Worker Node

Consider random sparse matrices $\mathbf{A} \in \mathbb{R}^{t\times r}$ and $\mathbf{B} \in \mathbb{R}^{t\times w}$ where the probability that any entry is non-zero is η. In our proposed approach in Alg. 2, the respective weights of the assigned submatrices of $\mathbf{A}$ and $\mathbf{B}$ are ω_A and ω_B. Thus, when η is small, the probability of any entry in an encoded submatrix of $\mathbf{A}$ to be non zero is

$$1 - \prod_{i=1}^{\omega_A}(1-\eta) = 1 - (1-\eta)^{\omega_A} \approx 1 - (1 - \omega_A\eta) = \omega_A\eta; \quad (4)$$

For example, (i) if $\eta = 0.01$ and $\omega_A = 6$, we have $1 - (1 - \eta)^{\omega_A} = 1 - (1 - 0.01)^6 = 0.0585 \approx \omega_A\eta$, or (ii) if $\eta = 0.02$ and $\omega_B = 4$, we have $1 - (1 - \eta)^{\omega_B} = 1 - (1 - 0.02)^4 = 0.077 \approx \omega_B\eta$. Thus, in any encoded submatrix of $\mathbf{A}$ or $\mathbf{B}$, the probability of any entry being non-zero is can be approximated by $\omega_A\eta$ or $\omega_B\eta$, respectively.

In this work, we consider the scenarios where $\mathbf{A}$ and $\mathbf{B}$ are sparse, hence η is small. Therefore, the computational complexity for any node is $\mathcal{O}(\omega_A\eta \times \omega_B\eta \times t \times \frac{rw}{k_A k_B}) = \mathcal{O}(\omega_A\omega_B\eta^2 \times \frac{rwt}{k_A k_B})$. On the other hand, in a dense coded approach [4], [11] which assign linear combination of k_A and k_B submatrices, the computational complexity is approximately $\mathcal{O}(k_A\eta \times k_B\eta t \times \frac{rw}{k_A k_B}) = \mathcal{O}(\eta^2 \times rwt)$, which is $\frac{k_A k_B}{\omega_A\omega_B}$ times larger than our proposed method, as $\omega_A < k_A, \omega_B < k_B$.

The recent approach proposed in [8] can deal with sparse matrices with lesser per worker node computational complexity in comparison to the approaches in [4], [5], [11], [12]. The coding scheme in [8] sets the weight of matrix $\mathbf{B}$ as ζ which is given by $\zeta \geq 1 + k_B - \lceil \frac{k_B}{c} \rceil$ where $c = 1 + \lceil \frac{s}{k_B} \rceil$. In that case, the per worker node computational complexity is $O(\eta^2 \times rwt \times (\frac{\zeta}{n} + \frac{\zeta s}{nk_B}))$. Thus in order to compare our approach against [8], we consider the ratio

$$\frac{\left(\frac{\zeta}{n} + \frac{\zeta s}{nk_B}\right)}{\frac{\omega_A\omega_B}{k_A k_B}} = \frac{\zeta k_A(k_B + s)}{n\omega_A\omega_B} = \frac{k_A(k_B+s)}{n} \times \frac{\zeta}{\omega_A\omega_B}. \quad (5)$$

Thus, our proposed approach involves less computational complexity whenever $\frac{k_A(k_B+s)}{n} \times \frac{\zeta}{\omega_A\omega_B} > 1$. In the following, we discuss such examples.

Example 5: Consider a scenario, where $k_A = 8, k_B = 6$ and $s = 3$. In our approach, we set $\omega_A = \omega_B = 2$. On the other hand, the approach in [8] sets $c = 2$ and $\zeta \geq 4$. Thus, according to (5), the ratio of the per worker node computational complexity for the approach in [8] and that in this work is 72/51, which indicates a 30% reduction in our case compared to the method in [8].

Remark 3: The weights in our proposed approach depend on the number of stragglers (s) since we just need to satisfy the inequality, $\omega_A\omega_B > s$. On the other hand, in the approach in [8], the corresponding weights depend on k_A and k_B. Thus, for fixed s, in our approach, the weights remain fixed, whereas for the method in [8], the weights increase with the increase of k_A and k_B.

TABLE II
Comparison of the Worst Case Condition Number, κ_{worst} for Different Distributions for Distributed Computation Over $n = 30$ Nodes With $s = 2$ Stragglers

Distributions	Worst case condition number, κ_{worst}	
	Matrix-vector	Matrix-matrix
rand(0,0.5)	5.53×10^4	1.60×10^4
rand(0,1)	2.43×10^4	1.10×10^4
rand(0,5)	3.61×10^4	1.63×10^4
unifrand(0,1)	2.87×10^4	1.45×10^4
unifrand(-1,1)	2.67×10^4	1.21×10^4
unifrand(-5,5)	2.84×10^4	1.38×10^4

Example 6: Consider the cases where k_A and k_B are even and $k_A = k_B \geq 6$, $s = 5$. In those cases, in our proposed approach, we set $\omega_A = 3$ and $\omega_B = 2$. Thus, the ratio in (5) becomes $\frac{k_A(k_A+s)}{k_A^2+s} \times \frac{\zeta}{6}$. When $k_A = k_B = 8$, this ratio is 1.25, which indicates a 20% reduction in the per worker node computational complexity for our approach compared to the method in [8]. Now, with the increase of $k_A = k_B$ (even), this ratio in (5) is greater than 1 and will increase ($\frac{k_A(k_A+s)}{k_A^2+s} \geq 1$ and ζ will increase), and thus, the gain of our method will always be significant for large n. When $k_A = k_B = 10$, the ratio is 1.43, which indicates a 30% gain over the approach in [8]. When $k_A = k_B = 12$, the ratio is 1.60, and so on.

B. Numerical Stability and Coefficient Determination Time

The condition number is often considered as an important metric for the numerical stability of a linear system [2], [11], [12]. In distributed computation, for a system of n workers and s stragglers, we define the worst case condition number (κ_{worst}) in the homogeneous system as the maximum of the condition numbers of the decoding matrices over all different choices of s stragglers. In the approaches where random coding is involved [2], [5], [12], the idea is to generate random coefficients several times (say, 20 trials), and keep the set of coefficients which provides the minimum κ_{worst}.

In this work, for the proofs of Theorems 1 and 2, we need the coefficients to be chosen i.i.d. at random from a continuous distribution. Now, we briefly explore a few continuous distributions with different parameters to observe the effect on the worst case condition number (κ_{worst}). We consider (i) Gaussian distribution (denoted as "rand(c, d)" if the mean is c and the standard deviation is d) and (ii) uniform distribution (denoted as "unifrand(lb, ub)" if the lower and the upper endpoints are lb and ub, respectively), for a distributed system of $n = 30$ nodes and $s = 2$ stragglers for both distributed matrix-vector and matrix-matrix multiplication. Table II gives the results, where we observe that κ_{worst} remains within an order of magnitude for the different distributions on each type of multiplication. In our numerical simulations in Section VI, we draw the linear coefficients at

TABLE III
COMPARISON OF WORKER COMPUTATION TIME AND COMMUNICATION DELAY FOR MATRIX-MATRIX MULTIPLICATION FOR $n = 39$, $\gamma_A = \gamma_B = \frac{1}{6}$ WHEN RANDOMLY CHOSEN 95%, 98% AND 99% ENTRIES OF MATRICES $\mathbf{A}$ AND $\mathbf{B}$ ARE ZERO

METHODS	WORKER COMP. TIME (IN S)			COMMUNICATION DELAY (IN S)		
	$\mu = 99\%$	$\mu = 98\%$	$\mu = 95\%$	$\mu = 99\%$	$\mu = 98\%$	$\mu = 95\%$
POLY. CODE [4]	1.61	5.13	8.91	0.76	1.41	2.39
ORTHO POLY CODE [11]	1.56	5.18	9.04	0.81	1.43	2.37
RKRP CODE [12]	1.58	5.09	8.95	0.78	1.38	2.35
SCS OPTIMAL SCHEME [5]	0.97	1.38	4.31	0.28	0.42	0.61
CLASS-BASED SCHEME [8]	0.52	0.85	3.42	0.23	0.34	0.55
PROPOSED SCHEME	0.34	0.53	2.24	0.16	0.25	0.42

i.i.d. from the standard normal distribution, since it gives the best performance out of the distributions considered here.[1]

In our proposed matrix-matrix multiplication approach in Alg. 2, we partition $\mathbf{A}$ and $\mathbf{B}$ into k_A and k_B block-columns, respectively, and we have the recovery threshold $\tau = k_A k_B$. Thus, in every trial we need to determine $\binom{n}{\tau}$ condition numbers of $\tau \times \tau$ sized (decoding) matrices, which has a total complexity of $\mathcal{O}(\binom{n}{\tau}\tau^3)$. On the other hand, the recent sparse matrix computation approaches in [5], [8] partition matrix $\mathbf{A}$ into $\Delta_A = \text{LCM}(n, k_A)$ block-columns. Thus, in every trial, they need to determine $\binom{n}{\tau}$ condition numbers of $\Delta_A k_B \times \Delta_A k_B$ sized matrices which has a total complexity of $\mathcal{O}(\binom{n}{\tau}\Delta_A^3 k_B^3)$. Since, Δ_A can be significantly larger than k_A, every trial involves considerably more complexity in comparison to ours. For example, consider a case where n and k_A are coprime. In that case $\Delta_A = nk_A$, and thus the complexity corresponding to the approaches [5], [8] are around $\mathcal{O}(n^3)$ times higher than ours. Similar result holds for the matrix-vector multiplication case in Alg. 1. Note that the approach in [12] involves similar computational complexity per trial as ours, and the approaches in [4], [11] do not require such coefficient search; however, these methods have very high computational complexity per worker node for sparse matrices, as discussed in Section V-A.

Remark 4: In case of distributed matrix-vector multiplication, our proposed Alg. 1 has a per worker node computational complexity $\mathcal{O}((s+1)\eta t \times \frac{r}{k_A}) = \mathcal{O}((s+1)\ \eta \times \frac{rt}{k_A})$. While it is significantly smaller than that of dense coded approaches, the approach in [8] has a per worker node computational complexity $\mathcal{O}((s+1)\ \eta \times \frac{rt}{n})$, slightly smaller than our proposed method (since $n = k_A + s$). However, it involves very high coefficient determination time which is confirmed by numerical experiments in Section VI. In addition, unlike the approaches in [5], [8], our approach is extended to heterogeneous systems.

Remark 5: It should be noted that our proposed approach requires the central node to invert a $k_A k_B \times k_A k_B$ sized decoding matrix. Thus, for $\mathbf{A} \in \mathbb{R}^{t\times r}$ and $\mathbf{B} \in \mathbb{R}^{t\times w}$, the corresponding decoding complexity is $\mathcal{O}(k_A^3 k_B^3 + rwk_A k_B)$. On the other hand, the SCS optimal approach in [5] and the class-based scheme in [8] involves a decoding complexity $\mathcal{O}(\Delta_A^3 k_B^3 + rw\Delta_A k_B)$, where $\Delta_A = \text{LCM}(n, k_A)$ could be significantly higher than k_A.

[1]Table II implies that the impact of the distribution parameters on the worst case condition number can be non-monotonic. Since our focus in this work is to provide resilience to the maximum number of stragglers irrespective of the coefficient distribution, we leave a more comprehensive investigation of how the choice of distribution impacts numerical stability to future work.

VI. NUMERICAL EXPERIMENTS

In this section, we compare the performance of our proposed approaches with different competing methods [4], [5], [8], [11], [12] via numerical experiments. We conduct our experiments on an AWS (Amazon Web Services) cluster with `t2.small` machines as the worker nodes. Note that the work in [24] is also suited for sparse matrix computations, however, it does not follow the storage constraints as mentioned in [4], [5], [8], [11], [12] and also does not meet the exact optimal recovery threshold. Therefore, we do not include [24] in our comparison.

We consider the case of matrix-matrix multiplication in a system with $n = 39$ workers, each of which can store $\gamma_A = \gamma_B = \frac{1}{6}$ fraction of matrices $\mathbf{A}$ and $\mathbf{B}$. We consider sparse input matrices $\mathbf{A}$ of size $20,000 \times 15000$ and $\mathbf{B}$ of size $20,000 \times 12000$. We assume three different cases where the sparsity of $\mathbf{A}$ and $\mathbf{B}$ are 95%, 98% and 99%, respectively, which indicate that randomly chosen 95%, 98% and 99% entries of matrix $\mathbf{A}$ are zero. There are many practical examples where the structure of data matrices exhibit this level of sparsity (see [13] for such examples). Note that we also carry out numerical simulations on matrix-vector multiplication. The results follow a similar trend of the matrix-matrix case and is discussed in [31].

Worker computation time: First we compare different methods in terms of worker computation time (the required time for a worker to complete its respective job) for our system of $n = 39$ workers and the results are shown in Table III. In this example, the approaches in [4], [11], [12] assign linear combinations of $k_A = k_B = 6$ submatrices to the worker nodes. Hence, the inherent sparsity of both $\mathbf{A}$ and $\mathbf{B}$ can be destroyed in the encoded submatrices. On the other hand, our proposed approach or the approaches in [5] or [8] assign linear combinations of less number of submatrices, and hence, are specifically suited for sparse matrices. Table III demonstrates that the worker node computations in these approaches are significantly faster than the dense coded approaches. In addition, if we compare our proposed approach against the approach in [8], we can see that the ratio in (5) is $\frac{6\times(6+3)}{39} \times \frac{4}{4} \approx 1.38$. It indicates around a 30% reduction of worker computational complexity in our proposed approach than the method in [8] which can be roughly verified from the results in Table III.

Communication delay: Now, the comparison among different approaches in terms of communication delay is also demonstrated in Table III. Here we define the communication

TABLE IV
COMPARISON AMONG DIFFERENT APPROACHES IN TERMS OF WORST CASE CONDITION NUMBER (κ_{worst}) AMONG ALL DIFFERENT CHOICES OF s STRAGGLERS AND THE CORRESPONDING REQUIRED TIME FOR 10 TRIALS TO FIND A GOOD SET OF COEFFICIENTS

METHODS	κ_{worst} FOR $n = 33, s = 3$	REQ. TIME 10 TRIALS	κ_{worst} FOR $n = 39, s = 3$	REQ. TIME 10 TRIALS	$\bar{\kappa}_{worst}$, $\bar{n} = 19$ $n = 30, s = 6$
POLY. CODE [4]	5.14×10^{14}	0	4.39×10^{19}	0	1.49×10^{13}
ORTHO-POLY [11]	7.23×10^{5}	0	1.81×10^{6}	0	1.74×10^{9}
RKRP CODE [12]	2.38×10^{5}	$5.45s$	3.43×10^{5}	$10.31s$	7.11×10^{6}
SCS OPT. SCH. [5]	5.39×10^{7}	$738s$	9.15×10^{7}	$3191s$	3.16×10^{8}
CLASS-BASED [8]	4.95×10^{7}	$1327s$	6.34×10^{7}	$5772s$	5.29×10^{8}
PROP. SCHEME	4.40×10^{5}	$5.87s$	2.21×10^{6}	$11.37s$	7.78×10^{7}

delay as the required time for the central node to transmit the coded submatrices to all the worker nodes. Thus, depending on the coding procedure, the central node may involve different communication delay for different approaches. Since the approaches in [4], [11], [12] assign dense linear combinations of the submatrices, to transmit these large number of non-zero entries, the system involves a considerable communication delay. On the other hand, the algorithm for our proposed scheme and the methods in [5] and [8] limit the number of non-zero entries; hence the delay is reduced significantly.

In addition, here we compare our proposed approach against the method in [8] in terms of the approximate number of non-zero entries. Consider the case when the matrices $\mathbf{A}$ and $\mathbf{B}$ have approximately 99% entries to be zero. Now, in our proposed approach, the number of non-zero entries to be sent to each worker node from the central node is approximately $\frac{20k\times 15k}{k_A} \times 0.01 \times \omega_A + \frac{20k\times 12k}{k_B} \times 0.01 \times \omega_B = 1.8 \times 10^6$. On the other hand, the number of the corresponding non-zero entries to be sent to each worker node in the approach [8] is approximately $\frac{20k\times 15k}{k_A} \times \frac{1}{13} \times (12 \times 0.01 + k_A \times 0.01) + \frac{20k\times 12k}{k_B} \times 0.01 \times \zeta \approx 2.3 \times 10^6$. Thus our proposed approach requires the central node to transmit approximately 20% less non-zero entries than the method in [8], which confirms the gain of our approach in Table III.

Numerical stability: Next we evaluate the numerical stability of the system for different distributed computation techniques. For any system of n workers and s stragglers, we find the condition numbers of the decoding matrices over all different choices of s stragglers and find the worst case condition number (κ_{worst}). We consider two different systems with different number of workers and stragglers. In system 1, we set $n = 33$ and $s = 3$, and in system 2, we set $n = 39$ and $s = 3$ and demonstrate the κ_{worst} values of different approaches in Table IV. As expected, the approach in [4] has a very high κ_{worst} which indicates its numerical instability. Among the numerically stable systems, our proposed approach provides smaller κ_{worst} values in comparison to the methods in [5] and [8] and also comparable with [11] and [12].

Coefficient determination time: Finally, we compare different approaches in terms of the required time for running 10 trials to find a "good" set of random coefficients that make the system numerically stable. Our proposed approach and the approach in [12] partition matrices $\mathbf{A}$ and $\mathbf{B}$ into k_A and k_B block-columns, which leads to $k_A k_B$ unknowns. On the other hand, the approaches in [5] and [8] partition matrices $\mathbf{A}$ and $\mathbf{B}$ into $\Delta_A = \text{LCM}(n, k_A)$ and k_B block-columns, which leads to $\Delta_A k_B$ unknowns. Now as discussed in Section V-B, Δ_A can be significantly higher than k_A and to find the condition numbers of these larger-sized matrices, approaches in [5] and [8] take much more time than ours. Table IV confirms more than $100\times$ speed gain for our proposed scheme over the methods in [5], [8] to find a "good" set of coefficients.

Thus, in summary, while the approaches in [5], [8] involve similar worker computation time as our proposed approach, they require significantly higher encoding time than ours.

Heterogeneous system: Next we consider a matrix-matrix multiplication over a heterogeneous system of $\bar{n} = 19$ worker nodes of $\lambda = 3$ different types of nodes. We assume that there are $\bar{n}_0 = 11$, $\bar{n}_1 = 5$ and $\bar{n}_2 = 3$ nodes of types $0, 1$ and 2, respectively, which are assigned 1, 2 and 3 block-columns each, respectively, hence $n = 11 \times 1 + 5 \times 2 + 3 \times 3 = 30$. We design the scheme according to Alg. 2, such that it is resilient to *any* $s = 6$ block-columns processing.

Now, in this heterogeneous setting, since different nodes are assigned different amounts of jobs depending on their corresponding types, the central node will wait until it receives the results of fastest $Q = 24$ block-column processing. In this regard, we define the worst case condition number in the heterogeneous system ($\bar{\kappa}_{worst}$) as the maximum of the condition numbers of the decoding matrices over all different choices of s block-columns. Table IV shows the comparison among different approaches in terms of ($\bar{\kappa}_{worst}$): our proposed approach provides significantly smaller ($\bar{\kappa}_{worst}$) values than the approaches in [4], [5], [8], [11] and also provides competitive $\bar{\kappa}_{worst}$ value in compared to the approach in [12].

VII. CONCLUSION

In this work, we have developed distributed matrix computation schemes which preserve sparsity properties of the inputs while remaining resilient to the maximum number of stragglers for given storage constraints. We saw how existing dense coded approaches [2], [4], [11], [12] suffer from a huge communication and computation delay in case of sparse matrices. Since our proposed approach allows very limited amounts of coding within the submatrices, it preserves the inherent sparse structure of the input matrix $\mathbf{A}$ (and $\mathbf{B}$) up to certain level. Thus, the worker computation delay and the communication delay were seen to be significantly reduced in comparison to those dense

TABLE V
NOTATION TABLE

NOTATION	DEFINITION	DESCRIPTION
$\mathbf{A}, \mathbf{B}$	SPARSE LARGE-SIZED MATRICES	$\mathbf{A} \in \mathbb{R}^{t\times r}, \mathbf{B} \in \mathbb{R}^{t\times w}$
γ_A, γ_B	STORAGE FRACTION FOR $\mathbf{A}$ AND $\mathbf{B}$, RESPECTIVELY	$\gamma_A = \frac{1}{k_A}, \gamma_B = \frac{1}{k_B}$
n	NUMBER OF TOTAL WORKER NODES	$n \geq k_A k_B$
s	NUMBER OF MAXIMUM POSSIBLE STRAGGLERS	$s = n - k_A k_B$
W_i	WORKER NODE WITH INDEX i	$0 \leq i \leq n-1$
Δ_A, Δ_B	NUMBER OF BLOCK-COLUMNS THAT $\mathbf{A}$ AND $\mathbf{B}$, RESPECTIVELY, ARE PARTITIONED INTO	$\Delta_A = k_A$ AND $\Delta_B = k_B$
Δ	TOTAL NUMBER OF UNKNOWNS THAT NEED TO BE RECOVERED	$\Delta = \Delta_A \Delta_B$
κ_{worst}	WORST CASE CONDITION NUMBER OVER ALL $\binom{n}{s}$ STRAGGLERS	–
τ	RECOVERY THRESHOLD OF THE SCHEME	$\tau = k_A k_B$
Q	NUMBER OF SUBMATRIX PRODUCTS THAT HAVE TO BE COMPUTED IN THE WORST CASE TO RECOVER THE INTENDED RESULT	$Q \geq \Delta$
ω_A, ω_B	WEIGHTS FOR THE ENCODING OF $\mathbf{A}$ AND $\mathbf{B}$	$\omega_A \omega_B > s$

coded approaches. There are some sparsely coded approaches in [5], [8] which have been developed specifically to deal with sparse matrices; however, our proposed approach was seen to provide three-fold gains over them. Overall, we showed analytically and experimentally that our proposed approach (i) provides significant gain in worker computation and communication delay, (ii) saves considerable amount of time to find a "good" set of coefficients to make the system numerically stable, and (iii) is applicable to the systems where the worker nodes are heterogeneous in nature.

There are a number of directions for the future work of this paper. While there are several secure distributed matrix computation schemes [14], [15], [16], [17] which protect the system against privacy leakage, most of them add dense random matrices to the coded submatrices which destroy the sparsity of the assigned submatrices. Thus, a straggler resilient secure coded scheme needs to be developed which is particularly suitable for sparse input matrices. Another direction can be developing a scheme for a server-less architecture, where there is no such central node to encode the matrices and the worker nodes may communicate among them to establish straggler resilience. This can be particularly helpful for distributed learning or federated learning methods [32]. We could also aim for improving the performance in the heterogeneous setting by assigning multiple jobs with varying weights as guided in [33], [34]. Moreover, we need to develop schemes where true knowledge about the worker nodes in the heterogeneous setting may not be available prior to the assignment of the jobs.

APPENDIX A
NOTATION TABLE

In this section, for the ease of the readers, we provide an overview of the notations used in this work in Table V. This table also includes very brief definitions of the corresponding notations.

APPENDIX B
PROOF OF LEMMA 1

Proof: As discussed in Section IV-A, we denote the minimum number of participating unknowns in $\tilde{\mathcal{M}}_i$ by ρ_i when $0 \leq i \leq k_A - 1$. The trivial lower bound for ρ_i is *zero* when $k_A - \omega_A + 1 \leq i \leq k_A - 1$, hence, $\sum_{i=0}^{k_A-1} \rho_i \geq \sum_{i=0}^{k_A-\omega_A} \rho_i$. Thus, in order to prove the lemma, we need to show that

$$\sum_{i=0}^{k_A-\omega_A} \rho_i \geq \sum_{i=0}^{k_A-1} \tilde{\delta}_i. \tag{6}$$

Now we provide the following definition for the next part of the proof.

Definition 2: We say that $\tilde{\mathcal{M}}_i$ covers itself, if $\rho_i \geq \tilde{\delta}_i$. Next, we say that a set of $\tilde{\mathcal{M}}_i$'s, denoted by $\mathcal{U}$, covers itself and another set of $\tilde{\mathcal{M}}_j$'s, denoted by $\mathcal{V}$, (where $\mathcal{U}$ and $\mathcal{V}$ are disjoint) if

$$\sum_{i:\tilde{\mathcal{M}}_i \in \mathcal{U}} \rho_i = \sum_{i:\tilde{\mathcal{M}}_i \in \mathcal{U}} \tilde{\delta}_i + \sum_{j:\tilde{\mathcal{M}}_j \in \mathcal{V}} \tilde{\delta}_j.$$

Claim 3: Every $\tilde{\mathcal{M}}_j$ covers itself, for $0 \leq j \leq k_A - \omega_A$.

Proof: From (1), we have $\rho_0 \geq \omega_A \times (\omega_B + \tilde{\delta}_0 - 2) \geq \tilde{\delta}_0$, since $\omega_A, \omega_B \geq 2$. So, $\tilde{\mathcal{M}}_0$ covers itself. Next from (2), for $1 \leq j \leq k_A - \omega_A$, we have $\rho_j \geq \tilde{\delta}_j + \omega_B - 2 \geq \tilde{\delta}_j$. So, $\tilde{\mathcal{M}}_j$ covers itself. ∎

Now since (6) leads to $\sum_{i=0}^{k_A-\omega_A} \rho_i \geq \sum_{i=0}^{k_A-\omega_A} \tilde{\delta}_i + \sum_{i=k_A-\omega_A+1}^{k_A-1} \tilde{\delta}_i$, we first denote a set $\mathcal{V} = \{\tilde{\mathcal{M}}_{k_A-\omega_A+1}, \tilde{\mathcal{M}}_{k_A-\omega_A+2}, \ldots, \tilde{\mathcal{M}}_{k_A-1}\}$ (thus, $|\mathcal{V}| = \omega_A - 1$) and in order to satisfy (6), we always need to find an appropriate $\mathcal{U}$ which can also cover $\mathcal{V}$ along with itself. To do so, we need to find those $\tilde{\mathcal{M}}_i$'s where $\rho_i - \tilde{\delta}_i > 0$. In this proof, we define $\mathcal{U}_\lambda = \{\tilde{\mathcal{M}}_0, \tilde{\mathcal{M}}_1, \tilde{\mathcal{M}}_2, \ldots, \tilde{\mathcal{M}}_\lambda\}$. Now we consider the following two cases for each of which we show that (6) is true.

Case 1 ($1 \leq \tilde{\delta}_0 \leq k_B$): If, $\tilde{\delta}_0 = 1$, using (1), we have $\rho_0 = \omega_A \omega_B > \omega_A \tilde{\delta}_0$, since $\omega_B \geq 2$. Moreover, if $1 < \tilde{\delta}_0 \leq k_B$, using (1), we have $\rho_0 \geq \omega_A \tilde{\delta}_0$. Thus, when $1 \leq \tilde{\delta}_0 \leq k_B$, we have $\rho_0 \geq \omega_A \tilde{\delta}_0 = \tilde{\delta}_0 + (\omega_A - 1)\tilde{\delta}_0 \geq \tilde{\delta}_0 + \sum_{i=k_A-\omega_A+1}^{k_A-1} \tilde{\delta}_i$, since $\tilde{\delta}_i$'s are arranged in a non-increasing order. In this case, we can set $\mathcal{U} = \mathcal{U}_0 = \{\tilde{\mathcal{M}}_0\}$ which covers $\mathcal{V}$ along with itself. Now, since each of the other $\tilde{\mathcal{M}}_i$'s (for $i \leq 1 \leq k_A - \omega_A$) covers itself (according to Claim 3), (6) is true.

Case 2 ($\tilde{\delta}_0 = k_B + 1$): Assume that $\tilde{\delta}_0 = \tilde{\delta}_1 = \cdots = \tilde{\delta}_{\alpha-1} = k_B + 1$. Now, there can be at most $s \leq \min(\omega_A \omega_B - 1, k_A)$ such

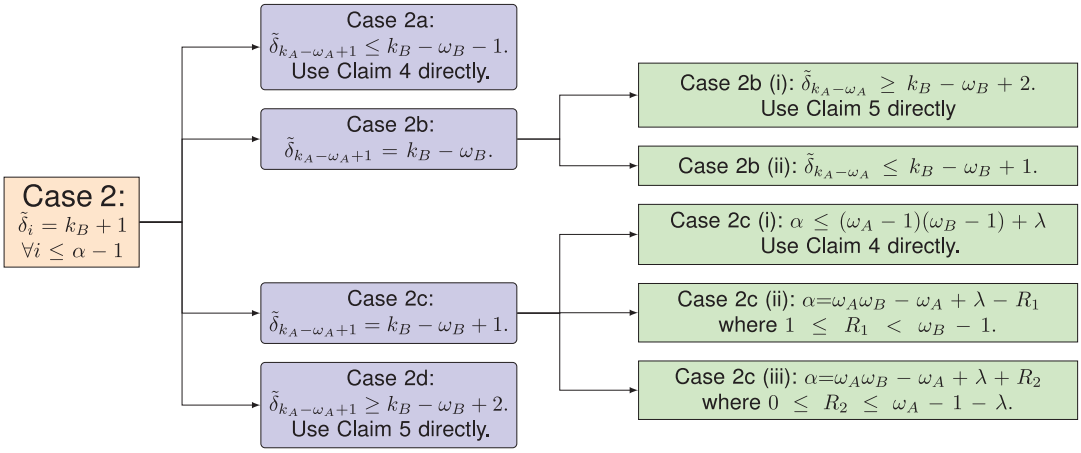

Fig. 4. An overview of Case 2.

$\tilde{\mathcal{M}}_i$'s which have cardinality k_B+1. Thus, α is upper bounded by $\min(\omega_A\omega_B-1, k_A)$. Now, before moving into details in this case, we state the following claims. The corresponding proofs are given in [31].

Claim 4: For $\alpha \geq 1$, if $\sum_{i=k_A-\omega_A+1}^{k_A-1} \tilde{\delta}_i \leq (\omega_A - 1)\ k_B - \alpha$, then (6) is true by setting $\mathcal{U} = \mathcal{U}_{\alpha-1}$.

Remark 6: While we cannot assume $\omega_A \geq \omega_B$ without loss of generality, note that ω_A and ω_B are design parameters. The constraint that we have on the number of stragglers in terms of ω_A and ω_B is that $s \leq \omega_A\omega_b - 1$. Since the upper bound is symmetric in terms of ω_A and ω_B, we can always set $\omega_A \geq \omega_B$ in our design to be resilient to the same number of stragglers.

Claim 5: Assume that $\alpha \geq 1$ and let us define κ as the minimum i such that $\tilde{\delta}_i \leq k_B - 1$. (a) If $\kappa > k_A - \omega_A$, then $\mathcal{U} = \mathcal{U}_{\alpha-1}$ will cover $\mathcal{V}$ along with itself. (b) If $\kappa \leq k_A - \omega_A$ and $\tilde{\delta}_i \geq k_B - \omega_B + 2$ for all $i \leq k_A - \omega_A$, then $\mathcal{U} = \mathcal{U}_{\alpha-1} \cup \tilde{\mathcal{U}}_\kappa$ will cover $\mathcal{V}$ along with itself, where $\tilde{\mathcal{U}}_\kappa := \{\tilde{\mathcal{M}}_\kappa, \tilde{\mathcal{M}}_{\kappa+1}, \ldots, \tilde{\mathcal{M}}_{k_A-\omega_A}\}$.

Our main idea is to perform an exhaustive case analysis on the value of $\tilde{\delta}_{k_A-\omega_A+1}$, an overview of which is depicted in Fig. 4. In all cases, we find the appropriate $\mathcal{U}$ such that (6) holds.

Case 2a ($\tilde{\delta}_{k_A-\omega_A+1} \leq k_B - \omega_B - 1$): Now, $\tilde{\delta}_i$'s are in non-increasing order, thus, in this case

$$\begin{aligned}&\textstyle\sum_{i=k_A-\omega_A+1}^{k_A-1} \tilde{\delta}_i \leq (k_B - \omega_B - 1)(\omega_A - 1)\\ =\ & k_B(\omega_A - 1) - (\omega_B + 1)(\omega_A - 1) \leq k_B(\omega_A - 1) - \alpha,\end{aligned}$$

since $\omega_A \geq \omega_B$ and $\alpha \leq \omega_A\omega_B - 1$. Thus, according to Claim 4, we are done by setting $\mathcal{U} = \mathcal{U}_{\alpha-1}$.

Case 2b ($\tilde{\delta}_{k_A-\omega_A+1} = k_B - \omega_B$): To prove (6), here we consider the following two subcases.

Case 2b (i) ($\tilde{\delta}_{k_A-\omega_A} \geq k_B - \omega_B + 2$): In this scenario, if $\kappa > k_A - \omega_A$, then we are done using Claim 5(a), and, if $\kappa \leq k_A - \omega_A$, then we are done using Claim 5(b) since $\tilde{\delta}_i \geq k_B - \omega_B + 2$ for all $i \leq k_A - \omega_A$. We can find κ in Claim 5 as the minimum value of i when $\tilde{\delta}_i < k_B$.

Case 2b (ii) ($\tilde{\delta}_{k_A-\omega_A} \leq k_B - \omega_B + 1$): Since $\tilde{\delta}_{k_A-\omega_A+1} = k_B - \omega_B$, then $\sum_{i=k_A-\omega_A+1}^{k_A-1} \tilde{\delta}_i \leq (k_B - \omega_B)(\omega_A - 1)$. Now in this scenario, $\sum_{i=0}^{\alpha-1} \rho_i + \rho_{k_A-\omega_A} = \omega_A k_B + (\alpha-1)k_B + \rho_{k_A-\omega_A}$, and,

$$\begin{aligned}\sum_{i=0}^{\alpha-1} \tilde{\delta}_i + \sum_{i=k_A-\omega_A}^{k_A-1} \tilde{\delta}_i &\leq \tilde{\delta}_{k_A-\omega_A} + (k_B - \omega_B)(\omega_A - 1) + \alpha\\ &\quad + \alpha k_B = \omega_A k_B\\ &\quad + (\alpha - 1)k_B + \tilde{\delta}_{k_A-\omega_A} + \alpha - \omega_B(\omega_A - 1).\end{aligned}$$

If $\left|\tilde{\mathcal{M}}_{k_A-\omega_A}\right| = k_B$, then according to (2), $\rho_{k_A-\omega_A} - \tilde{\delta}_{k_A-\omega_A} \geq \omega_B - 1$. Since $\alpha \leq \omega_A\omega_B - 1$, we are done by setting $\mathcal{U} = \mathcal{U}_{\alpha-1} \cup \{\tilde{\mathcal{M}}_{k_A-\omega_A}\}$ to cover $\mathcal{V}$ along with itself. Otherwise, if $\left|\tilde{\mathcal{M}}_{k_A-\omega_A}\right| = k_B + 1$, we have $\rho_{k_A-\omega_A} - \tilde{\delta}_{k_A-\omega_A} \geq \omega_B - 2$. However, in that case $\alpha \leq \omega_A\omega_B - 2$ since $\left|\tilde{\mathcal{M}}_{k_A-\omega_A}\right|$ is already set as $k_B + 1$. Thus, we can again set $\mathcal{U} = \mathcal{U}_{\alpha-1} \cup \{\tilde{\mathcal{M}}_{k_A-\omega_A}\}$.

Case 2c ($\tilde{\delta}_{k_A-\omega_A+1} = k_B - \omega_B + 1$): In this case, if $\tilde{\delta}_{k_A-\omega_A} \geq k_B - \omega_B + 2$, according to Claim 5(b), we are done by setting $\mathcal{U} = \mathcal{U}_{\alpha-1} \cup \tilde{\mathcal{U}}_\kappa$. We can find κ in as the minimum value of i when $\tilde{\delta}_i < k_B$. Now, we consider the only remaining scenario where $\tilde{\delta}_{k_A-\omega_A} = k_B - \omega_B + 1$. Since, $\tilde{\delta}_{k_A-\omega_A+1} = k_B - \omega_B + 1$, we assume $\sum_{i=k_A-\omega_A+1}^{k_A-1} \tilde{\delta}_i = (\omega_A - 1)\ (k_B - \omega_B + 1)\ - \lambda_2 = (\omega_A - 1)\ k_B - (\omega_A - 1)(\omega_B - 1)\ - \lambda_2$. Now, we consider the following three subcases.

Case 2c (i) ($\alpha \leq (\omega_A - 1)\ (\omega_B - 1)\ + \lambda_2$): Here, we are done by setting $\mathcal{U} = \mathcal{U}_{\alpha-1}$ (Claim 4).

Case 2c (ii) ($\alpha = \omega_A\omega_B - \omega_A + \lambda_2 - R_1$, where $1 \leq R_1 < \omega_B - 1$): Here, $\sum_{i=0}^{\alpha-1} \rho_i + \rho_{k_A-\omega_A} = \omega_A k_B + (\alpha-1)k_B + \rho_{k_A-\omega_A}$, and,

$$\begin{aligned}\sum_{i=0}^{\alpha-1} \tilde{\delta}_i + \sum_{i=k_A-\omega_A}^{k_A-1} \tilde{\delta}_i &= \alpha(k_B + 1) + \tilde{\delta}_{k_A-\omega_A}\\ &\quad + (k_B - \omega_B + 1)(\omega_A - 1) - \lambda_2 = \omega_A k_B + (\alpha - 1)k_B\\ &\quad + \tilde{\delta}_{k_A-\omega_A} + \alpha - \{(\omega_A - 1)(\omega_B - 1) + \lambda_2\}.\end{aligned}$$

Since, $\tilde{\delta}_{k_A-\omega_A} = k_B - \omega_B + 1$, according to (2), $\rho_{k_A-\omega_A} - \tilde{\delta}_{k_A-\omega_A} \geq \omega_B - 2$. In addition, $\alpha - [(\omega_A - 1)\ (\omega_B - 1)\ + \lambda_2] = \omega_B - 1 - R_1$. Thus, we are done by setting $\mathcal{U} = \mathcal{U}_{\alpha-1} \cup \{\tilde{\mathcal{M}}_{k_A-\omega_A}\}$.

Case 2c (iii) ($\alpha = \omega_A\omega_B - \omega_A + \lambda_2 + R_2$, where $0 \leq R_2 \leq \omega_A - 1 - \lambda_2$): Since we know, $\tilde{\delta}_{k_A-\omega_A} = k_B - \omega_B + 1$, we assume that $\tilde{\delta}_{k_A-\omega_A} = \tilde{\delta}_{k_A-\omega_A-1} = \cdots = \tilde{\delta}_{k_A-\omega_A-\beta+1} =$

$k_B - \omega_B + 1$, where $\beta \geq 1$. We define the set $\mathcal{H} = \{\tilde{\mathcal{M}}_{k_A-\omega_A}, \tilde{\mathcal{M}}_{k_A-\omega_A-1}, \ldots, \tilde{\mathcal{M}}_{k_A-\omega_A-\beta+1}\}$.

Now, we consider two scenarios depending on the value of ω_B. If $\omega_B \geq 3$, we assume that $\tilde{\delta}_i = k_B - \omega_B + 2 + \bar{\sigma}_i$ with $0 \leq \bar{\sigma}_i \leq \omega_B - 3$ where $\kappa \leq i \leq k_A - \omega_A - \beta$. Thus, in this scenario,

$$\begin{aligned}\sum_{i=0}^{k_A-\omega_A} \rho_i &= \sum_{i=0}^{\alpha-1} \rho_i + \sum_{i=\alpha}^{\kappa-1} \rho_i + \sum_{i=\kappa}^{k_A-\omega_A-\beta} \rho_i + \sum_{i=k_A-\omega_A-\beta+1}^{k_A-\omega_A} \rho_i \\ &= \alpha k_B + k_B(\omega_A - 1) + (\kappa - \alpha)k_B \\ &\quad + (k_A - \omega_A - \beta - \kappa + 1)k_B + \sum_{i=k_A-\omega_A-\beta+1}^{k_A-\omega_A} \rho_i \quad (7)\end{aligned}$$

In the remaining scenario where $\omega_B = 2$, we have $\tilde{\delta}_{k_A-\omega_A-\beta+1} = k_B - 1$, thus $\tilde{\delta}_i = k_B$, when $\alpha \leq i \leq k_A - \omega_A - \beta$. So,

$$\begin{aligned}\sum_{i=0}^{k_A-\omega_A} \rho_i &= \sum_{i=0}^{\alpha-1} \rho_i + \sum_{i=\alpha}^{k_A-\omega_A-\beta} \rho_i + \sum_{i=k_A-\omega_A-\beta+1}^{k_A-\omega_A} \rho_i \\ &= \alpha k_B + k_B(\omega_A - 1) + (k_A - \omega_A - \beta - \alpha + 1)k_B \\ &\quad + \sum_{i=k_A-\omega_A-\beta+1}^{k_A-\omega_A} \rho_i \quad (8)\end{aligned}$$

Thus, from (7) and (8), for any $\omega_B \geq 2$, we have

$$\sum_{i=0}^{k_A-\omega_A} \rho_i = (k_A - \beta)k_B + \sum_{i=k_A-\omega_A-\beta+1}^{k_A-\omega_A} \rho_i. \quad (9)$$

Now, since $\lambda_2 + R_2 \leq \omega_A - 1$ and $\sum_{i=k_A-\omega_A+1}^{k_A-1} \tilde{\delta}_i = (\omega_A - 1)(k_B - \omega_B + 1) - \lambda_2$, the number of $\tilde{\mathcal{M}}_i$'s in $\mathcal{V}$ having $\tilde{\delta}_i$ to be less than $k_B - \omega_B + 1$ is upper bounded by λ_2. Thus, the number of $\tilde{\mathcal{M}}_i$'s left in $\mathcal{V}$ with $\tilde{\delta}_i = k_B - \omega_B + 1$ is lower bounder by $\omega_A - 1 - \lambda_2$, we denote the set of such $\tilde{\mathcal{M}}_i$'s as $\tilde{\mathcal{V}}$. Besides, we have $\beta \geq 1$ more $\tilde{\delta}_i$'s with the same value, $k_B - \omega_B + 1$. Thus, $\tilde{\delta}_i = k_B - \omega_B + 1$ when $i = k_A - \omega_A - \beta + 1, \ldots, k_A - \omega_A, k_A - \omega_A + 1, \ldots k_A - 1 - \lambda_2$.

Note that, $\tilde{\mathcal{M}}_i \in \mathcal{H}$ when $i = k_A - \omega_A - \beta + 1, \ldots, k_A - \omega_A$ and $\tilde{\mathcal{M}}_i \in \tilde{\mathcal{V}}$ when $i = k_A - \omega_A + 1, \ldots k_A - 1 - \lambda_2$. But, the total number of $\tilde{\mathcal{M}}_i$'s with cardinality $k_B + 1$ is upper bounded by $\omega_A\omega_B - 1$. Since we have already taken account $\alpha = \omega_A\omega_B - \omega_A + \lambda_2 + R_2$ of such $\tilde{\mathcal{M}}_i$'s, we have at most $\omega_A - 1 - \lambda_2 - R_2$ of $\tilde{\mathcal{M}}_i$'s left which belong to $\tilde{\mathcal{U}}_\kappa \cup \mathcal{V}$. Thus, we can have at most $\omega_A - 1 - \lambda_2 - R_2$ of such $\tilde{\mathcal{M}}_i$'s in $\mathcal{H} \cup \tilde{\mathcal{V}}$, since $\mathcal{H} \subseteq \tilde{\mathcal{U}}_\kappa$ and $\tilde{\mathcal{V}} \subseteq \mathcal{V}$.

Now, since $\tilde{\delta}_i = k_B - \omega_B + 1$ when $\tilde{\mathcal{M}}_i \in \mathcal{H} \cup \tilde{\mathcal{V}}$, and $\left|\tilde{\mathcal{V}}\right| \geq \omega_A - 1 - \lambda_2 - R_2$, according to our arrangement of $\tilde{\mathcal{M}}_i$'s (in Section IV-A2(ii)), all such $\tilde{\mathcal{M}}_i$'s (i.e., $|\tilde{\mathcal{M}}_i| = k_B + 1$ and $\tilde{\delta}_i = k_B - \omega_B + 1$) belong to $\tilde{\mathcal{V}}$. Therefore, $\left|\tilde{\mathcal{M}}_i\right| = k_B$ when $\tilde{\mathcal{M}}_i \in \mathcal{H}$ and $\rho_i = (\omega_B + \tilde{\delta}_i - 1, k_B) = k_B$, since $\tilde{\delta}_i = k_B - \omega_B + 1$. Thus we are done using (9), since $\sum_{i=0}^{k_A-\omega_A} \rho_i = k_A k_B \geq \sum_{i=0}^{k_A-1} \tilde{\delta}_i$.

Case 2d ($\tilde{\delta}_{k_A-\omega_A+1} \geq k_B - \omega_B + 2$): In this case, if $\kappa > k_A - \omega_A$, then we are done using Claim 5(a), and, if $\kappa \leq k_A - \omega_A$, then we are done using Claim 5(b) since $\tilde{\delta}_i \geq k_B - \omega_B + 2$ for all $i \leq k_A - \omega_A$. We can find κ (from Claim 5) as the minimum value of i when $\tilde{\delta}_i < k_B$. ■

REFERENCES

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

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

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

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

[5] A. B. Das and A. Ramamoorthy, "Coded sparse matrix computation schemes that leverage partial stragglers," *IEEE Trans. Inf. Theory*, vol. 68, no. 6, pp. 4156–4181, Jun. 2022.

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

[7] R. Tandon, Q. Lei, A. G. Dimakis, and N. Karampatziakis, "Gradient coding: Avoiding stragglers in distributed learning," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2017, pp. 3368–3376.

[8] A. B. Das and A. Ramamoorthy, "A unified treatment of partial stragglers and sparse matrices in coded matrix computation," *IEEE J. Sel. Area Inf. Theory*, vol. 3, no. 2, pp. 241–256, Jun. 2022.

[9] A. K. Pradhan, A. Heidarzadeh, and K. R. Narayanan, "Factored LT and factored raptor codes for large-scale distributed matrix multiplication," *IEEE J. Sel. Area Inf. Theory*, vol. 2, no. 3, pp. 893–906, Sep. 2021.

[10] S. Dutta, M. Fahim, F. Haddadpour, H. Jeong, V. Cadambe, and P. Grover, "On the optimal recovery threshold of coded matrix multiplication," *IEEE Trans. Inf. Theory*, vol. 66, no. 1, pp. 278–301, Jan. 2020.

[11] M. Fahim and V. R. Cadambe, "Numerically stable polynomially coded computing," *IEEE Trans. Inf. Theory*, vol. 67, no. 5, pp. 2758–2785, May 2021.

[12] A. M. Subramaniam, A. Heidarzadeh, and K. R. Narayanan, "Random Khatri-Rao-product codes for numerically-stable distributed matrix multiplication," in *Proc. Annu. Allerton Conf. Commun. Control Comput.*, Sep. 2019, pp. 253–259.

[13] "SuiteSparse matrix collection." Accessed: Sep. 9, 2023. [Online]. Available: https://sparse.tamu.edu/

[14] M. Xhemrishi, R. Bitar, and A. Wachter-Zeh, "Distributed matrix-vector multiplication with sparsity and privacy guarantees," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2022, pp. 1028–1033.

[15] J. Li and C. Hollanti, "Private and secure distributed matrix multiplication schemes for replicated or MDS-coded servers," *IEEE Trans. Inf. Forensics Security*, vol. 17, pp. 659–669, 2022.

[16] M. Aliasgari, O. Simeone, and J. Kliewer, "Private and secure distributed matrix multiplication with flexible communication load," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 2722–2734, 2020.

[17] W.-T. Chang and R. Tandon, "On the capacity of secure distributed matrix multiplication," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2018, pp. 1–6.

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

[19] A. B. Das and A. Ramamoorthy, "Distributed matrix-vector multiplication: A convolutional coding approach," in *Proc. IEEE Intl. Symp. Inf. Theory*, 2019, pp. 3022–3026.

[20] 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 Signal Process. Mag.*, vol. 37, no. 3, pp. 136–145, May 2020.

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

[22] B. Hasırcıoğlu, J. Gómez-Vilardebó, and D. Gündüz, "Bivariate polynomial coding for efficient distributed matrix multiplication," *IEEE J. Sel. Area. Inf. Theory*, vol. 2, no. 3, pp. 814–829, Sep. 2021.

[23] A. Ramamoorthy and L. Tang, "Numerically stable coded matrix computations via circulant and rotation matrix embeddings," *IEEE Trans. Inf. Theory*, vol. 68, no. 4, pp. 2684–2703, Apr. 2022.

[24] S. Wang, J. Liu, and N. Shroff, "Coded sparse matrix multiplication," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2018, pp. 5152–5160.
[25] R. Ji, A. Heidarzadeh, and K. R. Narayanan, "Sparse random Khatri-Rao product codes for distributed matrix multiplication," in *Proc. IEEE Inf. Theory Workshop*, 2022, pp. 416–421.
[26] A. Reisizadeh, S. Prakash, R. Pedarsani, and A. S. Avestimehr, "Coded computation over heterogeneous clusters," *IEEE Trans. Inf. Theory*, vol. 65, no. 7, pp. 4227–4242, Jul. 2019.
[27] S. Kianidehkordi, N. Ferdinand, and S. C. Draper, "Hierarchical coded matrix multiplication," *IEEE Trans. Inf. Theory*, vol. 67, no. 2, pp. 726–754, Feb. 2021.
[28] M. Hall, *Combinatorial Theory*. Hoboken, NJ, USA: Wiley, 1986.
[29] J. T. Schwartz, "Fast probabilistic algorithms for verification of polynomial identities," *J. ACM*, vol. 27, no. 4, pp. 701–717, 1980.
[30] V. Gupta, D. Carrano, Y. Yang, V. Shankar, T. Courtade, and K. Ramchandran, "Serverless straggler mitigation using local error-correcting codes," 2020, *arXiv:2001.07490*.
[31] A. B. Das, A. Ramamoorthy, D. J. Love, and C. G. Brinton, "Distributed matrix computations with low-weight encodings," 2023, *arXiv:2301.12685*.
[32] S. Prakash et al., "Coded computing for low-latency federated learning over wireless edge networks," *IEEE J. Sel. Area. Commun.*, vol. 39, no. 1, pp. 233–250, Jan. 2021.
[33] E. Ozfatura, S. Ulukus, and D. Gündüz, "Coded distributed computing with partial recovery," *IEEE Trans. Inf. Theory*, vol. 68, no. 3, pp. 1945–1959, Mar. 2022.
[34] E. Ozfatura, B. Buyukates, D. Gündüz, and S. Ulukus, "Age-based coded computation for bias reduction in distributed learning," in *Proc. IEEE Glob. Comm. Conf. (GLOBECOM)*, 2020, pp. 1–6.

Anindya Bijoy Das (Member, IEEE) received the B.Sc. degree from the Department of Electrical and Electronic Engineering, Bangladesh University of Engineering and Technology, Dhaka, Bangladesh, in 2014, and the M.Engg. and Ph.D. degrees from the Department of Electrical and Computer Engineering, Iowa State University, Ames, IA, USA, in 2018 and 2022, respectively. He is currently a Postdoctoral Researcher with the Department of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, USA. His research interests include federated learning, coding theory, and machine learning. He is the recipient of the Karas Award 2022 in Iowa State University for his outstanding thesis in the area of mathematical and physical sciences and engineering. He also received research excellence and teaching excellence awards from the Department of Electrical and Computer Engineering, Iowa State University.

Aditya Ramamoorthy (Senior Member, IEEE) received the B.Tech. degree in electrical engineering from the Indian Institute of Technology Delhi and the M.S. and Ph.D. degrees from the University of California at Los Angeles. He is a Professor of Electrical and Computer Engineering and (by courtesy) of Mathematics with Iowa State University. His research interests are in the areas of classical/quantum information theory and coding techniques with applications to distributed computation, content distribution networks, and machine learning. He is the recipient of the 2020 Mid-Career Achievement in Research Award, the 2019 Boast-Nilsson Educational Impact Award, and the 2012 Early Career Engineering Faculty Research Award from Iowa State University, the 2012 NSF CAREER Award, and the Harpole-Pentair Professorship in 2009 and 2010. He served as an Editor for the IEEE TRANSACTIONS ON INFORMATION THEORY from 2016 to 2019 and the IEEE TRANSACTIONS ON COMMUNICATIONS from 2011 to 2015.

David J. Love (Fellow, IEEE) received the B.S. (with Highest Hons.), M.S.E., and Ph.D. degrees in electrical engineering from the University of Texas at Austin in 2000, 2002, and 2004, respectively. Since 2004, he has been with the Elmore Family School of Electrical and Computer Engineering, Purdue University, where he is currently the Nick Trbovich Professor of Electrical and Computer Engineering. He holds 32 issued U.S. patents. His research interests are in the design and analysis of broadband wireless communication systems, beyond-5G wireless systems, multiple-input–multipleoutput (MIMO) communications, millimeter wave wireless, software-defined radios and wireless networks, coding theory, and MIMO array processing. He was a member of the Executive Committee for the National Spectrum Consortium.

Dr. Love won the best paper awards from the IEEE Communications Society (2016 Stephen O. Rice Prize and 2020 Fred W. Ellersick Prize), the IEEE Signal Processing Society (2015 IEEE Signal Processing Society Best Paper Award), and the IEEE Vehicular Technology Society (2010 Jack Neubauer Memorial Award), along with his coauthors. He is a Fellow of the American Association for the Advancement of Science and was named a Thomson Reuters Highly Cited Researcher in 2014 and 2015.

Christopher G. Brinton (Senior Member, IEEE) received the Ph.D. degree in electrical engineering from Princeton University in 2016. He is an Assistant Professor of Electrical and Computer Engineering with Purdue University. He is the recipient of the NSF CAREER Award, the ONR Young Investigator Program Award, the DARPA Young Faculty Award, and the Intel Rising Star Faculty Award.