Coded Sparse Matrix Computation Schemes That Leverage Partial Stragglers

Anindya Bijoy Das, *Graduate Student Member, IEEE*, and Aditya Ramamoorthy, *Senior Member, IEEE*

***Abstract*— Distributed matrix computations over large clusters can suffer from the problem of slow or failed worker nodes (called stragglers) which can dominate the overall job execution time. Coded computation utilizes concepts from erasure coding to mitigate the effect of stragglers by running "coded" copies of tasks comprising a job; stragglers are typically treated as erasures. While this is useful, there are issues with applying, e.g., MDS codes in a straightforward manner. Several practical matrix computation scenarios involve sparse matrices. MDS codes typically require dense linear combinations of submatrices of the original matrices which destroy their inherent sparsity. This is problematic as it results in significantly higher worker computation times. Moreover, treating slow nodes as erasures ignores the potentially useful partial computations performed by them. Furthermore, some MDS techniques also suffer from significant numerical stability issues. In this work we present schemes that allow us to leverage partial computation by stragglers while imposing constraints on the level of coding that is required in generating the encoded submatrices. This significantly reduces the worker computation time as compared to previous approaches and results in improved numerical stability in the decoding process. Exhaustive numerical experiments on Amazon Web Services (AWS) clusters support our findings.**

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

I. Introduction

DISTRIBUTED computation plays a major role in several problems in machine learning. For example, large scale matrix-vector multiplication is repeatedly used in gradient descent which in turn plays a key role in high dimensional machine learning problems. The size of the underlying matrices makes it impractical to perform the computation on a single computer (both from a speed and a storage perspective). Thus, the computation is typically subdivided into smaller tasks that are run in parallel across multiple worker nodes.

In these systems the overall execution time is typically dominated by the speed of the slowest worker. Thus, the presence of stragglers (as slow or failed workers are called) can negatively impact the performance of distributed computation. In recent years, techniques from coding theory (especially maximum-distance-separable (MDS) codes) [1]–[4] have been used to mitigate the effect of stragglers for problems such as matrix-vector and matrix-matrix multiplication. For instance, the work of [1] proposes to partition the computation of $\mathbf{A}^T\mathbf{x}$ by first splitting $\mathbf{A} = [\mathbf{A}_0 \mid \mathbf{A}_1]$ into two block-columns (with an equal number of column vectors) and assigning three workers, the task 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. Evidently, the computational load on each node is half of the original job. Furthermore, it is easy to see that $\mathbf{A}^T\mathbf{x}$ can be recovered as soon as any two workers complete their tasks (with some minimal post-processing). Thus, this system is resilient to one straggler. The work of [3], poses the multiplication of two matrices in a form that is roughly equivalent to a Reed-Solomon code. In particular, each worker node's task (which is multiplying smaller submatrices) can be imagined as a coded symbol. As long as enough tasks are complete, the master node can recover the matrix product by polynomial interpolation.

For such coded computing systems we can define a so-called recovery threshold. It is the minimum value of τ, such that the master node can recover the result as long as *any* τ workers complete their tasks. Thus, at the top level, in these systems stragglers are treated as the equivalent of erasures in coding theory, i.e., the assumption is that no useful information can be obtained from the stragglers.

While these are interesting ideas, there are certain issues that are ignored in the majority of prior work (see [5]–[9] for some exceptions). Firstly, several practical cases of matrix-vector or matrix-matrix multiplication involve sparse matrices. Using MDS coding strategies in a straightforward manner will often destroy the sparsity of the matrices being processed by the worker nodes. In fact, as noted in [7], this can cause the overall job execution time to actually go up rather than down. Secondly, in the distributed computation setting, we make the observation that it is possible to leverage partial computations that are performed by the stragglers. Thus, a slow worker may not necessarily be a useless worker. Fig. 1 (which also appears in [10]) shows the variation of speed of different `t2.micro` machines in AWS (Amazon Web Services) cluster, and it can be seen that for a particular job, even the slowest worker node may have approximately $60\% - 70\%$ of the speed of the fastest worker.

In this work we propose schemes which are not only resilient to full stragglers, but also can exploit slow workers

Manuscript received December 10, 2020; revised January 22, 2022; accepted February 1, 2022. Date of publication February 18, 2022; date of current version May 20, 2022. This work was supported in part by the National Science Foundation (NSF) under Grant CCF-1718470, Grant CCF-1910840, and Grant CCF-2115200. An earlier version of this paper was presented in part at the 2021 IEEE International Symposium on Information Theory [DOI: 10.1109/ISIT45174.2021.9518059] and in part at the 2018 IEEE Information Theory Workshop [DOI: 10.1109/ITW.2018.8613321]. *(Corresponding author: Aditya Ramamoorthy.)*

The authors are with the Department of Electrical and Computer Engineering, Iowa State University, Ames, IA 50011 USA (e-mail: abd149@iastate.edu; adityar@iastate.edu).

Communicated by C. Hollanti, Associate Editor for Coding Theory.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TIT.2022.3152827.

Digital Object Identifier 10.1109/TIT.2022.3152827

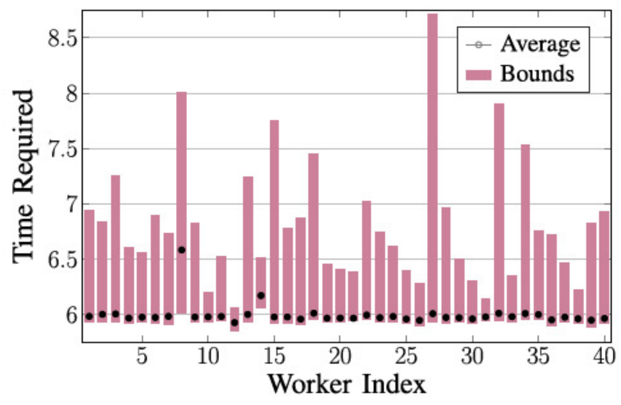

Fig. 1. Variation of worker speeds for the same job over 100 runs across 40 workers within AWS; the job involves multiplying two random matrices of size 4000×4000 twice. The average time is shown by the small circle for each worker. The upper and lower edges indicate the maximum and minimum time over the 100 runs. The required time exhibits a wide variation from 5.85 seconds to 8.71 seconds.

by utilizing their partially finished tasks. The works in [11] and [12] also address this issue but they are applicable only for matrix-vector multiplication whereas in this work, we propose schemes for matrix-matrix multiplication too. Furthermore, in several of our schemes we can specify the number of block-columns of the individual $\mathbf{A}$ and $\mathbf{B}$ matrices that are linearly combined to arrive at the encoded matrices. This is especially useful in the case of sparse matrices ($\mathbf{A}$ and $\mathbf{B}$) that often appear in practical settings. Thus, in short, our proposed approaches can leverage the partial computations of the stragglers and exploit the sparsity of the input matrices, both of which can enhance the overall speed of the whole system.

This paper is organized as follows. Section II describes the background and related work and summarizes the contributions of our work. Section III outlines some basic definitions and observations which are required for the subsequent presentation. Section IV discusses our proposed β-level coding schemes which constrain the level of coding in the encoded submatrices while leveraging partial computations. Following this, Section V proposes schemes for both matrix-vector and matrix-matrix multiplication which can be optimal in terms of resilience to full stragglers and can improve the utilization of the partial stragglers. Section VI discusses the experimental performance of our proposed methods and shows the comparison with other available approaches. We conclude the paper with a discussion about future work in Section VII.

II. Background and Related Work

Consider the case where a master node has a matrix $\mathbf{A}$ and either a matrix $\mathbf{B}$ or a vector $\mathbf{x}$ and needs to compute either $\mathbf{A}^T\mathbf{B}$ or $\mathbf{A}^T\mathbf{x}$. The computation needs to be carried out in a distributed fashion over n worker nodes. Each worker receives the equivalent of a certain fraction (denoted by γ_A and γ_B, respectively) of the columns of $\mathbf{A}$ and $\mathbf{B}$ or the whole vector $\mathbf{x}$. The node is responsible for computing its assigned submatrix-submatrix or submatrix-vector products.

We discuss the matrix-matrix scenario below where each worker node receives coded versions of submatrices of $\mathbf{A}$ and $\mathbf{B}$ respectively[1]. The corresponding matrix-vector case can be obtained as a special case. Consider a $p \times u$ and $p \times v$ block decomposition of $\mathbf{A}$ and $\mathbf{B}$ respectively as shown below.

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{0,0} & \dots & \mathbf{A}_{0,u-1} \\ \vdots & \ddots & \vdots \\ \mathbf{A}_{p-1,0} & \dots & \mathbf{A}_{p-1,u-1} \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} \mathbf{B}_{0,0} & \dots & \mathbf{B}_{0,v-1} \\ \vdots & \ddots & \vdots \\ \mathbf{B}_{p-1,0} & \dots & \mathbf{B}_{p-1,v-1} \end{bmatrix}. \tag{1}$$

The master node creates coded submatrices by computing appropriate scalar linear combinations of the $\mathbf{A}_{i,j}$ submatrices and respectively the $\mathbf{B}_{i,j}$ submatrices. This implies that the master node only performs scalar multiplications and additions. It is not responsible for any of the computationally intensive matrix operations. Following this, it sends the corresponding coded submatrices to each of the workers who perform the matrix operations.

In this work we only consider a decomposition of $\mathbf{A}$ and $\mathbf{B}$ into block-columns, which indicates that we set $p = 1$ in (1). Next we partition the columns of $\mathbf{A}$ and $\mathbf{B}$ into Δ_A and Δ_B block-columns, thus comparing with (1), we can say $u = \Delta_A$ and $v = \Delta_B$. These block columns are denoted as $\mathbf{A}_0, \mathbf{A}_1, \dots, \mathbf{A}_{\Delta_A - 1}$ and $\mathbf{B}_0, \mathbf{B}_1, \dots, \mathbf{B}_{\Delta_B - 1}$. We assume that the storage fraction γ_A (or γ_B) can be expressed as ℓ_A/Δ_A (likewise ℓ_B/Δ_B) where both ℓ_A and Δ_A (and ℓ_B and Δ_B) are integers. We also assume that $\mathbf{A}$ and $\mathbf{B}$ are large enough and satisfy divisibility constraints so that we can choose any large enough value of Δ_A and Δ_B. Now each node is assigned the equivalent of ℓ_A block-columns of $\mathbf{A}$ and ℓ_B block-columns of $\mathbf{B}$. Each of those ℓ_A block-columns from $\mathbf{A}$ will be multiplied with each of the ℓ_B block-columns from $\mathbf{B}$, so a particular worker node will compute, in total, $\ell = \ell_A \ell_B$ block-products for matrix-matrix multiplication. In case of matrix-vector multiplication, the worker node will compute $\ell = \ell_A$ block products, where each of ℓ_A blocks from $\mathbf{A}$ will be multiplied with $\mathbf{x}$.

The assignment can simply be subsets of $\{\mathbf{A}_0, \mathbf{A}_1, \dots, \mathbf{A}_{\Delta_A - 1}\}$ or $\{\mathbf{B}_0, \mathbf{B}_1, \dots, \mathbf{B}_{\Delta_B - 1}\}$; in this case we call the solution "uncoded". Alternatively, the assignment can be suitably chosen functions of $\{\mathbf{A}_0, \mathbf{A}_1, \dots, \mathbf{A}_{\Delta_A - 1}\}$ or $\{\mathbf{B}_0, \mathbf{B}_1, \dots, \mathbf{B}_{\Delta_B - 1}\}$; in this case we call the solution "coded". The assignment also specifies a sequential order from top to bottom in which each worker node needs to process its tasks. This implies that if a node is currently processing the i-th assignment ($0 \leq i \leq \ell - 1$), then it has already processed assignments 0 through $i - 1$. In this work, we assume that each time a node computes a product, it transmits the result to the master node. As we shall show, the processing order matters in this problem.

[1]A general formulation need not restrict the assignment to coded submatrices of $\mathbf{A}$ and $\mathbf{B}$. Nevertheless, all known schemes thus far and our proposed schemes work with equal-sized submatrices, so we present the formulation in this way.

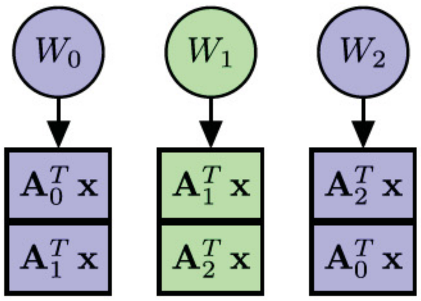
W_0
W_1
W_2
$\mathbf{A}_0^T\mathbf{x}$
$\mathbf{A}_1^T\mathbf{x}$
$\mathbf{A}_1^T\mathbf{x}$
$\mathbf{A}_2^T\mathbf{x}$
$\mathbf{A}_2^T\mathbf{x}$
$\mathbf{A}_0^T\mathbf{x}$

Fig. 2. Matrix $\mathbf{A}$ is partitioned into three submatrices. Each worker is assigned two of those uncoded submatrices. Here $Q = 4$.

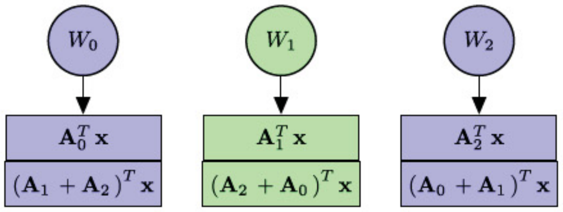
W_0
W_1
W_2
$\mathbf{A}_0^T\mathbf{x}$
$(\mathbf{A}_1+\mathbf{A}_2)^T\mathbf{x}$
$\mathbf{A}_1^T\mathbf{x}$
$(\mathbf{A}_2+\mathbf{A}_0)^T\mathbf{x}$
$\mathbf{A}_2^T\mathbf{x}$
$(\mathbf{A}_0+\mathbf{A}_1)^T\mathbf{x}$

Fig. 3. Matrix $\mathbf{A}$ is partitioned into three submatrices. Each worker is assigned one uncoded and one coded task. Here $Q = 3$.

There are two requirements that our system needs to have. The master node should be able to decode the intended result ($\mathbf{A}^T\mathbf{B}$ or $\mathbf{A}^T\mathbf{x}$) from any $n - s$ workers for s as large as possible. i.e., $n-s$ is the recovery threshold of the scheme [3]. The second requirement is that the master node should be able to recover $\mathbf{A}^T\mathbf{B}$ or $\mathbf{A}^T\mathbf{x}$ as long as it receives *any* Q products from the worker nodes. This formulation subsumes treating stragglers as non-working nodes. To our best knowledge, this second requirement has not been examined systematically within the coded computation literature, even though it is a natural constraint that allows for succinct treatment of recovery in distributed computing clusters where the workers have differing speeds.

Example 1: Consider a system with $n = 3$ worker nodes with $\gamma_A = 2/3$. We partition $\mathbf{A}$ into $\Delta_A = 3$ block-columns and the assignment of block-columns to each node is shown in Fig. 2 (this is an uncoded solution). We emphasize that the order of the computation also matters here, i.e., worker node W_0 (for example) computes $\mathbf{A}_0^T\mathbf{x}$ first and then $\mathbf{A}_1^T\mathbf{x}$. For the specific assignment it is clear that the computation is successful as long as any four block products are returned by the workers. Thus, for this system $Q = 4$.

On the other hand, Fig. 3 demonstrates a coded solution, where the bottom assignment in the workers are some suitably chosen functions of the elements of $\{\mathbf{A}_0^T\mathbf{x}, \mathbf{A}_1^T\mathbf{x}, \mathbf{A}_2^T\mathbf{x}\}$. For this assignment, it is obvious that the master node can recover $\mathbf{A}^T\mathbf{x}$ as long as any three block products are returned by the workers, so in this system $Q = 3$.

For any time t, we let $w_i(t)$ represent the state of computation of the i-th worker node, i.e., $w_i(t)$ is a non-negative integer such that $0 \leq w_i(t) \leq \ell$ which represents the number of tasks that have been processed by worker node i. Thus, our system requirement states as long as $\sum_{i=0}^{n-1} w_i(t) \geq Q$, the master node should be able to determine $\mathbf{A}^T\mathbf{B}$ or $\mathbf{A}^T\mathbf{x}$. As Δ, the number of unknowns to be recovered, is a parameter that can be chosen, our objective is to minimize the value of Q/Δ for such a system where we note that $Q \geq \Delta$. For matrix-vector multiplication, $\Delta = \Delta_A$, whereas for matrix-matrix multiplication, $\Delta = \Delta_A\Delta_B$. This formulation minimizes the worst case overall computation performed by the worker nodes.

A. Related Work

Several coded computation schemes have been proposed for matrix multiplication [1]–[3], [5]–[7], [11], [13]–[15], most of which are designed to mitigate full stragglers; see [16] for a tutorial overview. We illustrate the basic idea below using the polynomial code approach of [3] for a system with $n = 5$ workers where each of these worker nodes can store $\gamma_A = \frac{1}{2}$ fraction of matrix $\mathbf{A}$ and $\gamma_B = \frac{1}{2}$ fraction of matrix $\mathbf{B}$. Consider $u = v = 2$ and $p = 1$, thus we partition both $\mathbf{A}$ and $\mathbf{B}$ into two block-columns $\mathbf{A}_0, \mathbf{A}_1$ and $\mathbf{B}_0, \mathbf{B}_1$ respectively. Next, we define two matrix polynomials as

$$\mathbf{A}(z) = \mathbf{A}_0 + \mathbf{A}_1 z \quad \text{and} \quad \mathbf{B}(z) = \mathbf{B}_0 + \mathbf{B}_1 z^2;$$
$$\text{so } \mathbf{A}^T(z)\mathbf{B}(z) = \mathbf{A}_0^T\mathbf{B}_0 + \mathbf{A}_1^T\mathbf{B}_0 z + \mathbf{A}_0^T\mathbf{B}_1 z^2 + \mathbf{A}_1^T\mathbf{B}_1 z^3.$$

The master node evaluates these polynomial $\mathbf{A}(z)$ and $\mathbf{B}(z)$ at distinct real values $z_0, z_1, \ldots, z_{n-1}$, and sends the corresponding matrices to worker node W_i. Each worker node computes the product of its assigned submatrices. It follows that decoding at the master node is equivalent to decoding a degree-3 real-valued polynomial. Thus, the master node can recover $\mathbf{A}^T\mathbf{B}$ as soon as it receives the results from *any* four workers, i.e., in this example, the recovery threshold is, $\tau = 4$. When $\gamma_A = 1/k_A$ and $\gamma_B = 1/k_B$ and $p = 1$, the work of [13] shows that their scheme has a threshold $\tau = k_A k_B$ which is optimal. Random coding solutions for this problem were investigated in [17]. Approaches based on convolutional coding were presented in [10], [18]. In these schemes (analogous to linear block codes) there are systematic workers that only contain uncoded assignments and parity workers that contain coded assignments.

The case when $p > 1$ was considered in the work of [2], [13]–[15]. Structuring the computation in this manner increases the computational load on the workers and the communication load from the workers to the master node but can reduce the recovery threshold as compared to the case of $p = 1$.

It is well-recognized that in several practical situations the underlying matrices $\mathbf{A}$ and $\mathbf{B}$ are sparse. Computing the inner product $\mathbf{a}^T\mathbf{x}$ of n-length vectors $\mathbf{a}$ and $\mathbf{x}$ where $\mathbf{a}$ has around δn $(0 < \delta \ll 1)$ non-zero entries takes $\approx 2\delta n$ floating point operations (flops) as compared to $\approx 2\ n$ flops in the dense case. In general, the encoding process within coded computation increases the number of non-zero entries in the resultant encoded matrices. For instance, polynomial evaluations of degree d will increase the number of non-zero entries by approximately d times. This results in a d-fold increase in the worker computation times which can be unacceptably high. Thus, it is important to consider schemes

TABLE I
COMPUTATION TIME FOR SPARSE MATRIX MULTIPLICATION. WE CHOOSE MATRICES $\mathbf{A}$ AND $\mathbf{B}$ BOTH OF SIZE $10,000 \times 10,000$. BOTH OF THEM HAVE SPARSITY $\sigma = 3\%$, THUS RANDOMLY CHOSEN 3% ENTRIES OF $\mathbf{A}$ AND $\mathbf{B}$ ARE NON-ZERO

JOB	REQUIRED TIME
TO COMPUTE $\mathbf{A}^T\mathbf{B}$	9.41 SECONDS
TO COMPUTE (UNCODED) $\mathbf{A}_i^T\mathbf{B}_j$	0.58 SECONDS
TO COMPUTE (CODED) $\tilde{\mathbf{A}}_i^T\tilde{\mathbf{B}}_j$	2.13 SECONDS

where the encoding only combines a limited number of submatrices.

Example 2: Consider an example with two large sparse matrices $\mathbf{A}$ and $\mathbf{B}$ both of whose sizes are $10,000 \times 10,000$. Both of them have sparsity $\sigma = 3\%$, i.e., randomly chosen approximately 3% entries of $\mathbf{A}$ and $\mathbf{B}$ are non-zero (we have used `MATLAB` command `sprand` for this example). We partition matrices $\mathbf{A}$ and $\mathbf{B}$ into 4 and 5 block-columns, respectively. First we choose a block-column $\mathbf{A}_i$ and a block-column $\mathbf{B}_j$, and next we obtain two coded submatrices $\tilde{\mathbf{A}}_i$ and $\tilde{\mathbf{B}}_j$ which are random linear combinations of the uncoded block-columns of $\mathbf{A}$ and $\mathbf{B}$, respectively. Table I shows that it is around 4 times more expensive to compute the coded product than the uncoded product, although the sizes of the corresponding matrices are exactly the same. The reason is that the number of non-zero entries in the coded submatrices have gone up significantly.

An important aspect of coded computation is "numerical stability" of the recovered result. Indeed, while coded computation borrows techniques from classical coding theory (over finite fields), it differs in the sense that the coded submatrices and the decoding operates over the reals. Over finite fields, the invertibility of a matrix is sufficient to solve a system of equations. In contrast, over the reals if the corresponding matrix is ill-conditioned, then the recovery will in general be inaccurate. It is well-recognized that real Vandermonde matrices corresponding to polynomial interpolation have condition numbers that grow exponentially in the matrix sizes. This is a serious issue with the polynomial-based approaches of [3], [19]. There have been some works that have addressed these issues [10], [12], [17], [20]–[23] in part.

Yet another feature of the coded computation problem that distinguishes it from classical codes is the processing order. The worker nodes process the assigned tasks in a specific order, such that if a worker node is processing a given task, it has already completed the previously assigned tasks. Thus, at any given time the pattern of tasks that have been completed is restricted. Interestingly, codes for such systems have been investigated in [24], [25]. These ideas were adapted for the distributed matrix-vector multiplication problem in [11] and [12].

We note here that in principle using polynomial approaches can allow us to address both the optimal threshold and the optimal $Q/\Delta = 1$ by simply placing multiple evaluations of the polynomials at distinct points within each worker node. However, this approach is not practical, firstly because

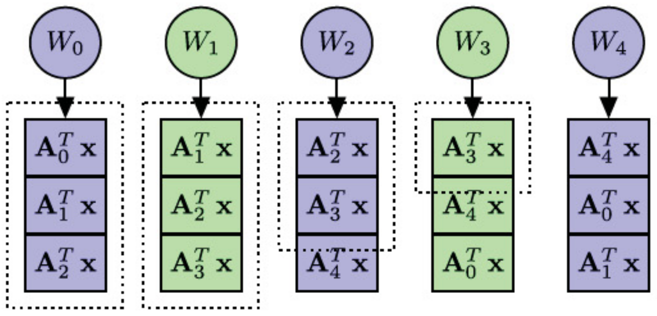
W_0
W_1
W_2
W_3
W_4
$\mathbf{A}_0^T\mathbf{x}$
$\mathbf{A}_1^T\mathbf{x}$
$\mathbf{A}_2^T\mathbf{x}$
$\mathbf{A}_1^T\mathbf{x}$
$\mathbf{A}_2^T\mathbf{x}$
$\mathbf{A}_3^T\mathbf{x}$
$\mathbf{A}_2^T\mathbf{x}$
$\mathbf{A}_3^T\mathbf{x}$
$\mathbf{A}_4^T\mathbf{x}$
$\mathbf{A}_3^T\mathbf{x}$
$\mathbf{A}_4^T\mathbf{x}$
$\mathbf{A}_0^T\mathbf{x}$
$\mathbf{A}_4^T\mathbf{x}$
$\mathbf{A}_0^T\mathbf{x}$
$\mathbf{A}_1^T\mathbf{x}$

Fig. 4. Partitioning matrix $\mathbf{A}$ into five submatrices and assigning three uncoded tasks in a cyclic fashion to the workers. The system is resilient to two stragglers and $Q = 10$. The tasks enclosed in dots can be processed without processing any copy of $\mathbf{A}_4^T\mathbf{x}$.

of numerical stability issues. Secondly, as discussed above when considering sparse $\mathbf{A}$ and $\mathbf{B}$ matrices, the polynomial approaches result in dense coded submatrices which can cause an unacceptable increase in the worker node computation times. Numerical experiments supporting these conclusions can be found in Section VI.

B. Summary of Contributions

The contributions of our work can be summarized as follows.

- We present a fine-grained model of the distributed matrix-vector and matrix-matrix multiplication that allows us to (i) leverage the slower workers using their partial computations and (ii) impose constraints on what extent coding is allowed in the solution. This allows us to capture a scenario where workers have differing speeds and the intended result can be recovered as long as the workers together complete a minimum number (Q) of the assigned tasks. This applies to the practically important case where the underlying matrices are sparse. The formulation leads to new questions within coded computing that to our best knowledge have not been investigated before systematically within the coded computing literature.
- We present systematic methods for both matrix-vector and matrix-matrix multiplication that address both the recovery threshold and the Q/Δ metric. For the uncoded assignment case, we present a lower bound on the performance of any scheme that our constructions are able to match.
- We have proposed two different schemes for distributed computations, first of which is named as β-level coding. In this approach, we have used resolvable combinatorial designs [26] to improve the recovery threshold and the Q/Δ metric over the uncoded approach. We have shown that the metrics can be further improved if we utilize certain relations among the blocks of different parallel classes within the resolvable designs.
- Prior work has demonstrated schemes with the optimal recovery threshold for certain storage fractions. In this work we present novel schemes that retain the optimal recovery threshold and also have low Q/Δ values.

TABLE II

COMPARISON BETWEEN OUR PROPOSED APPROACHES. HERE γ_A AND γ_B INDICATE THE STORAGE FRACTIONS FOR MATRICES $\mathbf{A}$ AND $\mathbf{B}$, RESPECTIVELY, AND n DENOTES THE TOTAL NUMBER OF WORKERS

APPROACH	PROPERTIES	PARAMETER REGIME	ADVANTAGES
β-LEVEL CODING	COMBINE β SUBMATRICES OF $\mathbf{A}$ (OR $\mathbf{B}$)	FOR $\gamma_A = \frac{a_1}{a_2}$ AND $\gamma_B = \frac{b_1}{b_2}$, WE NEED $n = ca_2b_2$	ASSIGNED SUBMATRICES ARE SPARSE
SCS OPTIMAL SCHEME	MAJORITY OF ASSIGNED SUBMATRICES ARE UNCODED	FOR $\gamma_A = \frac{1}{k_A}$ AND $\gamma_B = \frac{1}{k_B}$, WHERE k_A AND k_B ARE INTEGERS	OPTIMAL RECOVERY THRESHOLD (k_Ak_B)

TABLE III

COMPARISON WITH EXISTING WORKS

CODES	MAT-MAT MULT?	OPTIMAL THRESHOLD?	NUMERICAL STABILITY?	PARTIAL COMPUT?	SPARSELY CODED?
REPETITION CODES	✓	✗	✓	✗	✓
RATELESS CODES [6]	✗	✗	✓	✗	✗
PROD. CODES [27], FACTORED CODES [23]	✓	✗	✓	✗	✗
POLYNOMIAL CODES [3]	✓	✓	✗	✗	✗
BIV. HERMITIAN POLY. CODE [32]	✓	✓	✗	✓	✗
DYNAMIC HETERO.-AWARE CODE [30]	✗	✗	✓	✓	✗
ORTHOPOLY [21], RKRP CODE[17]	✓	✓	✓	✗	✗
CONV. CODE [10], CIRC. & ROT. MAT. [20]	✓	✓	✓	✗	✗
C^3LES [11]	✗	✗	✓	✓	✓
β-level Coding (proposed)	✓	✗	✓	✓	✓
SCS Optimal Scheme (proposed)	✓	✓	✓	✓	✓

- Finally, we present exhaustive experimental comparisons that demonstrate the benefit of our schemes while considering sparse matrices in terms of worker node computation times and numerical stability.

In Table II, we present a summary of the properties and the advantages of both of our proposed approaches, β-level coding and sparsely coded straggler (SCS) optimal scheme. Moreover, a detailed comparison of the properties of our methods with other available schemes is demonstrated in Table III.

It should be noted that there are other issues within various coded matrix computation approaches. Several approaches [6], [7], [27] are sub-optimal in terms of straggler resilience. Some other works [28], [29] have considered the issue of private computation along with the straggler mitigation issue. Here the goal is that no information about the matrices $\mathbf{A}$ or $\mathbf{B}$ can be obtained from any set of at most m workers. Another class of codes [30], [31] assumes the workers to be heterogeneous and time-varying, so that the system may have access to different number of workers at different moments where the workers may have different speeds and/or different storage capacities. These issues are out of the scope of this paper.

III. PRELIMINARIES

In this section we discuss some basic facts and observations that serve to explain our proposed distributed matrix computation schemes. Suppose that a given worker node is assigned encoded block-columns $\tilde{\mathbf{A}}_i, i = 0, 1, \ldots, \ell_A - 1$ and $\tilde{\mathbf{B}}_j, j = 0, 1, \ldots, \ell_B - 1$. The assignment also specifies a top to bottom order. For the matrix-vector problem, the node processes them simply in the order $\tilde{\mathbf{A}}_0^T\mathbf{x}, \tilde{\mathbf{A}}_1^T\mathbf{x}, \ldots, \tilde{\mathbf{A}}_{\ell_A-1}^T\mathbf{x}$. On the other hand for the matrix-matrix problem the node computes in the order $\tilde{\mathbf{A}}_0^T\tilde{\mathbf{B}}_0, \tilde{\mathbf{A}}_0^T\tilde{\mathbf{B}}_1, \ldots, \tilde{\mathbf{A}}_0^T\tilde{\mathbf{B}}_{\ell_B-1}, \tilde{\mathbf{A}}_1^T\tilde{\mathbf{B}}_0, \ldots, \tilde{\mathbf{A}}_1^T\tilde{\mathbf{B}}_{\ell_B-1}, \ldots, \tilde{\mathbf{A}}_{\ell_A-1}^T\tilde{\mathbf{B}}_0, \ldots, \tilde{\mathbf{A}}_{\ell_A-1}^T\tilde{\mathbf{B}}_{\ell_B-1}$.

Definition 1: A coding scheme for distributed matrix computation is said to be a β-level coding scheme if the assigned block-columns are a linear combination of exactly β block-columns of $\mathbf{A}$ and $\mathbf{B}$. The case of $\beta = 1$ represents an uncoded scheme.

Our constructions leverage the properties of combinatorial structures known as resolvable designs [26]. These have also been used in different contexts in [33]–[35].

Definition 2: [26] A resolvable design is a pair $(\mathcal{X}, \mathcal{A})$ where $\mathcal{X}$ is a set of elements (called points) and $\mathcal{A}$ is a family of non-empty subsets of $\mathcal{X}$ (called blocks) that have the same cardinality. A subset $\mathcal{P} \subset \mathcal{A}$ in a design $(\mathcal{X}, \mathcal{A})$ is called a parallel class if $\cup_{\{i:\mathcal{A}_i \in \mathcal{P}\}}\mathcal{A}_i = \mathcal{X}$ and if $\mathcal{A}_i \cap \mathcal{A}_j = \emptyset$ for $\mathcal{A}_i, \mathcal{A}_j \in \mathcal{P}$ when $i \neq j$. A partition of $\mathcal{A}$ into several parallel classes is called a resolution and $(\mathcal{X}, \mathcal{A})$ is said to be a resolvable design if $\mathcal{A}$ has at least one resolution.

A resolvable design always exists if the cardinality of a block divides $|\mathcal{X}|$.

Example 3: Let $\mathcal{X} = \{0, 1, 2, 3\}$ and $\mathcal{A} = \{\{0, 1\}, \{0, 2\}, \{0, 3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}\}$. Now $(\mathcal{X}, \mathcal{A})$ forms a resolvable design with parallel classes, $\mathcal{P}_0 = \{\{0, 1\}, \{2, 3\}\}$, $\mathcal{P}_1 = \{\{0, 2\}, \{1, 3\}\}$ and $\mathcal{P}_2 = \{\{0, 3\}, \{1, 2\}\}$.

We note that the specification of the "incidence relations" between the points and blocks of a design can also be shown by means of an incidence matrix.

Definition 3: The incidence matrix $\mathcal{N}$ of a design $(\mathcal{X}, \mathcal{A})$ is a $|\mathcal{X}| \times |\mathcal{A}|$ binary matrix such that the (i, j)-th entry is a 1 if the i-th point is a member of the j-th block and zero, otherwise.

For example, the incidence matrix for the resolvable design in Example 3 is given by

$$\mathcal{N} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}.$$

We will use a cyclic assignment of tasks extensively in our constructions. We illustrate this by means of the following matrix-vector multiplication example.

Example 4: Consider an example of computing $\mathbf{A}^T\mathbf{x}$, where we have $n = 5$ workers and each worker can process $\gamma = 3/5$ fraction of the total job. We partition $\mathbf{A}$ into $\Delta = 5$ block-columns: $\mathbf{A}_0, \mathbf{A}_1, \ldots, \mathbf{A}_4$. Let $\mathcal{X} = \{0, 1, 2, 3, 4\}$. If we do not incorporate any coding among the block-columns, then for $\beta = 1$, we have the trivial parallel class $\mathcal{P} = \{\{0\}, \{1\}, \ldots, \{4\}\}$. Fig. 4 shows a cyclic assignment of jobs where three uncoded submatrices are allocated to each of the workers in a cyclic fashion according to the indices of three elements of $\mathcal{P}$. It can be easily verified that the system is resilient to $s = 2$ stragglers. In the sequel, our assignment can be coded as well.

More generally, suppose that we have Δ symbols denoted $0, \ldots, \Delta - 1$, $n = \Delta$ worker nodes and ℓ symbols to be placed in each worker node where $\ell \leq \Delta$. The symbols can be encoded block-columns of $\mathbf{A}$ or the product of encoded block-columns of $\mathbf{A}$ and $\mathbf{B}$. A cyclic assignment in this case assigns the set $\{j, j+1, \ldots, j+\ell-1\} \pmod{\Delta}$ to worker W_j; symbol j appears at the top and sequentially symbol $(j+\ell-1)$ (the values are reduced modulo Δ) at the bottom. The node W_j processes the tasks specified by the symbols from top to bottom. Within a node, the position of a symbol is denoted by an integer between 0 and $\ell - 1$, where 0 denotes the top and $\ell - 1$ denotes the bottom.

Lemma 1: The cyclic assignment satisfies the following properties.

- Each symbol appears ℓ times across n worker nodes. Furthermore, it appears in each position $0, \ldots, \ell - 1$ exactly once, across all n workers.
- Let α_c be the maximum number of symbols that can be processed across all worker nodes such that a specific symbol j is processed exactly c times (where $0 \leq c \leq \ell$). Then, $\alpha_c = \Delta\ell - \frac{\ell(\ell+1)}{2} + \sum_{i=0}^{c-1}(\ell - i)$, independent of j.

Proof: The first claim follows since $\ell \leq \Delta = n$ and symbol j, where $0 \leq j \leq \Delta - 1$, appears in workers $j, j-1, j-2, \ldots, j-\ell+1$ (indices reduced modulo-Δ).

For the second claim we proceed by contradiction. Suppose that there is a symbol j for which the condition is violated. From part (a), symbol j appears once in positions $0, \ldots, \ell - 1$ across the workers. Thus, one can process at most $(\Delta - \ell)\ell + \sum_{i=0}^{\ell-1} i = \Delta\ell - \frac{\ell(\ell+1)}{2}$ symbols without processing any copy of j. Following this, any symbol processed will necessarily process symbol j. If we process the copy of j at position i, we can process another $\ell - 1 - i$ symbols without processing another copy of j. Therefore, the maximum number of symbols that can be processed such that c copies of j are processed are $\Delta\ell - \frac{\ell(\ell+1)}{2} + \sum_{i=0}^{c-1}(\ell - i)$. ∎

IV. β-Level Coding for Distributed Computations

We begin our discussion of β-level coding by considering the uncoded $\beta = 1$ case. In this scenario, the assignments are simply elements such as $\mathbf{A}_i^T\mathbf{x}$ (in the matrix-vector case) or elements such as $\mathbf{A}_i^T\mathbf{B}_j$ (in the matrix-matrix case). In the discussion below we refer the assignment of "symbols" to treat both cases together, where a symbol can either be of the form $\mathbf{A}_i^T\mathbf{x}$ or $\mathbf{A}_i^T\mathbf{B}_j$. Note that we can disregard the case when multiple copies of a symbol appear within the same worker node. Consider a $\langle n, \ell, \Delta, r\rangle$-uncoded system with n workers each of which can process $\ell \geq 1$ symbols out of a total of Δ symbols. We assume that each symbol appears r times across the different worker nodes, so $n\ell = \Delta r$. Now we show a lower bound on the value of Q for such a system.

Theorem 1: For a $\langle n, \ell, \Delta, r\rangle$-uncoded system we have $Q \geq \Delta r - \frac{r}{2}(\ell + 1) + 1$.

Proof: For the system under consideration, let Q_j represent the maximum number of symbols that are processed in the worst case without processing symbol j (see Fig. 4 for an example). It is evident in this case that $Q = \max_{j=0,\ldots,\Delta-1} Q_j + 1$.

Our strategy is to calculate the average $\overline{Q} = \frac{1}{\Delta}\sum_{j=0}^{\Delta-1} Q_j$ and use the simple bound $Q \geq \overline{Q} + 1$. Toward this end, note that for any uncoded solution, we can calculate $\sum_{j=0}^{\Delta-1} Q_j$ in a different way. For any worker i, there are ℓ assigned block-columns and the other $\Delta - \ell$ do not appear in it. Thus, in the calculation of $\sum_{j=0}^{\Delta-1} Q_j$, worker node i contributes

$$(\Delta - \ell)\ell + \sum_{k=1}^{\ell}(k - 1) \quad \text{symbols,}$$

which is clearly independent of i. Therefore,

$$\overline{Q} = n\,\frac{\left[\sum_{k=1}^{\ell}(k - 1) + (\Delta - \ell)\ell\right]}{\Delta} = n\ell - \frac{n\ell}{2\Delta}(\ell + 1).$$

Thus, we have the lower bound as

$$Q \geq \Delta r - \frac{r}{2}(\ell + 1) + 1 \tag{2}$$

since $n\ell = \Delta r$. ∎

Remark 1: In general, we are given the number of workers n and the storage fraction γ. The parameters Δ and ℓ can be treated as design parameters. In this setting, from (2), we have

$$\frac{Q}{\Delta} \geq r - \frac{r}{2}\frac{\ell}{\Delta} - \frac{r}{2\Delta} + \frac{1}{\Delta};$$

but $r = n\gamma$, thus $$\frac{Q}{\Delta} \geq n\gamma\left(1 - \frac{\gamma}{2}\right) + \left(1 - \frac{n\gamma}{2}\right)\frac{1}{\Delta}. \tag{3}$$

If $r = n\gamma > 2$, then the second term in the RHS above is negative and has an inverse dependence on Δ.

The lower bound in (2) is met with equality when we consider the cyclic assignment scheme. For instance, Fig. 4 shows an example where $\Delta = n = 5$, and it can be verified that $Q = 10$ and meets the lower bound in (2). A similar result holds for the matrix-matrix case. These results are discussed in the relevant parts of the remainder of this section.

A. Matrix-Vector Multiplication

We consider a β-level coding matrix-vector scenario where the storage fraction $\gamma = a_1/a_2$ for positive integers a_1 and a_2 with $a_1 \leq a_2$ such that $\gamma \leq \frac{1}{\beta}$. We assume that the number of worker nodes $n = ca_2$ where c is a positive integer.

We partition $\mathbf{A}$ into Δ block-columns where Δ is divisible by β. Next, we pick a resolvable design $(\mathcal{X}, \mathcal{A})$ where $\mathcal{X} = \{0, \dots, \Delta-1\}$. The size of the blocks in $\mathcal{A}$ is β. Let $\mathcal{P}_1, \mathcal{P}_2, \dots$ denote distinct parallel classes of this design. We will refer to the blocks of the design as meta-symbols (to avoid potential confusion with the term block-columns which we also have used extensively). Thus, the elements of a parallel class are meta-symbols.

The overall idea is to partition the set of worker nodes into c groups denoted $\mathcal{G}_0, \dots, \mathcal{G}_{c-1}$. For each group we pick a parallel class and place meta-symbols from the parallel class in a cyclic fashion. The parallel classes for the different groups can be the same as well. For each meta-symbol, we generate a coded block-column by choosing a random linear combination of the β block-columns within it. In the discussion below we refer to the block-columns as "unknowns" as they need to be decoded by the master nodes. A precise description appears in Algorithm 1. We illustrate it by means of an example below.

Algorithm 1 β-Level Coding Scheme for Distributed Matrix-Vector Multiplication

```
Input : Matrix A and vector x. Storage fraction
        $\gamma = \frac{a_1}{a_2} \leq \frac{1}{\beta}$, $\beta$-allowed coding level, and
        number of workers $n = ca_2$ where $c$ is a
        positive integer.
1  Set $\Delta = \beta a_2$. Partition A into $\Delta$ block-columns;
2  Number of assigned blocks per worker, $\ell = \Delta\gamma$;
3  Assume $\mathcal{X} = \{0, 1, 2, \dots, \Delta - 1\}$ and find $c$ parallel
   classes $\mathcal{P}_i$ having a block size $\beta$, $i = 0, 1, \dots, c-1$;
4  for $i \leftarrow 0$ to $c - 1$ do
5      Let the blocks of $\mathcal{P}_i$ be denoted as $p_0, p_1, \dots, p_{\frac{\Delta}{\beta}-1}$ ;
6      for $j \leftarrow 0$ to $\frac{\Delta}{\beta} - 1$ do
7          Assign meta-symbols $p_j, p_{j+1}, \dots, p_{j+\ell-1}$ from
           top to bottom (indices reduced modulo $a_2$) and
           vector x to worker $\frac{\Delta}{\beta}i + j$;
8          For each meta-symbol choose a random linear
           combination of length-$\beta$ of the constituent
           block-columns;
9      end
10 end
Output: $\beta$-level coding scheme for distributed
        matrix-vector multiplication.
```

Example 5: Consider a scenario with $n = 12$, $\gamma = 1/4$ and $\beta = 3$, and set $\Delta = 12$. We let $\mathcal{X} = \{0, 1, \dots, 11\}$ and pick $\mathcal{P} = \{\{0,1,2\}, \{3,4,5\}, \{6,7,8\}, \{9,10,11\}\}$. In this example, all three groups use the same parallel class $\mathcal{P}$. As shown in Fig. 5, in each group the meta-symbols are arranged in a cyclic fashion. For each meta-symbol a random linear combination is chosen, e.g. in worker W_0 the meta symbol $\{0,1,2\}$ will be replaced by $\tilde{\mathbf{A}}_0 = z_0\mathbf{A}_0 + z_1\mathbf{A}_1 + z_2\mathbf{A}_2$ where the z_i's are chosen at random. This implies that W_0 is responsible for computing $\tilde{\mathbf{A}}_0^T\mathbf{x}$, and the unknowns $\mathbf{A}_0^T\mathbf{x}$, $\mathbf{A}_1^T\mathbf{x}$ and $\mathbf{A}_2^T\mathbf{x}$ can be decoded if three copies of the meta-symbol $\{0,1,2\}$ are obtained from the workers as the corresponding equations are linearly independent with probability 1.

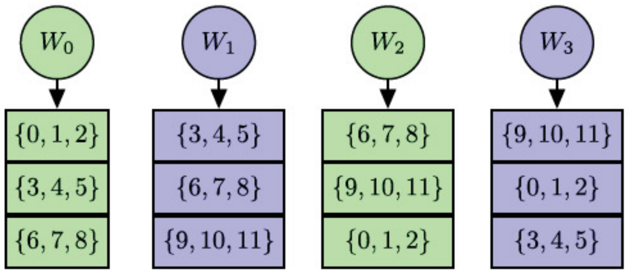

Fig. 5. Job assignment for worker group $\mathcal{G}_0$ for β-level matrix-vector multiplication scheme for $n = 12$ with $\gamma_A = \frac{1}{4}$ and $\Delta_A = 12$ using a single parallel class with $\beta = 3$. The indices $\{i, j, k\}$ indicates a random linear combination of the submatrices $\mathbf{A}_i, \mathbf{A}_j$ and $\mathbf{A}_k$. $\mathcal{G}_1$ and $\mathcal{G}_2$ are assigned the same symbols as workers $0 - 3$ but with different random coefficients.

Theorem 2: Consider a distributed matrix-vector multiplication scheme for $n = ca_2$ workers where each worker can store $\gamma = \frac{a_1}{a_2}$ fraction of matrix $\mathbf{A}$. Suppose that $c \geq \beta$ and we use the same parallel class $\mathcal{P}$ over all the worker groups. Then, the scheme described in Alg. 1 will be resilient to $s = c\ell - \beta$ stragglers, and $Q = n\ell - \frac{c\ell(\ell+1)}{2} + \ell(\beta - 1) + 1$.

Proof: Based on our construction we know that any meta-symbol $\in \mathcal{P}$ will appear in ℓ distinct workers in each worker group consisting of $\Delta/\beta = a_2$ workers (*cf.* Lemma 1). Thus there are $\frac{n}{a_2} = c$ such worker groups and it follows that there are a total of $c\ell$ appearances of that meta-symbol across all the worker nodes. Furthermore, each meta-symbol corresponds to a random linear combination of the corresponding unknowns (block-columns). As the choice of these random coefficients is made from a continuous distribution, as long as *any* β meta-symbols are processed across all the worker nodes, the constituent unknowns will be decodable with probability 1. Thus, the scheme is resilient to the failure of any $c\ell - \beta$ stragglers.

For the second claim, suppose that there exists a meta-symbol $\star \in \mathcal{P}$ that is processed at most $\beta - 1$ times when $n\ell - \frac{c\ell(\ell+1)}{2} + \ell(\beta-1) + 1$ meta-symbols have been processed. For each worker group, the meta-symbol $\star$ appears in all the positions $0, \dots, \ell - 1$. Suppose that $\star$ appears i times in η_i worker groups for $i = 1, \dots, y$. Thus, $\sum_{i=1}^{y} i\eta_i \leq \beta - 1$ and the maximum number of meta-symbols that can be processed is

$$Q' = \sum_{i=1}^{y} \eta_i\alpha_i + (c - \sum_{i=1}^{y} \eta_i)\alpha_0$$

where $\alpha_0 = \frac{\Delta}{\beta}\ell - \frac{\ell(\ell+1)}{2}$ and $\alpha_i = \alpha_0 + \sum_{j=0}^{i-1}(\ell - j) = \alpha_0 + i\ell - \frac{i(i-1)}{2}$ as specified in Lemma 1 (by setting the number of symbols to Δ/β). Thus,

$$Q' = c\alpha_0 + \ell\sum_{i=1}^{y} i\eta_i - \sum_{i=1}^{y} \eta_i \frac{i(i-1)}{2} \leq c\alpha_0 + \ell(\beta - 1)$$

(4)

since we have $\sum_{i=1}^{y} i\eta_i \leq \beta - 1$. Equality holds in (4) if we have $y = 1$ and $\eta_1 = \beta - 1$.

In the worst case therefore, we can process α_1 symbols from $\beta - 1$ groups and α_0 symbols from the remaining groups. This gives a total of

$$(\beta-1)\alpha_1 + (c-\beta+1)\alpha_0 = n\ell - \frac{c\ell(\ell+1)}{2} + \ell(\beta-1)$$

symbols, which is the same as the upper bound in (4). Thus if $Q \geq n\ell - \frac{c\ell(\ell+1)}{2} + \ell(\beta-1) + 1$ then we are guaranteed that every meta-symbol is processed at least β times. This concludes the proof. ■

It can be verified that the distributed matrix-vector multiplication scheme shown in Fig. 5 is resilient to $s = c\ell - \beta = 3 \times 3 - 3 = 6$ stragglers and has $Q = 25$. Theorem 2 provides the value for s and Q for distributed matrix-vector multiplication when $\beta \leq c$. In Appendix A, we show the calculation for s and Q for the case when $\beta > c$.

Remark 2: The proposed β-level coding scheme leads to an algorithm for uncoded matrix-vector multiplication when we set $\beta = 1$ (see Fig. 4 for an example). In that case, $Q = \Delta r - \frac{r}{2}(\ell+1) + 1$; since $\Delta = a_2$, $n = ca_2$ and $n\ell = \Delta r$. The ratio Q/Δ for the construction in Alg. 1 is lower in general as compared to the scheme in [11]. For instance, with $n = 10$ and $\gamma = 2/5$, Alg. 1 results in a scheme with $Q/\Delta = 3.0$, whereas the [11] scheme has $Q/\Delta = 3.1$. The reduction is due to the lower value of Δ (*cf.* Remark 1).

Remark 3: For $\beta > 1$ the Q/Δ ratio can be reduced significantly as compared to the uncoded ($\beta = 1$) case. To see this consider $n = ca_2$ and $\gamma = \frac{a_1}{a_2}$, where $c \geq \beta$. For the uncoded case, we set $\Delta_{unc} = a_2$, and we have $Q_{unc} = n\ell - \frac{c\ell(\ell+1)}{2} + 1$ where $\ell = a_1$. On the other hand for β-level coding, we set $\Delta_\beta = \beta a_2$, and we have $Q_\beta = n\ell - \frac{c\ell(\ell+1)}{2} + \ell(\beta-1) + 1$ where $\ell = \beta a_1$. This implies that

$$\frac{Q_{unc}}{\Delta_{unc}} - \frac{Q_\beta}{\Delta_\beta} = (\beta-1)\left(\gamma\left(\frac{ca_1}{2} - 1\right) + \frac{1}{\beta a_2}\right) > 0.$$

It turns out that the recovery threshold can be further reduced if we judiciously choose different parallel classes for the different worker groups in Alg. 1. Utilizing these parallel classes, we present a method that improves on Theorem 2 if we assume the property that the blocks among different parallel classes have intersection size to be at most one. Before stating the theorem, we discuss the decodabilty of the approach since this is not as straightforward as the single parallel class β-level coding.

To understand the decoding in this setting we consider a bipartite graph $\mathbf{G}_{dec} = \mathcal{U} \cup \mathcal{V}$ whose vertex set consists of the unknowns ($\mathcal{U}$) on the left and the processed meta-symbols ($\mathcal{V}$) on the right; an example is shown in Fig. 6. A meta-symbol is connected to its constituent unknowns. Note that $\mathbf{G}_{dec}$ specifies a system of equations in Δ unknowns and we need to argue that this system is invertible. In the argument below, suppose that the random linear coefficients of each meta-symbol are indeterminates and we argue that there exists a matching in $\mathbf{G}_{dec}$ where all the unknowns in $\mathcal{U}$ are matched.

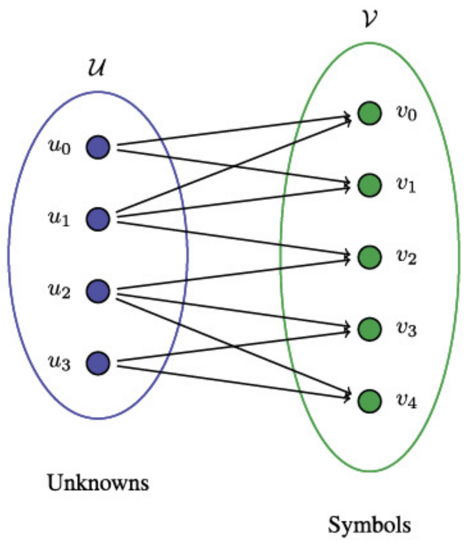

Fig. 6. For the case $\beta = 2$, every symbol is a random linear combination of two unknowns, thus there is a bipartite graph between the unknowns and symbols.

Consider a set $\tilde{\mathcal{U}}$ of d unknowns from $\mathbf{G}_{dec}$ and the corresponding neighborhood $\tilde{\mathcal{V}} = \mathcal{N}(\tilde{\mathcal{U}})$. Suppose we have a set of equations where these d unknowns, namely $u_0, u_1, \ldots, u_{d-1}$, participate in $\ell_0, \ell_1, \ldots, \ell_{d-1}$ equations. Thus the number of outgoing edges from $\tilde{\mathcal{U}}$ is $\sum_{i=0}^{d-1} \ell_i$. On the other hand, because of the structure of β-level coding approach, any symbol in $\tilde{\mathcal{V}}$ has a degree β, thus the number of incoming edges in $\tilde{\mathcal{V}}$ is $\beta|\tilde{\mathcal{V}}|$.

Now, suppose that a matching where all the elements of $\mathcal{U}$ are matched does not exist. Hall's marriage theorem [36] gives a necessary and sufficient condition for the existence of the matching. It says that there is a perfect $\mathcal{U}$-matching in $\mathbf{G}_{dec} = \mathcal{U} \cup \mathcal{V}$ if and only if $|\tilde{\mathcal{U}}| \leq |\mathcal{N}(\tilde{\mathcal{U}})|$ for every subset $\tilde{\mathcal{U}}$ of $\mathcal{U}$. Suppose that Hall's condition is violated for the set $\tilde{\mathcal{U}}$ (consisting of d unknowns), i.e., $|\tilde{\mathcal{V}}| \leq d - 1$. This means that

$$\sum_{i=0}^{d-1} \ell_i \;\leq\; \beta(d-1). \tag{5}$$

Lemma 2: Suppose that $\mathbf{G}_{dec}$ is such that each of Δ unknowns has at least degree 1 and at least $\Delta - 1$ unknowns have degree at least β each. Then, the master node can decode all the unknowns.

Proof: First, consider $d = 1$, so $|\tilde{\mathcal{U}}| = 1$. Since each unknown has at least degree 1, thus $\sum_{i=0}^{d-1} \ell_i \geq 1 > \beta(d-1)$. Next we consider any set of $d \geq 2$ unknowns, where we know that at least $(d-1)$ unknowns have degree at least β. In that case, $\sum_{i=0}^{d-1} \ell_i \geq 1 + \beta(d-1) > \beta(d-1)$. Thus (5) cannot be satisfied for any $d \geq 2$. So, there exists a matching in $\mathbf{G}_{dec}$ where all the unknowns are matched, hence the master node can decode all the unknowns. ■

Now we state the result when different parallel classes are utilized.

Theorem 3: Consider c distinct parallel classes (with block size β) such that the size of the intersection between any two blocks from different parallel classes is at most 1. Using Alg. 1, the distributed matrix-vector multiplication scheme can be resilient to at least $c\ell - \beta + \lambda$ stragglers, when $c \geq \beta + \lambda$ and $\lambda < \beta$.

Proof: The main idea is to find the scenario where Lemma 2 can be directly applicable. To establish that we consider two unknowns u_0 and u_1. The event that a pair of unknowns belong to the same meta-symbol can happen within at most one parallel class (in other words, within only one worker group) according to our choices of parallel classes. Thus in the remaining $(c-1)$ worker groups, those two unknowns exist in different meta-symbols. If they appear in the same meta-symbol, then there are ℓ workers within the worker group where they appear. On the other hand, if they appear in different meta-symbols, then there are at least $\ell + 1$ workers within the worker group where either u_0 or u_1 or both appear as part of a meta-symbol.

So, the unknowns u_0 or u_1 or both participate in different meta-symbols in at least in $\ell + (\ell+1)(c-1)$ workers. Now since we have $c\ell - \beta + \lambda$ stragglers, using $c \geq \beta + \lambda$, we have still

$$\ell + (\ell+1)(c-1) - (c\ell - \beta + \lambda) \geq 2\beta - 1$$

workers left. This means that either u_0 or u_1 or both exist in at least $2\beta - 1$ workers after the stragglers are removed. This in turn implies that the corresponding $\mathbf{G}_{dec}$ has at least $2\beta - 1$ edges emanating from the pair of unknowns u_0 and u_1, so that at least one of them has degree $\geq \beta$. Thus Lemma 2 is satisfied and we can decode all the unknowns. ■

Example 6: Consider a scenario with $n = 20$ workers with $\gamma = \frac{1}{5}$, thus $c = 4$, and we apply β-level coding approach with $\beta = 3$. In this case, we incorporate *four* different parallel classes of block size $\beta = 3$ obtained from the solution of the famous Kirkman's Schoolgirl problem [37]. These parallel classes are given by

$$\mathcal{P}_0 = \{\{0,1,2\},\{3,4,5\},\{6,7,8\},\{9,10,11\},\{12,13,14\}\},$$
$$\mathcal{P}_1 = \{\{0,3,6\},\{1,4,7\},\{2,9,12\},\{5,10,13\},\{8,11,14\}\},$$
$$\mathcal{P}_2 = \{\{0,4,14\},\{1,8,9\},\{2,3,13\},\{5,7,11\},\{6,10,12\}\},$$
$$\mathcal{P}_3 = \{\{0,8,12\},\{1,3,11\},\{2,4,10\},\{5,6,14\},\{7,9,13\}\},$$

where any two blocks from any two different parallel classes have an intersection size at most *one*. It can be verified that the distributed matrix-vector multiplication scheme will be resilient to at least $s = c\ell - \beta + 1 = 10$ stragglers whereas the straggler resilience if we used the single parallel class would have been 9.

Table IV compares experimental results for different matrix-vector multiplication approaches in terms of number of stragglers and Q/Δ values. For every case, we observe a significant improvement of the metrics if we incorporate multiple parallel classes instead of a single parallel class. We note here that the Q/Δ was computed via computer experiments.

The analysis in Theorem 3 above is somewhat loose as we only assume that the intersection sizes between blocks from different parallel classes is at most 1. Indeed, exploiting more structure in the choice of the parallel classes can yield better results, though the analysis becomes significantly harder. Here we present a method that improves on Theorem 2 when $c = \beta = 2, \ell \leq \Delta/2 - 2$ and $\Delta \geq 8$. Let $\mathcal{X} = \{0, 1, \ldots, \Delta - 1\}$ where $\Delta = n = 2a_2$. The block size of the design is two and the parallel classes are given as follows.

$$\begin{aligned} \mathcal{P}_0 &= \{\{0,1\},\{2,3\},\ldots,\{\Delta-2,\Delta-1\}\} \\ \text{and } \mathcal{P}_1 &= \{\{0,5\},\{2,7\},\ldots,\{\Delta-2,3\}\}. \end{aligned} \tag{6}$$

Thus, the i-th block in $\mathcal{P}_0$ and $\mathcal{P}_1$, for $0 \leq i \leq \Delta/2 - 1$ is given by $\{2i, 2i+1\}$ and $\{2i, 2i+5\} \pmod{\Delta}$, respectively. We follow the Alg. 1 for the specification of the coding scheme.

Theorem 4: Let $c = \beta = 2$, $\ell \leq \Delta/2 - 2$ and $\Delta \geq 8$. If we use the parallel classes in (6), then the matrix-vector scheme described in Alg. 1 will be resilient to $s = 2\ell - 1$ stragglers, and $Q = n\ell - \ell(\ell+1) + 1$.

Proof: The detailed proof is discussed in Appendix B. ■

B. Matrix-Matrix Multiplication

Now we consider the case of matrix-matrix multiplication, where we assume that each of the n worker nodes can store $\gamma_A = \frac{a_1}{a_2}$ and $\gamma_B = \frac{b_1}{b_2}$ fractions of matrices $\mathbf{A}$ and $\mathbf{B}$. In this case, we consider β_A and β_B-level coding for $\mathbf{A}$ and $\mathbf{B}$, respectively so that $\gamma_A \leq \frac{1}{\beta_A}$ and $\gamma_B \leq \frac{1}{\beta_B}$. We partition matrices $\mathbf{A}$ and $\mathbf{B}$ into Δ_A and Δ_B block-columns, respectively, and so, we have, in total, $\Delta = \Delta_A \Delta_B$ unknowns. Next we assign $\ell_A = \Delta_A \gamma_A$ block-columns from $\mathbf{A}$ and $\ell_B = \Delta_B \gamma_B$ block-columns of $\mathbf{B}$ to each of the workers. Thus, each worker computes $\ell = \ell_A \ell_B$ submatrix products according to the natural order discussed in Section III.

Once the matrices are decomposed into block-columns, we allow β_A-level and β_B-level coding for matrices $\mathbf{A}$ and $\mathbf{B}$, respectively. In this case we choose two separate resolvable designs with block sizes β_A and β_B supported on point sets $\{0, 1, \ldots, \Delta_A - 1\}$ and $\{0, 1, \ldots, \Delta_B - 1\}$ respectively. Furthermore, we assume that the number of worker nodes $n = c \times a_2 b_2$ where c is a positive integer.

Let $\mathcal{P}^A$ and $\mathcal{P}^B$ denote parallel classes for the matrices $\mathbf{A}$ and $\mathbf{B}$ respectively. As in the matrix-vector scheme, the coding scheme is specified by the meta-symbols (blocks) of $\mathcal{P}^A$ and $\mathcal{P}^B$. Let $\mathcal{N}_A$ and $\mathcal{N}_B$ denote the corresponding incidence matrices of these parallel classes. Recall that each meta-symbol is in one-to-one correspondence with the columns of the incidence matrices. Consider the matrix $\mathcal{N}_{AB}$ formed by considering pair-wise Kronecker products of columns from $\mathcal{N}_A$ and $\mathcal{N}_B$. Then the rows of $\mathcal{N}_{AB}$ correspond to unknowns of the form $\mathbf{A}_i^T \mathbf{B}_j$ and the columns correspond to the support of the random linear equations that are formed by considering the pairwise products. We will refer to the meta-symbols of $\mathcal{N}_{AB}$ as product meta-symbols and denote it by $\mathcal{P}^{AB}$. For example, suppose that $\beta_A = \beta_B = 2$ and consider two meta-symbols $\{0,1\} \in \mathcal{P}^A$ and $\{0,1\} \in \mathcal{P}^B$. If these symbols are placed in a worker, the corresponding product would be

TABLE IV
COMPARISON OF DIFFERENT METRICS FOR DIFFERENT APPROACHES

SYSTEM	METRICS	DENSE CODES [3], [17], [21]	β-LEVEL CODING (SINGLE PARALLEL CLASS)	β-LEVEL CODING (MULTIPLE PARALLEL CLASSES)
$n = 8, \gamma_A = \frac{1}{4}$ AND $\beta = 2$	s	4	2	3
	Q/Δ	–	13/8	11/8
$n = 8, \gamma_A = \frac{1}{4}$ AND $\beta = 3$	s	4	3	4
	Q/Δ	–	19/12	14/12
$n = 10, \gamma_A = \frac{1}{5}$ AND $\beta = 3$	s	5	3	4
	Q/Δ	–	25/15	20/15

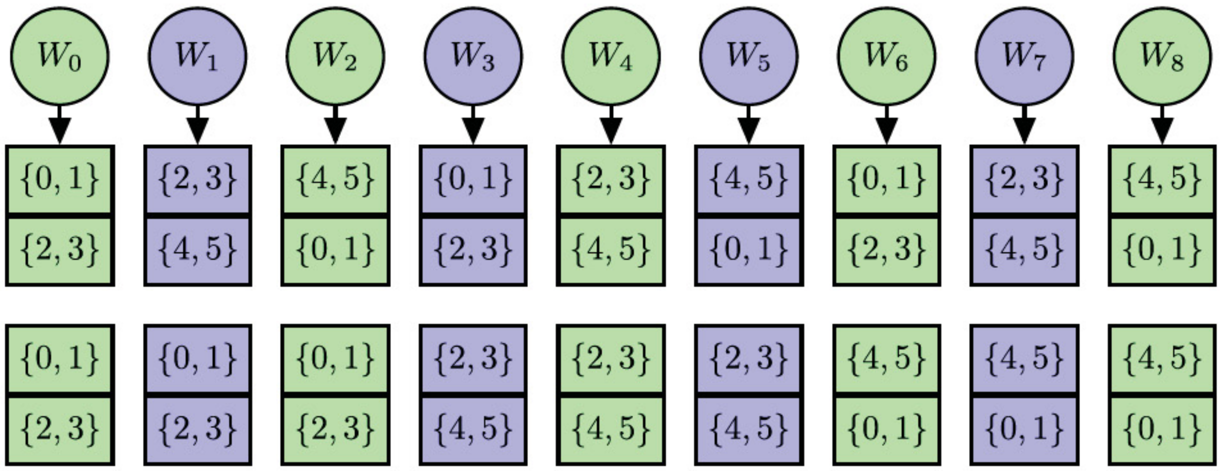

Fig. 7. Job assignment for worker group $\mathcal{G}_0$ for β-level matrix-matrix multiplication scheme with $n = 36$ with $\gamma_A = \gamma_B = \frac{1}{3}$ and $\Delta_A = \Delta_B = 6$ using a single parallel class with $\beta_A = \beta_B = 2$. The indices $\{i, j\}$ on top and bottom parts indicate random linear combinations of the submatrices of $\mathbf{A}$ and $\mathbf{B}$, respectively. $\mathcal{G}_1$, $\mathcal{G}_2$ and $\mathcal{G}_3$ are assigned the same symbols as workers $W_0 - W_8$, but with different random coefficients.

$(x_0\mathbf{A}_0^T + x_1\mathbf{A}_1^T)(y_0\mathbf{B}_0 + y_1\mathbf{B}_1) = x_0y_0\mathbf{A}_0^T\mathbf{B}_0 + x_0y_1\mathbf{A}_0^T\mathbf{B}_1 + x_1y_0\mathbf{A}_1^T\mathbf{B}_0 + x_1y_1\mathbf{A}_1^T\mathbf{B}_1$ where x_0, x_1, y_0, y_1 are chosen i.i.d. at random from a continuous distribution. Thus, the coefficients of the corresponding equation can be expressed as

$$[x_0 \; x_1] \otimes [y_0 \; y_1] \tag{7}$$

where $\otimes$ denotes the Kronecker product.

Claim 1: If $\mathcal{N}_A$ (of size $\Delta_A \times \Delta_A/\beta_A$) and $\mathcal{N}_B$ (of size $\Delta_B \times \Delta_B/\beta_B$) correspond to incidence matrices of parallel classes, then $\mathcal{N}_{AB}$ also forms a parallel class of size $\Delta_A\Delta_B \times \Delta_A\Delta_B/\beta_A\beta_B$.

Proof: Let $\mathbf{u}_i \otimes \mathbf{v}_i$ for $i = 0, 1$ denote two distinct columns of $\mathcal{N}_{AB}$ such that $\mathbf{u}_i$ and $\mathbf{v}_i$ are columns in $\mathcal{N}_A$ and $\mathcal{N}_B$ respectively. Then,

$$\begin{aligned}(\mathbf{u}_0 \otimes \mathbf{v}_0)^T(\mathbf{u}_1 \otimes \mathbf{v}_1) &= \mathbf{u}_0^T\mathbf{u}_1 \times \mathbf{v}_0^T\mathbf{v}_1 \\ &= 0\end{aligned}$$

since either $\mathbf{u}_0 \neq \mathbf{u}_1$ or $\mathbf{v}_0 \neq \mathbf{v}_1$. Moreover, there are $\frac{\Delta_A\Delta_B}{\beta_A\beta_B}$ distinct columns in $\mathcal{N}_{AB}$ each with a support of size $\beta_A\beta_B$. This implies that together all the product meta-symbols in $\mathcal{N}_{AB}$ cover all the $\Delta_A\Delta_B$ points. ■

As in the matrix-vector case, the scheme operates by placing cyclically shifted meta-symbols from $\mathcal{P}^A$ with ℓ_A meta-symbols in each worker for the first Δ_A/β_A workers. For these workers, the assignment of meta-symbols from $\mathcal{P}^B$ is the same. For the next set of Δ_A/β_A workers the assignment of meta-symbols from $\mathcal{P}^A$ repeats; however, we now employ a cyclic shift for the assignment of meta-symbols from $\mathcal{P}^B$. The complete algorithm is specified in Alg. 2 and an example is depicted in Fig. 7. As before, a group in this setting contains Δ/β workers and there a total of $\frac{n}{\Delta/\beta} = c$ groups denoted $\mathcal{G}_i, i = 0, 1, \ldots, c-1$. Let $\mathcal{X}_{AB} = \{\mathbf{A}_0^T\mathbf{B}_0, \mathbf{A}_0^T\mathbf{B}_1, \mathbf{A}_0^T\mathbf{B}_2, \ldots, \mathbf{A}_{\Delta_A-1}^T\mathbf{B}_{\Delta_B-1}\}$ denote the set of unknowns. The product of two assigned coded block-columns consists of a random linear combination of $\beta = \beta_A\beta_B$ unknowns from $\mathcal{X}_{AB}$.

Example 7: We consider an example with $n = 36$ workers in Fig. 7, each of which can store $\gamma_A = \gamma_B = \frac{1}{3}$ of each of matrices $\mathbf{A}$ and $\mathbf{B}$, and $\beta_A = \beta_B = 2$. We set $\Delta_A = \Delta_B = 6$, thus the cardinality of $\mathcal{X}_{AB}$ is 36. In terms of indices, we use the same parallel class, $\{\{0,1\},\{2,3\},\{4,5\}\}$ for both $\mathbf{A}$ and $\mathbf{B}$. Finally we use random vectors of length $\beta_A = \beta_B = 2$ to obtain the symbols from the submatrices of the elements of the parallel classes, $\mathcal{P}_i^A$ and $\mathcal{P}_i^B$ in any worker group $\mathcal{G}_i$, for $i = 0, 1, 2, 3$, as $c = 36/9 = 4$.

Lemma 3: The matrix-matrix multiplication scheme in Alg. 2 is such that there are $\ell = \ell_A\ell_B$ symbols corresponding to any product meta-symbol $\in \mathcal{P}_i^{AB}$ in a group $\mathcal{G}_i$. Furthermore, this product meta-symbol appears in all locations $0, 1, 2, \ldots, \ell - 1$ within $\mathcal{G}_i$.

Algorithm 2 β-Level Coding Scheme for Matrix-Matrix Multiplication

```
Input : Matrices $\mathbf{A}$ and $\mathbf{B}$, storage fractions of the workers $\gamma_A = \frac{a_1}{a_2}$ $\gamma_B = \frac{b_1}{b_2}$, $\beta_A, \beta_B$-coding level for $\mathbf{A}$ and $\mathbf{B}$, respectively, and number of worker nodes, $n = c \times a_2 b_2$, where $c$ is a positive integer.
1 Partition $\mathbf{A}$ into $\Delta_A = \beta_A a_2$ block-columns and partition $\mathbf{B}$ into $\Delta_B = \beta_B b_2$ block-columns;
2 $\Delta = \Delta_A \Delta_B$, $\ell_A = \Delta_A \gamma_A$, $\ell_B = \Delta_B \gamma_B$, $\beta = \beta_A \beta_B$;
3 Assume $\mathcal{X}_A = \{0, 1, 2, \dots, \Delta_A - 1\}$ and find parallel classes $\mathcal{P}_i^A$ having block size $\beta_A$, $i = 0, 1, \dots, c-1$;
4 Assume $\mathcal{X}_B = \{0, 1, 2, \dots, \Delta_B - 1\}$ and find parallel classes $\mathcal{P}_i^B$ having block size $\beta_B$, $i = 0, 1, \dots, c-1$;
5 for $i \leftarrow 0$ to $c - 1$ do
6     Let the blocks of $\mathcal{P}_i^A$ be denoted as $p_{A_0}, p_{A_1}, \dots, p_{A_{\alpha_A - 1}}$ ;
7     Let the blocks of $\mathcal{P}_i^B$ be denoted as $p_{B_0}, p_{B_1}, \dots, p_{B_{\alpha_B - 1}}$ ;
8     for $j \leftarrow 0$ to $\frac{\Delta}{\beta} - 1$ do
9         Assign sets $p_{A_j}, p_{A_{j+1}}, \dots, p_{A_{j+\ell_A-1}}$ from top to bottom (indices reduced modulo $a_2$) to worker $\frac{\Delta}{\beta} i + j$;
10        $k \leftarrow \left\lfloor \frac{j}{a_2} \right\rfloor$, and assign sets $p_{B_k}, p_{B_{k+1}}, \dots, p_{B_{k+\ell_B-1}}$ from top to bottom (indices reduced modulo $b_2$) to worker $\frac{\Delta}{\beta} i + j$;
11        Choose random linear combinations of the constituent block-columns of the meta-symbols of $\mathcal{P}_i^A$ and $\mathcal{P}_i^B$ of length $\beta_A$ and $\beta_B$ respectively;
12    end
13 end
Output: Distributed matrix-matrix multiplication scheme having $\beta$-level coding.
```

Proof: Any group $\mathcal{G}_i$ can be partitioned into $\alpha_B = \frac{\Delta_B}{\beta_B}$ disjoint subgroups each of which consists of $\alpha_A = \frac{\Delta_A}{\beta_A}$ workers. These subgroups are denoted as $\mathcal{H}_j$ where in terms of group worker indices, $\mathcal{H}_j = \{j\alpha_A, j\alpha_A + 1, \dots, (j+1)\alpha_A - 1\}$, for $j = 0, 1, \dots, \alpha_B - 1$.

If meta-symbols $x \in \mathcal{P}_i^A$ and $y \in \mathcal{P}_i^B$ appear at locations i_1 and j_1, respectively, $0 \le i_1 \le \ell_A - 1$ and $0 \le j_1 \le \ell_B - 1$, then the product meta-symbol $x \otimes y$ appears at location $i_1 \ell_B + j_1$ in the ordering. In our case, meta-symbol x appears ℓ_A times within subgroup $\mathcal{H}_j$ at distinct locations $0, \dots, \ell_A - 1$. Thus, if meta-symbol $y \in \mathcal{P}_i^B$ appears in $\mathcal{H}_j$ at location j_1 then the product meta-symbol $x \otimes y$ appears ℓ_A times at locations $j_1, \ell_B + j_1, 2\ell_B + j_1, \dots, (\ell_A - 1)\ell_B + j_1$. The result follows by realizing that there are ℓ_B subgroups where meta-symbol y appears. Moreover, $y \in \mathcal{P}_i^B$ appears at all locations $0, \dots, \ell_B - 1$ across these subgroups. ■

Theorem 5: If we use a single parallel class $\mathcal{P}_A$ for $\mathbf{A}$ and a single parallel class $\mathcal{P}_B$ for $\mathbf{B}$ across all the worker groups, then the scheme described in Alg. 2 will be resilient to $s = c\ell - \beta$ stragglers and will have, $Q = n\ell - \frac{c\ell(\ell+1)}{2} + \ell(\beta - 1) + 1$, where $\ell = \ell_A \ell_B$ and $\beta = \beta_A \beta_B \le c$.

Proof: The proof is very similar to the proof of Theorem 2 once we use the fact that each product meta-symbol appears in all locations $0, \dots, \ell - 1$ within the group in which it appears (*cf.* Lemma 3). ■

It can be verified that the distributed scheme shown in Fig. 7 is resilient to $s = c\ell - \beta = 4 \times 4 - 4 = 12$ stragglers and has $Q = 117$. Theorem 5 provides the value for s and Q for distributed matrix-matrix multiplication when $\beta \le c$. In Appendix A, we explicitly calculate the values for s and Q for the case when $\beta > c$.

Remark 4: Similar to the matrix-vector case, the uncoded matrix-matrix multiplication scheme can also be thought as a special case of β-level coding scheme with $\beta = 1$. The lower bound given in (2) is matched by the proposed scheme here with $\beta_A = \beta_B = 1$ (i.e., the uncoded scheme). An example appears in Fig. 8 where we have $n = 12$ workers and the master node can recover the final product as soon as it receives $Q = 52$ symbols across all the workers.

In the matrix-matrix case for $\beta_A = 2, \beta_B = 1$ we can show that using different parallel classes can improve the straggler resilience of the system. The corresponding Q analysis is harder to do and is part of future work.

Theorem 6: Let $\ell_A \le \frac{\Delta_A}{2} - 2$ and $\Delta_A \ge 8$. If we use the parallel classes in (6) for encoding $\mathbf{A}$, then the matrix-matrix multiplication scheme described in Alg. 2 will be resilient to $s = 2\ell - 1$ stragglers, when $\beta_A = 2$ and $\beta_B = 1$ such that $c = \beta = 2$.

Proof: Consider the set $\mathcal{B}_m = \{\mathbf{A}_0^T \mathbf{B}_m, \mathbf{A}_1^T \mathbf{B}_m, \dots, \mathbf{A}_{\Delta_A - 1}^T \mathbf{B}_m\}$, i.e., the set of all unknowns corresponding to $\mathbf{B}_m$, for $m = 0, 1, \dots, \Delta_B - 1$, so $|\mathcal{B}_m| = \Delta_A$. As $\mathbf{B}$ is uncoded, the equations consisting of the unknowns in $\mathcal{B}_m$ are disjoint of the equations consisting of the unknowns of $\mathcal{B}_p$, $(m \ne p)$. Thus, we can form $\mathcal{G}_{dec}^m$ using the unknowns corresponding to the set $\mathcal{B}_m$ and analyze the decoding using it. The rest of the argument follows analogous to the proof of Theorem 4. ■

It should be noted that there is certainly a trade-off between the coding parameters (β) and the properties of the scheme. For example, in case of matrix-vector multiplication, when the matrix has $\mu\%$ non-zero entries, the assigned submatrices can have approximately $\beta\mu\%$ non-zero entries (when μ is small) in case of β-level coding. It indicates that the worker computation time can increase by approximately β times in compared to the uncoded approach. On the other hand, the recovery threshold will reduce (according to Theorem 2) which indicates resilience to more number of stragglers than the uncoded case ($\beta = 1$).

C. Coded at Bottom Scheme

Intuitively, the β-level coding schemes can be improved if we allow for the inclusion of some densely coded block-columns. We now consider a variant of the uncoded scheme where such densely coded block-columns are added at the end of uncoded computations. This improves both the straggler resilience and the Q value of the scheme.

We now assume that each node receives $\gamma = \gamma_u + \gamma_c$ fraction of the columns of $\mathbf{A}$ and the vector $\mathbf{x}$. Here γ_u

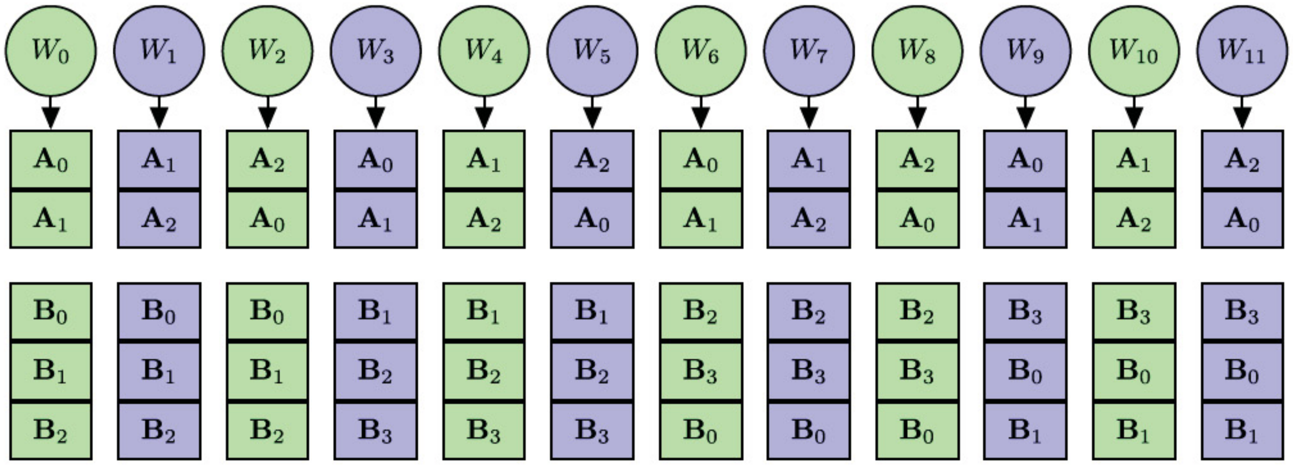

Fig. 8. Uncoded matrix-matrix multiplication with $n = 12$ and $s = 5$ with $\gamma_A = \frac{2}{3}$ and $\gamma_B = \frac{3}{4}$ where $\Delta_A = 3$ and $\Delta_B = 4$.

Algorithm 3 Cyclic Coded at the Bottom Scheme for Distributed Matrix-Vector Multiplication

```
Input  : Matrix $\mathbf{A}$ and vector x, $n$-number of worker
         nodes, total storage capacity fraction $\gamma$,
         replication factor for uncoded portion $r_u$.
1 Set $\Delta = n$, $\ell_u = r_u$, $\ell = \gamma\Delta$, $\ell_c = \ell - \ell_u$;
2 Partition $\mathbf{A}$ into $\Delta$ block-columns $\mathbf{A}_0, \mathbf{A}_1, \ldots, \mathbf{A}_{\Delta-1}$;
3 for $i \leftarrow 0$ to $n-1$ do
4     Define $T = \{i, i+1, \ldots, i+\ell_u-1\}$ (mod $\Delta$);
5     Assign all $\mathbf{A}_m$'s sequentially from top to bottom to
      worker node $i$, where $m \in T$;
6     Assign $\ell_c$ different random linear combinations of
      $\mathbf{A}_m$'s for $m \notin T$;
7 end
Output: Cyclic coded at the bottom scheme for
        matrix-vector multiplication.
```

corresponds to the storage fraction of the uncoded parts of $\mathbf{A}$, whereas γ_c corresponds to the coded portion. The coded blocks appear at the bottom of each node. Thus, under normal operating circumstances (no slow or failed nodes), the master node can simply decode the intended result from the uncoded computations. If some nodes are operating slower than normal, then the coded computations can be leveraged.

As in the uncoded setup let $\ell_u = \Delta\gamma_u$ be the number of uncoded block-columns and r_u be the replication factor. Likewise $\ell_c = \Delta\gamma_c$ represents the number of coded blocks in each worker. In this construction we set $\Delta = n$ so that $r_u = \ell_u$. In this case, the results from Theorem 2 immediately imply that $Q \geq \max(\Delta, \Delta r_u - \frac{r_u}{2}(\ell_u+1)+1)$. This follows by applying $\beta = 1$ to the uncoded part of the solution where $r_u = \ell_u$. A construction that meets these bounds is outlined in Algorithm 3. The algorithm uses a random matrix of dimension $n\ell_c \times \Delta$.

Theorem 7: The scheme in Alg.3 satisfies $Q = \max(\Delta, \Delta r_u - \frac{r_u}{2}(\ell_u+1)+1)$. Furthermore, it is resilient to $\left\lfloor \frac{n^2\gamma_c + n\gamma_u - 1}{n\gamma_c+1} \right\rfloor$ stragglers.

Proof: The detailed proof is discussed in Appendix C. ■

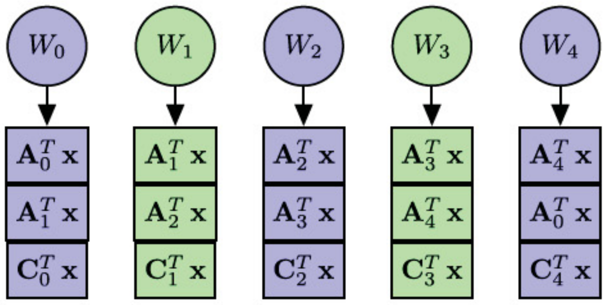
W_0
W_1
W_2
W_3
W_4
$\mathbf{A}_0^T\mathbf{x}$
$\mathbf{A}_1^T\mathbf{x}$
$\mathbf{C}_0^T\mathbf{x}$
$\mathbf{A}_1^T\mathbf{x}$
$\mathbf{A}_2^T\mathbf{x}$
$\mathbf{C}_1^T\mathbf{x}$
$\mathbf{A}_2^T\mathbf{x}$
$\mathbf{A}_3^T\mathbf{x}$
$\mathbf{C}_2^T\mathbf{x}$
$\mathbf{A}_3^T\mathbf{x}$
$\mathbf{A}_4^T\mathbf{x}$
$\mathbf{C}_3^T\mathbf{x}$
$\mathbf{A}_4^T\mathbf{x}$
$\mathbf{A}_0^T\mathbf{x}$
$\mathbf{C}_4^T\mathbf{x}$

Fig. 9. Partitioning matrix A into five submatrices and assigning two uncoded and one coded task to each of the five workers. The coded submatrix assigned to W_i is denoted as $\mathbf{C}_i$.

Example 8: Consider the setting where we have $n = 5$ workers with $\gamma = \frac{3}{5}$ where we set $\Delta = n = 5$. Fig. 4 shows the job assignments according to the uncoded scheme ($\beta = 1$). According to Theorem 2 in Section IV, the system is resilient to $\beta(n\gamma - 1) = 2$ stragglers and $Q = 5 \times 3 - \frac{3\times4}{2} + 1 = 10$ which can be verified from Fig. 4.

Now we assume that the whole storage fraction can be distributed into an uncoded storage fraction $\gamma_u = \frac{2}{5}$ and a coded storage fraction $\gamma_c = \frac{1}{5}$. Using the coded scheme, we get the job assignments shown in Fig. 9. This scheme is resilient to $\left\lfloor \frac{n^2\gamma_c + n\gamma_u - 1}{n\gamma_c+1} \right\rfloor = 3$ stragglers and it can be verified from that $\mathbf{A}^T\mathbf{x}$ can be computed once any $Q = \Delta r_u - \frac{r_u}{2}(\ell_u+1)+1 = 8$ block-columns have been processed. Thus, we can conclude that introducing a single coded block in each worker (at the bottom), helps to improve both Q and the straggler resilience of the system as compared to an uncoded system.

Similar schemes can be arrived at for the matrix-matrix case. We assume that the uncoded storage fraction for $\mathbf{A}$ is $\gamma_{Au} = \frac{a_u}{a_2}$ and the coded storage fraction is $\gamma_{Ac} = \frac{a_c}{a_2}$, so that the total storage fraction is $\gamma_A = \frac{a_1}{a_2}$. Each worker also receives $\gamma_B = \frac{b_1}{b_2}$ fraction of the uncoded columns of matrix B.

Theorem 8: The recovery threshold for the matrix-matrix multiplication scheme Alg. 4 is given by,

Algorithm 4 Cyclic Coded at the Bottom Scheme for Distributed Matrix-Matrix Multiplication

```
Input : Matrices $\mathbf{A}$ and $\mathbf{B}$, $n$-number of workers.
        Storage fractions $\gamma_{Au} = \frac{a_u}{a_2}$ and $\gamma_{Ac} = \frac{a_c}{a_2}$,
        so that $\gamma_A = \frac{a_1}{a_2}$ and $\gamma_B = \frac{b_1}{b_2}$.
1 Set $\Delta_A = a_2$, $\Delta_B = mb_2$, $m = \frac{n}{(a_2 \times b_2)}$. Partition $\mathbf{A}$ and
  $\mathbf{B}$ into $\Delta_A$ and $\Delta_B$ block-columns, respectively;
2 for $i \leftarrow 0$ to $n-1$ do
3     Define $T = \{i, i+1, \dots, i+a_u-1\}$ (mod $\Delta_A$);
4     Assign all $\mathbf{A}_m$'s sequentially from top to bottom to
      worker node $i$, where $m \in T$;
5     Assign $a_c$ different random linear combinations of
      $\mathbf{A}_m$'s for $m \notin T$;
6     $j \leftarrow \left\lfloor \frac{i}{a_2} \right\rfloor$ and assign $\mathbf{B}_j, \mathbf{B}_{j+1}, \dots, \mathbf{B}_{j+mb_1-1}$ from
      top to bottom (subscripts reduced modulo $\Delta_B$) to
      worker node $i$;
7 end
Output: Cyclic coded at the bottom scheme for
        matrix-matrix multiplication.
```

$\tau = n - ma_2b_1 + \kappa_{min}$, where κ_{min} is the minimum positive integer for κ satisfying the inequality

$$\left\lceil \frac{\kappa}{mb_1} \right\rceil + \kappa a_c \geq a_2 - a_u + 1.$$

Proof: The detailed proof is discussed in Appendix D. ■

Example 9: We consider the scenario as before, where $\gamma_A = \frac{2}{3}$ and $\gamma_B = \frac{3}{4}$, and $n = 12$, so $m = \frac{12}{3\times4} = 1$. According to Alg. 4, we set $\ell_A = 2$, $\Delta_A = 3$ and $\ell_B = 3$, $\Delta_B = 4$. So, we need to recover $\Delta = \Delta_A\Delta_B = 12$ block products. Figs. 8 and 10 show the job assignments to the workers for the uncoded case and the proposed coded scheme, respectively. For the coded scheme, we assume $\gamma_{Au} = \frac{1}{3}$ and $\gamma_{Ac} = \frac{1}{3}$, and on the other hand, for the uncoded scheme, we have $\gamma_{Ac} = 0$, so $a_c = 0$.

Now for the uncoded case, according to Theorem 5, the recovery threshold is $\tau = n - (c\ell - \beta) = 12 - (1 \times \ell_A\ell_B - 1) = 7$. On the other hand, according to Theorem 8, the recovery threshold for the coded case is, $\tau = n - ma_2b_1 + \kappa = 12 - 3 \times 3 + 2 = 5$ since the minimum positive integer κ that satisfies $\left\lceil \frac{\kappa}{3} \right\rceil + \kappa \geq 3$ is 2.

We expect that the benefits of having densely coded block-columns at the bottom should extend for the case of general $\beta > 1$ and the Q/Δ analysis should be possible to perform for the matrix-matrix case. However, this appears to be more challenging and will be investigated as part of future work.

V. Sparsely Coded Straggler (SCS) Optimal Matrix Computations

In this section, we develop schemes for distributed matrix computations which perform optimally in terms of straggler resilience. For example, in matrix-matrix multiplication case, if the storage fractions of each worker node are $\gamma_A = 1/k_A$ and $\gamma_B = 1/k_B$ then it can be shown the lowest possible threshold is k_Ak_B [3]. Similarly, for the matrix-vector multiplication case the optimal threshold is k_A. Prior work has also demonstrated schemes that achieve these thresholds. In what follows, we present schemes that are similar in spirit to our constructions in Section IV which are suitable for sparse matrices while continuing to enjoy the optimal threshold k_Ak_B. Moreover, unlike the previously available dense coded approaches, our proposed sparsely coded straggler (SCS) optimal scheme can utilize the partial computations of the slow workers and can provide significantly small Q/Δ.

Algorithm 5 SCS Optimal Scheme for Matrix-Vector Multiplication

```
Input : Matrix $\mathbf{A}$ and vector $\mathbf{x}$, $n$-number of worker
        nodes, storage fraction $\gamma_A = \frac{1}{k_A}$.
1 Set $\Delta = \mathrm{LCM}(n, k_A)$. Partition $\mathbf{A}$ into $\Delta$ block-columns
  $\mathbf{A}_0, \mathbf{A}_1, \dots, \mathbf{A}_{\Delta-1}$;
2 Number of coded submatrices of $\mathbf{A}$ in each worker node,
  $\ell_c = \frac{\Delta}{k_A} - \frac{\Delta}{n}$;
3 for $i \leftarrow 0$ to $n-1$ do
4     $u \leftarrow i \times \frac{\Delta}{n}$;
5     Define $T = \left\{u, u+1, \dots, u+\frac{\Delta}{n}-1\right\}$ (mod $\Delta$);
6     Assign all $\mathbf{A}_m$'s sequentially from top to bottom to
      worker node $i$, where $m \in T$;
7     Assign $\ell_c$ different random linear combinations of
      $\mathbf{A}_m$'s for $m \notin T$;
8 end
Output: $\langle n, \gamma_A \rangle$ SCS optimal-scheme for matrix-vector
        multiplication with $Q/\Delta = 1$.
```

A. Matrix-Vector Multiplication

In our proposed scheme in Alg. 5, we set $\Delta = \mathrm{LCM}(n, k_A)$ and assign the uncoded jobs in such a way that all the workers are assigned the uncoded jobs in an equal manner and the replication factor of the uncoded symbols over all n workers is, $r_u = 1$. Thus, each of the workers is assigned Δ/n uncoded jobs and the rest $\ell_c = \frac{\Delta}{k_A} - \frac{\Delta}{n}$ jobs are assigned using a random linear encoding matrix, $\mathcal{R}$ of size $n\ell_c \times \Delta$. Since any $(\Delta - \lambda) \times (\Delta - \lambda)$ submatrix of $\mathcal{R}$ is full rank with probability 1, the master node can decode all the unknowns if it receives any λ uncoded symbols and any $\Delta - \lambda$ coded symbols from all the workers. Thus we can say that $Q = \Delta$, and since each worker stores Δ/k_A block-columns, we have the recovery threshold, $\tau = \frac{\Delta}{\Delta/k_A} = k_A$.

Example 10: We consider an example in Fig. 11 with $n = 6$ and $\gamma = \frac{1}{4}$, so $k_A = 4$. We set $\Delta = \mathrm{LCM}(6,4) = 12$, and $\ell_c = \frac{12}{4} - \frac{12}{6} = 1$. Thus, we assign two uncoded jobs and one coded job to each worker where the coded job assignment would be incorporated using a random matrix $\mathcal{R}$ of size 6×12. In this case, $Q = 12$, thus $Q/\Delta = 1$, and $\tau = 4$.

Remark 5: On the surface Fig. 11 may appear equivalent to a systematic version of the RKRP coded scheme [17] with the same number of matrix partitions. However, there is a significant difference that the idea in the RKRP coded scheme is to assign the systematic versions to some workers and the coded versions to other workers, whereas we assign the jobs

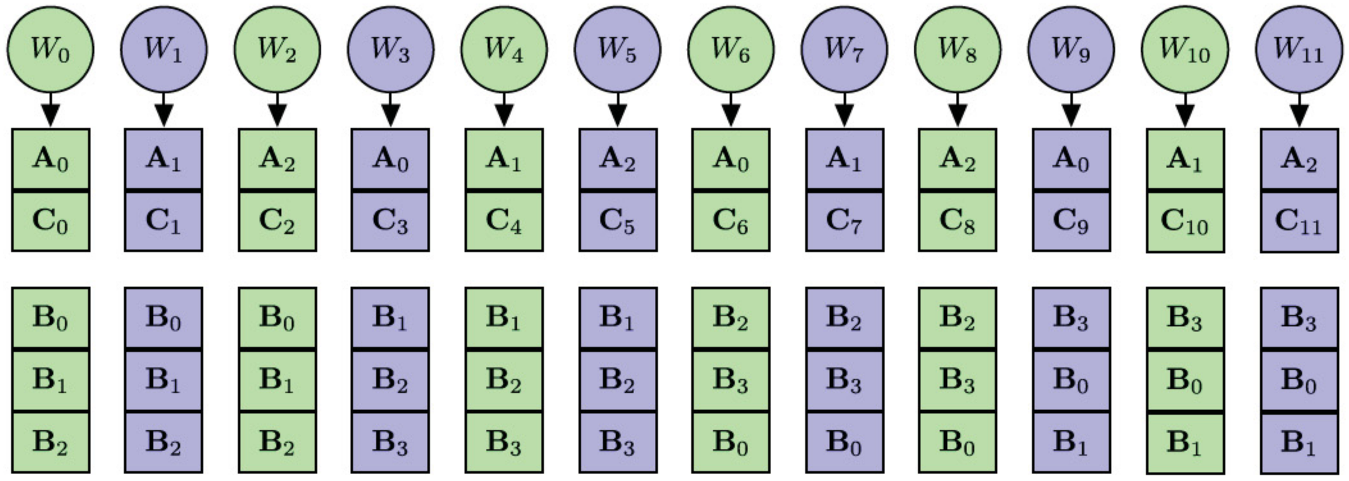
W_0 W_1 W_2 W_3 W_4 W_5 W_6 W_7 W_8 W_9 W_{10} W_{11}
$\mathbf{A}_0$ $\mathbf{A}_1$ $\mathbf{A}_2$ $\mathbf{A}_0$ $\mathbf{A}_1$ $\mathbf{A}_2$ $\mathbf{A}_0$ $\mathbf{A}_1$ $\mathbf{A}_2$ $\mathbf{A}_0$ $\mathbf{A}_1$ $\mathbf{A}_2$
$\mathbf{C}_0$ $\mathbf{C}_1$ $\mathbf{C}_2$ $\mathbf{C}_3$ $\mathbf{C}_4$ $\mathbf{C}_5$ $\mathbf{C}_6$ $\mathbf{C}_7$ $\mathbf{C}_8$ $\mathbf{C}_9$ $\mathbf{C}_{10}$ $\mathbf{C}_{11}$
$\mathbf{B}_0$ $\mathbf{B}_0$ $\mathbf{B}_0$ $\mathbf{B}_1$ $\mathbf{B}_1$ $\mathbf{B}_1$ $\mathbf{B}_2$ $\mathbf{B}_2$ $\mathbf{B}_2$ $\mathbf{B}_3$ $\mathbf{B}_3$ $\mathbf{B}_3$
$\mathbf{B}_1$ $\mathbf{B}_1$ $\mathbf{B}_1$ $\mathbf{B}_2$ $\mathbf{B}_2$ $\mathbf{B}_2$ $\mathbf{B}_3$ $\mathbf{B}_3$ $\mathbf{B}_3$ $\mathbf{B}_0$ $\mathbf{B}_0$ $\mathbf{B}_0$
$\mathbf{B}_2$ $\mathbf{B}_2$ $\mathbf{B}_2$ $\mathbf{B}_3$ $\mathbf{B}_3$ $\mathbf{B}_3$ $\mathbf{B}_0$ $\mathbf{B}_0$ $\mathbf{B}_0$ $\mathbf{B}_1$ $\mathbf{B}_1$ $\mathbf{B}_1$

Fig. 10. Coded matrix-matrix multiplication with $n = 12$ with $\gamma_{Au} = \frac{1}{3}$, $\gamma_{Ac} = \frac{1}{3}$ and $\gamma_B = \frac{3}{4}$ where $\Delta_A = 3$ and $\Delta_B = 4$. The coded submatrix for $\mathbf{A}$ assigned to W_i is denoted as $\mathbf{C}_i$.

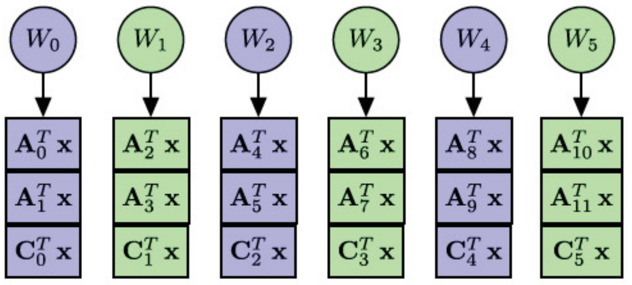
W_0 W_1 W_2 W_3 W_4 W_5
$\mathbf{A}_0^T\mathbf{x}$ $\mathbf{A}_2^T\mathbf{x}$ $\mathbf{A}_4^T\mathbf{x}$ $\mathbf{A}_6^T\mathbf{x}$ $\mathbf{A}_8^T\mathbf{x}$ $\mathbf{A}_{10}^T\mathbf{x}$
$\mathbf{A}_1^T\mathbf{x}$ $\mathbf{A}_3^T\mathbf{x}$ $\mathbf{A}_5^T\mathbf{x}$ $\mathbf{A}_7^T\mathbf{x}$ $\mathbf{A}_9^T\mathbf{x}$ $\mathbf{A}_{11}^T\mathbf{x}$
$\mathbf{C}_0^T\mathbf{x}$ $\mathbf{C}_1^T\mathbf{x}$ $\mathbf{C}_2^T\mathbf{x}$ $\mathbf{C}_3^T\mathbf{x}$ $\mathbf{C}_4^T\mathbf{x}$ $\mathbf{C}_5^T\mathbf{x}$

Fig. 11. Partitioning matrix A into $\Delta = 12$ submatrices and assigning to $n = 6$ workers each of which has been assigned two uncoded and one coded task to be resilient to $s = 2$ stragglers. The coded submatrix assigned to W_i is denoted as $\mathbf{C}_i$.

in a symmetric fashion so that every worker receives same number of uncoded and same number of coded jobs. If the input matrices are sparse, then the parity workers in the RKRP coded scheme will be significantly slower than the systematic workers.

B. Matrix-Matrix Multiplication

We propose a matrix-matrix multiplication scheme in Alg. 6 with storage fractions $\gamma_A = 1/k_A$ and $\gamma_B = 1/k_B$ and recovery threshold $k_A k_B$. Furthermore, $Q/\Delta = 1 + (k_B - 1)\ell_c/\Delta$, where ℓ_c is the number of coded-coded matrix-matrix products assigned to each worker node.

Theorem 9: Alg. 6 proposes a distributed matrix-matrix multiplication scheme being resilient to $s = n - k_A k_B$ stragglers.

Proof: According to this scheme, we know that every worker is assigned $\frac{\Delta_A}{k_A}$ block-columns (uncoded and coded) from $\mathbf{A}$ and one coded block-column from $\mathbf{B}$, which indicates that we can obtain, in total, $\frac{\Delta_A}{k_A}$ products from each of the workers. Thus from any $k_A k_B$ workers, the master node can obtain $\frac{\Delta_A}{k_A} \times k_A k_B = \Delta_A k_B = \Delta_A \Delta_B = \Delta$ products. A simple counting argument applied to Alg. 6 shows that any uncoded block-column of $\mathbf{A}$ appears exactly k_B times over all n workers.

Algorithm 6 SCS Optimal Scheme for Distributed Matrix-Matrix Multiplication

```
   Input : Matrices A and B, n-number of worker nodes,
           storage fraction $\gamma_A = \frac{1}{k_A}$ and $\gamma_B = \frac{1}{k_B}$. So,
           $s = n - k_A k_B$.
 1 Set $\Delta_A = \mathrm{LCM}(n, k_A)$ and $\Delta_B = k_B$;
 2 Partition A and B into $\Delta_A$ and $\Delta_B$ block-columns, and
   $\Delta = \Delta_A \Delta_B$;
 3 Number of coded submatrices of A in each worker node,
   $\ell_c = \frac{\Delta_A}{k_A} - \frac{\Delta}{n}$;
 4 for $i \leftarrow 0$ to $n-1$ do
 5     $u \leftarrow i \times \frac{\Delta_A}{n}$;
 6     Define $T = \{u, u+1, \dots, u + \frac{\Delta}{n} - 1\}$ (modulo $\Delta_A$);
 7     Assign all $\mathbf{A}_m$'s sequentially from top to bottom to
       worker node $i$, where $m \in T$;
 8     Assign $\ell_c$ different random linear combinations of
       $\mathbf{A}_m$'s for $m \notin T$;
 9     Assign a single random linear combination of all
       block-columns of B;
10 end
   Output: $\langle n, \gamma_A, \gamma_B \rangle$ SCS optimal-scheme for distributed
           matrix-matrix multiplication.
```

In what follows we show that each of these block products corresponds to a linearly independent equation where the variables are $\mathbf{A}_i^T \mathbf{B}_j$ for $i = 0, 1, \dots, \Delta_A - 1, j = 0, 1, \dots, \Delta_B - 1$. Let e_i denote the i-th unit vector of length Δ_A, $i = 0, \dots, \Delta_A - 1$. It follows that the product $(\sum_{i=0}^{\Delta_A - 1} u_i \mathbf{A}_i)^T (\sum_{j=0}^{\Delta_B - 1} v_j \mathbf{B}_j)$ corresponds to the vector $\sum_{i=0}^{\Delta_A - 1} u_i (e_i \otimes v)$, where v is the vector $[v_0 \; v_1 \; \dots \; v_{\Delta_B - 1}]^T$ (*cf.* discussion around (7)).

Now, suppose that we consider a subset of $k = k_A k_B$ workers indexed by the set $\mathcal{I} = \{i_0, i_1, \dots, i_{k-1}\}$. Within this worker node set, let $\mathcal{J}_i$ denote the index set of the worker nodes where $\mathbf{A}_i$ appears uncoded. The random encoding vectors for $\mathbf{A}$ and $\mathbf{B}$ in worker W_ℓ are denoted by $u^{(\ell,j)}$

(of length Δ_A) for $j = 0, 1, \ldots, \ell_c - 1$ and $v^{(\ell)}$ (of length Δ_B) respectively.

It follows that the products involving the uncoded block-column $\mathbf{A}_i$ can be expressed as

$$e_i \otimes v^{(\ell)} \text{ for } \ell \in \mathcal{J}_i.$$

Our first observation is that the collection of vectors $\{e_i \otimes v^{(\ell)}\}$ for $\ell \in \mathcal{J}_i, i = 0, \ldots, \Delta_A - 1$ is linearly independent. This follows because any linear combination of these vectors can equivalently be expressed as

$$\sum_{i=0}^{\Delta_A - 1} \left(e_i \otimes \sum_{\ell \in \mathcal{J}_i} \alpha_\ell^{(i)} v^{(\ell)} \right)$$

where $\alpha_\ell^{(i)}$'s are the linear combination coefficients and each term in the above sum needs to be forced to zero. Note that $|\mathcal{J}_i| \leq k_B$. Therefore, the vectors $v^{(\ell)}$ for $\ell \in \mathcal{J}_i$ are linearly independent with probability 1, since $v^{(\ell)}$ has length $\Delta_B = k_B$. Thus, there is no setting of $\alpha_\ell^{(i)}$'s for which the above sum can be forced to the zero vector.

The product of the coded $\mathbf{A}$ and $\mathbf{B}$ matrices can be represented by $u^{(\ell,j)} \otimes v^{(\ell)}$ for $j = 0, 1, \ldots, \ell_c - 1$ and $\ell \in \mathcal{I}$. We will now show that the overall collection of vectors that we obtain is linearly independent with probability 1. To see this suppose that there exist coefficients $\alpha_\ell^{(i)}$'s and $\kappa_\ell^{(j)}$'s not all zero such that

$$\begin{aligned}\sum_{i=0}^{\Delta_A - 1} e_i \otimes \sum_{\ell \in \mathcal{J}_i} \alpha_\ell^{(i)} v^{(\ell)} &= \sum_{\ell \in \mathcal{I}} \sum_{j=0}^{\ell_c - 1} \kappa_\ell^{(j)} u^{(\ell,j)} \otimes v^{(\ell)} \\ &= \sum_{\ell \in \mathcal{I}} \sum_{j=0}^{\ell_c - 1} \kappa_\ell^{(j)} \sum_{j_1=0}^{\Delta_A - 1} u_{j_1}^{(\ell,j)} e_{j_1} \otimes v^{(\ell)}.\end{aligned}$$

It can be observed that this decouples into finding solutions for

$$e_i \otimes \sum_{\ell \in \mathcal{J}_i} \alpha_\ell^{(i)} v^{(\ell)} = e_i \otimes \sum_{\ell \in \mathcal{I}} \sum_{j=0}^{\ell_c - 1} \kappa_\ell^{(j)} u_i^{(\ell,j)} v^{(\ell)} \tag{8}$$

where the $\alpha_\ell^{(i)}$ values on the LHS can be chosen freely given the RHS. For a given choice of the $\kappa_\ell^{(j)}$'s the above equation can definitely be satisfied if $|\mathcal{J}_i| = k_B$. If we $|\mathcal{J}_i| < k_B$ then this may not be true depending on the values of the $\kappa_\ell^{(j)}$'s.

The $n - k_A k_B$ stragglers together contain $(n - k_A k_B)\Delta_A k_B / n$ uncoded block-columns of $\mathbf{A}$. It is not too hard to see that not all $\mathbf{A}_i$'s that appear within the stragglers appear k_B times within the stragglers (see Appendix F). Thus, the number of $\mathbf{A}_i$'s with $|\mathcal{J}_i| < k_B$ is $\geq (n - k_A k_B)\Delta_A / n + 1$.

In the argument below we only consider the $\alpha_\ell^{(i)}$'s corresponding to these uncoded block-columns and suppose that there is an assignment of $\alpha_\ell^{(i)}$'s that satisfy (8). In this case the problem of finding the corresponding $\kappa_\ell^{(j)}$'s is equivalent to solving a block system of equations described below.

Let $\mathbf{A}_\delta$ be an uncoded block-column that appears less than k_B times in $\mathcal{I}$. The block row corresponding to it (*cf.* (8)) is given by $\tilde{\mathbf{V}} \odot \tilde{\mathbf{U}}$ where

$$\begin{aligned}\tilde{\mathbf{V}} &= [\overbrace{v^{(i_0)} \ \ldots \ v^{(i_0)}}^{\ell_c} | \ \ldots \ | \overbrace{v^{(i_{k-1})} \ \ldots \ v^{(i_{k-1})}}^{\ell_c}], \text{ and} \\ \tilde{\mathbf{U}} &= [u_\delta^{(i_0,0)} \ \ldots \ u_\delta^{(i_0,\ell_c-1)} | \ \ldots \ | u_\delta^{(i_{k-1},0)} \ \ldots \ u_\delta^{(i_{k-1},\ell_c-1)}]\end{aligned}$$

where $\odot$ represents the Khatri-Rao product that corresponds to column-wise Kronecker products.

Appendix E shows that the concatenation of block rows in $\tilde{\mathbf{V}} \odot \tilde{\mathbf{U}}$ corresponding to the different $\mathbf{A}_\delta$'s is such that any $\ell_c k_A k_B \times \ell_c k_A k_B$ matrix is full rank with probability-1. This implies that from the first $\ell_c k_A$ block rows we can decode all the $\kappa_\ell^{(i)}$'s.

On the other hand the equations in (8) need to be satisfied for at least $(n - k_A k_B)\Delta_A / n + 1$ different $\mathbf{A}_i$'s based on the argument above. However

$$\begin{aligned}(n - k_A k_B)\frac{\Delta_A}{n} &= \Delta_A - \frac{k_A k_B \Delta_A}{n} \\ &= k_A \left(\frac{\Delta_A}{k_A} - \frac{\Delta}{n} \right) = \ell_c k_A,\end{aligned}$$

and thus $(n - k_A k_B)\Delta_A / n + 1 > \ell_c k_A$. This implies that there is at least one equation that need to be satisfied with a fixed choice of the $\kappa_\ell^{(i)}$'s. But this probability is zero since each of the remaining equations involve random $u_\delta^{(\ell,i)}$ values that have not appeared in the first $\ell_c k_A$ block rows. ■

Theorem 10: Alg. 6 proposes a distributed matrix-matrix multiplication scheme with $Q = \Delta + (k_B - 1)\ell_c$.

Proof: As in the proof of the previous result, we let $u^{(\ell,j)}$ for $j = 0, \ldots, \ell_c - 1$ denote the j-th random encoding vector for $\mathbf{A}$ in worker W_ℓ and $v^{(\ell)}$ the corresponding random encoding vector for $\mathbf{B}$. We will demonstrate that the system of equations that corresponding to decoding the $\mathbf{A}_i^T \mathbf{B}_j$'s is nonsingular with probability 1. Let e_i denote the i-th unit vector of length Δ_A. For a given $\mathbf{A}_i$, suppose that it appears uncoded in $\mathcal{J}_i$ worker nodes where $|\mathcal{J}_i| \leq k_B$ we obtain certain equations from the uncoded part which correspond to $e_i \otimes v^{(\ell)}$ for $\ell \in \mathcal{J}_i$. If $|\mathcal{J}_i| < k_B$ then it needs to use the coded-coded products for decoding the unknowns corresponding to $\mathbf{A}_i$.

The block system of equations under consideration corresponds to a $\Delta_A k_B \times \Delta_A k_B$ square matrix with random entries. For $\mathbf{A}_i$ such that $|\mathcal{J}_i| = k_B$ the matrix consists of a $k_B \times k_B$ block on the diagonal with k_B distinct vectors $v^{(\ell)}$. This block is nonsingular with probability-1 owing to the random choice of the $v^{(\ell)}$'s.

For the other $\mathbf{A}_i$'s where $|\mathcal{J}_i| < k_B$ we will demonstrate a setting of the $u^{(\ell,j)}$'s such that the entire matrix is a block diagonal matrix with $k_B \times k_B$ blocks of distinct $v^{(\ell)}$ vectors. This demonstrates that there exists a choice of random coefficients for which the system of equations is nonsingular. Following this the result holds with probability-1 when the choice is made at random.

Towards this end, suppose that the pattern of obtained products is such that we get $\Delta - \lambda$ uncoded-coded products and $\lambda + (k_B - 1)\ell_c$ coded-coded products. Without loss of generality we assume that we need to decode the products that involve $\mathbf{A}_0, \mathbf{A}_1, \ldots, \mathbf{A}_{\delta-1}$ using the coded-coded products. Furthermore we suppose that $\mathbf{A}_i$ appears $k_B - \eta_i$ times within the uncoded-coded products, so that $\eta_0 + \eta_1 + \cdots + \eta_{\delta-1} = \lambda$.

Under this setting, there are at least $(k_B - 1)\ell_c + \lambda - (k_B - \eta_0)\ell_c = (\eta_0 - 1)\ell_c + \lambda$ coded-coded products that can be obtained from worker nodes that do not contain an uncoded copy of $\mathbf{A}_0$. Furthermore, these are spread out in at least

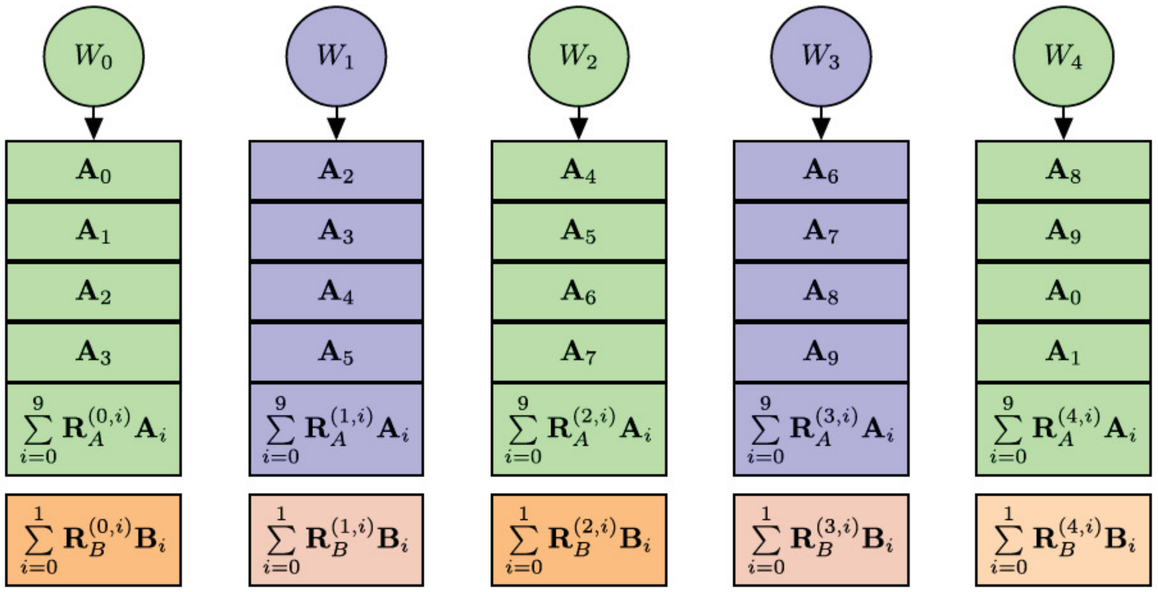
W_0
W_1
W_2
W_3
W_4
$\mathbf{A}_0$ $\mathbf{A}_1$ $\mathbf{A}_2$ $\mathbf{A}_3$ $\sum_{i=0}^{9} \mathbf{R}_A^{(0,i)}\mathbf{A}_i$ $\sum_{i=0}^{1} \mathbf{R}_B^{(0,i)}\mathbf{B}_i$
$\mathbf{A}_2$ $\mathbf{A}_3$ $\mathbf{A}_4$ $\mathbf{A}_5$ $\sum_{i=0}^{9} \mathbf{R}_A^{(1,i)}\mathbf{A}_i$ $\sum_{i=0}^{1} \mathbf{R}_B^{(1,i)}\mathbf{B}_i$
$\mathbf{A}_4$ $\mathbf{A}_5$ $\mathbf{A}_6$ $\mathbf{A}_7$ $\sum_{i=0}^{9} \mathbf{R}_A^{(2,i)}\mathbf{A}_i$ $\sum_{i=0}^{1} \mathbf{R}_B^{(2,i)}\mathbf{B}_i$
$\mathbf{A}_6$ $\mathbf{A}_7$ $\mathbf{A}_8$ $\mathbf{A}_9$ $\sum_{i=0}^{9} \mathbf{R}_A^{(3,i)}\mathbf{A}_i$ $\sum_{i=0}^{1} \mathbf{R}_B^{(3,i)}\mathbf{B}_i$
$\mathbf{A}_8$ $\mathbf{A}_9$ $\mathbf{A}_0$ $\mathbf{A}_1$ $\sum_{i=0}^{9} \mathbf{R}_A^{(4,i)}\mathbf{A}_i$ $\sum_{i=0}^{1} \mathbf{R}_B^{(4,i)}\mathbf{B}_i$

Fig. 12. Matrix-matrix multiplication with $n = 5$ and $s = 1$ with $\gamma_A = \gamma_B = \frac{1}{2}$. Here $\mathbf{R}_A$ and $\mathbf{R}_B$ are random matrices whose superscripts indicate their corresponding rows and columns.

η_0 distinct worker nodes. Next, we pick η_0 encoding vectors for $\mathbf{A}$ from the η_0 distinct workers and set them all to e_0. With this setting we obtain a $k_B \times k_B$ block (corresponding to decoding $\mathbf{A}_0^T\mathbf{B}_j, j = 0, \ldots, \Delta_B - 1$) that consists of distinct $v^{(\ell)}$ vectors that are nonsingular with probability 1.

At this point we are left with $(k_B - 1)\ell_c + \lambda - \eta_0$ coded-coded products. The argument can be repeated for $\mathbf{A}_1$ since there are at least $(\eta_1 - 1)\ell_c + \lambda - \eta_0$ coded-coded products that can be obtained from workers where $\mathbf{A}_1$ does not appear, which in turn correspond to at least η_1 distinct workers. In this case we will set the η_1 encoding vectors to e_1. The process can be continued in this way until the coded-coded products are assigned to each of $\mathbf{A}_0, \mathbf{A}_1, \ldots, \mathbf{A}_{\delta-1}$.

At the end of the process we can claim that we have a block diagonal matrix where each block is a $k_B \times k_B$ square matrix with distinct $v^{(\ell)}$ vectors. Thus, each block and consequently the entire system of equations is nonsingular.

Finally, as there exists a choice of random values that makes the system of equations nonsingular, it continues to be nonsingular with probability 1 under a random choice. ■

To summarize, Theorems 9 and 10 demonstrate that our proposed scheme has the optimal threshold $k_A k_B$ and

$$\begin{aligned}\frac{Q}{\Delta} &= 1 + \frac{(k_B - 1)\ell_c}{\Delta} = 1 + \frac{(k_B - 1)\left(\frac{\Delta_A}{k_A} - \frac{\Delta}{n}\right)}{\Delta} \\ &= 1 + \frac{\Delta(k_B - 1)\left(\frac{1}{k_A k_B} - \frac{1}{n}\right)}{\Delta} \\ &= 1 + \frac{(k_B - 1)s}{n k_A k_B} \approx 1 + \frac{s}{n k_A};\end{aligned}$$

if k_B is significantly larger than 1. Moreover in the practical cases, we usually have $s \ll n k_A$, thus in this SCS optimal scheme, we have $Q/\Delta \approx 1$. Thus the proposed SCS optimal scheme can exploit the partial stragglers quite efficiently irrespective of whether the 'input' matrices are sparse or not. While being resilient to optimal number of stragglers as [3], [17], [21], this is clearly a significant advantage of our propose method over those approaches.

Example 11: We consider an example in Fig. 12 with $n = 5$ and $k_A = k_B = 2$, so the system is resilient $s = 5 - 4 = 1$ straggler. We set $\Delta_A = \text{LCM}(n, k_A) = 10$ and $\Delta_B = k_B = 2$, and in this example, $Q = 21$, thus $Q/\Delta = 1.05$.

VI. Numerical Experiments and Comparisons

In this section, we discuss the results of the numerical experiments for our proposed approaches and compare them with other available methods. First we compare all the approaches in terms of number of stragglers that a scheme can be resilient to, and in terms of Q values. Next we compare the approaches in terms of the worker computation time, overall computation time and numerical stability during the decoding process. Software code for recreating these experiments can be found at [38].

A. Comparison With the Proposed β-Level Coding Scheme

Here we choose examples for distributed matrix-vector and matrix-matrix multiplications; and compare our proposed β-level coding schemes with other available methods in terms of different metrics.

1) Number of Stragglers and Q Value: Table V shows the comparison for matrix-vector multiplication for $n = 30$ workers, each of which can store $\gamma_A = \frac{1}{10}$ fraction of matrix $\mathbf{A}$. For the convolutional code approach, we assume $s = 15$ so that $n - s = 15 > \frac{1}{\gamma_A} = 10$ which satisfies the required condition in [10]. And for the coded at bottom approach, we assume $\gamma_u = \frac{1}{15}$ and $\gamma_c = \frac{1}{30}$, so that $\gamma = \gamma_u + \gamma_c$. Similarly, Table VI shows the comparison for different approaches for matrix-matrix multiplication for

TABLE V

COMPARISON OF NUMBER OF STRAGGLERS, Q VALUES, WORKER COMPUTATION TIME (IN ms) AND WORST CASE CONDITION NUMBER(κ_{worst}) FOR MATRIX-VECTOR MULTIPLICATION FOR $n = 30$ AND $\gamma_A = \frac{1}{10}$ (*FOR CONVOLUTIONAL CODE, WE ASSUMED $s = 15$)

METHODS	STRAGGLERS	$\frac{Q}{\Delta}$ VALUE	WORKER COMPUTATION TIME		κ_{worst}
			SPARSITY 98%	SPARSITY 95%	
POLYNOMIAL CODE [3]	20	N/A	62.8	87.1	5.99×10^9
ORTHO-POLY CODE [21]	20	N/A	62.3	86.4	4.34×10^{11}
RKRP CODE[17]	20	N/A	62.9	86.8	5.44×10^8
CONVOLUTIONAL CODE* [10]	15	N/A	63.1	87.7	6.24×10^4
UNCODED [11]	2	85/30	19.1	32.9	1.7321
UNCODED (PROPOSED)	2	84/30	19.2	33.1	1.7321
β-LEVEL CODING ($\beta = 2$)	4	81/30	25.3	40.2	242.89
β-LEVEL CODING ($\beta = 3$)	6	79/30	29.2	47.9	1.53×10^3
CODED AT BOTTOM	15	58/30	24.1	37.8	1.41×10^3

TABLE VI

COMPARISON OF NUMBER OF STRAGGLERS, Q VALUES, WORKER COMPUTATION TIME (IN SECONDS) AND WORST CASE CONDITION NUMBER (κ_{worst}) FOR MATRIX-MATRIX MULTIPLICATION FOR $n = 18$ AND $\gamma_A = \gamma_B = \frac{1}{3}$ (*FOR CONVOLUTIONAL CODE, $k_A = k_B = 4$)

METHODS	STRAGGLERS	$\frac{Q}{\Delta}$ VALUE	WORKER COMPUTATION TIME		κ_{worst}
			SPARSITY 98%	SPARSITY 95%	
POLYNOMIAL CODE [3]	9	N/A	2.58	10.16	7.33×10^6
ORTHO-POLY CODE [21]	9	N/A	2.51	10.08	1.33×10^7
RKRP CODE[17]	9	N/A	2.63	10.23	2.15×10^5
CONVOLUTIONAL CODE* [10]	2	N/A	2.44	10.19	1.82×10^3
UNCODED (PROPOSED)	1	17/9	0.69	1.96	1.41
β-LEVEL CODING ($\beta_A = \beta_B = 2$)	4	16/9	1.02	3.68	8.89×10^3

$n = 18$ workers, each of which can store $\gamma_A = \frac{1}{3}$ and $\gamma_B = \frac{1}{3}$ fraction of matrices $\mathbf{A}$ and $\mathbf{B}$ respectively. Here we assume $k_A = k_B = 4 > \frac{1}{\gamma_A} = \frac{1}{\gamma_B} = 3$ for the approach in [10].

In case of both matrix-vector and matrix-matrix multiplications, we know that the dense coded approaches [3], [17], [10] and [21] are MDS but they do not consider the partial computations of the slower workers. On the other hand, our proposed approaches are able to utilize the partial computations of the stragglers for both matrix-vector and matrix-matrix multiplications. We can see that the β-level coding approaches, with $\beta = 2$ or 3, have smaller Q/Δ values than the uncoded approaches, one of which is introduced in [11] and the other is a special case of our proposed β-level coding where $\beta = 1$. We emphasize that a larger value of β or a larger value of γ_c will provide smaller values of Q/Δ for our proposed β-level coding approach and the coded-at the bottom scheme, respectively. It should be noted that the approach in [10] requires the condition $n - s > \frac{1}{\gamma}$ to hold to be resilient to s stragglers, so as mentioned in Tables V and VI, this convolutional code-based approach is resilient to less number of stragglers than the other dense coded approaches.

2) Worker Computation Time: Now we compare the worker computation time required by different approaches. Worker computation time indicates the required time for a single worker node to finish all its assigned tasks. We performed the experiments on an Amazon Web Services (AWS) cluster where we choose a `t2.2xlarge` machine as the master node and `t2.small` machines as the worker nodes, which are, in fact, responsible for computing the submatrix products.

For matrix-vector multiplication, We choose a matrix $\mathbf{A}$ of size $40,000 \times 17,640$ and a vector $\mathbf{x}$ of length $40,000$, and the job is to compute $\mathbf{A}^T\mathbf{x}$ in a distributed fashion. We assume that the matrix $\mathbf{A}$ is sparse, which indicates that the most of the entries of $\mathbf{A}$ are $zero$. For example, the sparsity of $\mathbf{A}$ can be 98% (or 95%), which indicates that randomly chosen 2% (or 5%) entries of matrix $\mathbf{A}$ are non-zero. We consider the same scenario where we have $n = 30$ workers, each of which can store $\gamma_A = \frac{1}{10}$ fraction of matrix $\mathbf{A}$. The comparison among different approaches for different sparsity values is shown in Table V. Next a similar experiment is carried out with $n = 18$ workers with $\gamma_A = \gamma_B = \frac{1}{3}$ for matrix-matrix multiplication where both $\mathbf{A}$ and $\mathbf{B}$ are sparse and of sizes 12000×13680 and 12000×10260, respectively, and the corresponding results are shown in Table VI.

From the experimental results shown in Tables V and VI, we can see that the workers require much more time to complete their assigned tasks in case of the dense coded approaches ([3], [17], [10] and [21]) than our proposed approaches. The reason is that the dense coded approaches cannot preserve the sparsity of the matrices $\mathbf{A}$ or $\mathbf{B}$, so the corresponding coded submatrices are quite dense even if A and B are sparse. On the other hand, our proposed approaches can preserve the sparsity in the submatrices, and can complete the jobs $3 \sim 4$ times faster than the available approaches. It should be noted that a smaller value of β or a smaller value of γ_c will lead to less worker computation time for our proposed β-level coding approach and the coded at the bottom scheme, respectively.

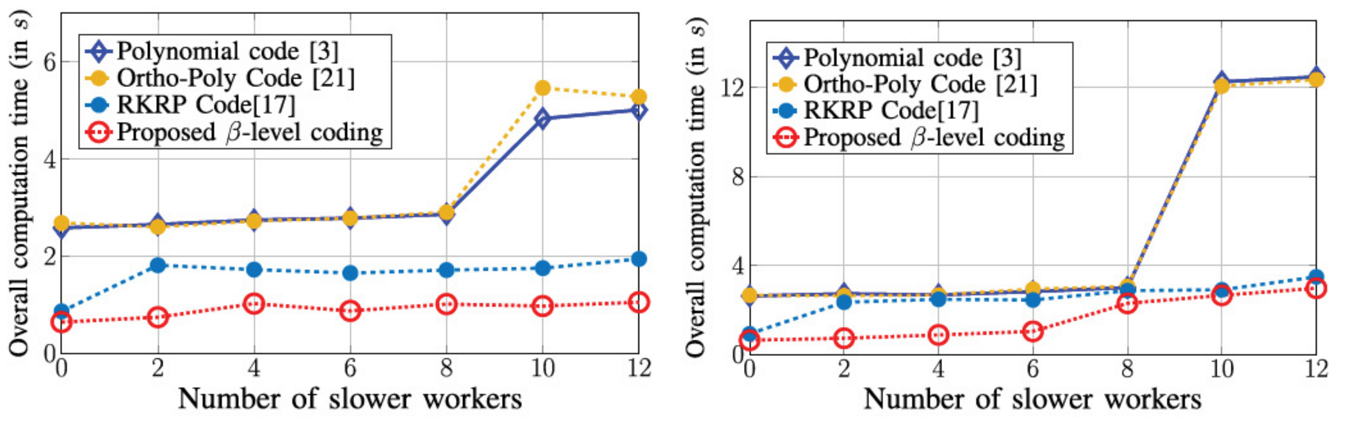

Fig. 13. Comparison among the dense coded approaches and the proposed β-level coding approach (with $\beta = \beta_A \beta_B = 4$) in terms of overall computation time for different number of slower worker nodes when the 'input' matrices are 98% sparse. The system has $n = 18$ worker nodes and the slower workers are simulated in such a way so that they have half (left) or one-fifth (right) of the speed of the non-straggling workers.

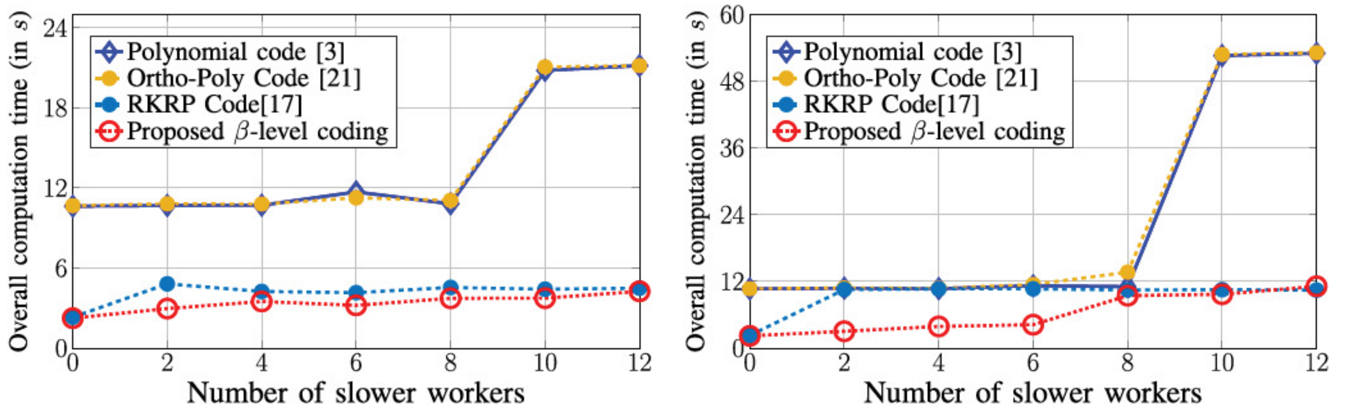

Fig. 14. Comparison among the dense coded approaches and the proposed β-level coding approach (with $\beta = \beta_A \beta_B = 4$) in terms of overall computation time for different number of slower worker nodes when the 'input' matrices are 95% sparse. The system has $n = 18$ worker nodes and the slower workers are simulated in such a way so that they have half (left) or one-fifth (right) of the speed of the non-straggling workers.

We note here that while there is a significant difference between the required time of the dense coded approaches and our proposed approaches, this difference can be much higher. For example, in Table V, we can see that the polynomial code approach is around $3 \sim 4$ times slower than the uncoded approach, but the gap according to the theoretical analysis should be as large as 10 times, since $\gamma_A = 1/10$. The reason underlying the smaller gap is the use of two different commands in `Python` to compute products between the matrix and the vector. Since the proposed uncoded or the β-level coding approaches can preserve the sparsity up to certain level, we have leveraged the sparse matrix-multiplication commands in these cases, whereas for the dense coded approaches which cannot preserve the sparsity, regular matrix-multiplication command provided better results. A more optimized sparse matrix-multiplication scheme could result in bigger multiplicative gaps between these approaches. Furthermore, the difference of the required time would be certainly higher and more significant if the matrix sizes were higher (for example, in millions). However, owing to the memory limitations of the machines that we are using (in this case, `t2.small`), we cannot conduct experiments with such large matrices.

3) Overall Computation Time: We consider the same example of $n = 18$ workers with $\gamma_A = \gamma_B = \frac{1}{3}$, and compare all these matrix-matrix multiplication approaches in terms of overall computation time to recover $\mathbf{A}^T\mathbf{B}$. Overall computation time is the time required by the worker nodes to compute the products so that the master node is able to decode all the unknowns corresponding to $\mathbf{A}^T\mathbf{B}$. It should be noted that the overall computation time is different than the worker computation time discussed above. For example, let these 18 worker nodes in polynomial code approach [3] (having recovery threshold to be 9) require $t_0, t_1, t_2, \ldots, t_{17}$ to compute their respective block-products, then the overall computation time for this approach is t_8.

In Figs. 13 and 14, we compare the overall computation time where the 'input' matrices are 98% and 95% sparse. We simulate the speed of worker nodes such that the slower workers operate at either $\frac{1}{2}$ or $\frac{1}{5}$-th of the speed of the standard worker nodes. From the figures, it is clear that the proposed β-level coding scheme takes significantly less overall computation time than [3], [21] and [17] for different numbers of slower workers. It should be noted that the approaches in [3] and [21] have a spike in overall worker computation time when there are 10 slower worker nodes, since the recovery threshold

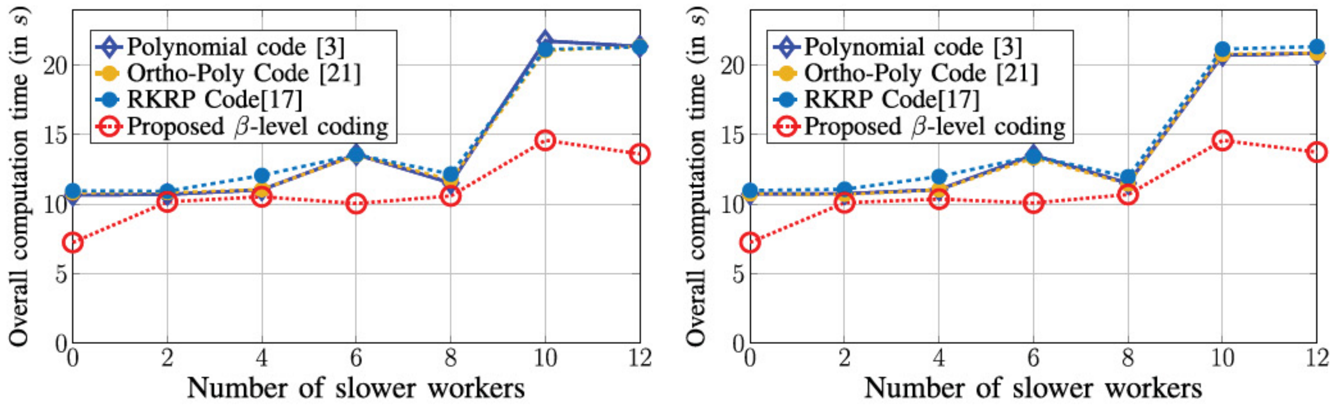

Fig. 15. Comparison among the dense coded approaches and the proposed β-level coding approach (with $\beta = \beta_A \beta_B = 4$) in terms of overall computation time for different number of slower worker nodes when the 'input' matrices are 50% sparse (left) or fully dense (right). The system has $n = 18$ worker nodes and the slower workers are simulated in such a way so that they have half of the speed of the non-straggling workers.

is 9; and they cannot leverage the partial computations. In case of [17], the message workers can leverage the whole sparsity of the 'input matrices' in absence of slower worker nodes. Thus there is a spike in overall worker computation time for [17], in Figs. 13 and 14 when there is a slower worker node.

Next in Fig. 15, we compare the overall computation time where the 'input' matrices are 50% sparse and fully dense. Again, we simulate the speed of worker nodes in such a way so that the standard worker nodes are twice faster than the slower worker nodes. In this figure, we can see the effect of utilizing the partial computations in β-level coding scheme which takes less overall computation time than [3], [21] and [17] for different numbers of slower workers.

4) Numerical Stability: Now we do another experiment to compare the numerical stability of different schemes. We know that for decoding a system of equations, errors in the input can get amplified by the condition number (ratio of maximum and minimum singular values) of the associated decoding matrix; hence, a low condition number is critical [10], [20]. For example, let us consider the polynomial codes [3] for matrix vector multiplication, where each of n workers can store $\gamma = \frac{1}{k}$ fraction of matrix $\mathbf{A}$. Now partitioning $\mathbf{A}$ into $\Delta = k$ submatrices lead to Δ unknowns, $\mathbf{A}_0^T\mathbf{x}, \mathbf{A}_1^T\mathbf{x}, \dots, \mathbf{A}_{\Delta-1}^T\mathbf{x}$. Now in order to assign the coded jobs to n workers, we need to choose a polynomial of degree $k-1$ and n evaluation points, thus the coding matrix is of size $n \times k$. Since the recovery threshold here is $\tau = k$, we are interested in all choices of $k \times k$ submatrices of that $n \times k$ coding matrix. It can be shown that the system will be numerically more stable in the worst case if the evaluation points are chosen uniformly spaced in $[-1, 1]$, rather than choosing the integers $1, 2, \dots, n$ [15]. In other words, choosing interpolation points uniformly spaced in $[-1, 1]$ will lead to a smaller worst case condition number (κ_{worst}).

In this experiment we compare the condition numbers for different approaches in case of the worst choice of full stragglers. Tables V and VI show the comparison of worst case condition numbers (κ_{worst}) for matrix-vector and matrix-matrix multiplication, respectively, for the previously chosen scenario. We can see that the dense coded approaches ([3], [17] and [21]) have a very high worst case condition number, thus suffer from numerical instability which leads to erroneous results. On the other hand, our proposed β-level coding approach has a much smaller worst case condition number. The reason is that even in the worst case, the decoding of some β unknowns depends on a $\beta \times \beta$ system matrix whose entries are randomly chosen. Thus a smaller β leads to a smaller κ_{worst}, for example, we can see that the uncoded case (same as the case with $\beta = 1$) is the scheme having the smallest κ_{worst}.

We point out that in this work, we do not have analytical results on the numerical stability of our schemes; the comparisons are based on exhaustive experiments. A comprehensive analysis of numerical stability of our schemes is part of future work.

B. Comparison With the Proposed SCS Optimal Scheme

In this experiment, we compare the dense coded approaches with our proposed SCS optimal coding scheme in terms of different metrics, for example, Q/Δ values, worker computation time etc.

1) Number of Stragglers and Q Value: First we do the comparison for matrix-vector multiplication where we choose a square sparse matrix $\mathbf{A}$ of size $27,720 \times 27,720$, and a vector $\mathbf{x}$ of length $27,720$. The job is to compute $\mathbf{A}^T\mathbf{x}$ in a distributed system of $n = 18$ workers, each of which can store $\gamma_A = \frac{1}{15}$ fraction of matrix $\mathbf{A}$. We consider two different choices of matrix $\mathbf{A}$. In the first case, $\mathbf{A}$ is a band matrix [39] where the entries are non-zero along the principal diagonal and in 1000 other k-diagonals just above and below the principal diagonal. In the second case, the entries are non-zero along the principal diagonal and in 2000 other randomly chosen k-diagonals. The comparison is shown in Table VII where we can see that all the competing schemes are resilient to $s = n - 1/\gamma_A = 3$ stragglers. While the proposed SCS optimal scheme is also resilient to 3 stragglers, it can also efficiently leverage the partial computations done by the slower workers, since in this scheme, $Q/\Delta = 1$.

TABLE VII
COMPARISON OF Q VALUES, WORKER COMPUTATION TIME (IN ms) AND WORST CASE CONDITION NUMBER (κ_{worst}) FOR MATRIX-VECTOR MULTIPLICATION FOR $n = 18, \gamma_A = \frac{1}{15}$ (*FOR CONVOLUTIONAL CODE, WE ASSUME $\gamma_A = \frac{1}{10}$)

METHODS	NO OF STRAGGLERS	$\frac{Q}{\Delta}$ VALUE	WORKER COMPUTATION TIME		κ_{worst}
			BAND	RANDOM	
POLYNOMIAL CODE [3]	3	N/A	29.7	30.2	4.03×10^7
ORTHO-POLY CODE [21]	3	N/A	30.1	29.8	2.13×10^4
RKRP CODE[17]	3	N/A	29.3	30.0	6.35×10^3
CONVOLUTIONAL CODE* [10]	3	N/A	35.2	34.7	1.21×10^3
SCS OPTIMAL SCHEME	3	1	14.8	20.3	6.81×10^4

TABLE VIII
COMPARISON OF Q VALUES, WORKER COMPUTATION TIME (IN SECONDS) AND WORST CASE CONDITION NUMBER (κ_{worst}) FOR MATRIX-MATRIX MULTIPLICATION FOR $n = 24, \gamma_A = \frac{1}{4}$ AND $\gamma_B = \frac{1}{5}$ (*FOR CONVOLUTIONAL CODE, WE ASSUME $\gamma_A = \frac{2}{5}$ AND $\gamma_B = \frac{1}{3}$). THE VALUES IN THE PARENTHESES FOR THE SCS OPTIMAL SCHEME SHOWS THE TIME REQUIRED FOR UNCODED AND CODED PORTIONS, RESPECTIVELY

METHODS	NO OF STRAGGLERS	$\frac{Q}{\Delta}$ VALUE	WORKER COMPUTATION TIME		κ_{worst}
			SPARSITY 98%	SPARSITY 95%	
POLYNOMIAL CODE [3]	4	N/A	3.11	8.29	2.40×10^{10}
ORTHO-POLY CODE [21]	4	N/A	3.08	8.16	1.96×10^6
RKRP CODE[17]	4	N/A	3.15	8.22	2.83×10^5
CONVOLUTIONAL CODE* [10]	4	N/A	5.16	10.92	2.65×10^4
SCS OPTIMAL SCHEME	4	31/30	1.93 $(0.91 + 1.02)$	4.76 $(2.71 + 2.05)$	4.93×10^6

Next to show an example for distributed matrix-matrix multiplication, we choose two random sparse matrices $\mathbf{A}$ and $\mathbf{B}$ of sizes 12000×15000 and 12000×13500, where randomly chosen any 2% and 5% entries are non-zero. We consider a distributed system having $n = 24$ workers, each of which can store $\gamma_A = \frac{1}{4}$ fraction of matrix $\mathbf{A}$ and $\gamma_B = \frac{1}{5}$ fraction of matrix $\mathbf{B}$. The comparison is shown in Table VII where we can see that, similar to the most of the dense coded approaches [3], [21], [17], our proposed SCS optimal scheme is also resilient to $s = n - k_A k_B$ stragglers, where $\gamma_A = \frac{1}{k_A}$ and $\gamma_B = \frac{1}{k_B}$. Moreover, our proposed approach leverages the partial computations done by the slower workers quite efficiently, since in this scheme, $Q/\Delta = 31/30$, very close to 1. We point out that we did not compare with the approach in [7] since their approach does not respect the storage constraints for the matrices at each worker node and only has a high-probability guarantee on the recovery threshold. Similarly we did not compare with [32] which assumes heterogeneous workers, but we note that this approach provides with a better value of Q/Δ (can be even 1) compared to the proposed SCS optimal scheme for the example shown in Table VIII.

2) Worker Computation Time: We show the comparison of worker computation time for distributed matrix-matrix multiplication in Table VIII. It confirms the superiority of the proposed SCS optimal scheme in terms of workers' computation speeds. The major reason behind the enhancement of the speed in the SCS optimal scheme lies in its ability to leverage the sparsity of the matrices up to certain level, whereas the approaches in [3] or [21] use the dense linear combinations of the submatrices which destroy the sparsity. The approaches in [10] and [17] consider some parity worker nodes where all the assigned submatrices are dense, which leads to high worker computation time for those workers. On the other hand, in the proposed SCS optimal scheme the submatrices, obtained from dense linear combinations, are assigned uniformly within the workers. This removes the asymmetry between the worker node computation times.

Similarly we can see the comparison of worker computation time for distributed matrix-vector multiplication in Table VII. It is clear that the proposed SCS optimal scheme requires less time from the worker nodes in comparison to the other dense coded approaches, which in fact, cannot leverage the sparsity of matrix $\mathbf{A}$.

3) Overall Computation Time: Now we compare all these matrix-matrix multiplication approaches in terms of overall computation time. In Figs. 16 and 17, we show the simulation results where the 'input' matrices are 98% and 95% sparse, respectively. We simulate the speed of worker nodes such that the slower workers operate at either $\frac{1}{2}$ or $\frac{1}{5}$-th of the speed of the standard worker nodes. From the figures, it is clear that the proposed SCS optimal scheme takes significantly less overall computation time than [3] and [21] for different numbers of slower workers. It should be noted that the message workers in the approach [17] do not involve any coding (so the message worker computation time is less), hence there are certain cases (for example, no slower workers) where it takes less overall computation time than the proposed SCS optimal scheme. However, if the slower workers have a very slow speed that the situation changes and the SCS optimal scheme takes less time than [17]. Thus, unlike [17], our proposed approach can be fast even in the presence of full stragglers (failure of worker nodes).

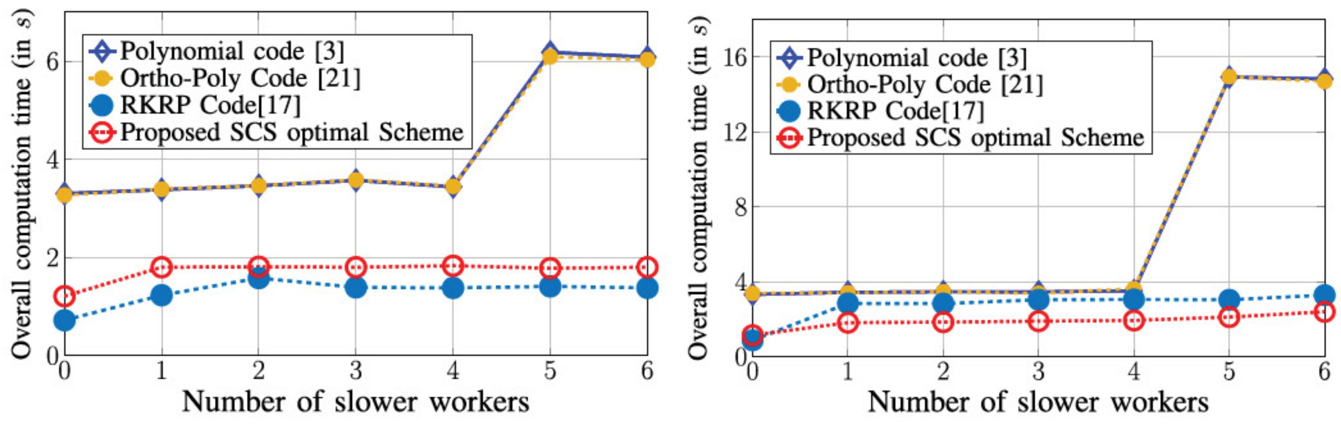

Fig. 16. Comparison among the dense coded approaches and the proposed SCS optimal scheme in terms of overall computation time for different number of slower worker nodes when the 'input' matrices are 98% sparse. The system has $n = 24$ worker nodes and the slower workers are simulated in such a way so that they have half (left) or one-fifth (right) of the speed of the non-straggling workers.

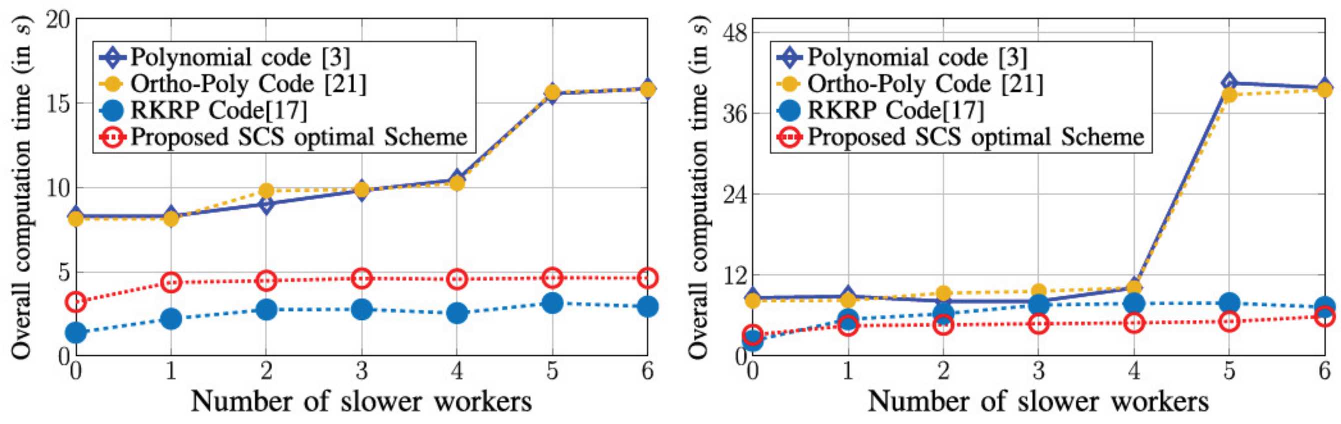

Fig. 17. Comparison among the dense coded approaches and the proposed SCS optimal scheme in terms of overall computation time for different number of slower worker nodes when the 'input' matrices are 95% sparse. The system has $n = 24$ worker nodes and the slower workers are simulated in such a way so that they have half (left) or one-fifth (right) of the speed of the non-straggling workers.

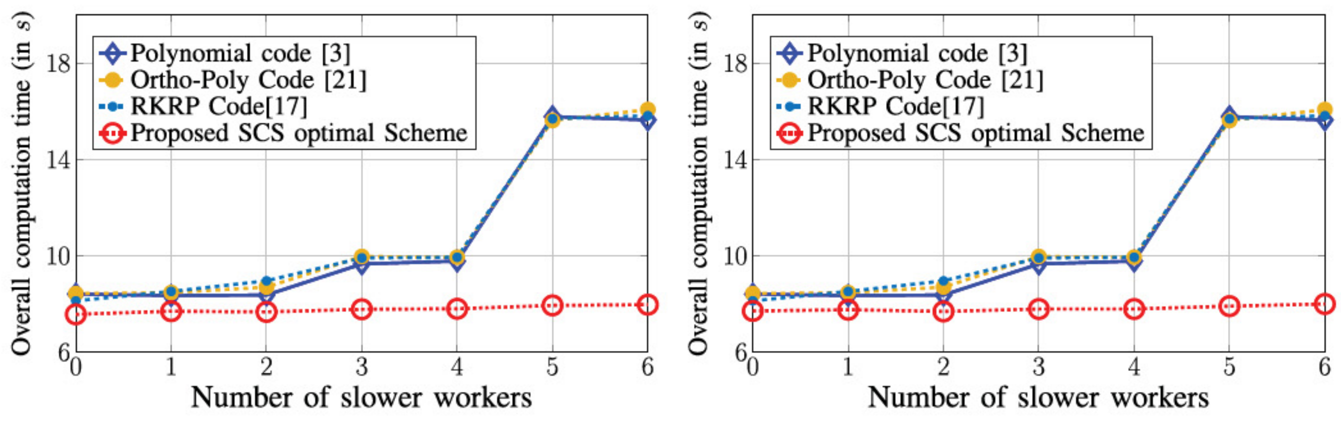

Fig. 18. Comparison among the dense coded approaches and the proposed SCS optimal scheme in terms of overall computation time for different number of slower worker nodes when the 'input' matrices are 50% sparse (left) or fully dense (right). The system has $n = 24$ worker nodes and the slower workers are simulated in such a way so that they have half of the speed of the non-straggling workers.

Next in Fig. 18, we compare the overall computation time where the 'input' matrices are 50% sparse and fully dense. Again, we simulate the speed of worker nodes in such a way so that the standard worker nodes are twice faster than the slower worker nodes. In this figure, we can see the effect of utilizing the partial computations in SCS optimal scheme which takes less overall computation time than [3], [21] and [17] for different numbers of slower workers.

4) Numerical Stability: Tables VII and VIII show the comparisons of worst case condition number (κ_{worst}) among different approaches for matrix-vector and matrix-matrix multiplication, respectively. In both cases, we can see that the

TABLE IX
COMPARISON OF THE Q/Δ VALUES, WORKER COMPUTATION TIME (IN SECONDS) AND WORST CASE CONDITION NUMBERS FOR MATRIX-MATRIX MULTIPLICATION FOR $n = 10, \gamma_A = \gamma_B = \frac{1}{3}$, SO $s = 1$

METHODS	W/O PARTIAL COMPUTATIONS			W/ PARTIAL COMPUTATIONS		
	$\frac{Q}{\Delta}$ VALUE	κ_{worst}	WORKER TIME	$\frac{Q}{\Delta}$ VALUE	κ_{worst}	WORKER TIME
POLY CODE [3]	N/A	8.8×10^3	2.46	1	1.86×10^{13}	7.09
ORTHO-POLY [21]	N/A	16.66	2.49	1	4.33×10^5	7.06
RKRP CODE[17]	N/A	11.96	2.41	1	1.16×10^4	7.14
SCS OPTIMAL SCHEME	-	-	-	1.02	2.15×10^3	2.04

approach in [3] provides very high κ_{worst} values; whereas our proposed SCS optimal scheme has κ_{worst} values in a similar range as [21] and [17]. It indicates that our proposed scheme will be numerically stable even in the worst choice of s stragglers.

It should be noted that in the dense coded approaches, we can decode all $\Delta = k_A k_B$ unknowns from any $k_A k_B$ submatrix block products, and in that sense we have $\frac{Q}{\Delta} = 1$. But it does not necessarily mean that those schemes can utilize the partial computations done by the slower workers, since in those cases the master requires $k_A k_B$ workers to finish their jobs, and discard the computations done by others.

However, one can still use those approaches to utilize the partial computations, by partitioning the matrices into more submatrices. We can consider an example of $n = 10$ workers with $\gamma_A = \gamma_B = 1/3$ and 98% sparse matrices $\mathbf{A}$ and $\mathbf{B}$, both having size $12,000 \times 12,000$. Now we can partition matrix $\mathbf{A}$ into $\Delta_A = 3$ or $\Delta_A = 9$ submatrices for the dense coded approaches. We can see the comparison of κ_{worst} and worker computation time in Table IX for these two values of Δ_A. In case of $\Delta_A = 9$, we will require polynomials of higher degrees (for [3] or [21]) or more random coefficients (for [17]) than in the case of $\Delta_A = 3$. It leads to a very high condition number ($\approx 10^{13}$) which will make the whole system numerically unstable. Besides, a larger Δ_A would make the submatrices even denser, which will lead to higher worker computation time for the workers. The case is similar for the work in [32] which uses larger Δ_A and Δ_B to utilize the partial computations. On the other hand, in the proposed SCS optimal scheme, uncoded submatrices are placed at the top, and coded submatrices are placed at the bottom. Moreover the coded jobs are allocated uniformly among all the workers which does not let the worker computation time go high for any particular worker.

VII. CONCLUSION AND FUTURE WORK

In this work we have presented several coded matrix computation schemes that (i) leverage partial computations by stragglers and (ii) impose constraints on the extent to which coding is allowed in the solution.

The second feature is especially valuable in the practical case of computations with sparse matrices and provides significant reductions in worker node computation time and better numerical stability as compared to the previous schemes. Prior work has demonstrated schemes with optimal recovery threshold in certain cases. We present schemes that match the optimal threshold while enjoying lower worker node computation times and improved numerical stability. Exhaustive numerical experiments corroborate our findings.

There are several opportunities for future work. We have demonstrated that carefully chosen different parallel classes provide improved recovery thresholds and Q/Δ metrics for the matrix-vector problem. We expect that this should help even in the case of matrix multiplication. Schemes that apply for a larger range of storage fractions are also of interest. Analytical results quantifying the numerical stability of our constructions may be an interesting direction to pursue. In this work we defined the value of Q as the worst case number of symbols that allows for recovering the intended result. Analysis and constructions for the random case would be natural extensions.

APPENDIX A
PROPERTIES OF β-LEVEL CODING WHEN $\beta > c$

In Section IV, we have discussed β-level coding for distributed matrix computations when $c \geq \beta$ and here we prove the properties of β-level coding when $\beta > c$. The difference is that the constraint $c \geq \beta$ ensures that we will have at least β worker groups, whereas it is not the case when $\beta > c$.

A. Matrix-Vector Multiplication

Suppose that we have $n = ca_2$ workers, each of which can store $\gamma = \frac{a_1}{a_2}$ fraction of matrix $\mathbf{A}$. To incorporate β-level coding, matrix $\mathbf{A}$ is partitioned into $\Delta = \beta a_2$ block-columns, and thus each worker will be assigned $\ell = \Delta\gamma = \beta a_1$ jobs. It should be noted that we have $\frac{n}{\Delta/\beta} = c$ worker groups among the workers because of the cyclic fashion of job assignments.

Lemma 4: If we use a single parallel class in Alg. 1, then the number of stragglers will be $s = c\ell - \beta$ and we will have

$$Q = c\left[\frac{\Delta}{\beta}\ell - \frac{\ell(\ell+1)}{2}\right] + c\sum_{i=0}^{c_1-1}(\ell - i) + c_2(\ell - c_1) + 1$$

where $c_1 = \left\lfloor \frac{\beta-1}{c} \right\rfloor$ and $c_2 = \beta - 1 - cc_1$.

Proof: The straggler resilience follows similar to the proof of Theorem 2 by counting the number of occurrences of the meta-symbols.

For the Q analysis, assume that there exists a meta-symbol $\star$ that appears at most $\beta - 1$ times among the acquired Q symbols where Q is defined in the theorem statement. We have c worker groups and in each group, $\star$ appears in positions $0, 1, 2, \ldots, \ell - 1$.

Now we know that we can process $\alpha_0 = \left[\frac{\Delta}{\beta}\ell - \frac{\ell(\ell+1)}{2}\right]$ meta-symbols from each of the worker groups without processing $\star$. Any additional processing will necessarily process $\star$. Suppose we choose any particular worker, where the position index of $\star$ is i. In that case, we can acquire at most $\ell - 1 - i$ more symbols from that particular worker without any more appearances of $\star$. Thus, the maximum number of meta-symbols that can be processed for each additional appearance of $\star$ can be expressed by the following vector.

$$\mathbf{z} = (\underbrace{\ell, \ell, \dots, \ell}_{c}, \underbrace{\ell-1, \dots, \ell-1}_{c}, \dots, \underbrace{1, 1, \dots, 1}_{c}).$$

Here $\mathbf{z}$ is a non-increasing sequence, so in order to obtain the maximum number of symbols where the meta-symbol $\star$ appears at most $\beta - 1$ times, we need to acquire symbols sequentially as mentioned in $\mathbf{z}$. Let $c_1 = \left\lfloor \frac{\beta-1}{c} \right\rfloor$ and $c_2 = \beta - 1 - cc_1$. Thus we can choose the first $cc_1 + c_2 = \beta - 1$ workers (as mentioned in $\mathbf{z}$) so that we can have $Q^{'}$ symbols where $\star$ appears exactly $\beta - 1$ times, so

$$Q^{'} = c\alpha_0 + c\sum_{i=0}^{c_1-1}(\ell - i) + c_2(\ell - c_1);$$

which indicates that $Q = Q^{'} + 1$ symbols ensures that $\star$ will appear at least β times. This leads to a contradiction and concludes the proof. ■

B. Matrix-Matrix Multiplication

The argument is almost the same for the matrix-matrix case with appropriate definitions for ℓ and β. Specifically, recall that $n = c \times a_2 b_2$, and $\Delta_A = \beta_A a_2$ and $\Delta_B = \beta_B b_2$. Thus, we have $\frac{n}{\Delta/\beta} = c$ worker groups, where $\Delta = \Delta_A \Delta_B$ and $\beta = \beta_A \beta_B$. In each worker, we assign $\ell_A = \Delta_A \gamma_A$ and $\ell_B = \Delta_B \gamma_B$ coded submatrices of $\mathbf{A}$ and $\mathbf{B}$, respectively and set $\ell = \ell_A \ell_B$. Following this, we can obtain the number of stragglers as $s = c\ell - \beta$ and

$$Q = c\left[\frac{\Delta}{\beta}\ell - \frac{\ell(\ell+1)}{2}\right] + c\sum_{i=0}^{c_1-1}(\ell - i) + c_2(\ell - c_1) + 1.$$

Appendix B
Proof of Theorem 4

Proof: **Straggler Resilience:** To prove the straggler resilience, we note that if there are at $2\ell - 1$ stragglers it is evident that $\mathbf{G}_{dec}$ formed by the remaining meta-symbols is such that each unknown has degree at least one. Let X_i and X_j denote the subset of worker nodes where unknowns $\mathbf{A}_i^T\mathbf{x}$ and $\mathbf{A}_j^T\mathbf{x}$ appear within a meta-symbol, so that $|X_i| = |X_j| = 2\ell$. Furthermore, $|X_i \cap X_j| \leq 2\ell - 2$. To see this we note that if $\{i, j\}$ appear together w.l.o.g. in $\mathcal{G}_0$ then $|X_i \cap X_j| = \ell + \ell - 2$ as this implies that they appear together in exactly $\ell - 2$ workers in $\mathcal{G}_1$ (since $\ell \leq \Delta/2 - 2$). On the other hand if i and j do not appear together in either $\mathcal{G}_0$ or $\mathcal{G}_1$ then they appear together in the workers of each group at most $\ell - 1$ times, so the claim holds. Thus,

$$\begin{aligned}|X_i \cup X_j| &= |X_i| + |X_j| - |X_i \cap X_j| \\ &\geq 2\ell + 2.\end{aligned}$$

Now suppose by way of contradiction that we have two unknowns $\mathbf{A}_i^T\mathbf{x}$ and $\mathbf{A}_j^T\mathbf{x}$ (where $i < j$) both of which appear exactly once across the remaining $n - 2\ell + 1$ workers. The preceding argument shows that if $2\ell - 1$ workers are stragglers then unknowns $\mathbf{A}_i^T\mathbf{x}$ or $\mathbf{A}_j^T\mathbf{x}$ or both appear in at least three nodes, i.e., at least one of them appears at least twice. This contradicts our original assumption. By Lemma 2 the decoding is successful.

Value of Q: Note that $\alpha_0 = \frac{\Delta}{2}\ell - \frac{\ell(\ell+1)}{2}$ denotes the maximum number of meta-symbols that can be processed within a group such that a specific meta-symbol is not processed (*cf.* Lemma 1). This implies that at most $2\alpha_0$ meta-symbols can be processed without processing any specific unknown. Let ρ_0 and ρ_1 denote the number of meta-symbols processed in the two groups $\mathcal{G}_0$ and $\mathcal{G}_1$ where we assume w.l.o.g. that $\rho_0 \geq \rho_1$.

- Case 1: Suppose that $\rho_0 \geq \alpha_0 + \ell + 1$. Lemma 1 implies that each meta-symbol $\in \mathcal{P}_0$ is processed at least twice in $\mathcal{G}_0$. Then by Lemma 2, the decoding is successful.
- Case 2: If $\alpha_0 + 2 \leq \rho_0 \leq \alpha_0 + \ell$, we claim that at most one meta-symbol in $\mathcal{G}_0$ is processed once. The other meta-symbols are processed at least twice. To see this, consider two meta-symbols $(2i, 2i+1)$ and $(2j, 2j+1)$ in $\mathcal{G}_0$ such that $j > i$ such that $(2i, 2i+1)$ is processed only once. If $j - i \geq 2$ then there are at least two workers in $\mathcal{G}_0$ where the meta-symbol $(2j, 2j+1)$ appears but $(2i, 2i+1)$ does not. Therefore, if at least $\alpha_0 + 1$ meta-symbols are processed in $\mathcal{G}_0$, then $(2j, 2j+1)$ appears at least twice. On the other hand if $j = i + 1$ then there is only one worker where $(2j, 2j+1)$ appears but $(2i, 2i+1)$ does not. Thus, if $\alpha_0 + 1$ meta-symbols are processed then we have processed $(2j, 2j+1)$ at least once. The $\alpha_0 + 2$-th meta-symbol cannot be $(2i, 2i+1)$ since by assumption it is processed only once, thus it has to be $(2j, 2j+1)$ (since $j = i + 1$).
  Now, we argue either unknown $\mathbf{A}_{2i}^T\mathbf{x}$ or $\mathbf{A}_{2i+1}^T\mathbf{x}$ appear within the meta-symbols in $\mathcal{G}_1$. Towards this end, we note that there are exactly two workers in $\mathcal{G}_1$ where $\mathbf{A}_{2i+1}^T\mathbf{x}$ appears but $\mathbf{A}_{2i}^T\mathbf{x}$ does not. Therefore, at most $\alpha_0 - (2\ell - 1)$ meta-symbols can be processed in $\mathcal{G}_1$ while avoiding both the unknowns $\mathbf{A}_{2i}^T\mathbf{x}$ and $\mathbf{A}_{2i+1}^T\mathbf{x}$.
  This implies that the total number of meta-symbols that can be processed such that at least two unknowns appear only once in G_{dec} is at most $2\alpha_0 - \ell + 1 < Q$.
- Case 3: If $\rho_0 = \alpha_0 + 1$, then we can have two meta-symbols $(2i, 2i+1)$ and $(2i+2, 2i+3)$ that appear exactly once in $\mathcal{G}_0$. It can be verified that none of the unknowns $\mathbf{A}_{2i}^T\mathbf{x}, \dots, \mathbf{A}_{2i+3}^T\mathbf{x}$ appear together in a meta-symbol in $\mathcal{G}_1$ since $\Delta \geq 8$. Thus, if we process $\rho_1 = \alpha_0$ symbols in $\mathcal{G}_1$, then we can avoid at most one unknown from the set $\{\mathbf{A}_{2i}^T\mathbf{x}, \dots, \mathbf{A}_{2i+3}^T\mathbf{x}\}$. It follows that at most one unknown appears once in G_{dec} and by Lemma 2, the decoding is successful.

■

Appendix C
Proof of Theorem 7

Proof: We need to show that for any pattern of Q symbols the master node can decode $\mathbf{A}^T\mathbf{x}$. Towards this end, from

Theorem 2 (setting $\beta = 1$ and $\ell = \ell_u = r_u$), we know that any pattern of Q uncoded symbols allows the recovery of all Δ unknowns. In other words for any computation state vector $\mathbf{w}(t) = [w_0(t)\ w_2(t)\ \dots\ w_{n-1}(t)]$ such that $w_i(t) \le \ell_u$ and $\sum_{i=0}^{n-1} w_i(t) \ge Q$, the master node can decode. Now, consider a vector $\mathbf{w}'(t)$ such that (w.l.o.g.) $w'_0(t), \dots, w'_{\alpha-1}(t) \ge \ell_u + 1$ and $w'_\alpha(t), \dots, w'_{n-1}(t) \le \ell_u$ and $\sum_{i=0}^{n-1} w'_i(t) \ge Q$, i.e., the first α worker nodes process coded blocks whereas the others do not. It is not too hard to determine a different vector $\tilde{\mathbf{w}}(t)$ with the following properties.

$$\tilde{w}_i(t) = \begin{cases} \ell_u & 1 \le i \le \alpha, \\ w'_i(t) + \beta_i & \alpha + 1 \le i \le n, \end{cases}$$

where β_i's are positive integers such that $w'_i(t) + \beta_i \le \ell_u$ and $\sum_{i=0}^{n-1} \tilde{w}_i(t) = Q$. Thus, $\tilde{\mathbf{w}}(t)$ corresponds to a pattern of Q uncoded blocks that recovers Δ distinct blocks.

Now, we compare the vectors $\mathbf{w}'(t)$ and $\tilde{\mathbf{w}}(t)$. Let the uncoded symbols in $\mathbf{w}'(t)$ be denoted by the set $\mathcal{A}$. Then the set of uncoded symbols in $\tilde{\mathbf{w}}(t)$ can be expressed as $\mathcal{A} \cup \mathcal{B}$ where the set $\mathcal{B}$ results from the transformation above. It is evident that for computation state vector $\mathbf{w}'(t)$ the master node has $\sum\limits_{i=0}^{\alpha-1} (w'_i(t) - \ell_u)$ equations with $\Delta - |\mathcal{A}|$ variables. Now,

$$\sum_{i=0}^{\alpha-1} (w'_i(t) - \ell_u) \ge |\mathcal{B}| \ge |\mathcal{B} \setminus \mathcal{A}| = \Delta - |\mathcal{A}|.$$

In particular, this establishes that we have at least as many equations as variables. Since any square submatrix of a random matrix is invertible with probability 1, we have the required result.

Next, we establish the straggler resilience of our scheme. Consider worker nodes $0 \le i_1 < i_2 < \dots < i_k \le n-1$; each of these worker nodes has ℓ_u uncoded symbols. Consider the case that $i_t - i_{t-1} < \ell_u$ for $t = 2, 3, \dots, k$. We claim that these worker nodes contain at least $\min(\ell_u + k - 1, \Delta)$ distinct uncoded symbols. To see this we proceed inductively. Let X_{i_j} denote the symbols in worker i_j. If $k = 2$, then $|X_{i_1} \cup X_{i_2}| = |X_{i_1}| + |X_{i_2}| - |X_{i_1} \cap X_{i_2}| \ge 2\ell_u - (\ell_u - 1) = \ell_u + 1$. We assume the inductive hypothesis, i.e, $|X_{i_1} \cup \dots \cup X_{i_{k-1}}| \ge \min(\ell_u + k - 2, \Delta)$.

Now consider $|X_{i_1} \cup \dots \cup X_{i_{k-1}} \cup X_{i_k}|$. It can be observed that if $i_{k-1} + \ell_u - 1 - \Delta < i_1$ then there exists at least one symbol in X_{i_k} that does not exist in $X_{i_1} \cup \dots \cup X_{i_{k-1}}$. Thus, in this case $|X_{i_1} \cup \dots \cup X_{i_{k-1}} \cup X_{i_k}| \ge \ell_u + k - 1$.

On the other hand if $i_{k-1} + \ell_u - 1 - \Delta \ge i_1$ then $X_{i_k} \subseteq X_{i_{k-1}} \cup X_{i_1}$. Let δ be the smallest integer such that $i_\delta + \ell_u - 1 - \Delta \ge i_1$ but $i_{\delta-1} + \ell_u - 1 - \Delta < i_1$. In this case, we have $|X_{i_1} \cup X_{i_2} \dots \cup X_{i_{\delta-1}}| = \ell_u + \sum_{x=2}^{\delta-1} (i_x - i_{x-1})$. Furthermore, X_{i_δ} contributes another $\Delta + i_1 - (i_{\delta-1} + \ell_u - 1) - 1$ symbols so that

$$\begin{aligned}|X_{i_1} \cup \dots \cup X_{i_{\delta-1}} \cup X_{i_\delta}| &= \ell_u + \sum_{x=2}^{\delta-1} (i_x - i_{x-1}) + \Delta \\ &\quad + i_1 - (i_{\delta-1} + \ell_u - 1) - 1 = \Delta.\end{aligned}$$

Thus, there is nothing to prove in this case.

On the other hand, suppose that $1 \le \alpha \le k$ is the least value such that $i_\alpha - i_{\alpha-1} \ge \ell_u$. In this case, we know from the above claim that $|X_{i_1} \cup \dots \cup X_{i_{\alpha-1}}| \ge \ell_u + \alpha - 2$. It follows that $X_{i_\alpha}, \dots X_{i_k}$ each contribute at least one new symbol, namely $i_\alpha, \dots, i_k$. Therefore $|X_{i_1} \cup \dots \cup X_{i_k}| \ge \ell_u + \alpha - 2 + k - \alpha + 1 = \ell_u + k - 1$.

Thus, if we think about choosing k workers, then we need to ensure that

$$\ell_u + (k-1) + k(\ell - \ell_u) \ge \Delta$$

which further implies

$$k \ge \frac{n - \ell_u + 1}{\ell - \ell_u + 1} = \frac{n - n\gamma_u + 1}{n\gamma - n\gamma_u + 1}$$

as $n = \Delta$. So, if the system is resilient to s stragglers then

$$s \le \left\lfloor n - \frac{n - n\gamma_u + 1}{n\gamma - n\gamma_u + 1} \right\rfloor = \left\lfloor \frac{n^2\gamma_c + n\gamma_u - 1}{n\gamma_c + 1} \right\rfloor.$$

It should be noted that setting $\gamma_c = 0$ leads to the uncoded case which is resilient to $(n\gamma - 1)$ workers (same as setting $\beta = 1$ in Theorem 2). ■

Appendix D
Proof of Theorem 8

Proof: To prove the theorem by contradiction, we assume that there exists an unknown $\mathbf{A}_i^T\mathbf{B}_j$, which cannot be decoded from a particular set of τ workers where τ is defined in the theorem statement. We consider the set $\mathcal{B}_j = \{\mathbf{A}_0^T\mathbf{B}_j, \mathbf{A}_1^T\mathbf{B}_j, \dots, \mathbf{A}_i^T\mathbf{B}_j, \dots, \mathbf{A}_{\Delta_A-1}^T\mathbf{B}_j\}$, i.e, the set of all unknowns corresponding to $\mathbf{B}_j$, for $j = 0, 1, \dots, \Delta_B - 1$, thus $|\mathcal{B}_j| = \Delta_A = a_2$. It should be noted that the equations consisting of the unknowns of $\mathcal{B}_j$ are disjoint with the equations consisting of the unknowns of $\mathcal{B}_m$, $(j \ne m)$ since the assigned submatrices from $\mathbf{B}$ are uncoded.

Let $\mathcal{S}_j$ denote the set of workers where $\mathbf{B}_j$ does not appear in the assignments and $\mathcal{T}_j$ denote the set of workers where it appears. According to the scheme in Alg. 4, there are ma_2b_1 workers each of which has an uncoded copy of $\mathbf{B}_j$. Thus, $|\mathcal{S}_j| = n - ma_2b_1$.

Next, partition the workers of $\mathcal{T}_j$ into $\ell_B = mb_1$ worker groups, within each of which, all a_2 uncoded block-columns of $\mathbf{A}$ appear in a cyclic fashion. From the proof of Theorem 7, we know that any k workers within a group will provide $\min(a_u + k - 1, a_2)$ uncoded symbols corresponding to $\mathcal{B}_i$. Now we have ℓ_B such worker groups which indicates that we have ℓ_B workers of $\mathcal{T}_j$ which have the same uncoded job assignments. Thus, from any κ workers of $\mathcal{T}_j$, we will obtain $\min(a_u + \left\lceil \frac{\kappa}{mb_1} \right\rceil - 1, a_2)$ uncoded symbols, and κa_c coded symbols. So in order to be able to decode the elements of $\mathcal{B}_j$, we need to find the minimum positive integer for κ (which is denoted as κ_{min}) such that

$$a_u + \left\lceil \frac{\kappa}{mb_1} \right\rceil - 1 + \kappa a_c \ge a_2.$$

It indicates that any κ_{min} workers of $\mathcal{T}_j$ are enough to recover all the elements of $\mathcal{B}_j$ including $\mathbf{A}_i^T\mathbf{B}_j$. But $\tau - |\mathcal{S}_j| = \kappa_{min}$, which leads to a contradiction and hence concludes the proof. ■

APPENDIX E
CONCATENATION OF BLOCK ROWS IN $\tilde{\mathbf{V}} \odot \tilde{\mathbf{U}}$

Let $\mathbf{U}$ denote a $\ell_c k_A \times \ell_c k_A k_B$ matrix whose δ-th row is given by $[u_\delta^{(i_0,0)} \quad \dots u_\delta^{(i_0,\ell_c-1)}| \dots |u_\delta^{(i_{k-1},0)} \quad \dots \quad u_\delta^{(i_{k-1},\ell_c-1)}]$ where we recall that each entry of $\mathbf{U}$ is chosen i.i.d. at random from a continuous distribution and $k = k_A k_B$. The matrix $\mathbf{U}$ can be written as

$$\mathbf{U} = [\mathbf{U}_0 \mid \mathbf{U}_1 | \dots |\mathbf{U}_{k-1}]$$

where each $\mathbf{U}_j$ is of dimension $\ell_c k_A \times \ell_c$. We wish to show that

$$\left[\mathbf{U}_0 \otimes v^{(i_0)} | \mathbf{U}_1 \otimes v^{(i_1)} | \dots | \mathbf{U}_{k-1} \otimes v^{(i_{k-1})}\right]$$

is full-rank with probability 1.

Note that the vectors $v^{(i_\ell)}$'s are also chosen at random and any collection of k_B such vectors is full rank with probability 1. In the argument below we show a specific choice of $\mathbf{U}$ that yields a full-rank matrix. This implies that the matrix continues to be full-rank under the random choice. Towards this end, we pick the first ℓ_c rows of $\mathbf{U}$ to be

$$\left[\mathbf{I}_{\ell_c} \quad \dots \quad \mathbf{I}_{\ell_c} \quad 0 \quad \dots \quad 0\right],$$

i.e., the first k_B block-columns are identity matrices. It can be seen that these result in $\ell_c k_B$ linearly independent rows. The next block row of $\mathbf{U}$ is a k_B block-column shifted version of the first block row, i.e., it is

$$\left[0 \quad \dots \quad 0 \quad \mathbf{I}_{\ell_c} \quad \dots \quad \mathbf{I}_{\ell_c} \quad 0 \quad \dots \quad 0\right].$$

This yields another $\ell_c k_B$ linearly independent block rows. This process can be repeated k_A times to provide the required result.

APPENDIX F
NUMBER OF $\mathbf{A}_i$'S THAT APPEAR LESS THAN k_B TIMES WITHIN THE STRAGGLERS

In the setting of Theorem 6, suppose that we have $n - k_A k_B$ stragglers that together contain $(n - k_A k_B)\Delta_A k_B / n$ uncoded block-columns of $\mathbf{A}$. We want to show that not all $\mathbf{A}_i$'s appear k_B times within the stragglers. To see this consider a bipartite graph that specifies the placement of the uncoded block-columns of $\mathbf{A}$. It contains vertices denoting the $\mathbf{A}_i$'s and the worker nodes. An edge connects $\mathbf{A}_i$ and W_j if $\mathbf{A}_i$ appears in W_j. Thus each $\mathbf{A}_i$ has degree k_B. It can be seen that this graph is connected as any two neighboring workers W_j and W_{j+1} (indices reduced modulo-n) have block-columns in common. Suppose that the stragglers are such that each $\mathbf{A}_i$ that appears within the stragglers also appears k_B times within the stragglers. This implies that the subgraph induced by the stragglers is such that it disconnected from the remaining workers. This is a contradiction.

ACKNOWLEDGMENT

The authors acknowledge interesting conversations with Dr. Li Tang and his participation in [11].

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

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

[4] 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.

[5] S. Kiani, N. Ferdinand, and S. C. Draper, "Exploitation of stragglers in coded computation," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2018, pp. 1988–1992.

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

[7] S. Wang, J. Liu, and N. Shroff, "Coded sparse matrix multiplication," in *Proc. 35th Int. Conf. Mach. Learn. (ICML)*, 2018, pp. 5152–5160.

[8] 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.

[9] E. Ozfatura, S. Ulukus, and D. Gunduz, "Distributed gradient descent with coded partial gradient computations," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2019, pp. 3492–3496.

[10] 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.

[11] A. B. Das, L. Tang, and A. Ramamoorthy, "C^{3}LES: Codes for coded computation that leverage stragglers," in *Proc. IEEE Inf. Theory Workshop (ITW)*, Nov. 2018, pp. 1–5.

[12] A. Ramamoorthy, L. Tang, and P. O. Vontobel, "Universally decodable matrices for distributed matrix-vector multiplication," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2019, pp. 1777–1781.

[13] 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.

[14] 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.

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

[16] 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.

[17] A. M. Subramaniam, A. Heidarzadeh, and K. R. Narayanan, "Random Khatri-Rao-product codes for numerically-stable distributed matrix multiplication," in *Proc. 57th Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Sep. 2019, pp. 253–259.

[18] A. B. Das and A. Ramamoorthy, "Distributed matrix-vector multiplication: A convolutional coding approach," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2019, pp. 3022–3026.

[19] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Straggler mitigation in distributed matrix multiplication: Fundamental limits and optimal coding," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2018, pp. 2022–2026.

[20] A. Ramamoorthy and L. Tang, "Numerically stable coded matrix computations via circulant and rotation matrix embeddings," *IEEE Trans. Inf. Theory*, early access, Dec. 21, 2021, doi: 10.1109/TIT.2021.3137266.

[21] M. Fahim and V. R. Cadambe, "Numerically stable polynomially coded computing," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2019, pp. 3017–3021.

[22] A. M. Subramaniam, A. Heidarzadeh, A. K. Pradhan, and K. R. Narayanan, "Product Lagrange coded computing," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2020, pp. 197–202.

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

[24] M. Y. Rosenbloom and M. A. Tsfasman, "Codes for the m-metric," *Probl. Inf. Transm.*, vol. 33, no. 1, pp. 45–52, 1997.

[25] A. Ganesan and P. O. Vontobel, "On the existence of universally decodable matrices," *IEEE Trans. Inf. Theory*, vol. 53, no. 7, pp. 2572–2575, Jul. 2007.

[26] D. Stinson, *Combinatorial Designs: Constructions and Analysis*. Springer, 2007.

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

[28] 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.

[29] B. Hasircioglu, J. Gomez-Vilardebo, and D. Gunduz, "Speeding up private distributed matrix multiplication via bivariate polynomial codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2021, pp. 1853–1858.

[30] Y. Keshtkarjahromi, Y. Xing, and H. Seferoglu, "Dynamic heterogeneity-aware coded cooperative computation at the edge," in *Proc. IEEE 26th Int. Conf. Netw. Protocols (ICNP)*, Sep. 2018, pp. 23–33.

[31] E. Vedadi and H. Seferoglu, "Adaptive coding for matrix multiplication at edge networks," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2021, pp. 1064–1069.

[32] B. Hasircioglu, J. Gomez-Vilardebo, and D. Gunduz, "Bivariate Hermitian polynomial coding for efficient distributed matrix multiplication," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2020, pp. 1–6.

[33] A. Tripathy and A. Ramamoorthy, "Sum-networks from incidence structures: Construction and capacity analysis," *IEEE Trans. Inf. Theory*, vol. 64, no. 5, pp. 3461–3480, May 2018.

[34] O. Olmez and A. Ramamoorthy, "Fractional repetition codes with flexible repair from combinatorial designs," *IEEE Trans. Inf. Theory*, vol. 62, no. 4, pp. 1565–1591, Apr. 2016.

[35] K. Konstantinidis and A. Ramamoorthy, "Resolvable designs for speeding up distributed computing," *IEEE/ACM Trans. Netw.*, vol. 28, no. 4, pp. 1657–1670, Aug. 2020.

[36] M. Hall, *Combinatorial Theory*. Hoboken, NJ, USA: Wiley, 1998.

[37] F. N. Cole, "Kirkman parades," *Bull. Amer. Math. Soc.*, vol. 28, no. 9, pp. 435–437, 1922.

[38] *Leveraging Partial Stragglers Codes*. Accessed: Feb. 23, 2022. [Online]. Available: https://github.com/anindyabijoydas/LeveragePartialStragglers

[39] K. E. Atkinson, *An Introduction to Numerical Analysis*. Hoboken, NJ, USA: Wiley, 2008.

Anindya Bijoy Das (Graduate Student Member, IEEE) received the B.Sc. degree in electrical and electronic engineering from the Bangladesh University of Engineering and Technology, Dhaka, in 2014, and the M.Eng. degree in electrical engineering from Iowa State University, Ames, IA, USA, in 2018. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering (ECpE), Iowa State University. His research interests include coding theory and machine learning. He was a recipient of the Teaching Excellence Award in Fall 2020 and the Research Excellence Award in Fall 2021 from the ECpE Department, 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, Los Angeles (UCLA).

He is currently a Professor of electrical and computer engineering and (by courtesy) of mathematics at Iowa State University. His research interests include classical/quantum information theory and coding techniques with applications to distributed computation, content distribution networks, and machine learning. He was a recipient of the 2020 Mid-Career Achievement in Research Award, the 2019 Boast-Nilsson Educational Impact Award, 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.