Coded sparse matrix computation schemes that leverage partial stragglers

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

2021 IEEE International Symposium on Information Theory (ISIT) | 978-1-5386-8209-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/ISIT45174.2021.9518059

***Abstract*—Coded matrix computation utilizes concepts from erasure coding to mitigate the effect of slow worker nodes (stragglers) in the distributed setting. While this is useful, there are issues with applying, e.g., MDS codes in a straightforward manner for this problem. Several practical scenarios involve sparse matrices. MDS codes typically require dense linear combinations of submatrices of the original matrices which destroy their inherent sparsity; this leads to significantly higher worker computation times. Moreover, treating slow nodes as erasures ignores the potentially useful partial computations performed by them. 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 support our findings.**

I. Introduction

Coded computation is an emerging area that uses ideas from coding theory to make distributed computation resilient to stragglers (slow or failed workers). In particular, ideas from MDS codes have been successfully used for distributed matrix computations [1], [2], [3], [4], [5], [6]. For these systems, we define a so-called recovery threshold which is the minimum value of τ, such that the master node can recover the desired result as long as *any* τ out of n workers complete their tasks.

While these are interesting ideas, there are certain issues that are ignored in the majority of prior work (see [7], [8], [9], [10], [11], [12] for some exceptions). Firstly, several practical cases of matrix computations involve sparse matrices. Using MDS coding strategies in a straightforward manner will often destroy the sparsity of the encoded matrices being processed by the worker nodes. In fact, this can cause the overall job execution time to actually go up rather than down [9]. Secondly, in the distributed computation setting, we observe that it is possible to leverage partial computations performed by the stragglers. Thus, a slow worker may not necessarily be a useless worker.

In this work, we propose distributed matrix multiplication schemes which can exploit slow workers by utilizing their partially finished tasks. In particular for computing $\mathbf{A}^T\mathbf{B}$, we partition matrices $\mathbf{A}$ and $\mathbf{B}$ into Δ_A and Δ_B block-columns which leads to $\Delta = \Delta_A\Delta_B$ pairwise block-products of the form $\mathbf{A}_i^T\mathbf{B}_j, i \in [\Delta_A], j \in [\Delta_B]$ ($[m]$ denotes $\{0, \ldots, m-1\}$). For any time t, we let $w_i(t)$ represent the state of computation of the i-th worker node, $(0 \leq w_i(t) \leq \ell)$, which represents the number of tasks that have been processed by worker node i (where ℓ is the total number of assigned jobs to that worker). Thus, our system requirement states that as long as $\sum_{i=0}^{n-1} w_i(t) \geq Q$, the master node should be able to determine $\mathbf{A}^T\mathbf{B}$, and our objective is to minimize the value of Q/Δ. This formulation (introduced in our prior work [13] for matrix-vector multiplication) subsumes treating stragglers as non-working nodes. 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 can be much lower than competing schemes [3], [14], [15] and [16]. Thus, our schemes ensure that the encoded matrices are not much denser than the original matrices. This is especially useful in the case of sparse matrices that often appear in practical settings. Owing to space limitations, most of the proofs appear in the long version of the paper [17].

This work was supported in part by the National Science Foundation (NSF) under Grant CCF-1718470 and Grant CCF-1910840.

II. Preliminaries

Suppose that a given worker node is assigned encoded block-columns $\tilde{\mathbf{A}}_i, i \in [\ell_A]$ and $\tilde{\mathbf{B}}_j, j \in [\ell_B]$. The assignment specifies a top to bottom order to compute $\ell = \ell_A\ell_B$ block-products, $\tilde{\mathbf{A}}_0^T\tilde{\mathbf{B}}_0, \tilde{\mathbf{A}}_0^T\tilde{\mathbf{B}}_1, \ldots, \tilde{\mathbf{A}}_0^T\tilde{\mathbf{B}}_{\ell_B-1}, \tilde{\mathbf{A}}_1^T\tilde{\mathbf{B}}_0, ..., \tilde{\mathbf{A}}_1^T\tilde{\mathbf{B}}_{\ell_B-1}, \tilde{\mathbf{A}}_2^T\tilde{\mathbf{B}}_0, \ldots, \tilde{\mathbf{A}}_{\ell_A-1}^T\tilde{\mathbf{B}}_0, \ldots, \tilde{\mathbf{A}}_{\ell_A-1}^T\tilde{\mathbf{B}}_{\ell_B-1}$, sequentially.

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

Our constructions leverage the properties of combinatorial structures known as resolvable designs [18].

Definition 2. 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 $\mathcal{P} \subset \mathcal{A}$ 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 $i \neq j$.

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.

In the discussion below we refer to the blocks of a design as "meta-symbols" (to avoid confusion with block-columns).

Cyclic Assignment: We will use a cyclic assignment of tasks [13] extensively in our constructions. We illustrate this by an example for matrix-vector multiplication.

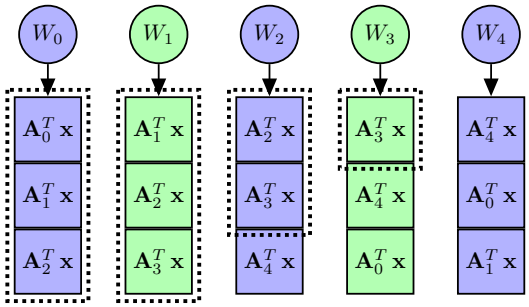

Fig. 1: Partitioning matrix $\mathbf{A}$ into five submatrices and assigning three uncoded tasks in a cyclic fashion to the workers, which leads to $s = 2$ and $Q = 10$ (for example, at most *nine* tasks, enclosed in dotted lines, can be processed without processing any copy of $\mathbf{A}_4^T\mathbf{x}$).

Example 1. 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 the matrix $\mathbf{A}$ into $\Delta = 5$ block-columns as $\mathbf{A}_0, \mathbf{A}_1, \ldots, \mathbf{A}_4$. Now we consider $\mathcal{X} = \{0, 1, 2, 3, 4\}$. If we do not incorporate any coding among the block-columns, then for block length $\beta = 1$, we have the trivial parallel class $\mathcal{P} = \{\{0\}, \{1\}, \ldots, \{4\}\}$. Fig. 1 shows that we allocate three uncoded submatrices to each worker in a cyclic fashion according to the indices of three elements of $\mathcal{P}$. It can be verified that the system is resilient to $s = 2$ stragglers. In the sequel, our assignment can be coded as well.

Suppose, 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, e.g., the product of encoded block-columns of $\mathbf{A}$ and $\mathbf{B}$. A cyclic scheme assigns the set $\{j, j+1, \ldots, j+\ell-1\} \pmod{\Delta}$ to worker node W_j; symbol j appears at the top and sequentially symbol $(j + \ell - 1)$ (values reduced modulo Δ) at the bottom. W_j processes the tasks specified by the symbols from top to bottom. The position of a symbol in a node is denoted by an integer between 0 and $\ell - 1$, where 0 denotes the top and $\ell - 1$ denotes the bottom.

Lemma 1. [17] The cyclic scheme has the following properties.

- Each symbol appears ℓ times across n worker nodes. Furthermore, it appears in each position $0, 1, \ldots, \ell - 1$ 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, $\alpha_c = \Delta\ell - \frac{\ell(\ell+1)}{2} + \sum_{i=0}^{c-1}(\ell - i)$, which is independent of j.

Example 1 shows an application of Lemma 1 where a maximum of $\alpha_0 = 9$ tasks can be processed (as depicted in Fig. 1) while task $\mathbf{A}_4^T\mathbf{x}$ is processed zero times. This is true for any task, so for this uncoded scheme, we have $Q = 10$.

III. β-LEVEL CODING FOR DISTRIBUTED COMPUTATIONS

For matrix-matrix multiplication, most prior methods [3], [14], [15], [16] consider $\gamma_A = 1/k_A, \gamma_B = 1/k_B$ and create the encoded matrices by linearly combining k_A and k_B block-columns of $\mathbf{A}$ and $\mathbf{B}$ respectively. This can make the encoded matrices k_A and k_B times denser than the input matrices.

In this section, we present our approach where we allow β_A and β_B level coding for the input matrices; β_A and β_B are user-specified integers and can be much smaller than k_A and k_B. Thus, the encoded matrices are at most β_A and β_B times denser than the input matrices. Furthermore, our approach can leverage partial computations by the slower worker nodes.

Consider n worker nodes each of which can store $\gamma_A = \frac{a_1}{a_2}$ and $\gamma_B = \frac{b_1}{b_2}$ fractions of matrices $\mathbf{A}$ and $\mathbf{B}$ which are partitioned into Δ_A and Δ_B block-columns respectively; there are $\Delta = \Delta_A\Delta_B$ unknowns of the form $\mathbf{A}_i^T\mathbf{B}_j$. We allow β_A-level and β_B-level coding for matrices $\mathbf{A}$ and $\mathbf{B}$, where $\gamma_A \leq \frac{1}{\beta_A}$ and $\gamma_B \leq \frac{1}{\beta_B}$ and set $\Delta_A = \beta_A a_2, \Delta_B = \beta_B b_2$.

To specify the scheme, we choose two separate resolvable designs with block sizes β_A and β_B, as $(\mathcal{X}_\mathbf{A}, \mathcal{A})$ where $\mathcal{X}_\mathbf{A} = \{0, 1, \ldots, \Delta_\mathbf{A} - 1\}$, and $(\mathcal{X}_\mathbf{B}, \mathcal{B})$ where $\mathcal{X}_\mathbf{B} = \{0, 1, \ldots, \Delta_\mathbf{B} - 1\}$. Let $\mathcal{P}_0^A, \mathcal{P}_1^A, \ldots$ and $\mathcal{P}_0^B, \mathcal{P}_1^B, \ldots$ denote distinct parallel classes of these designs, respectively. We assume that the number of workers $n = c \times a_2 b_2$ where c is a positive integer.

The overall idea is to partition the set of n worker nodes into c disjoint groups denoted $\mathcal{G}_0, \ldots, \mathcal{G}_{c-1}$, each of which consists of $\frac{n}{c} = a_2 b_2$ workers. For each group $\mathcal{G}_i$, we pick parallel classes $\mathcal{P}_i^A$ and $\mathcal{P}_i^B$; these parallel classes can also be the same for the different groups. The scheme operates by placing cyclically shifted meta-symbols from $\mathcal{P}_i^A$ with ℓ_A meta-symbols in each worker for the first $\Delta_A/\beta_A = a_2$ workers. For these workers, the assignment of ℓ_B meta-symbols from $\mathcal{P}_i^B$ is the same. For the next set of a_2 workers, the assignment of meta-symbols from $\mathcal{P}_i^A$ repeats; however, now we employ a cyclic shift for the assignment of meta-symbols from $\mathcal{P}_i^B$. For each assigned meta-symbol, we generate a coded block-column by choosing a random linear combination of the β_A (respectively β_B) block-columns within it. Thus, the product of two assigned coded block-columns within a worker consists of a random linear combination of $\beta = \beta_A\beta_B$ unknowns and each worker computes ℓ submatrix products sequentially. The complete algorithm is specified in Alg. 1.

Example 2. Fig. 2 depicts an example with $n = 36, \gamma_A = \gamma_B = \frac{1}{3}$ and $\beta_A = \beta_B = 2$ so that $\Delta_A = \Delta_B = 6$. We use the same parallel class, $\{\{0, 1\}, \{2, 3\}, \{4, 5\}\}$ for both $\mathbf{A}$ and $\mathbf{B}$ for all four groups. We use random vectors of length $\beta_A = \beta_B = 2$ to obtain the encoded matrices, e.g., W_0 will first compute $(x_0\mathbf{A}_0^T + x_1\mathbf{A}_1^T)(y_0\mathbf{B}_0 + y_1\mathbf{B}_1)$ where x_0, x_1, y_0 and y_1 are chosen i.i.d. at random from a continuous distribution.

In contrast, the works in [3], [14], [15], [16] require linearly combining three block-columns of $\mathbf{A}$ and $\mathbf{B}$, respectively.

Let $\mathcal{N}_A$ and $\mathcal{N}_B$ denote the corresponding incidence matrices of two parallel classes $\mathcal{P}^A$ and $\mathcal{P}^B$. Consider the matrix $\mathcal{N}_{AB}$ formed by considering all 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 $\mathcal{P}^{AB}$. It can be shown that $\mathcal{N}_{AB}$ also forms a parallel class of size $\Delta \times \Delta/\beta$, where $\Delta = \Delta_A\Delta_B$ and $\beta = \beta_A\beta_B$.

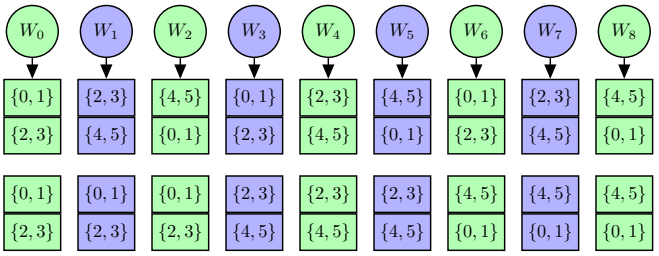

Fig. 2: Job assignment for worker group $\mathcal{G}_0$ for β-level matrix-matrix multiplication scheme with $n = 36$ with $\gamma_A = \gamma_B = \frac{1}{3}$ and $\Delta_A = \Delta_B = 6$ using a single parallel class with $\beta_A = \beta_B = 2$. The indices $\{i, j\}$ on top and bottom parts indicate random linear combinations of the submatrices of $\mathbf{A}$ and $\mathbf{B}$, respectively. Here $\beta = \beta_A\beta_B = 4$, and groups $\mathcal{G}_1$, $\mathcal{G}_2$ and $\mathcal{G}_3$ are assigned the same symbols as workers $W_0 - W_8$, but with different random coefficients.

Algorithm 1: β-level coding scheme for distributed matrix-matrix multiplication

Input : Matrices $\mathbf{A}$ and $\mathbf{B}$, storage fractions of the workers $\gamma_A = \frac{a_1}{a_2}$ $\gamma_B = \frac{b_1}{b_2}$, β_A, β_B-coding level for $\mathbf{A}$ and $\mathbf{B}$, respectively, and number of worker nodes, $n = c \times a_2 b_2$, where $c \in \mathbb{Z}^+$.

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;

$\Delta = \Delta_A\Delta_B$, $\ell_A = \Delta_A\gamma_A$, $\ell_B = \Delta_B\gamma_B$, $\beta = \beta_A\beta_B$;

Assume $\mathcal{X}_A = \{0, 1, 2, \ldots, \Delta_A - 1\}$ and find parallel classes $\mathcal{P}_i^A$ having block size β_A, $i = 0, 1, \ldots, c-1$;

Assume $\mathcal{X}_B = \{0, 1, 2, \ldots, \Delta_B - 1\}$ and find parallel classes $\mathcal{P}_i^B$ having block size β_B, $i = 0, 1, \ldots, c-1$;

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

  Denote $\mathcal{P}_i^A = \{p_{A_0}, p_{A_1}, \ldots, p_{A_{a_2-1}}\}$;

  Denote $\mathcal{P}_i^B = \{p_{B_0}, p_{B_1}, \ldots, p_{B_{b_2-1}}\}$;

  for $j \leftarrow 0$ **to** $\frac{\Delta}{\beta} - 1$ **do**

    Assign sets $p_{A_j}, p_{A_{j+1}}, \ldots, p_{A_{j+\ell_A-1}}$ from top to bottom (indices mod a_2) to worker $\frac{\Delta}{\beta}i + j$;

    $k \leftarrow \lfloor \frac{j}{a_2} \rfloor$, and assign sets $p_{B_k}, p_{B_{k+1}}, \ldots, p_{B_{k+\ell_B-1}}$ (indices reduced modulo b_2) from top to bottom to worker $\frac{\Delta}{\beta}i + j$;

    Choose random linear combinations of the constituent block-columns of meta-symbols of $\mathcal{P}_i^A$ and $\mathcal{P}_i^B$ of length β_A and β_B respectively;

  end

end

Output : Distributed matrix-matrix multiplication scheme having β-level coding.

Theorem 1. 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 c worker groups, then the scheme described in Alg. 1 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 $\beta = \beta_A\beta_B \leq c$.

Proof. It can be shown that any product meta-symbol will appear in ℓ distinct workers in each of c worker groups [17]. Thus there are a total of $c\ell$ appearances of that product meta-symbol across all the worker nodes. Furthermore, each such product meta-symbol corresponds to a product of random linear combination of $\beta = \beta_A\beta_B$ corresponding unknowns (block-columns). As the choice of these random coefficients is made i.i.d. from a continuous distribution, as long as *any* β product meta-symbols are processed across all the worker nodes, the constituent unknowns will be decodable with probability 1. Thus, the scheme is resilient to any $c\ell - \beta$ stragglers.

For the second claim, suppose that there exists a product meta-symbol $\star$ that is processed at most $\beta - 1$ times when $n\ell - \frac{c\ell(\ell+1)}{2} + \ell(\beta-1) + 1$ product meta-symbols have been processed. For each worker group, $\star$ appears in all the positions $0, \ldots, \ell-1$. Suppose that $\star$ appears i times in η_i worker groups for $i = 1, \ldots, y$. Thus, $\sum_{i=1}^{y} i\eta_i \leq \beta - 1$ and the maximum number of product 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 $\alpha_i = \alpha_0 + \sum_{j=0}^{i-1}(\ell - i) = \alpha_0 + i\ell - \frac{i(i-1)}{2}$ as specified in Lemma 1 (by setting the number of symbols to Δ/β). Thus,

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

since we have $\sum_{i=1}^{y} i\eta_i \leq \beta - 1$. Equality holds in (1) 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, same as the bound in (1). Thus if $Q \geq n\ell - \frac{c\ell(\ell+1)}{2} + \ell(\beta-1)+1$, we are guaranteed that every product meta-symbol is processed at least β times. This concludes the proof. ■

Extensions of this result when $\beta > c$ can be found in [17].

Remark 1. It can be verified that the distributed scheme shown in Fig. 2 is resilient to $s = c\ell - \beta = 4 \times 4 - 4 = 12$ stragglers and has $Q = 117$. On the other hand, the works of [3], [14] and [15] will be resilient to 27 stragglers. However, the worker node computation time for their approaches can be much higher, e.g., 2.5 times, for 98% sparsity (see Table I).

It should be noted that the parallel class structure within the resolvable design ensures that there is no repetition of any unknown within any particular worker. Moreover, a judicious choice of different parallel classes in different groups can lead the scheme to be resilient to more stragglers with a smaller Q/Δ, as discussed in details in [17].

Remark 2. For $\beta > 1$ the Q/Δ ratio can be reduced significantly as compared to the uncoded ($\beta = 1$) case. To see this consider, $n = ca_2b_2$ where $\gamma_A = \frac{a_1}{a_2}$, $\gamma_B = \frac{b_1}{b_2}$ and $c \geq \beta$. For the uncoded case with $\beta = 1$, we have $\Delta_{unc} = a_2b_2$, and $Q_{unc} = n\ell - \frac{c\ell(\ell+1)}{2} + 1$ where $\ell = a_1b_1$. On the other hand for β-level coding, we have $\Delta_\beta = \beta a_2b_2$, and $Q_\beta = n\ell - \frac{c\ell(\ell+1)}{2} + \ell(\beta-1) + 1$ where $\ell = \beta a_1b_1$. Thus

$$\frac{Q_{unc}}{\Delta_{unc}} - \frac{Q_\beta}{\Delta_\beta} = (\beta-1)\left[\gamma\left(\frac{ca_1b_1}{2} - 1\right) + \frac{1}{\beta a_2b_2}\right] > 0.$$

Algorithm 2: Sparsely coded straggler (SCS) optimal scheme for distributed matrix-matrix multiplication

Input : Matrices $\mathbf{A}$ and $\mathbf{B}$, n-number of worker nodes, storage fractions $\gamma_A = \frac{1}{k_A}$, $\gamma_B = \frac{1}{k_B}$.

Set $\Delta_A = \text{LCM}(n, k_A)$ and $\Delta_B = k_B$, $\Delta = \Delta_A \Delta_B$;

Partition $\mathbf{A}$ and $\mathbf{B}$ into Δ_A and Δ_B block-columns;

Number of coded submatrices of $\mathbf{A}$ in each worker node, $\ell_c = \frac{\Delta_A}{k_A} - \frac{\Delta}{n}$;

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

  $u \leftarrow i \times \frac{\Delta_A}{n}$, and define $T = \left\{u, u+1, \ldots, u + \frac{\Delta}{n} - 1\right\}$ (modulo Δ_A);

  Assign all $\mathbf{A}_m$'s sequentially from top to bottom to worker node i, where $m \in T$;

  Assign ℓ_c different random linear combinations of $\mathbf{A}_m$'s for $m \notin T$;

  Assign a single random linear combination of all block-columns of $\mathbf{B}$;

end

Output : $\langle n, \gamma_A, \gamma_B \rangle$ SCS optimal scheme.

IV. Sparsely Coded Straggler Optimal Matrix Computations

In the previous section, we demonstrated β_A and β_B level coding techniques that ensure that the encoded matrices are not too dense. However, this comes at a corresponding penalty in the straggler-resilience of the scheme. In this section we propose a sparsely coded straggler (SCS) optimal scheme in Alg. 2, which is suitable for sparse matrices and enjoys the optimal threshold $k_A k_B$ [3] when $\gamma_A = 1/k_A, \gamma_B = 1/k_B$. Moreover, unlike previous "dense" coded approaches [3], [14], [15], our scheme can utilize the partial computations of the slow workers and can provide significantly small Q/Δ. As before, we split $\mathbf{A}$ and $\mathbf{B}$ into Δ_A and Δ_B block-columns. We set $\Delta_B = k_B$ and place only one random linear combination of the block-columns of $\mathbf{B}$ in each worker so that $\gamma_B = 1/k_B$. On the other hand, each worker receives a carefully chosen number (ℓ_u) of uncoded block-columns of $\mathbf{A}$, followed by ℓ_c random linear combinations of block-columns of $\mathbf{A}$, such that $\gamma_A = 1/k_A = (\ell_u + \ell_c)/\Delta_A$. The uncoded assignment follows a block-cyclic shift across the workers (see Fig. 3 for an example).

Theorem 2. [17] Alg. 2 proposes a distributed matrix-matrix multiplication scheme which is resilient to $s = n - k_A k_B$ stragglers and has $Q = \Delta + (k_B - 1)\ell_c$.

Proof. The proof of straggler resilience appears in [17]. We discuss the calculation of Q here. Let $u^{(\ell,j)}$ for $j = 0, 1, \ldots, \ell_c - 1$ denote the j-th random encoding vector for $\mathbf{A}$ in worker node W_ℓ and $v^{(\ell)}$ the corresponding random encoding vector for $\mathbf{B}$, for $\ell = 0, 1, \ldots, n-1$. Once Q block-products are received, we will demonstrate that there exists a system of Δ equations (out of those Q) that corresponds to decoding the $\mathbf{A}_i^T \mathbf{B}_j$'s which is nonsingular with probability 1, for $i = 0, 1, \ldots, \Delta_A - 1$ and $j = 0, 1, \ldots, \Delta_B - 1$. 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 the worker nodes of $\mathcal{J}_i$ where $|\mathcal{J}_i| \leq k_B$ (A simple counting argument applied to Alg. 2 shows that any uncoded block-column of A appears exactly k_B times over all n workers). We obtain certain equations from the uncoded part which correspond to $e_i \otimes v^{(\ell)}$ ($\otimes$ denotes the Kronecker product) for $\ell \in \mathcal{J}_i$; we refer to these as uncoded-coded block products. Thus, if $|\mathcal{J}_i| < k_B$ then we need 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 from a continuous distribution.

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 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 η_0 distinct worker nodes. Next, we pick η_0 encoding vectors for $\mathbf{A}$ from 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. ∎

Example 3. Consider an example in Fig. 3 with $n = 5$ and $k_A = k_B = 2$, so we have $s = 1$. We set $\Delta_A = \text{LCM}(n, k_A) = 10$ and $\Delta_B = k_B = 2$. Thus, $Q = 21$, and $Q/\Delta = 1.05$.

TABLE I: 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 [14]	9	N/A	2.51	10.08	1.33×10^7
RANDOM KR CODE[15]	9	N/A	2.63	10.23	2.15×10^5
CONVOLUTIONAL CODE* [16]	2	N/A	2.44	10.19	1.82×10^3
PROPOSED UNCODED ($\beta_A = \beta_B = 1$)	1	17/9	0.69	1.96	1.41
PROPOSED β-LEVEL CODING ($\beta_A = \beta_B = 2$)	4	16/9	1.02	3.68	8.89×10^3

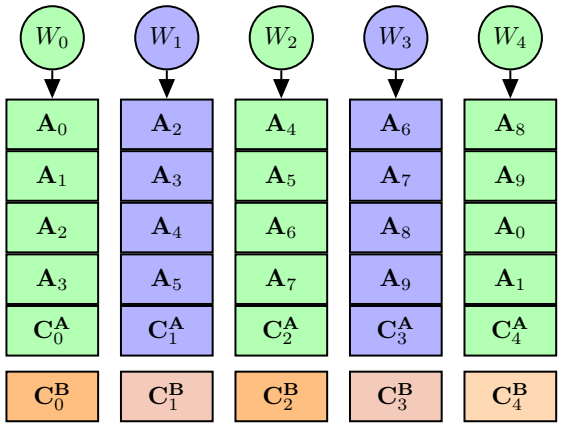

Fig. 3: Matrix-matrix multiplication with $n = 5$ and $s = 1$ with $\gamma_A = \gamma_B = \frac{1}{2}$. $\mathbf{C}_i^{\mathbf{A}}$ and $\mathbf{C}_i^{\mathbf{B}}$ are the coded submatrices assigned to worker W_i, given by $\sum_{j=0}^{9} \mathbf{R}_A^{(i,j)} \mathbf{A}_i$ and $\sum_{j=0}^{1} \mathbf{R}_B^{(i,j)} \mathbf{B}_i$, respectively.

TABLE II: 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 $\gamma_B = \frac{1}{3}$) for 95% and 98% sparsity.

METHODS	$\frac{Q}{\Delta}$	COMP. TIME (S)		κ_{worst}
		98%	95%	
POLY CODE [3]	N/A	3.11	8.29	2.40×10^{10}
ORTHO-POLY [14]	N/A	3.08	8.16	1.96×10^6
RKRP CODE[15]	N/A	3.15	8.22	2.83×10^5
CONV CODE* [16]	N/A	5.16	10.92	2.65×10^4
SCS OPTIMAL APPR.	7/6	1.93	4.76	4.93×10^6

V. NUMERICAL EXPERIMENTS AND COMPARISONS

We performed numerical experiments over a Amazon Web Services (AWS) to compare the different approaches. We chose a `t2.2xlarge` machine is used as the master node and `t2.small` machines as the worker nodes.

A. Comparison with the proposed β-level coding scheme

First we chose a system with $n = 18$ and $\gamma_A = \gamma_B = 1/3$, where $\mathbf{A}$ and $\mathbf{B}$ are sparse matrices of dimensions 12000×13680 and 12000×10260, respectively. The sparsity level of $\mathbf{A}$ and $\mathbf{B}$ can be 98% (or 95%), which indicates that randomly chosen 2% (or 5%) entries of matrices $\mathbf{A}$ and $\mathbf{B}$ are non-zero.

Table I shows the comparison for different approaches in terms of different metrices. The dense coded approaches [3], [14], [15] and [16] are MDS but they do not consider the partial computations of the slower workers[1]. On the other hand, our proposed approaches are capable of utilizing the partial computations of the stragglers which are quantified by the Q/Δ values. We can see that the β-level coding approaches, with $\beta_A = \beta_B = 2$, have smaller Q/Δ values than the uncoded approach where $\beta = 1$. A larger value of β would provide smaller value of Q/Δ.

Next, we note that the worker node computation times for other methods are $2 \sim 4$ times higher than our schemes; this is because our schemes are able to preserve the sparsity of the encoded matrices better than other methods. A more optimized sparse matrix-multiplication scheme in `Python` could result in bigger multiplicative gaps between these approaches. Lastly, the numerical stability of our schemes is also much better than [3], [14] and [15], which is verified by the smaller worst case condition number (κ_{worst}) over all choices of stragglers. A reason behind a smaller κ_{worst} may be that even in the worst case, the decoding of some β unknowns depends on a $\beta \times \beta$ system matrix whose entries are chosen randomly. Thus a smaller choice of β may lead to a smaller κ_{worst}.

B. Comparison with the proposed SCS optimal approach

In this experiment, we compare the dense coded approaches with our proposed sparsely coded straggler (SCS) optimal approach in terms of Q/Δ values, worker computation time and κ_{worst} values. For this construction we work with random sparse matrices $\mathbf{A}$ and $\mathbf{B}$ of dimensions 12000×15000 and 12000×13500. The system has $n = 24$ workers, with $\gamma_A = \frac{1}{4}$ and $\gamma_B = \frac{1}{5}$ so that the optimal threshold is 20. As shown in Table II, our worker node computation times are significantly lower than other prior works. Next, although all schemes have the same threshold (20), our proposed SCS optimal scheme can leverage the partial computations done by the slower workers and provides a significantly small Q/Δ value. Moreover, it can be observed that the worst case condition number of our scheme also much lower than [3] and comparable with the other schemes.

[1]In principle one can extend [3] for partial computation with $Q/\Delta = 1$ by placing multiple polynomial evaluations within a worker. However, this is impractical owing to numerical instability and higher computation times [17]

REFERENCES

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

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

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

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

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

[6] S. Dutta, M. Fahim, F. Haddadpour, H. Jeong, V. Cadambe, and P. Grover, "On the optimal recovery threshold of coded matrix multiplication," *IEEE Trans. on Info. Th.*, vol. 66, no. 1, pp. 278–301, 2019.

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

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

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

[10] S. Kiani, N. Ferdinand, and S. C. Draper, "Hierarchical coded matrix multiplication," *IEEE Transactions on Information Theory*, 2020.

[11] E. Ozfatura, S. Ulukus, and D. Gündüz, "Distributed gradient descent with coded partial gradient computations," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 3492–3496.

[12] B. Hasırcıoğlu, J. Gómez-Vilardebó, and D. Gündüz, "Bivariate hermitian polynomial coding for efficient distributed matrix multiplication," in *IEEE Global Communications Conference*. IEEE, 2020, pp. 1–6.

[13] A. B. Das, L. Tang, and A. Ramamoorthy, "C^3LES : Codes for coded computation that leverage stragglers," in *IEEE Info. Th. Workshop*, 2018.

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

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

[16] A. B. Das, A. Ramamoorthy, and N. Vaswani, "Efficient and robust distributed matrix computations via convolutional coding," preprint, 2020, [Online] Available: https://arxiv.org/abs/1907.08064.

[17] A. B. Das and A. Ramamoorthy, "Coded sparse matrix computation schemes that leverage partial stragglers," *arXiv preprint arXiv:2012.06065*, 2020.

[18] D. Stinson, *Combinatorial designs: constructions and analysis*. Springer Science & Business Media, 2007.