Tessellated Distributed Computing of Non-Linearly Separable Functions

Ali Khalesi
Institut Polytechnique des Sciences Avancées (IPSA)
Paris, France
Email: ali.khalesi@ipsa.fr

Ahmad Tanha, Derya Malak, and Petros Elia
Communication Systems Department, EURECOM
Biot, France
Email: {tanha, malak, elia}@eurecom.fr

***Abstract*—The work considers the N-server distributed computing setting with K users requesting functions that are arbitrary multi-variable polynomial evaluations of L real (potentially non-linear) basis subfunctions of a certain degree. We aim to reduce both the computational cost at the servers and the communication load between the servers and the users. To do so, we introduce a novel approach, which involves transforming our distributed computing problem into a sparse tensor factorization problem $\bar{\mathcal{F}} = \bar{\mathcal{E}} \times_1 \mathbf{D}$, where tensor $\bar{\mathcal{F}}$ represents the requested non-linearly-decomposable jobs expressed as the mode-1 product between tensor $\bar{\mathcal{E}}$ and matrix $\mathbf{D}$, where $\mathbf{D}$ and $\bar{\mathcal{E}}$ respectively define the communication and computational assignment, and where their sparsity respectively allows for reduced communication and computational costs. We here design an achievable scheme, designing $\bar{\mathcal{E}}, \mathbf{D}$ by utilizing novel fixed-support SVD-based tensor factorization methods that first split $\bar{\mathcal{F}}$ into properly sized and carefully positioned subtensors, and then decompose them into properly designed subtensors of $\bar{\mathcal{E}}$ and submatrices of $\mathbf{D}$. For the zero-error case and under basic dimensionality assumptions, this work establishes an achievable system rate K/N, given a specific communication and computational load.**

I. Introduction

There is a growing demand for massive parallel computing to efficiently distribute the computations across servers [1], [2]. To address this challenge, numerous works have proposed methods, focusing on scalability [3]–[5], privacy and security [6]–[10], completion time, latency, or straggler mitigation [11]–[13], among others. For comprehensive surveys of the related works, the reader is referred to [14], [15].

Beyond these considerations, the renowned computation-versus-communication tradeoff lies at the heart of distributed computing as a foundational principle with far-reaching implications. This tradeoff emerges as a critical limiting factor in numerous distributed computing scenarios [16]–[18], including the practical settings of *multi-user, multi-server distributed computation of linearly-separable functions*, addressing several classes of computing problems, e.g., gradient coding [19]–[21], linear-transform computation [22]–[24], matrix multiplication [25], or multivariate polynomial computation [11], [26]–[29], also training of large-scale machine learning algorithms and deep neural networks with massive data [16], [30], where communication and computation are the interdependent bottlenecks that significantly shape overall performance.

The advent of large language models (LLMs), whose unprecedented scale necessitates distributed training and inference, has intensified the need for communication/computation-efficient distributed systems. Particularly, training and inference in LLMs rely on non-linear transformations [31], most prominently, activation functions [32] and attention mechanisms [33], that capture multiplicative interactions beyond linearly separable functions. While non-linear computation has been extensively studied in contexts such as neural networks [34], approximation theory [35], and non-linear optimization [36], a systematic characterization of the fundamental limits of the distributed computation for non-linearly separable functions remains unexplored. Although existing frameworks, such as matrix factorization in [37], can compute non-linearly separable functions, they require an excessive number of basis subfunctions, leading to substantial resource overhead. In contrast, the proposed tensor factorization approach in this work captures non-linearly separable functions more efficiently by folding the power ranges of substantially fewer basis subfunctions into high-order tensors, thereby significantly reducing resource requirements.

Summary of our contributions: We study the problem of multi-user distributed computing of non-linearly separable functions in an N-server, K-user setting, where each user requests arbitrary multivariate polynomial evaluations of L real (potentially non-linear) basis subfunctions, represented by a full-rank tensor $\bar{\mathcal{F}}$, detailed in Section II. The sparse factorization $\bar{\mathcal{F}} = \bar{\mathcal{E}} \times_1 \mathbf{D}$ of user requests provides a natural framework for reducing both computation and communication costs. Here, the sparse encoding tensor $\bar{\mathcal{E}}$ determines assigning a specific range of exponents of special subsets of basis subfunctions to a selected subset of servers to compute, while the sparse decoding matrix $\mathbf{D}$ specifies a selected subset of users to which servers transmit. This formulation, detailed in Section III, establishes a direct link between distributed computing and tensor factorization, identifying fixed-support tensor factorization as a central component of our real-valued multi-user distributed computing problem. It generalizes distributed gradient coding [20], which arises as the special case of linearly separable tasks. Unlike gradient coding, our framework also handles non-linear separations, permits arbitrary (not

This research was partially supported by European Research Council ERC-StG Project SENSIBILITÉ under Grant 101077361, the ERC-PoC Project LIGHT under Grant 101101031, by the Huawei France-Funded Chair Toward Future Wireless Networks, and by the Program "PEPR Networks of the Future" of France 2030.

just cyclic) task allocations, and supports multiple users. In this paper, we only focus on the lossless case and establish an achievable scheme for the single-shot[1] system rate (see Theorem 1 in Section IV) by employing novel concepts and algorithms for fixed-support sparse matrix factorization and multilinear singular value decomposition (SVD) introduced in [38], [39], respectively, that allows us first to split $\bar{\mathcal{F}}$ into properly sized and carefully positioned subtensors, and then decompose them into properly designed subtensors of $\bar{\mathcal{E}}$ and submatrices of $\mathbf{D}$. Finally, we discuss the concluding remarks and our future directions in Section V.

Notations. For a positive integer n, we let $[n] = [1 : n] \triangleq \{1, \ldots, n\}$. For $a, b \in \mathbb{Z}^+$ such that $a < b$, $[[a : b]]$ denotes an ordered set of integers, ranging from a to b, and $a \mid b$ denotes a divides b. We will use supp(.) to represent the support of an array, describing the set of indices of non-zero elements. $[\mathbf{A}, \mathbf{B}]$ indicates the horizontal concatenation of the two matrices. For any matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$, then $\mathbf{X}(i, j)$, $i \in [m]$, $j \in [n]$, represents the entry in the i^{th} row and j^{th} column, while $\mathbf{X}(i, :)$, $i \in [m]$ and $\mathbf{X}(:, j)$, $j \in [n]$ represent its i^{th} row and j^{th} column, respectively. For two index sets $\mathcal{I} \subset [m], \mathcal{J} \subset [n]$, then $\mathbf{X}(\mathcal{I}, \mathcal{J})$ represents the submatrix comprised of the rows in $\mathcal{I}$ and columns in $\mathcal{J}$. All the above matrix notations are extended to tensors. $\mathbb{1}(.)$ is the indicator function, returning 1 for the corresponding condition. For a vector $\mathbf{x} \in \mathbb{R}^N$, $\|\mathbf{x}\|_0$ denotes the number of non-zero elements while $\|\mathbf{x}\|$ is the Euclidean norm.

We next briefly review the essential concepts of tensors.

Tensor Notation and Operations. *An order-N tensor* $\bar{\mathcal{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ is a multi-way array with N modes. The $(i_1, \ldots, i_N)$-th scalar entry of $\bar{\mathcal{X}}$ is denoted by $\bar{\mathcal{X}}(i_1, \ldots, i_N)$, where the mode $n \in [N]$ has a dimensionality $i_n \in [I_n]$.

A mode-n unfolding of a tensor $\bar{\mathcal{X}}$ is the mapping of its elements to a matrix. For instance, mode-1 unfolding of $\bar{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is represented as $\bar{\mathcal{X}}_{(1)} \in \mathbb{R}^{I_1 \times I_2 I_3 \cdots I_N}$.

The stacking operation corresponds to grouping of order-N tensor samples $\bar{\mathcal{X}}_j \in \mathbb{R}^{I_1 \times \cdots \times I_N}$, $j \in [J]$ to form an order-$(N+1)$ tensor $\bar{\mathcal{Y}} = \textit{stack}_{N+1}\left(\bar{\mathcal{X}}_1, \ldots, \bar{\mathcal{X}}_J\right) \in \mathbb{R}^{I_1 \times \cdots \times I_N \times J}$.

The *mode-n product* takes as input an order-N tensor, $\bar{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, and a matrix $\mathbf{A} \in \mathbb{R}^{J \times I_n}$, to produce another tensor, $\bar{\mathcal{Y}}$, of the same order as the original tensor $\bar{\mathcal{X}}$. The operation is denoted by $\bar{\mathcal{Y}} = \bar{\mathcal{X}} \times_n \mathbf{A}$, and provides a direct generalization of matrix multiplication.

Tensor Contraction Product (TCP) generalizes the mode-n product to product of tensors with possibly different orders. Given tensors $\bar{\mathcal{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ and $\bar{\mathcal{Y}} \in \mathbb{R}^{J_1 \times \cdots \times J_M}$, with common modes $I_n = J_m$, then their (n, m)-contraction, denoted by $\times_n^m$, yields an order-$(N+M-2)$ tensor $\bar{\mathcal{Z}} = \bar{\mathcal{X}} \times_n^m \bar{\mathcal{Y}} \in \mathbb{R}^{I_1 \times \cdots \times I_{n-1} \times I_{n+1} \times \cdots \times I_N \times J_1 \times \cdots \times J_{m-1} \times J_{m+1} \times \cdots \times J_M}$.

Generalized TCP contracts a whole *ordered block of modes* (a multi-index) between two tensors. Given tensors $\bar{\mathcal{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ and $\bar{\mathcal{Y}} \in \mathbb{R}^{J_1 \times \cdots \times J_M}$, assume the tails match pairwise, i.e., $(I_n, \ldots, I_N) = (J_m, \ldots, J_M)$ and $N - n + 1 = M - m + 1$. Then, their $([[n : N]], [[m : M]])$-contraction, denoted by $\times_{[[n:N]]}^{[[m:M]]}$, yields an order-$(n + m - 2)$ tensor $\bar{\mathcal{Z}} = \bar{\mathcal{X}} \times_{[[n:N]]}^{[[m:M]]} \bar{\mathcal{Y}} \in \mathbb{R}^{I_1 \times \cdots \times I_{n-1} \times J_1 \times \cdots \times J_{m-1}}$.

A comprehensive description of the tensor-related definitions can be found in Appendix A.

II. Multi-User Non-Linearly Separable Distributed Computing

We focus on the very broad and arguably practical N-*server distributed computing framework* with K users requesting arbitrary multi-variable polynomial evaluations of L real (potentially non-linear) basis subfunctions of a certain degree from distributed servers, as we will detail below.

Our setting, as depicted in Figure 1, initially considers a *master node* that coordinates, in three phases, a set of N distributed *servers* that compute functions requested by the K *users*. During the initial *demand* phase, each user $k \in [K]$ independently requests the computed output of a single real function $F_k(.)$. Under the real-valued non-linear separability assumption[2], these functions take the basic form

$$F_k(.) = \sum_{\underline{\mathbf{p}} \in \prod_{\ell \in [L]} [P_\ell]} f_{k,\underline{\mathbf{p}}} \prod_{\ell \in [L]} W_\ell^{p_\ell - 1} \tag{1}$$

where $W_\ell \triangleq f_\ell(x), x \in \mathcal{D}$, denotes the real-valued *output file* of $f_\ell(\cdot)$ for a multi-variate input x from any domain set $\mathcal{D}$, $f_\ell(\cdot)$ denotes a *basis subfunction*, and $f_{k,\underline{\mathbf{p}}} \in \mathbb{R}$ denotes a real-valued *basis coefficient*, where $\underline{\mathbf{p}} \triangleq (p_1, \ldots, p_L)$ describes the exponent vector of each demand. If $\underline{\mathbf{p}} \in \{(p_1, \ldots, p_L) \mid |\text{supp}(\mathbf{I}_{\underline{\mathbf{p}}})| > \Gamma\}$, where $\mathbf{I}_{\underline{\mathbf{p}}} \triangleq (\mathbb{1}(p_1 > 1), \mathbb{1}(p_2 > 1), \ldots, \mathbb{1}(p_L > 1))$, then $f_{k,\underline{\mathbf{p}}} = 0$. Subsequently, during the *computing phase*, the master assigns to each server $n \in [N]$, a set of basis subfunctions $\mathcal{S}_n \subseteq [L]$ to be computed locally and also a set of indices of multiplicative terms $\mathcal{P}_n \in 2^{\prod_{\ell \in [L]} [P_\ell]}$, representing the set of exponent vectors of server n, to generate the corresponding multiplicative terms $\prod_{\ell \in [L]} W_\ell^{p_\ell - 1}$ for each $\underline{\mathbf{p}} \in \mathcal{P}_n$ from the computed output subfunctions, which determines how to compute the powers of different basis subfunctions, and how to multiply the output subfunctions computed locally on each server. Then, during the *communication phase*, each server n forms signals

$$z_n \triangleq \sum_{\underline{\mathbf{p}} \in \mathcal{P}_n} e_{n,\underline{\mathbf{p}}} \prod_{\ell \in [L]} W_\ell^{p_\ell - 1}, \forall n \in [N] \tag{2}$$

which are multivariate maps as dictated by the *encoding coefficients* $e_{n,\underline{\mathbf{p}}} \in \mathbb{R}, n \in [N]$, and we will design later on. Note that $p_\ell = 1$, $\ell \notin \mathcal{S}_n$, reflects the absence of certain basis subfunctions and their powers in the multiplicative terms for computation by server n. Subsequently, server n proceeds to transmit z_n to a subset of users $\mathcal{T}_n \subseteq [K]$, via an error-free

[1] In the single-shot scenario, a server broadcasts a single non-linear combination of the output files to its connected users. While in the multi-shot settings, a server can broadcast multiple non-linear combinations of the output files, not necessarily to the same set of users.

[2] This nicely captures non-linearly separable functions, where each $F_k(.)$, taking L subfunctions as input, can be written as a non-linear combination of L *univariate* basis subfunctions, each with a specific range of exponents.

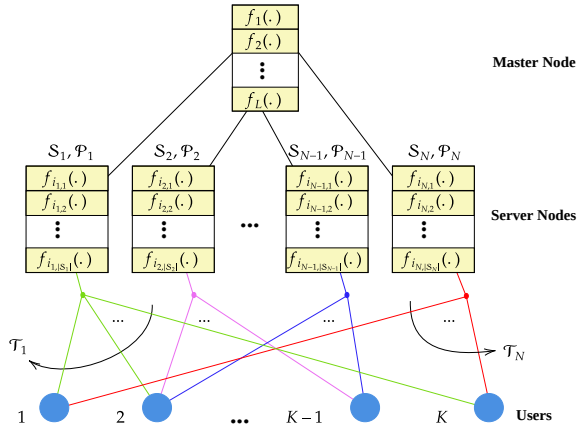

Fig. 1. The lossless $(K, N, L, \Gamma, \Delta, \{P_\ell, \Lambda_\ell\}_{\ell\in[L]})$ distributed computing setting with N servers and K users. Each server n computes the basis subfunctions $\{f_\ell(.), \forall \ell \in \mathcal{S}_n, \ell \triangleq i_{n,j}, j \in [|\mathcal{S}_n|]\}$ and produces multiplicative terms $\{W_\ell^{p_\ell - 1}, \forall \ell \in \mathcal{S}_n, \ell \triangleq i_{n,j}, j \in [|\mathcal{S}_n|]\}$, where the exponent vector of server n takes the form $(p_\ell)_{\ell\in[L]} \in \prod_{\ell\in[L]}[P_\ell]$, where $p_\ell = 1$ if $\ell \notin \mathcal{S}_n$. Furthermore, each server communicates a linear combination of the multiplicative terms to users in $\mathcal{T}_n$, under computational constraint $|\mathcal{S}_n| \leq \Gamma \leq L$, communication constraint $|\mathcal{T}_n| \leq \Delta \leq K$, and multiplication constraint $\Lambda_\ell \leq P_\ell$, $\ell \in [L]$, which bounds the number of self multiplication of each basis subfunction.

shared link. Finally, during the decoding part of the last phase, user k linearly combines its received signals to get

$$F'_k \triangleq \sum_{n\in[N]} d_{k,n} z_n \tag{3}$$

as dictated by the *decoding coefficients* $d_{k,n} \in \mathbb{R}, k \in [K], n \in [N]$. Naturally, $d_{k,n} = 0, \forall k \notin \mathcal{T}_n$, simply because user k does not receive any symbols from server n.

Furthermore, we consider computation, communication, and multiplication costs

$$\Gamma \triangleq \max_{n\in[N]} |\mathcal{S}_n| \ , \ \Delta \triangleq \max_{n\in[N]} |\mathcal{T}_n| \ , \tag{4}$$

$$\Lambda_\ell \triangleq \max_{\underline{\mathbf{p}}\in\mathcal{P}_n} p_\ell - \min_{\underline{\mathbf{p}}\in\mathcal{P}_n} p_\ell + 1 \tag{5}$$

respectively representing the maximum number of basis subfunctions to be locally computed at any server[3], the maximum number of users that a server can communicate to, and finally, the power range of basis subfunctions that must be computed, which in turn determines the maximum number of multiplications required locally at each server[4] among all multiplicative terms in (1). Simplifying the multiplication cost in (5) for $\Lambda_\ell = P_\ell = 2$, our system model reduces to the special case of linearly separable functions in [37].

We next determine the cost of evaluating the required powers of each basis subfunction in (1) using (5). Consider a power term W_ℓ^α in the demanded functions, where the exponent α is decomposed as $\alpha \triangleq q\Lambda_\ell + r$, $q \in \mathbb{N}$, $r \in [[0 : \Lambda_\ell - 1]]$. Each server computes the specific assigned range $[[q\Lambda_\ell + 1 : (q+1)\Lambda_\ell]]$ of exponents. The cost of evaluating W_ℓ^α is $\lfloor \log_2(q\Lambda_\ell + 1) \rfloor + (r - 1)$, where the logarithmic cost is due to repeated squaring, i.e., computing successive powers of two until reaching the desired anchor exponent, which is negligible compared to the $(r-1)$ multiplications required within the range of exponents. Hence, the overall complexity induced by multiplications per server for obtaining exponents of ℓ^{th} basis subfunction is in order $\mathcal{O}(\Lambda_\ell)$, versus $\mathcal{O}(\alpha)$ for naive repeated multiplications. For example, with $\alpha = 851$, $\Lambda_\ell = 100$, and $P_\ell = 1000$, the computation requires only ≈ 50 multiplications, compared to 850 in the naive approach.

In a system parametrized by $(K, N, L, \Gamma, \Delta, \{P_\ell, \Lambda_\ell\}_{\ell\in[L]})$, our goal is to find schemes that can recover any set of desired functions without error, with the smallest possible computation, communication, and multiplication loads. Accordingly, for each server n, we must specify the basis subfunctions $\mathcal{S}_n$, the exponent vectors $\mathcal{P}_n$, and the users $\mathcal{T}_n$ it communicates with. Having to serve many users with fewer servers naturally places a burden on the system, bringing to the fore the concept of the *system rate*

$$R \triangleq \frac{K}{N} \tag{6}$$

and the corresponding *system capacity* C, representing the supremum of all rates.

Toward analysing our non-linearly separable distributed computing problem, the desired functions in (1) are fully represented by a tensor $\bar{\mathcal{F}} \in \mathbb{R}^{K\times P_1\times\ldots\times P_L}$ of the coefficients $f_{k,\underline{\mathbf{p}}}$. With $\bar{\mathcal{F}}$ in place, we must decide on the computation assignment (encoding) and the communication protocol (decoding). As we have seen in [39], for the error-free case, this task is equivalent — directly from (2),(3)— to solving a (sparse) tensor factorization problem of the form

$$\bar{\mathcal{F}} = \bar{\mathcal{E}} \times_1 \mathbf{D} \, . \tag{7}$$

where, as we will specify later on, the $N \times P_1 \times \ldots \times P_L$ *computing-and-encoding tensor* $\bar{\mathcal{E}}$ holds the coefficients $e_{n,\underline{\mathbf{p}}}$ from (2), while the $K \times N$ *communication matrix* $\mathbf{D}$ holds the decoding coefficients $d_{k,n}$ from (3). Furthermore, we aim to decompose a given tensor $\bar{\mathcal{F}}$ as a mode-1 product of a tensor $\bar{\mathcal{E}}$ and a matrix $\mathbf{D}$, satisfying sparsity constraints Γ, Δ, and Λ_ℓ. Particularly, we leverage tensors as high-dimensional arrays to precisely capture the multiplicative terms associated with each basis subfunction in each user's demand.

We next present the formulation of our multi-user non-linearly separable distributed computing problem in the lossless case, establishing its connection to tensor factorization.

III. Problem Formulation

We here describe in detail the main parameters of our model, defining matrix $\mathbf{D}$ and tensors $\bar{\mathcal{E}}, \bar{\mathcal{F}}$ and the various metrics, rigorously linking the distributed computing problem and the tensor factorization in (7) for our error-free case.

[3]We highlight that the constraints Γ, Δ, and Λ_ℓ are strict, meaning they must be satisfied for every instance of the problem.

[4]With uniform server computational capacity, the maximum number of multiplications of basis subfunction ℓ with itself is bounded by Λ_ℓ. If servers had heterogeneous capacities, the bound would instead depend on n.

Let us consider

$$\mathbf{f} \triangleq [F_1, F_2, \ldots, F_K]^\intercal \in \mathbb{R}^K\,, \tag{8}$$

$$\bar{\mathcal{F}}_k(\underline{\mathbf{p}}) \triangleq f_{k,\underline{\mathbf{p}}},\ \bar{\mathcal{F}}_k \in \mathbb{R}^{P_1 \times P_2 \times \ldots \times P_L},\ k \in [K]\,, \tag{9}$$

$$\bar{\mathcal{W}}(\underline{\mathbf{p}}) \triangleq W_1^{p_1-1} W_2^{p_2-1} \ldots W_L^{p_L-1},\ \bar{\mathcal{W}} \in \mathbb{R}^{P_1 \times P_2 \times \ldots \times P_L}\,,\ \forall \underline{\mathbf{p}} \in [P_1] \times [P_2] \times \ldots \times [P_L] \tag{10}$$

where $\mathbf{f}$ represents the vector of desired function outputs F_k, $\bar{\mathcal{F}}_k$ is the tensor of function coefficients $f_{k,\underline{\mathbf{p}}}$ for the function requested by user k, and $\bar{\mathcal{W}}$ denotes the tensor of multiplicative terms of the output files $W_\ell = f_\ell(\cdot)$, all from (1). Then, recalling the encoding coefficients $e_{n,\underline{\mathbf{p}}}$ and transmitted signals z_n from (2), as well as the decoding coefficients $d_{k,n}$ and decoded functions F'_k from (3), we have

$$\bar{\mathcal{E}}_n(\underline{\mathbf{p}}) \triangleq e_{n,\underline{\mathbf{p}}},\ \bar{\mathcal{E}}_n \in \mathbb{R}^{P_1 \times P_2 \times \ldots \times P_L}\,, \tag{11}$$

$$\mathbf{d}_k \triangleq [d_{k,1}, d_{k,2}, \ldots, d_{k,N}]^\intercal \in \mathbb{R}^N,\ k \in [K] \tag{12}$$

and thus from (8), we have the output vector

$$\mathbf{f} = \mathit{stack}_1(\bar{\mathcal{F}}_1, \ldots, \bar{\mathcal{F}}_K) \times_{[[2:L+1]]}^{[[1:L]]} \bar{\mathcal{W}} \tag{13}$$

as well as the transmitted signal by server n, taking the form

$$z_n = \bar{\mathcal{E}}_n \times_{[[1:L]]}^{[[1:L]]} \bar{\mathcal{W}}\ . \tag{14}$$

This allows us to form the tensors

$$\bar{\mathcal{F}} \triangleq \mathit{stack}_1(\bar{\mathcal{F}}_1, \ldots, \bar{\mathcal{F}}_K) \in \mathbb{R}^{K \times P_1 \times P_2 \times \ldots \times P_L}, \tag{15}$$

$$\bar{\mathcal{E}} \triangleq \mathit{stack}_1(\bar{\mathcal{E}}_1, \ldots, \bar{\mathcal{E}}_N) \in \mathbb{R}^{N \times P_1 \times P_2 \times \ldots \times P_L}, \tag{16}$$

$$\mathbf{D} \triangleq [\mathbf{d}_1, \ldots, \mathbf{d}_K]^\intercal \in \mathbb{R}^{K \times N} \tag{17}$$

where $\bar{\mathcal{F}}$ denotes the $K \times P_1 \times \ldots \times P_L$ tensor of all function coefficients across all the users, $\bar{\mathcal{E}}$ represents the aforementioned $N \times P_1 \times \ldots \times P_L$ *computing tensor*, capturing the computing and linear encoding tasks of servers, and $\mathbf{D}$ represents the $K \times N$ *communication matrix*, capturing the communication and linear decoding task done by each user.

To see the transition to the tensor factorization problem, we first note that from (2) and (14), the overall transmitted vector $\mathbf{z} \triangleq [z_1, z_2, \ldots, z_N] \in \mathbb{R}^N$ takes the form

$$\mathbf{z} = \bar{\mathcal{E}} \times_{[[2:L+1]]}^{[[1:L]]} \bar{\mathcal{W}} \tag{18}$$

and then that given the decoding phase from (3), each retrieved function takes the form

$$F'_k = \mathbf{d}_k^\intercal \mathbf{z} \tag{19}$$

thus resulting in the vector of all retrieved functions taking the form $\mathbf{f}' = [\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_K]^\intercal \mathbf{z}$. We aim to set the recovery error to zero as follows.

$$\|\mathbf{f}' - \mathbf{f}\|^2 = 0\ . \tag{20}$$

Directly from the above and as in (7), we see that for the error-free case, resolving our distributed computing problem requires that $\bar{\mathcal{F}}$ be decomposed as $\bar{\mathcal{F}} = \bar{\mathcal{E}} \times_1 \mathbf{D}$.

In terms of the corresponding connection to the sparsity of $\mathbf{D}$ and $\bar{\mathcal{E}}$, we recall from (4) our metrics Γ, Δ, and Λ_ℓ, which directly from (12)–(14) and from (15)–(17), imply the computation constraint as

$$\max_{n \in [N]} \sum_{\ell \in [L]} \Big| \mathbb{1}\Big(\mathrm{supp}\big(\bar{\mathcal{E}}(n, \lhd, [2:P_\ell], \rhd)\big) \neq \emptyset \Big) \Big| \leq \Gamma$$

where $\lhd \triangleq \underbrace{:, \ldots, :}_{\ell - 1 \text{ terms}}$ and $\rhd \triangleq \underbrace{:, \ldots, :}_{L - \ell \text{ terms}}$, the communication constraint as

$$\max_{n \in [N]} \big| \mathrm{supp}\big(\mathbf{D}(:, n)\big) \big| \leq \Delta$$

and the multiplication constraint as

$$\|\bar{\mathcal{E}}(n, p_1, p_2, \ldots, p_{\ell-1}, :, p_{\ell+1}, \ldots, p_L)\|_0 \leq \Lambda_\ell\,, \quad \forall \ell \in [L],\ p_\ell \in [P_\ell]\,.$$

IV. Main Result

We present the achievable rate result for the proposed lossless distributed computing setting, parametrized by $(K, N, L, \Gamma, \Delta, \{P_\ell, \Lambda_\ell\}_{\ell \in [L]})$, in Theorem 1. All the results hold without any restriction on the dimensions, provided that each subtensor of $\bar{\mathcal{F}}$ has full rank, a condition that is easily justified in our real-valued function settings.

Theorem 1. *The achievable rate of the lossless $(K, N, L, \Gamma, \Delta, \{P_\ell, \Lambda_\ell\}_{\ell \in [L]})$ distributed computing system, under $(\Delta | K,\ \Lambda_\ell | P_\ell)$, takes the form $R = K/N$, where*

$$N \leq \frac{K}{\Delta} \sum_{i=1}^{\binom{L}{\Gamma}} \min\big(\Delta, \prod_{\ell \in \mathcal{Q}_i} \Lambda_\ell\big) \times \prod_{\ell \in \mathcal{Q}_i} \frac{P_\ell}{\Lambda_\ell} \tag{21}$$

where $\mathcal{Q}_i = \{i_1, \ldots, i_\Gamma\} \in \mathit{subset}([L], \Gamma), \forall i \in [\binom{L}{\Gamma}]$ denotes the subsets of basis subfunctions with cardinality Γ.

Proof. Due to a lack of space (partly because our proofs require a sizable set of definitions), we present the proof in the supplementary document (cf. Appendix C), which describes the achievability of the corresponding decomposition $\bar{\mathcal{E}} \times_1 \mathbf{D} = \bar{\mathcal{F}}$, and how this is translated into our distributed computing setting while abiding by the constraints on communication (Δ), computation (Γ), and multiplication (Λ_ℓ). To provide some insights, we next present a proof sketch.

Sketch of the proof: The proof consists of an achievable scheme, consisting of three main steps. In the first step, the master node divides $\bar{\mathcal{F}}$ into properly sized hypercubic subtensors (corresponding to 'tiles') where the tiles must cover all elements of $\bar{\mathcal{F}}$, and maintain a width limit Δ and modal constraint Λ_ℓ, $\ell \in [L]$ for Γ modes in subset $\mathcal{Q}_i$ of L modes (cf. Definition 2 in Appendix A). In the second step, the master node performs a multilinear SVD for each tile (the aforementioned subtensors of $\bar{\mathcal{F}}$), which defines the way each tile is the mode-1 product of a subtensor by a submatrix (the so-called right and left factors of the multilinear SVD, as detailed in Appendix B). Finally, in the third step, the right and left factors are carefully positioned to form the desired $\bar{\mathcal{E}}$ and $\mathbf{D}$, which in turn, define the computation and communication protocols of the distributed computing

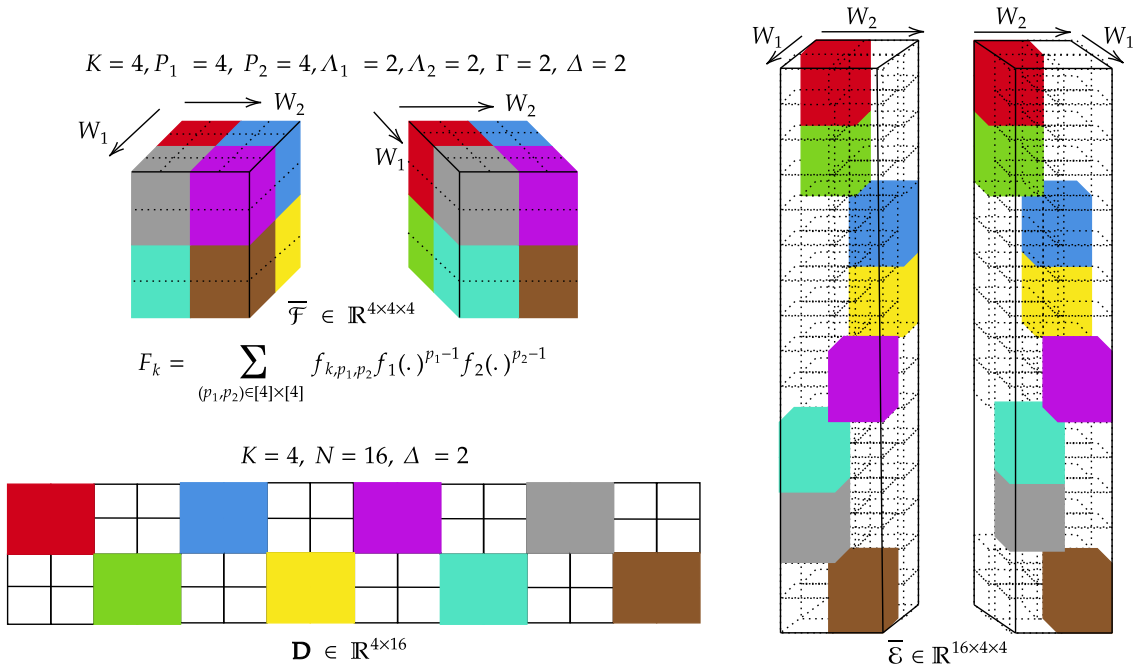

Fig. 2. Corresponding to Example 1, this figure illustrates the partitioning of $\bar{\mathcal{F}}$ into 8 tiles of size $(\Delta \times \Lambda_1 \times \Lambda_2) = (2 \times 2 \times 2)$, and also illustrates the sparse tiling of $\mathbf{D}$ and $\bar{\mathcal{E}}$ with tiles $\mathbf{L}_j$ and $\bar{\mathcal{R}}_j$, respectively, resulting in the full tiling of $\bar{\mathcal{F}} = \bar{\mathcal{E}} \times_1 \mathbf{D}$, which is covered by the 8 tiles $\bar{\mathcal{S}}_j = \bar{\mathcal{R}}_j \times_1 \mathbf{L}_j, j \in [8]$, guaranteeing the sparsity constraints $\frac{\Delta}{K} = \frac{1}{2}$ for $\mathbf{D}$, $\frac{\Lambda_1}{P_1} = \frac{\Lambda_2}{P_2} = \frac{1}{2}$, and $\frac{\Gamma}{L} = 1$ for $\bar{\mathcal{E}}$, thus satisfying the per-server communication and computing constraints, while yielding lossless reconstruction of $\bar{\mathcal{F}}$ and thus of the desired functions.

problem, respectively. To elaborate on the number of required servers for the proposed scheme, we consider a mode-1 matrix unfolding[5] of tensor $\bar{\mathcal{F}}$ and utilize its rank properties. □

We next present a basic example of a multi-user non-linearly decomposable problem and compare the performance of our scheme with the matrix factorization approach in [37].

Example 1. Consider our setting, parametrized by $(K = 4, N, L = 2, \Gamma = 2, \Delta = 2, \{P_\ell = 4, \Lambda_\ell = 2\}_{\ell\in[2]})$, where N servers are tasked with computing functions requested by $K = 4$ users. Each function is a non-linear combination of $L = 2$ basis subfunctions that takes the form

$$F_k(.) = \sum_{\underline{\mathbf{p}} \triangleq (p_1,p_2)\in[4]\times[4]} f_{k,\underline{\mathbf{p}}}\, W_1^{p_1-1} W_2^{p_2-1},\ k \in [4]\ . \quad (22)$$

The coefficients $f_{k,\underline{\mathbf{p}}}$ are described by tensor $\bar{\mathcal{F}}$, as demonstrated in Figure 2. Given the design constraints, we seek the minimum number of servers needed to guarantee lossless reconstruction of (22) at the users, and for this, we directly use (21) to conclude that we need $N = 16$ servers. To tackle this challenge of reconstruction, we need to construct

1) The $(N \times P_1 \times P_2) = (16 \times 4 \times 4)$ computing tensor $\bar{\mathcal{E}}$, specifying the computational tasks of each server.
2) The $(K \times N) = (4 \times 16)$ communication matrix $\mathbf{D}$, determining the server-user connections.

[5]Unfolding, also referred to as matricization or flattening, is the process of rearranging the elements of a multi-dimensional array into a matrix format.

These originate from the decomposition of $(K \times P_1 \times P_2) = (4 \times 4 \times 4)$ tensor $\bar{\mathcal{F}}$ (cf. (15)) as $\bar{\mathcal{F}} = \bar{\mathcal{E}} \times_1 \mathbf{D}$, representing the requested functions. The solution is then as follows.

1) Initially, we partition $\bar{\mathcal{F}}$ into $\frac{K}{\Delta} \times \prod_{\ell\in[L]} \frac{P_\ell}{\Lambda_\ell} = 2 \cdot 2 \cdot 2 = 8$ disjoint $2 \times 2 \times 2$ subtensors $\bar{\mathcal{S}}_j \in \mathbb{R}^{\Delta\times\Lambda_1\times\Lambda_2} = \mathbb{R}^{2\times2\times2}$, where j ranges from 1 to 8. This is illustrated in Figure 2.
2) Next, using the standard tensor decomposition form (cf. Appendix B), we decompose each $\bar{\mathcal{S}}_j$ as $\bar{\mathcal{S}}_j = \bar{\mathcal{R}}_j \times_1 \mathbf{L}_j$, where $\bar{\mathcal{R}}_j \in \mathbb{R}^{2\times2\times2}, \mathbf{L}_j \in \mathbb{R}^{2\times2}$ for all $j \in [8]$, noting that such full decomposition is feasible since the maximum rank of each $\bar{\mathcal{S}}_j$ is $\min(\Delta, \Lambda_1\Lambda_2) = 2$.
3) Finally, we construct $\mathbf{D} \in \mathbb{R}^{4\times16}$ and $\bar{\mathcal{E}} \in \mathbb{R}^{16\times4\times4}$ by tiling them with $\mathbf{L}_j$ and $\bar{\mathcal{R}}_j$, respectively, as illustrated in Figure 2. Thus, for example, the upper left 2×2 submatrix of $\mathbf{D}$ is equal to $\mathbf{L}_1$, the 2×2 tile to the right of that is zero, while the lower right corner of $\bar{\mathcal{E}}$ is equal to $\bar{\mathcal{R}}_8$.

The above example provides a glimpse, albeit a partial illustration, of the general principle behind creating our achievable scheme. In brief, corresponding to (21), we begin by splitting our $K \times P_1 \times P_2$ tensor $\bar{\mathcal{F}}$, into $\frac{K}{\Delta}\frac{P_1}{\Lambda_1}\frac{P_2}{\Lambda_2}$ subtensors of size $\Delta \times \Lambda_1 \times \Lambda_2$. We decompose (using the tensor decomposition form) each subtensor into the $\mathbf{L}_j, j \in [8]$ part that becomes a tile of $\mathbf{D}$, and into the $\bar{\mathcal{R}}_j, j \in [8]$ part that becomes a tile of $\bar{\mathcal{E}}$. The tile placement must respect the sparsity constraints from $\Gamma, \Delta, \Lambda_\ell$ and must yield $\bar{\mathcal{E}} \times_1 \mathbf{D} = \bar{\mathcal{F}}$. Regarding the required number of servers, the general rule is that N is simply the number of subtensors, multiplied by the rank of each subtensor.

Since we had 8 subtensors, each of rank 2, we employed $N = 16$ servers.

To solve this problem with the matrix factorization approach of [37], $L_M = \prod_{\ell \in [L]} P_\ell - 1 = 15$ basis subfunctions are needed to capture all multiplicative terms. The number of required servers with the same computation and communication cost as the tensor-based approach is thus $\frac{K}{\Delta}\Big[\big\lfloor \frac{L_M}{\Gamma} \big\rfloor \times \min(\Delta, \Gamma) + \min(\Delta, \mathrm{mod}(L_M, \Gamma))\Big] = 30$. This means our proposed tensor-based approach, for Example 1, has a gain $\approx 47\%$ over the matrix-based approach in [37].

V. Conclusion

In this work, we investigate the performance of lossless multi-user distributed computing for real-valued multi-variable polynomials, establishing clear connections to fixed-support tensor factorization and high-dimensional tessellation theory. Theorem 1 specifies an achievable rate $\frac{K}{N}$ by characterizing an upper bound on the number of required servers for the error-free reconstruction of users' demands, given the computation and communication limits to support a specific set of users in evaluating the range of exponents of the basis subfunctions.

Our future work will focus on deriving an optimal solution by establishing a converse to Theorem 1. We will also extend the framework to the multi-shot setting, which expands the span of transmitted signals and reduces the number of required servers. Moreover, we will investigate the lossy setting, where users can recover the requested functions within a bounded error, thus characterizing the system capacity and the fundamental tradeoff between communication and computation for distributed computing of arbitrary polynomials.

References

[1] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[2] M. Zaharia *et al.*, "Spark: Cluster computing with working sets," in *2nd USENIX Wksh. Hot Topics Cloud Comput. (HotCloud)*, Sydney, Australia, Aug. 2010.

[3] S. Li *et al.*, "A scalable framework for wireless distributed computing," *IEEE/ACM Trans. Netw.*, vol. 25, no. 5, pp. 2643–2654, 2017.

[4] M. Soleymani, H. Mahdavifar, and A. S. Avestimehr, "Analog lagrange coded computing," *IEEE J. Sel. Areas Inf. Theory*, vol. 2, no. 1, pp. 283–295, 2021.

[5] F. Brunero *et al.*, "Coded distributed computing for sparse functions with structured support," in *Proc., Inf. Theory Wksh. (ITW)*, Saint-Malo, France, Apr. 2023, pp. 474–479.

[6] T. Jahani-Nezhad *et al.*, "Swiftagg+: Achieving asymptotically optimal communication loads in secure aggregation for federated learning," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 4, pp. 977–989, 2023.

[7] A. Khalesi, M. Mirmohseni, and M. A. Maddah-Ali, "The capacity region of distributed multi-user secret sharing," *IEEE J. Sel. Areas Inf. Theory*, vol. 2, no. 3, pp. 1057–1071, 2021.

[8] M. Soleymani, H. Mahdavifar, and A. S. Avestimehr, "Privacy-preserving distributed learning in the analog domain," *arXiv preprint arXiv:2007.08803*, 2020.

[9] M. Soleymani *et al.*, "List-decodable coded computing: Breaking the adversarial toleration barrier," *IEEE J. Sel. Areas Inf. Theory*, vol. 2, no. 3, pp. 867–878, 2021.

[10] J. So *et al.*, "Securing secure aggregation: Mitigating multi-round privacy leakage in federated learning," in *Proc. AAAI Conf. Artif. Intell.*, vol. 37, Washington, DC, USA, Feb. 2023, pp. 9864–9873.

[11] Z. Jia and S. A. Jafar, "Cross subspace alignment codes for coded distributed batch computation," *IEEE Trans. Inf. Theory*, vol. 67, no. 5, pp. 2821–2846, 2021.

[12] E. Parrinello, E. Lampiris, and P. Elia, "Coded distributed computing with node cooperation substantially increases speedup factors," in *Proc., IEEE Int. Symp. Inf. Theory (ISIT)*, 2018, pp. 1291–1295.

[13] T. Jahani-Nezhad and M. A. Maddah-Ali, "Berrut approximated coded computing: Straggler resistance beyond polynomial computing," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 1, pp. 111–122, Feb. 2022.

[14] J. S. Ng *et al.*, "A comprehensive survey on coded distributed computing: Fundamentals, challenges, and networking applications," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 3, pp. 1800–1837, 2021.

[15] S. Li and S. A. Avestimehr, "Coded computing: Mitigating fundamental bottlenecks in large-scale distributed computing and machine learning," *Found. Trends Commun. Inf. Theory*, vol. 17, no. 1, pp. 1–148, 2020.

[16] J. Verbraeken *et al.*, "A survey on distributed machine learning," *ACM Comput. Surv.*, vol. 53, no. 2, pp. 1–33, 2020.

[17] Q. Yan, S. Yang, and M. Wigger, "Storage-computation-communication tradeoff in distributed computing: Fundamental limits and complexity," *IEEE Trans. Inf. Theory*, vol. 68, no. 8, pp. 5496–5512, 2022.

[18] S. Ulukus *et al.*, "Private retrieval, computing and learning: Recent progress and future challenges," *IEEE J. Sel. Areas Commun.*, 2022.

[19] M. Ye and E. Abbe, "Communication-computation efficient gradient coding," in *Proc., Int. Conf. Mach. Learn. (ICML)*, PMLR, Stockholm, Sweden, Jul. 2018, pp. 5610–5619.

[20] R. Tandon *et al.*, "Gradient coding: Avoiding stragglers in distributed learning," in *Proc., Int. Conf. Mach. Learn. (ICML)*, PMLR, Sydney, Australia, Aug. 2017, pp. 3368–3376.

[21] N. Charalambides, H. Mahdavifar, and A. O. Hero, "Generalized fractional repetition codes for binary coded computations," *IEEE Trans. Inf. Theory*, 2025.

[22] S. Dutta, V. Cadambe, and P. Grover, "Short-dot: Computing large linear transforms distributedly using coded short dot products," *Proc., Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 29, Dec. 2016.

[23] S. Wang *et al.*, "Fundamental limits of coded linear transform," *arXiv preprint arXiv:1804.09791*, 2018.

[24] S. Vithana and S. Ulukus, "Private read-update-write with controllable information leakage for storage-efficient federated learning with top-r sparsification," *IEEE Trans. Inf. Theory*, vol. 70, no. 5, pp. 3669–3692, 2023.

[25] O. Ordentlich and Y. Polyanskiy, "Optimal quantization for matrix multiplication," *arXiv preprint arXiv:2410.13780*, 2025.

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

[27] A. Khalesi and P. Elia, "Multi-user linearly-separable distributed computing," *IEEE Trans. Inf. Theory*, vol. 69, no. 10, pp. 6314–6339, 2023.

[28] A. Khalesi and P. Elia, "Perfect multi-user distributed computing," in *Proc., IEEE Int. Symp. Inf. Theory (ISIT)*, Athens, Greece, Jul. 2024, pp. 1349–1354.

[29] A. Tanha and D. Malak, "The influence of placement on transmission in distributed computing of Boolean functions," in *Proc., IEEE Int. Wksh. Signal Process. Adv. Wireless Commun. (SPAWC)*, Lucca, Italy, Sep. 2024.

[30] M. Kavian *et al.*, "Heterogeneity matters even more in distributed learning: Study from generalization perspective," *arXiv preprint arXiv:2503.01598*, 2025.

[31] S. M. Jayakumar *et al.*, "Multiplicative interactions and where to find them," in *Proc., Int. Conf. Learn. Represent. (ICLR)*, Addis Ababa, Ethiopia, 2020.

[32] D. Haziza *et al.*, "2:4 sparsity for activation functions in LLMs," in *Proc., Int. Conf. Learn. Represent. (ICLR)*, Singapore, May 2025.

[33] A. Hassani *et al.*, "Generalized neighborhood attention: Multi-dimensional sparse attention at the speed of light," *arXiv preprint arXiv:2504.16922*, 2025.

[34] K. Xu *et al.*, "How powerful are graph neural networks?" In *Proc., Int. Conf. Learn. Represent. (ICLR)*, New Orleans, LA, USA, May 2019.

[35] D. Elbrächter *et al.*, "Deep neural network approximation theory," *IEEE Trans. Inf. Theory*, vol. 67, no. 5, pp. 2581–2623, 2021.

[36] D. P. Bertsekas, *Nonlinear Programming*. Athena Scientific, 1999.

[37] A. Khalesi and P. Elia, "Tessellated distributed computing," *IEEE Trans. Inf. Theory*, vol. 71, no. 6, pp. 4754–4784, 2025.

[38] Q.-T. Le, E. Riccietti, and R. Gribonval, "Spurious valleys, np-hardness, and tractability of sparse matrix factorization with fixed support," *SIAM J. Matrix Anal. Appl.*, vol. 44, no. 2, pp. 503–529, 2023.

[39] L. De Lathauwer, B. De Moor, and J. Vandewalle, "A multilinear singular value decomposition," *SIAM J. Matrix Anal. Appl.*, vol. 21, no. 4, pp. 1253–1278, 2000.

[40] S. V. Dolgov and D. V. Savostyanov, "Alternating minimal energy methods for linear systems in higher dimensions," *SIAM J. Sci. Comput.*, vol. 36, no. 5, A2248–A2271, 2014.

APPENDIX

Supplementary Notations. The matrix $\mathbb{I}(\mathcal{X},\mathcal{Y}) \in \mathbb{R}^{m\times n}$ has all entries equal to zero, except at positions $(i,j), i \in \mathcal{X}, j \in \mathcal{Y}$. Supp$(\mathbf{A}) \in \{0,1\}^{m\times n}$ is a binary matrix, representing the locations of non-zero elements of $\mathbf{A} \in \mathbb{R}^{m\times n}$. For a real number $x \in \mathbb{R}$, $\lceil x \rceil, \lfloor x \rfloor$, represent respectively the ceiling and floor function of x. $\mathbb{1}(\ell \in \mathcal{Q})$ is the indicator function, returning 1 for $\ell \in \mathcal{Q}$. We use $\odot$ to represent the Hadamard product of tensors of the same dimensions, i.e., the elementwise product. $\vee$ denotes a logical "OR" operator.

A. Basic Tensor Definitions

We here review the basic concepts of tensors.

Definition 1. *An order-N tensor*, $\bar{\mathcal{X}} \in \mathbb{R}^{I_1\times\cdots\times I_N}$, is a multi-way array with N modes, with the n^{th} mode of dimensionality I_n, for $n \in [N]$. Special cases of tensors include matrices as order-2 tensors (e.g., $\mathbf{X} \in \mathbb{R}^{I_1\times I_2}$), vectors as order-1 tensors (e.g., $\mathbf{x} \in \mathbb{R}^{I_1}$), and scalars as order-0 tensors (e.g., $x \in \mathbb{R}$).

Definition 2. A *mode-n unfolding* of a tensor is the procedure of mapping the elements from a multidimensional array to a two-dimensional array (matrix). Conventionally, such a procedure is associated with stacking mode-n fibers (modal vectors) as column vectors of the resulting matrix. For instance, mode-1 unfolding of $\bar{\mathcal{X}} \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$ is represented as $\bar{\mathcal{X}}_{(1)} \in \mathbb{R}^{I_1\times I_2 I_3\cdots I_N}$, and given by

$$\bar{\mathcal{X}}_{(1)}\big(i_1, \overline{i_2 i_3 \dots i_N}\big) = \bar{\mathcal{X}}(i_1, i_2, \dots, i_N) \ . \tag{23}$$

Note that the overlined subscripts refer to linear indexing (or Little-Endian) [40], is given by

$$\begin{aligned}\overline{i_1 i_2 \dots i_N} &= 1 + \sum_{n=1}^{N} (i_n - 1) \prod_{k=1}^{n-1} I_k \\ &= 1 + i_1 + (i_2 - 1) I_1 + \cdots + (i_N - 1) I_1 \dots I_{N-1}.\end{aligned} \tag{24}$$

Definition 3. Any given vector $\mathbf{x} \in \mathbb{R}^{I_1 I_2 \dots I_N}$ can be *folded* into an N^{th} order tensor, $\bar{\mathcal{X}} \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$, with the relation between their entries defined by

$$\bar{\mathcal{X}}(i_1, i_2, \dots, i_N) = \mathbf{x}(i), \ \forall i_n \in [I_n] \tag{25}$$

where $i = 1 + \sum_{n=1}^{N} (i_n - 1) \prod_{k=1}^{n-1} I_k$.

Definition 4. Consider grouping J order-N tensor samples $\bar{\mathcal{X}}_j \in \mathbb{R}^{I_1\times\cdots\times I_N}$, $j \in [J]$, so as to form an order-$(N+1)$ data tensor, $\bar{\mathcal{Y}} \in \mathbb{R}^{I_1\times\cdots\times I_N\times J}$. This *stacking* operation is denoted by

$$\bar{\mathcal{Y}} = \mathit{stack}_{N+1}\left(\bar{\mathcal{X}}_1, \dots, \bar{\mathcal{X}}_J\right) \ . \tag{26}$$

In other words, the combined tensor samples introduce another dimension, the $(N+1)^{th}$ mode of $\bar{\mathcal{Y}}$, such that its mode-$(N+1)$ unfolding $\bar{\mathcal{Y}}_{(N+1)} \in \mathbb{R}^{(I_1 I_2\cdots I_N)\times J}$ is

$$\bar{\mathcal{Y}}_{(N+1)}\big(\overline{i_1 i_2 i_3 \dots i_N}, j\big) \ = \ \big[\mathbf{x}_1, \dots, \mathbf{x}_J\big] \tag{27}$$

where $\mathbf{x}_j \in \mathbb{R}^{I_1 I_2\cdots I_N}$ denotes the vectorization of the tensor $\bar{\mathcal{X}}_j$, obtained by stacking all its entries into a single column vector in Little-Endian order consistent with (24).

Definition 5. The *mode-n product* takes as input an order-N tensor, $\bar{\mathcal{X}} \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$, and a matrix $\mathbf{A} \in \mathbb{R}^{J\times I_n}$, to produce another tensor, $\bar{\mathcal{Y}}$, of the same order as the original tensor $\bar{\mathcal{X}}$. The operation is denoted by

$$\bar{\mathcal{Y}} = \bar{\mathcal{X}} \times_n \mathbf{A} \tag{28}$$

where $\bar{\mathcal{Y}} \in \mathbb{R}^{I_1\times\cdots\times I_{n-1}\times J\times I_{n+1}\times\cdots\times I_N}$. The mode-$n$ product is comprised of 3 consecutive steps:

$$\begin{aligned}\bar{\mathcal{X}} &\to \bar{\mathcal{X}}_{(n)} \ , \\ \bar{\mathcal{Y}}_{(n)} &= \mathcal{A}\bar{\mathcal{X}}_{(n)} \ , \\ \bar{\mathcal{Y}}_{(n)} &\to \bar{\mathcal{Y}} \ .\end{aligned} \tag{29}$$

Definition 6. *Tensor Contraction Product* (TCP) is at the core of tensor decompositions, an operation similar to the mode-n product, but the arguments of which are multidimensional arrays that can be of a different order. For instance, given an N^{th} order tensor $\bar{\mathcal{X}} \in \mathbb{R}^{I_1\times\cdots\times I_N}$ and another M^{th} order tensor $\bar{\mathcal{Y}} \in \mathbb{R}^{J_1\times\cdots\times J_M}$, with common modes $I_n = J_m$, then their (n,m)-contraction denoted by $\times_n^m$, yields a third tensor $\bar{\mathcal{Z}} \in \mathbb{R}^{I_1\times\cdots\times I_{n-1}\times I_{n+1}\times\cdots\times I_N\times J_1\times\cdots\times J_{m-1}\times J_{m+1}\times\cdots\times J_M}$ of order $(N+M-2)$, $\bar{\mathcal{Z}} = \bar{\mathcal{X}} \times_n^m \bar{\mathcal{Y}}$ with the entries

$$\begin{aligned}&\bar{\mathcal{Z}}(i_1, \dots, i_{n-1}, i_{n+1}, \dots, i_N, j_1, \dots, j_{m-1}, j_{m+1}, \dots, j_M) \\ &= \sum_{i_n \in [I_n]} \bar{\mathcal{X}}(i_1, \dots, i_{n-1}, i_n, i_{n+1}, \dots, i_N) \\ &\times \bar{\mathcal{Y}}(j_1, \dots, j_{m-1}, i_n, j_{m+1}, \dots, j_M) \, .\end{aligned} \tag{30}$$

The overwhelming indexing associated with the TCP operation in (30) becomes unmanageable for larger tensor networks, whereby multiple TCPs are carried out across a large number of tensors. Manipulation of such expressions is prone to errors and prohibitive to the manipulation of higher-order tensors.

We next generalize the TCP operation for the cases where there is more than one common mode between two tensors.

Definition 7. *Generalized TCP* is an operation similar to TCP, but it contracts a whole *ordered block of modes* (a multi-index) between two tensors. For instance, given an N^{th} order tensor $\bar{\mathcal{X}} \in \mathbb{R}^{I_1\times\cdots\times I_N}$ and an M^{th} order tensor $\bar{\mathcal{Y}} \in \mathbb{R}^{J_1\times\cdots\times J_M}$, assume the tails match pairwise, i.e., $(I_n, \dots, I_N) = (J_m, \dots, J_M)$ and $N - n + 1 = M - m + 1$. Then, their $([[n : N]], [[m : M]])$-contraction, denoted by $\times_{[[n:N]]}^{[[m:M]]}$, yields a third tensor $\bar{\mathcal{Z}} \in \mathbb{R}^{I_1\times\cdots\times I_{n-1}\times J_1\times\cdots\times J_{m-1}}$, where

$$\bar{\mathcal{Z}} \ = \ \bar{\mathcal{X}} \times_{[[n:N]]}^{[[m:M]]} \bar{\mathcal{Y}}$$

of order $(n+m-2)$, with entries

$$\begin{aligned}&\bar{\mathcal{Z}}(i_1, \dots, i_{n-1}, \, j_1, \dots, j_{m-1}) \\ &= \sum_{\underline{\mathbf{i}} \in \mathcal{I}} \bar{\mathcal{X}}(i_1, \dots, i_{n-1}, \, i_n, \dots, i_N) \\ &\times \bar{\mathcal{Y}}(j_1, \dots, j_{m-1}, \, i_n, \dots, i_N)\end{aligned}$$

where $\mathcal{N} \triangleq [[n : N]]$ denotes the range of contracted modes, $\underline{\mathbf{i}} \triangleq (i_n)_{n\in\mathcal{N}} = (i_n, \dots, i_N)$ refers to their corresponding indices, and $\mathcal{I} \triangleq \prod_{n\in\mathcal{N}} [I_n]$. The sum ranges over all $\underline{\mathbf{i}} \in \mathcal{I}$

and contracts $N-n+1$ paired modes (from n to N), resulting in an order-$(n+m-2)$ tensor $\bar{\mathcal{Z}}$.

Before proceeding with the scheme, we provide a brief reminder of the fundamental concepts related to SVD decompositions for high-dimensional data.

B. A Primer on Multilinear SVD

The multilinear SVD (MLSVD) extends the concept of the matrix SVD into the multilinear domain [39]. This decomposition provides a powerful tool for analyzing tensors and obtaining low-rank multilinear approximations.

For matrices, the SVD is well-known and expressed as

$$\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\intercal} \tag{31}$$

where $\mathbf{M} \in \mathbb{R}^{J_1 \times J_2}$ is an arbitrary real-valued matrix, $\mathbf{\Sigma} \in \mathbb{R}^{I_1 \times I_2}$ is a diagonal matrix with the entries $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_{\min(I_1,I_2)} \geq 0$ in descending order, and $\mathbf{U} \in \mathbb{R}^{J_1 \times I_1}$ and $\mathbf{V} \in \mathbb{R}^{J_2 \times I_2}$ are orthogonal matrices.

Using mode-n tensor-matrix products, we have:

$$\mathbf{M} = \mathbf{\Sigma} \times_1 \mathbf{U} \times_2 \mathbf{V}^{\intercal} \ . \tag{32}$$

The MLSVD generalizes this decomposition to higher-order tensors. In the literature, e.g., [39], it is also referred to as the higher-order SVD or Tucker decomposition, though "Tucker decomposition" has evolved into a broader term. The MLSVD of a N^{th} order tensor is represented as

$$\bar{\mathcal{T}} = \bar{\mathcal{S}} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \ldots \times_N \mathbf{U}^{(N)} \tag{33}$$

where $\bar{\mathcal{T}} \in \mathbb{R}^{J_1 \times J_2 \times \ldots \times J_N}$, $\bar{\mathcal{S}} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$, and $\mathbf{U}^{(n)} \in \mathbb{R}^{J_n \times I_n}$, $n \in [N]$. Similar to the matrix case, where $\mathbf{U}$ and $\mathbf{V}$ serve as orthonormal bases for the column and row spaces, the MLSVD computes N orthonormal mode matrices $\mathbf{U}^{(n)} \in \mathbb{R}^{I_n \times J_n}$, $n \in [N]$, each spanning the subspace of mode-n vectors. These mode matrices are directly analogous to the singular vector bases in the matrix SVD. Concretely, the MLSVD unfolds the tensor along each mode, applies the matrix SVD to the resulting unfolding, and collects the mode-n singular vectors to form the factor matrices, yielding an orthogonal decomposition of the tensor into a core tensor and its mode matrices. The subtensors $\bar{\mathcal{S}}_{i_n=\alpha}$ of $\bar{\mathcal{S}}$ are obtained by fixing the n^{th} index to α with the following properties:

- *All-orthogonality*[6]: Subtensors $\bar{\mathcal{S}}_{i_n=\alpha}$ and $\bar{\mathcal{S}}_{i_n=\beta}$ are orthogonal for all possible n, α, β if

$$\langle \bar{\mathcal{S}}_{i_n=\alpha}, \bar{\mathcal{S}}_{i_n=\beta} \rangle = 0 \quad \text{when} \quad \alpha \neq \beta \tag{34}$$

  where the scalar product of two tensors $\bar{\mathcal{T}}, \bar{\mathcal{S}} \in \mathbb{R}^{I_1 \times \ldots \times I_N}$ is defined as

$$\langle \bar{\mathcal{T}}, \bar{\mathcal{S}} \rangle \triangleq \sum_{i_1} \ldots \sum_{i_N} t_{i_1 \ldots i_N} s_{i_1 \ldots i_N} \tag{35}$$

  where t and s represent the elements of tensors $\bar{\mathcal{T}}$ and $\bar{\mathcal{S}}$, respectively.

[6]Arrays with a scalar product of zero are considered orthogonal.

- *Ordering*:

$$\sigma_1^{(n)} \geq \sigma_2^{(n)} \geq \ldots \geq \sigma_{I_n}^{(n)} \geq 0 \tag{36}$$

  where symbols $\sigma_i^{(n)}$ represent the n-mode singular values of $\bar{\mathcal{S}}$, and are equal to the Frobenius-norms $\|\bar{\mathcal{S}}_{i_n=i}\|$, $i \in [I_n]$, where the Frobenius-norm of tensor $\bar{\mathcal{T}}$ is described by

$$\|\bar{\mathcal{T}}\| \triangleq \langle \bar{\mathcal{T}}, \bar{\mathcal{T}} \rangle \ . \tag{37}$$

The mode-n rank (or n-rank) of a tensor is defined using the matrix-based methods. Specifically, it is the rank of the mode-n unfolding of the tensor, i.e., the dimension of the subspace spanned by its mode-n vectors. The mode-n vectors of $\bar{\mathcal{T}}$ are precisely the column vectors of its mode-n matrix unfolding $\bar{\mathcal{T}}_{(n)}$. We thus have:

$$\text{rank}_n(\bar{\mathcal{T}}) = \text{rank}(\bar{\mathcal{T}}_{(n)}) \ . \tag{38}$$

With the above in place, we proceed to describe the achievable scheme for the lossless case.

C. Scheme for Lossless Reconstruction and Achievability Proof for Theorem 1

We proceed to describe the design of the communication matrix $\mathbf{D}$ and the computing tensor $\bar{\mathcal{E}}$ that yield $\bar{\mathcal{E}} \times_1 \mathbf{D} = \bar{\mathcal{F}}$, while maintaining N as well as a maximum of Δ non-zero elements in any column of $\mathbf{D}$ and a maximum of Λ_ℓ non-zero elements in ℓ^{th} dimension of subtensors of $\bar{\mathcal{E}}$ for any subset of L subfunctions with maximum cardinality Γ. These constraints guarantee that each of the N servers can locally calculate up to a range Λ_ℓ of exponents of Γ basis subfunctions and can engage in communication with at most Δ users. The design borrows concepts and definitions from the blockwise SVD approach of [38], generalized to the multilinear scenario in [39] by utilizing tensors.

In this proof, we first, in Part C1, define the n^{th} rank-one contribution support, representative rank-one support, or tile, along with their related parameters. These tiles represent classes of rank-one contribution supports, which show how any support constraint on $\mathbf{D}$ and $\bar{\mathcal{E}}$ contributes to the supports on $\bar{\mathcal{E}} \times_1 \mathbf{D}$. The correspondence above is shown via Lemma 1. Then in Part C2, we show how to design the tiles for the product matrix $\bar{\mathcal{E}} \times_1 \mathbf{D}$, create and fill the non-zero tiles in $\mathbf{D}$ and $\bar{\mathcal{E}}$, and place the filled tiles in $\mathbf{D}$ and in $\bar{\mathcal{E}}$ for single-shot settings. The number of servers required for the single-shot scenario is obtained by associating the rank of each tile with the servers and then summing the ranks over all tiles [7].

1) Basic Concepts and Definitions: We first present the following definitions, which capture the basic concepts and preliminaries of our proposed scheme.

[7]Quickly recall that for a $(L+1)$- dimensional tensor $\bar{\mathcal{E}}$, then $\bar{\mathcal{E}}(n, :, \ldots, :)$ represents a L dimensional subtensor and $\mathbf{D}(:, n)$ for a matrix $\mathbf{D}$ is its n^{th} column, and Supp($\mathbf{D}$) (Supp($\bar{\mathcal{E}}$)) is a binary matrix (tensor), indicating the support of $\mathbf{D}$ ($\bar{\mathcal{E}}$). Also, when we refer to a support constraint, this will be in the form of a binary matrix (tensor) that indicates the support (the position of the allowed non-zero elements) of a matrix (tensor) of interest.

Definition 8. Given two support constraints $\mathbf{I} \in \{0,1\}^{K\times N}$ and $\bar{\mathcal{J}} \in \{0,1\}^{N\times P_1\times P_2\times\ldots\times P_L}$, then for any $n \in [N]$, we refer to $\bar{\mathcal{S}}_n(\mathbf{I},\bar{\mathcal{J}}) \triangleq \bar{\mathcal{J}}(n,:,\ldots,:)\times_1 \mathbf{I}(:,n)$ as the n^{th} *rank-one contribution support*.

We note that when the supports are implied, we may shorten $\bar{\mathcal{S}}_n(\mathbf{I},\bar{\mathcal{J}})$ to just $\bar{\mathcal{S}}_n$. The aforementioned $\mathbf{I}$ and $\bar{\mathcal{J}}$ will generally represent the support of $\mathbf{D}$ and $\bar{\mathcal{E}}$ respectively, while $\bar{\mathcal{S}}_n(\mathbf{I},\bar{\mathcal{J}})$ will generally capture some of the support of $\bar{\mathcal{E}}\times_1\mathbf{D}$ and thus of $\bar{\mathcal{F}}$. We have the following lemma for this.

Lemma 1. For $\mathbf{I} \triangleq \text{supp}(\mathbf{D})$ and $\bar{\mathcal{J}} \triangleq \text{supp}(\bar{\mathcal{E}})$, then

$$\cup_{n=1}^{N}\bar{\mathcal{S}}_n(\mathbf{I},\bar{\mathcal{J}}) = \cup_{n=1}^{N}\bar{\mathcal{J}}(n,:,\ldots,:)\times_1\mathbf{I}(:,n) \supseteq \text{Supp}(\bar{\mathcal{E}}\times_1\mathbf{D})\ . \tag{39}$$

Proof. The above follows from Definition 8 and from the fact that $\bar{\mathcal{E}}\times_1\mathbf{D} = \sum_{n=1}^{N}\bar{\mathcal{E}}(n,:,\ldots,:)\times_1\mathbf{D}(:,n)$. □

We also need the following definitions.

Definition 9. Given two supports $\mathbf{I} \in \{0,1\}^{K\times N}$ and $\bar{\mathcal{J}} \in \{0,1\}^{N\times P_1\times P_2\times\ldots\times P_L}$, the *equivalence classes of rank-one supports* are defined by the equivalence relation $i \sim j$ on $[N]$, which holds if and only if $\bar{\mathcal{S}}_i = \bar{\mathcal{S}}_j$, as represented in Figure 3.

The above splits the columns of $\mathbf{D}$ (and correspondingly the rows of $\bar{\mathcal{E}}$) into equivalence classes such that $i \sim j$ holds if and only if $\bar{\mathcal{J}}(i,:,\ldots,:)\times_1\mathbf{I}(:,i) = \bar{\mathcal{J}}(j,:,\ldots,:)\times_1\mathbf{I}(:,j)$.

Definition 10. For two supports $\mathbf{I} \in \{0,1\}^{K\times N}, \bar{\mathcal{J}} \in \{0,1\}^{N\times P_1\times P_2\times\ldots\times P_L}$, and for $\mathcal{C}$ being the collection of equivalence classes as in Definition 9, then for each class $\mathcal{P}\in\mathcal{C}$, we call $\bar{\mathcal{S}}_{\mathcal{P}}$ to be the *representative rank-one support* of class $\mathcal{P}$, which will also be called the *tile* labeled[8] by $\mathcal{P}$. Furthermore, for each tile $\bar{\mathcal{S}}_{\mathcal{P}}$, let $\mathbf{c}_{\mathcal{P}} \triangleq \mathbf{I}(:,n), n\in\mathcal{P}$ (resp. $\mathbf{r}_{\mathcal{P}} \triangleq \bar{\mathcal{J}}(n,:,\ldots,:), n\in\mathcal{P}$) be the corresponding *component column* (resp. *component row*) of the representative rank-one support. Finally, $\mathcal{C}_{\mathcal{P}} \triangleq \text{supp}(\mathbf{c}_{\mathcal{P}}) \subset [K]$ describes the set of the indices of the non-zero elements in $\mathbf{c}_{\mathcal{P}}$, while $\mathcal{R}_{\mathcal{P}} \triangleq \text{supp}(\mathbf{r}_{\mathcal{P}}) \subset [P_1]\times\ldots\times[P_L]$ describes the set of indices of the non-zero elements in $\mathbf{r}_{\mathcal{P}}$. This is illustrated in Figure 4.

Definition 11. For every set $\mathcal{C}'\subseteq\mathcal{C}$ of equivalence classes, let us define the *union of the representative rank-one supports* $\mathcal{S}_{\mathcal{C}'} \triangleq \cup_{\mathcal{P}\in\mathcal{C}'}\bar{\mathcal{S}}_{\mathcal{P}}$ to be the point-wise logical OR of the corresponding $\bar{\mathcal{S}}_{\mathcal{P}}$.

In the above, $\mathcal{S}_{\mathcal{C}'}$ is simply the area of the product tensor covered by all the tiles $\mathcal{P}$ in $\mathcal{C}'$. Furthermore, we have the following definition.

Definition 12 [38]**.** The *maximum rank of a representative rank-one support of class* $\mathcal{P}\subset\mathcal{C}$ is

$$r_{\mathcal{P}} \triangleq \min(|\mathcal{C}_{\mathcal{P}}|, |\mathcal{R}_{\mathcal{P}}|)\ . \tag{40}$$

The above simply says that for the case of $\mathbf{I} = \text{supp}(\mathbf{D})$ and $\bar{\mathcal{J}} = \text{supp}(\bar{\mathcal{E}})$, then the part of tensor $\bar{\mathcal{E}}\times_1\mathbf{D}$ covered by tile $\bar{\mathcal{S}}_{\mathcal{P}}$ can have rank which is at most $r_{\mathcal{P}}$.

[8]Note that $\bar{\mathcal{S}}_n = \bar{\mathcal{S}}_{\mathcal{P}}$ for any $n\in\mathcal{P}$. Furthermore, the term *tile* will be used interchangeably to represent both $\bar{\mathcal{S}}_{\mathcal{P}}$ and $\mathcal{P}$.

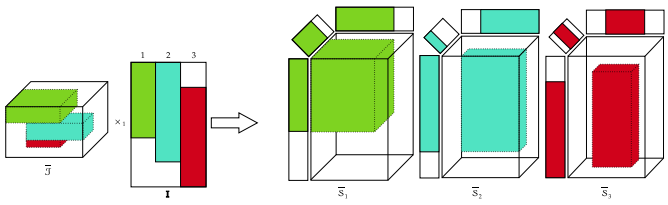

Fig. 3. The figure on the left illustrates the support constraints $\mathbf{I}$ and $\bar{\mathcal{J}}$ on $\mathbf{D}$ and $\bar{\mathcal{E}}$ respectively. The constraints $\mathbf{I}(:,1)$ and $\bar{\mathcal{J}}(1,:,:)$ on the columns and rows of $\mathbf{D}$ and $\bar{\mathcal{E}}$ respectively are colored green, $\mathbf{I}(:,2)$ and $\bar{\mathcal{J}}(2,:,:)$ are colored cyan and $\mathbf{I}(:,3)$ and $\bar{\mathcal{J}}(3,:,:)$ are colored red. The product of a column with a row of the same color yields the corresponding rank-one contribution support $\bar{\mathcal{S}}_n(\mathbf{I},\bar{\mathcal{J}}), n=1,2,3$, as described in Definition 8, and as illustrated on the right side of the figure.

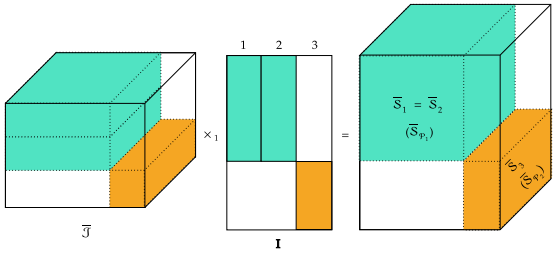

Fig. 4. This figure illustrates three different rank-one contribution supports $\bar{\mathcal{S}}_1,\bar{\mathcal{S}}_2,\bar{\mathcal{S}}_3$, where the first two fall into the same equivalence class $\bar{\mathcal{S}}_{\mathcal{P}_1} = \bar{\mathcal{S}}_1 = \bar{\mathcal{S}}_2$, while $\bar{\mathcal{S}}_{\mathcal{P}_2} = \bar{\mathcal{S}}_3$.

With the above in place, we proceed to describe the method used to design the matrix $\mathbf{D}$ and the tensor $\bar{\mathcal{E}}$.

2) Construction of $\mathbf{D}$, $\bar{\mathcal{E}}$*:* The steps will involve

- Designing the sizes of the tiles for the product tensor $\bar{\mathcal{E}}\times_1\mathbf{D}$, and placing them.
- Creating and filling the non-zero tiles in $\mathbf{D}$ and $\bar{\mathcal{E}}$.
- Placing the filled tiles in $\mathbf{D}$ and in $\bar{\mathcal{E}}$

3) Designing $\mathbf{D}$, $\bar{\mathcal{E}}$*:*

a) Step 1: Designing the sizes of the tiles for the $K\times P_1\times\ldots\times P_L$ *product tensor* $\bar{\mathcal{E}}\times_1\mathbf{D}$*, and placing the tiles:* We initially partition the set of equivalent classes $\mathcal{C}$ (cf. Definition 10) into the following set of equivalence classes

$$\begin{aligned}
&\mathcal{C} \triangleq \{\mathcal{P}_{\mathcal{Q}_i} \mid \mathcal{C}_{\mathcal{P}_{\mathcal{Q}_i}} = [1+(k-1)\Delta : \min(k\Delta, K)]\,,\\
&\mathcal{R}_{\mathcal{P}_{\mathcal{Q}_i}} = \prod_{\ell\in[L]}[1]\cup(\mathbb{1}(\ell\in\mathcal{Q}_i)\times[1+(j_\ell-1)\Lambda_\ell :\\
&\min(j_\ell\Lambda_\ell, P_\ell)])\,,\\
&\forall(k,j_1,\ldots,j_L)\in[\lceil\frac{K}{\Delta}\rceil]\times[\lceil\frac{P_1}{\Lambda_1}\rceil]\times\ldots\times[\lceil\frac{P_L}{\Lambda_L}\rceil]\,,\\
&\mathcal{Q}_i = \{i_1,\ldots,i_\Gamma\}\subseteq[L],\ |\mathcal{Q}_i| = \Gamma,\ \forall i\in[\binom{L}{\Gamma}]\}
\end{aligned} \tag{41}$$

where $\mathcal{C}_{\mathcal{P}_{\mathcal{Q}_i}}$ are the indices of the corresponding columns of $\mathbf{I}$, $\mathcal{R}_{\mathcal{P}_{\mathcal{Q}_i}}$ are the indices of the corresponding rows of $\bar{\mathcal{J}}$, which are defined in Definition 10, and j_ℓ is the corresponding index to the ℓ^{th} basis subfunction in tensor $\bar{\mathcal{F}}$. To cover all the possibilities for our tensor-based tiling problem, we consider the broad scenario where $P_\ell \geq \Lambda_\ell, \forall\ell\in[L]$ and find the general form for the upper bound on the required number of

servers N as the sum of the total number of tiles with different maximum representative rank-one supports.

To decompose the high-dimensional tiles in this problem, we consider the multilinear SVD approach, detailed in Appendix B, which is based on the matrix SVD on mode-1 unfolding of Γ-dimensional tiles $\bar{\mathcal{S}}_{\mathcal{P}}$, denoted as $\bar{\mathcal{S}}_{\mathcal{P}_{(1)}}$. We then use the rank properties of the matrix unfolding of a tensor, where we have $\mathrm{rank}(\bar{\mathcal{S}}_{\mathcal{P}}) = \mathrm{rank}(\bar{\mathcal{S}}_{\mathcal{P}_{(1)}})$.

The maximum rank of each representative rank-one support for such tiles, i.e., of each tile $\mathcal{P}$ of $\bar{\mathcal{E}} \times_1 \mathbf{D}$, from (40) and Definition 12, therefore takes the form

$$r_{\mathcal{P}} = r_{\mathcal{P}_{(1)}} = \tag{42}$$
$$\begin{cases} \min\Big(\Delta, \prod\limits_{\ell\in\mathcal{Q}_i}\Lambda_\ell\Big), \quad \mathcal{P}\in\mathcal{C}_1, \\ (k, j_1, \ldots, j_L) \in [\lfloor \frac{K}{\Delta} \rfloor] \times \prod\limits_{\ell\in[L]} [1] \cup \Big(\mathbb{1}(\ell\in\mathcal{Q}_i) \times [\lfloor \frac{P_\ell}{\Lambda_\ell} \rfloor]\Big), \\ \min\Big(\mathrm{mod}(K,\Delta), \prod\limits_{\ell\in\mathcal{Q}_i}\Lambda_\ell\Big), \quad \mathcal{P}\in\mathcal{C}_2, \\ (k, j_1, \ldots, j_L) \in \prod\limits_{\ell\in[L]} [1] \cup (\mathbb{1}(\ell\in\mathcal{Q}_i) \times [\lfloor \frac{P_\ell}{\Lambda_\ell} \rfloor]), \\ \min\Big(\Delta, \prod_{\ell,\ell'\in\mathcal{Q}_i:\ \Lambda_\ell\mid P_\ell,\ \Lambda_{\ell'}\nmid P_{\ell'}} \Lambda_\ell \prod_{\ell'} \mathrm{mod}(P_{\ell'}, \Lambda_{\ell'})\Big), \quad \mathcal{P}\in\mathcal{C}_3, \\ (k, j_1, \ldots, j_L) \in [\lfloor \frac{K}{\Delta} \rfloor] \times \prod\limits_{\ell\in[L]} [1] \cup \Big(\mathbb{1}(\ell\in\mathcal{Q}_i)\times \\ \big(\{[\lceil \frac{P_\ell}{\Lambda_\ell} \rceil]\} \cup \{\mathbb{1}(\Lambda_\ell\mid P_\ell) \times [\lfloor \frac{P_\ell}{\Lambda_\ell} \rfloor]\}\big)\Big), \\ \min\Big(\mathrm{mod}(K,\Delta), \prod_{\ell,\ell'\in\mathcal{Q}_i:\ \Lambda_\ell\mid P_\ell,\ \Lambda_{\ell'}\nmid P_{\ell'}} \Lambda_\ell \prod_{\ell'} \mathrm{mod}(P_{\ell'}, \Lambda_{\ell'})\Big), \quad \mathcal{P}\in\mathcal{C}_4, \\ (k, j_1, \ldots, j_L) \in \prod\limits_{\ell\in[L]} [1] \cup \Big(\mathbb{1}(\ell\in\mathcal{Q}_i)\times \\ \big(\{[\lceil \frac{P_\ell}{\Lambda_\ell} \rceil]\} \cup \{\mathbb{1}(\Lambda_\ell\mid P_\ell) \times [\lfloor \frac{P_\ell}{\Lambda_\ell} \rfloor]\}\big)\Big) \end{cases}$$

where the equivalence classes $\mathcal{C}_i$, $i \in [4]$ follow from Definition 9 for all possible scenarios where $\Delta \mid K$, $\Lambda_\ell \mid P_\ell$, $\Delta \nmid K$, and $\Lambda_\ell \nmid P_\ell$. We note that for the cases $\Delta \mid K$ and $\Lambda_\ell \mid P_\ell$, we divide each $(\Gamma+1)$-dimensional space corresponding to each $\mathcal{Q}_i$, $\forall i \in [\binom{L}{\Gamma}]$ to $\frac{K}{\Delta} \times \prod\limits_{\ell\in\mathcal{Q}_i} \frac{P_\ell}{\Lambda_\ell}$ tiles with dimensions $\Delta \times \Lambda_{i_1} \times \ldots \times \Lambda_{i_\Gamma}$. A similar procedure holds for the residual space for the general cases $\Delta \nmid K$ and $\Lambda_\ell \nmid P_\ell$.

The number of representative rank-one supports for tiles $\mathcal{P}$ in each equivalence class is therefore

$$|\mathcal{C}_1| = \lfloor \frac{K}{\Delta} \rfloor \sum_{i=1}^{\binom{L}{\Gamma}} \prod_{\ell\in\mathcal{Q}_i} \lfloor \frac{P_\ell}{\Lambda_\ell} \rfloor,$$
$$|\mathcal{C}_2| = \sum_{i=1}^{\binom{L}{\Gamma}} \prod_{\ell\in\mathcal{Q}_i} \lfloor \frac{P_\ell}{\Lambda_\ell} \rfloor,$$
$$|\mathcal{C}_3| = \lfloor \frac{K}{\Delta} \rfloor \sum_{i=1}^{\binom{L}{\Gamma}} \prod_{\ell\in\mathcal{Q}_i:\ \Lambda_\ell\mid P_\ell} \lfloor \frac{P_\ell}{\Lambda_\ell} \rfloor \times \prod_{\ell'\in\mathcal{Q}_i:\ \Lambda_{\ell'}\nmid P_{\ell'}} \lceil \frac{P_{\ell'}}{\Lambda_{\ell'}} \rceil,$$
$$|\mathcal{C}_4| = \sum_{i=1}^{\binom{L}{\Gamma}} \prod_{\ell\in\mathcal{Q}_i:\ \Lambda_\ell\mid P_\ell} \lfloor \frac{P_\ell}{\Lambda_\ell} \rfloor \times \prod_{\ell'\in\mathcal{Q}_i:\ \Lambda_{\ell'}\nmid P_{\ell'}} \lceil \frac{P_{\ell'}}{\Lambda_{\ell'}} \rceil. \tag{43}$$

The above information will be essential in enumerating our equivalence classes and associating each such class with a collection of servers.

b) Step 2: Filling tiles in $\bar{\mathcal{E}} \times_1 \mathbf{D}$ as a function of $\bar{\mathcal{F}}$: Recall that we have a tile $\bar{\mathcal{S}}_{\mathcal{P}}(\mathcal{R}_{\mathcal{P}}, \mathcal{C}_{\mathcal{P}})$ corresponding to the non-zero elements of $\bar{\mathcal{S}}_{\mathcal{P}}$. This tile is "empty" in the sense that all non-zero entries of $\bar{\mathcal{S}}_{\mathcal{P}}$ are equal to 1.

To avoid assigning any entry of $\bar{\mathcal{F}}$ to more than one tile, we fix a strict total ascending order $\prec$ on the tiles $\{\mathcal{P}\}$, e.g., lexicographic on $(k, j_1, \ldots, j_L, \mathcal{Q}_i)$, and maintain a binary mask $\mathcal{M}$ (same size as $\bar{\mathcal{F}}$), initialized to zero. We process tiles in the order $\prec$, and for each tile $\mathcal{P}$ we *zero–force* previously owned positions:

$$\bar{\mathcal{S}}^{\circ}_{\mathcal{P}} \triangleq \bar{\mathcal{S}}_{\mathcal{P}} \odot (\mathbf{1} - \mathcal{M}).$$

Consequently, all entries already assigned by earlier tiles are set to zero. If $\bar{\mathcal{S}}^{\circ}_{\mathcal{P}} = \mathbf{0}$, we skip $\mathcal{P}$. Otherwise, we define the induced index sets

$$\mathcal{C}^{\circ}_{\mathcal{P}} \triangleq \{k : \exists\, \underline{\mathbf{p}} \text{ s.t. } (k, \underline{\mathbf{p}}) \in \mathrm{Supp}(\bar{\mathcal{S}}^{\circ}_{\mathcal{P}})\}, \tag{44}$$
$$\mathcal{R}^{\circ}_{\mathcal{P}} \triangleq \{\underline{\mathbf{p}} : \exists\, k \text{ s.t. } (k, \underline{\mathbf{p}}) \in \mathrm{Supp}(\bar{\mathcal{S}}^{\circ}_{\mathcal{P}})\} \tag{45}$$

to form the cropped subtensor, factored as

$$\hat{\bar{\mathcal{F}}}_{\mathcal{P}} \triangleq \big(\bar{\mathcal{F}} \odot \bar{\mathcal{S}}^{\circ}_{\mathcal{P}}\big)\big(\mathcal{R}^{\circ}_{\mathcal{P}}, \mathcal{C}^{\circ}_{\mathcal{P}}\big)$$

and set the rank budget $r^{\circ}_{\mathcal{P}} = \min\big(|\mathcal{C}^{\circ}_{\mathcal{P}}|, |\mathcal{R}^{\circ}_{\mathcal{P}}|\big)$. We then compute the MLSVD of $\hat{\bar{\mathcal{F}}}_{\mathcal{P}}$ as

$$\hat{\bar{\mathcal{F}}}_{\mathcal{P}} = \bar{\mathcal{R}}_{\mathcal{P}} \times_1 \mathbf{L}_{\mathcal{P}}$$

with $\mathbf{L}_{\mathcal{P}} \in \mathbb{R}^{|\mathcal{C}^{\circ}_{\mathcal{P}}| \times r^{\circ}_{\mathcal{P}}}$ and $\bar{\mathcal{R}}_{\mathcal{P}} \in \mathbb{R}^{r^{\circ}_{\mathcal{P}} \times |\mathcal{R}^{\circ}_{\mathcal{P}}(1)| \times \cdots \times |\mathcal{R}^{\circ}_{\mathcal{P}}(L)|}$. The columns of $\mathbf{D}$ reserved for tile $\mathcal{P}$ (one column per rank-1 component) are filled with $\mathbf{L}_{\mathcal{P}}$ on the rows indexed by $\mathcal{C}^{\circ}_{\mathcal{P}}$ and zeros elsewhere. The matching frontal slices of $\bar{\mathcal{E}}$ are filled with $\mathcal{R}_{\mathcal{P}}$ on the indices $\mathcal{R}^{\circ}_{\mathcal{P}}$ and zeros elsewhere. Finally, we update the mask by

$$\mathcal{M} \leftarrow \mathcal{M} \vee \mathrm{Supp}(\bar{\mathcal{S}}^{\circ}_{\mathcal{P}}).$$

By construction, the owned supports $\{\mathrm{Supp}(\bar{\mathcal{S}}^{\circ}_{\mathcal{P}})\}_{\mathcal{P}}$ are pairwise disjoint, hence no entry of $\bar{\mathcal{F}}$ is assigned twice. Therefore, the later tiles automatically see zeros on intersections.

c) Step 3: Creating and filling the non-zero tiles in $\mathbf{D}$ and $\bar{\mathcal{E}}$: This step starts with MLSVD (as described in Section B) of the cropped tile $\bar{\mathcal{F}}_{\mathcal{P}}$, where this decomposition takes the form

$$\bar{\mathcal{F}}_{\mathcal{P}} = \bar{\mathcal{E}}_{\mathcal{P}} \times_1 \mathbf{D}_{\mathcal{P}} \tag{46}$$

where $\mathbf{D}_{\mathcal{P}} \in \mathbb{R}^{|\mathcal{C}^{\circ}_{\mathcal{P}}| \times r^{\circ}_{\mathcal{P}}}$ and $\bar{\mathcal{E}}_{\mathcal{P}} \in \mathbb{R}^{r^{\circ}_{\mathcal{P}} \times |\mathcal{R}^{\circ}_{\mathcal{P}}|}$. In particular, $\bar{\mathcal{F}}_{\mathcal{P}}, \mathbf{D}_{\mathcal{P}}$, and $\bar{\mathcal{E}}_{\mathcal{P}}$ are associated to $\bar{\mathcal{T}}$, $\mathbf{U}^{(1)}$, and $\bar{\mathcal{S}}$ in all complete SVD decomposition of (33). Naturally, $\mathrm{rank}(\bar{\mathcal{F}}_{\mathcal{P}}) \leq r^{\circ}_{\mathcal{P}}$.

Let

$$\bigcup_{i=1}^{4} \mathcal{C}_i \triangleq \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_m\},\ m \in \mathbb{N} \tag{47}$$

describe the enumeration we give to each tile. Then, the position that each cropped tile takes inside $\mathbf{D}$, is given by

$$\mathcal{C}^{\circ}_{\mathcal{P}_j},\ \Big[\sum_{i=1}^{j-1} r^{\circ}_{\mathcal{P}_i} + 1 \,:\, \sum_{i=1}^{j} r^{\circ}_{\mathcal{P}_i}\Big], \quad \forall\, \mathcal{P}_j \in \bigcup_{i=1}^{4} \mathcal{C}_i \tag{48}$$

and the position of each cropped tile in $\bar{\mathcal{E}}$ is given by

$$\Big[\sum_{i=1}^{j-1} r^{\circ}_{\mathcal{P}_i} + 1 \,:\, \sum_{i=1}^{j} r^{\circ}_{\mathcal{P}_i}\Big],\ \mathcal{R}^{\circ}_{\mathcal{P}_j}, \quad \forall\, \mathcal{P}_j \in \bigcup_{i=1}^{4} \mathcal{C}_i\,. \tag{49}$$

In particular, these yield

$$\mathbf{D}(\mathcal{C}^{\circ}_{\mathcal{P}_j}, [\sum_{i=1}^{j-1} r^{\circ}_{\mathcal{P}_i} + 1, \sum_{i=1}^{j} r^{\circ}_{\mathcal{P}_i}]) = \mathbf{D}_{\mathcal{P}_j} \tag{50}$$

and

$$\bar{\mathcal{E}}([\sum_{i=1}^{j-1} r^{\circ}_{\mathcal{P}_i} + 1, \sum_{i=1}^{j} r^{\circ}_{\mathcal{P}_i}], \mathcal{R}^{\circ}_{\mathcal{P}_j}) = \bar{\mathcal{E}}_{\mathcal{P}_j} \tag{51}$$

while naturally the remaining non-assigned elements of $\mathbf{D}$ and $\bar{\mathcal{E}}$ are zero.

Moreover, the total number of required servers is obtained as $N \leq \sum_{\mathcal{P}} r^{\circ}_{\mathcal{P}}$ while the design constraints $(\Gamma, \Delta, \{\Lambda_\ell\}_{\ell\in[L]})$ are satisfied at each server. Finally, as one can readily verify, the above design corresponds to

$$\begin{aligned}
N &\leq \sum_{i\in[4]} \sum_{\mathcal{P}\in\mathcal{C}_i} r^{\circ}_{\mathcal{P}} \\
&= \lfloor \frac{K}{\Delta} \rfloor \sum_{i=1}^{\binom{L}{\Gamma}} \min\Big(\Delta, \prod_{\ell\in\mathcal{Q}_i} \Lambda_\ell\Big) \times \prod_{\ell\in\mathcal{Q}_i} \lfloor \frac{P_\ell}{\Lambda_\ell} \rfloor \\
&+ \sum_{i=1}^{\binom{L}{\Gamma}} \min\Big(\mathrm{mod}(K,\Delta), \prod_{\ell\in\mathcal{Q}_i} \Lambda_\ell\Big) \times \prod_{\ell\in\mathcal{Q}_i} \lfloor \frac{P_\ell}{\Lambda_\ell} \rfloor \\
&+ \lfloor \frac{K}{\Delta} \rfloor \sum_{i=1}^{\binom{L}{\Gamma}} \min\Big(\Delta, \prod_{\substack{\ell \\ \ell,\ell'\in\mathcal{Q}_i:\ \Lambda_\ell | P_\ell,\ \Lambda_{\ell'} \nmid P_{\ell'}}} \Lambda_\ell \prod_{\ell'} \mathrm{mod}(P_{\ell'}, \Lambda_{\ell'})\Big) \\
&\times \prod_{\ell\in\mathcal{Q}_i:\ \Lambda_\ell | P_\ell} \lfloor \frac{P_\ell}{\Lambda_\ell} \rfloor \times \prod_{\ell'\in\mathcal{Q}_i:\ \Lambda_{\ell'} \nmid P_{\ell'}} \lceil \frac{P_{\ell'}}{\Lambda_{\ell'}} \rceil \\
&+ \sum_{i=1}^{\binom{L}{\Gamma}} \min\Big(\mathrm{mod}(K,\Delta), \prod_{\substack{\ell \\ \ell,\ell'\in\mathcal{Q}_i:\ \Lambda_\ell | P_\ell,\ \Lambda_{\ell'} \nmid P_{\ell'}}} \Lambda_\ell \prod_{\ell'} \mathrm{mod}(P_{\ell'}, \Lambda_{\ell'})\Big) \\
&\times \prod_{\ell\in\mathcal{Q}_i:\ \Lambda_\ell | P_\ell} \lfloor \frac{P_\ell}{\Lambda_\ell} \rfloor \times \prod_{\ell'\in\mathcal{Q}_i:\ \Lambda_{\ell'} \nmid P_{\ell'}} \lceil \frac{P_{\ell'}}{\Lambda_{\ell'}} \rceil
\end{aligned} \tag{52}$$

corresponding to Theorem 1 when we consider the special case where $\Delta \mid K$ and $\Lambda_\ell \mid P_\ell$, $\forall \ell \in [L]$.