Low-Latency Coded Tensor–Matrix Multiplication for Distributed Signal Processing Systems

Ahmad Tanha
EURECOM & Sorbonne University, France
Email: ahmad.tanha@eurecom.fr

Ali Khalesi
IPSA & LINCS Lab, Paris, France
Email: ali.khalesi@ipsa.fr

***Abstract*—Large-scale signal processing systems, including aeronautical and aerospace platforms, increasingly rely on tensor operations, whose distributed execution is constrained by communication, memory, and latency bottlenecks. In this paper, we propose a tensor-aware coded computation framework for distributed mode-1 tensor–matrix multiplication that operates directly on tensor subtensors, avoiding explicit unfolding and preserving multi-dimensional structure. For a fixed partitioning configuration, the proposed tensor-PolyDot scheme achieves the same recovery threshold, communication cost, worker-side computation, and memory requirements as conventional matrix-based PolyDot schemes. However, by leveraging multivariate encoding, it replaces high-degree univariate interpolation at the fusion node with structured low-dimensional decoding. This leads to a substantial reduction in decoding complexity and latency, without affecting any other system metric. Additionally, the tensor-based approach improves memory locality and enables parallel decoding across tensor modes, making it well-suited to modern hardware architectures. Numerical results show decoding speedups proportional to the tensor partitioning factor, reaching up to an order-of-magnitude improvement in practical settings. These advantages make the proposed framework particularly relevant to latency-critical aeronautical and aerospace distributed signal processing systems.**

***Index Terms*—Distributed matrix multiplication, tensor contraction, scalable machine learning, distributed signal processing, low-latency coded computing.**

I. Introduction

Modern signal processing systems increasingly rely on multi-dimensional data representations, where signals are naturally modeled as tensors rather than vectors or matrices. Such representations arise in a wide range of applications, including multi-antenna wireless communications, hyperspectral imaging, video processing, and distributed learning systems [1], [2]. In these settings, tensor operations, such as tensor–matrix products, tensor contractions, and tensor decompositions, form the computational backbone of many signal processing pipelines [3]. Beyond these domains, tensor-based processing is also fundamental in modern aeronautics and aerospace systems, where large-scale, multi-dimensional data must be processed under stringent latency and reliability constraints. For example, airborne radar, synthetic aperture radar, and satellite-based remote sensing inherently generate high-dimensional data capturing spatial, temporal, and spectral structures [4]. Similarly, autonomous aerial vehicles and advanced avionics rely on multi-sensor fusion, integrating heterogeneous data sources, such as radar, LiDAR, and communication signals, into tensor-structured representations that require efficient distributed processing schemes [5].

In such environments, computation is often distributed across onboard processors, edge devices, and ground stations, where communication resources are limited and real-time decision-making is critical. Consequently, decoding at the fusion node frequently becomes a performance bottleneck. This motivates the need for distributed computation strategies that reduce decoding latency without compromising robustness. The proposed tensor-aware coded computation framework addresses this challenge by exploiting multi-dimensional structure to significantly reduce decoding complexity while preserving resilience to stragglers, making it particularly well-suited for next-generation aeronautical and aerospace distributed signal processing systems.

Among tensor operations, mode-n tensor–matrix multiplication plays a central role, enabling dimensionality reduction, feature extraction, and mode-wise filtering. In particular, the mode-1 tensor–matrix product is critical in many applications. For instance, in multi-antenna wireless communication systems, channel state information can be represented as tensors capturing spatial, temporal, and frequency dimensions; processing along the antenna dimension corresponds to mode-1 multiplication and is essential for tasks such as beamforming and channel estimation. Likewise, in hyperspectral imaging, spectral filtering and dimensionality reduction are naturally expressed as tensor–matrix multiplications along specific modes [4].

As the scale of such data continues to grow, executing tensor operations on a single machine becomes increasingly impractical, necessitating distributed implementations across multiple worker nodes. However, distributed computation introduces several system-level challenges, including communication bottlenecks, limited memory at worker nodes, and the presence of *stragglers*, i.e., slow or unresponsive workers that delay the overall computation [6]. These challenges are particularly critical in real-time signal processing systems, where latency constraints are stringent.

To address stragglers, *coded computation* techniques have recently emerged as a powerful approach [7]–[9]. By introducing structured redundancy, coded schemes enable recovery of the final result from a subset of worker outputs. Notable examples include Polynomial codes, MatDot codes, and PolyDot codes, which achieve efficient tradeoffs between computation, communication, and recovery threshold for distributed matrix

multiplication [10]–[13]. Recent works have also explored structured distributed computation more broadly [14], including sparse tensor factorization [15] and secrecy-constrained coded matrix multiplication [16].

Despite these advances, most of the existing coded computation frameworks are inherently matrix-centric: they rely on reshaping or unfolding tensor data into matrices prior to processing. This abstraction overlooks the inherent multi-dimensional structure of tensors, often leading to increased memory traffic, loss of data locality, and additional preprocessing overhead. More importantly, it results in decoding procedures based on high-degree univariate polynomial interpolations, a significant computational bottleneck at the fusion node, given the limited computational resources. This motivates the need for tensor-aware coded computation schemes that directly exploit the multi-dimensional structure to improve system efficiency.

In this work, we address these limitations by proposing a tensor-aware coded computation framework for distributed mode-1 tensor–matrix multiplication that operates directly on tensor subtensors, without requiring explicit unfolding. The key idea is to exploit the multi-dimensional structure of tensors to design *multivariate* encoding schemes, which enable decoding via a collection of lower-dimensional interpolation problems. This work demonstrates that exploiting tensor structure can substantially alleviate decoding bottlenecks in distributed coded computation. Compared to conventional matrix-based approaches, the proposed tensor-based framework preserves all fundamental system metrics—such as recovery threshold, communication cost, and worker-side computation and memory—while significantly reducing decoding complexity at the fusion node. This reduction directly translates into lower latency, which is critical in practical distributed and real-time signal processing systems.

Our contributions. The main contributions of this work are summarized as follows.

- **Tensor-aware coded framework:** We propose a coded computation framework for distributed mode-1 tensor–matrix multiplication that operates directly on tensor subtensors, avoiding matrix unfolding and preserving the inherent multi-dimensional structure of the data.
- **Multivariate coding for efficient decoding:** We introduce a multivariate encoding and decoding strategy that replaces high-degree univariate interpolation with structured low-dimensional interpolation, reducing decoding complexity by a factor proportional to the tensor partitioning factor.
- **Preservation of system guarantees:** We show that the proposed scheme preserves all fundamental system metrics—including recovery threshold, communication cost, and worker-side computation and memory—matching those of matrix-based PolyDot schemes.
- **Unified performance characterization:** We provide a comprehensive analysis of recovery threshold, computation, communication, memory, and decoding complexity, revealing a new tradeoff enabled by tensor-aware coding.
- **Practical gains in latency:** Through numerical evaluations, we demonstrate that the proposed approach achieves substantial decoding speedups (up to an order of magnitude), translating into significant reductions in end-to-end latency in distributed signal processing systems.

Notations. Scalars are denoted by lowercase or uppercase letters, matrices by bold uppercase letters, and tensors by calligraphic uppercase letters. For $m \in \mathbb{N}$, let $[m] \triangleq \{1, \dots, m\}$. The mode-1 matrix unfolding of $\mathcal{A}$ is denoted by $\mathcal{A}_{(1)}$, and the mode-1 tensor–matrix product is written as $\mathcal{Y} = \mathcal{A} \times_1 \mathbf{B}$. We use $M \triangleq \prod_{n=2}^{N} I_n$, where s is the partition factor along mode 1, u_n is the partition factor along mode n, and $t \triangleq \prod_{n=2}^{N} u_n$. The recovery threshold is denoted by K.

The rest of this paper is organized as follows. In Section II, we introduce the problem formulation and the distributed computation model. Section III presents the coded computation framework and provides a unified analysis of performance metrics, including recovery threshold, computation, communication, and decoding complexity. Section IV presents numerical results illustrating the advantages of the proposed approach. Finally, Section V concludes the paper and outlines future research directions.

II. Problem Setup and System Model

Let $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ denote an N-mode tensor, and let $\mathbf{B} \in \mathbb{R}^{P \times I_1}$ be a matrix. We consider the computation of the mode-1 tensor–matrix product

$$\mathcal{Y} = \mathcal{A} \times_1 \mathbf{B}, \tag{1}$$

which produces a tensor $\mathcal{Y} \in \mathbb{R}^{P \times I_2 \times \cdots \times I_N}$.

a) Element-wise interpretation.: The mode-1 product corresponds to multiplying the tensor along its first mode. More explicitly, each entry of $\mathcal{Y}$ is given by

$$\mathcal{Y}(p, i_2, \dots, i_N) = \sum_{i_1=1}^{I_1} \mathbf{B}(p, i_1)\, \mathcal{A}(i_1, i_2, \dots, i_N) \tag{2}$$

for all $p \in [P]$ and $(i_2, \dots, i_N) \in [I_2] \times \cdots \times [I_N]$.

b) Unfolding interpretation.: Let $\mathcal{A}_{(1)} \in \mathbb{R}^{I_1 \times M}$ denote the mode-1 matrix unfolding of $\mathcal{A}$, where

$$M \triangleq \prod_{n=2}^{N} I_n. \tag{3}$$

Then, the tensor–matrix product can be equivalently written as a matrix multiplication of the form

$$\mathcal{Y}_{(1)} = \mathbf{B}\mathcal{A}_{(1)} \in \mathbb{R}^{P \times M}. \tag{4}$$

While this equivalence is useful for analysis, explicitly forming $\mathcal{A}_{(1)}$ may incur significant memory overhead and disrupt data locality. This motivates our tensor-aware approach that avoids explicit unfolding.

c) Distributed computation setting.: We consider a distributed system where the input tensor $\mathcal{A}$ and matrix $\mathbf{B}$ are stored at a master node, and the computation is offloaded to multiple worker nodes.

Our objective is to design a coded distributed computation scheme that

- Minimizes the number of worker responses required for recovery (recovery threshold),
- Balances computation and memory across workers,
- Reduces communication overhead, and
- Minimizes decoding complexity at the fusion node.

d) Partitioning along mode-1.: We divide the first mode of $\mathcal{A}$ into s equal parts, assuming $s \mid I_1$. Accordingly, the matrix $\mathbf{B}$ is partitioned column-wise into s blocks:

$$\mathbf{B} = [\mathbf{B}_0\ \mathbf{B}_1\ \cdots\ \mathbf{B}_{s-1}], \quad \mathbf{B}_i \in \mathbb{R}^{P \times \frac{I_1}{s}}. \tag{5}$$

e) Partitioning along remaining modes: For each mode $n \in \{2, \ldots, N\}$, we partition the dimension I_n into u_n segments, assuming $u_n \mid I_n$. This results in a total of

$$t \triangleq \prod_{n=2}^{N} u_n. \tag{6}$$

subtensors, each of size

$$\frac{I_1}{s} \times \frac{I_2}{u_2} \times \cdots \times \frac{I_N}{u_N}.$$

f) Block structure in unfolded form: Under the mode-1 matrix unfolding, each subtensor corresponds to a matrix block of $\mathcal{A}_{(1)}$ of size

$$\frac{I_1}{s} \times \frac{M}{t}.$$

g) Worker computation: Each worker is assigned encoded combinations of these blocks. After decoding the linear combinations, each worker effectively computes

$$\left(P \times \frac{I_1}{s}\right) \times \left(\frac{I_1}{s} \times \frac{M}{t}\right), \tag{7}$$

producing an output block of size

$$P \times \frac{M}{t}.$$

h) System model and workflow:

- **Encoding (Master):** The master encodes $\mathcal{A}$ and $\mathbf{B}$ and broadcasts them.
- **Computation (Workers):** Each worker performs a single local matrix multiplication.
- **Decoding (User):** The user reconstructs $\mathcal{Y}$ from a subset of results.

i) Straggler model: We assume the presence of stragglers, and design the system such that the result can be recovered from K suitably selected worker responses.

j) Design objective: We aim to jointly optimize the recovery threshold K, computation and memory at workers, communication cost, and decoding complexity at the user.

III. Coded Schemes and Performance Analysis

In this section, we present two coded computation strategies for distributed mode-1 tensor–matrix multiplication: a baseline matrix-based approach and a proposed tensor-aware scheme. We then provide a unified analysis of their performance in terms of recovery threshold, computation, memory, communication, and decoding complexity.

A. Matrix-Based PolyDot (Baseline [17])

A natural approach is to reduce the tensor computation to a matrix multiplication by unfolding $\mathcal{A}$ along mode-1, yielding $\mathcal{A}_{(1)} \in \mathbb{R}^{I_1 \times M}$. The problem then reduces to computing (4).

PolyDot coding is applied by partitioning $\mathbf{B}$ into s column blocks and $\mathcal{A}_{(1)}$ into $s \times t$ blocks. These blocks are encoded into a single-variable polynomial, where each worker evaluates the polynomial at a distinct point, performs a matrix multiplication, and returns the result.

While this approach achieves efficient distributed computation and robustness to stragglers, it relies on:

- explicit tensor unfolding, which increases memory traffic,
- high-degree univariate polynomial interpolation at the decoder.

B. Tensor-PolyDot (Proposed)

We instead operate directly on tensor subtensors without unfolding. The tensor $\mathcal{A}$ is partitioned into $s \times t$ subtensors across its modes, while $\mathbf{B}$ is partitioned along its columns.

The key idea is to encode:

- mode-1 partitions using a variable x,
- partitions along other modes using variables $y_2, \ldots, y_N$,

resulting in a *multivariate polynomial encoding*.

Each worker evaluates the encoded inputs at a tuple $(x, y_2, \ldots, y_N)$, performs a local multiplication, and returns the result. The fusion node then reconstructs the output using multivariate interpolation, which decomposes into several low-dimensional interpolation problems that can be performed in parallel.

This approach preserves tensor structure, avoids data reshaping, and significantly reduces decoding complexity.

C. Recovery Threshold

For both schemes, the recovery threshold is determined by the total number of distinct polynomial evaluations required to recover all coefficients. This yields $K = (2s-1)t$, which means that both approaches achieve identical robustness to stragglers.

D. Computation and Memory

Each worker computes a matrix multiplication of size

$$\left(P \times \frac{I_1}{s}\right) \times \left(\frac{I_1}{s} \times \frac{M}{t}\right), \tag{8}$$

leading to

$$\text{flops/worker} \ \sim\ P \frac{I_1}{s} \frac{M}{t}. \tag{9}$$

The peak memory requirement per worker, including input and output blocks, is equivalent to

$$\text{memory} \ \sim\ P \frac{I_1}{s} + \frac{I_1}{s} \frac{M}{t} + P \frac{M}{t}. \tag{10}$$

These quantities are identical for both matrix- and tensor-based schemes, since they depend only on partition sizes.

E. Communication Cost

Each worker returns an output block of size $P \times \frac{M}{t}$. Consequently, the total communication required for the user to recover the result is obtained as

$$\text{Worker} \rightarrow \text{User}: \quad (2s-1)PM. \tag{11}$$

This quantity is independent of t, highlighting that partitioning across additional modes does not increase total communication.

F. Decoding Complexity and Latency

A key difference between the two schemes lies in the decoding stage, which is detailed below.

a) Matrix-based decoding (univariate): After variable folding, decoding requires interpolation of a univariate polynomial of degree $(2s-1)t-1$, leading to a complexity

$$C_{\text{mat}} \sim PM(2s-1)^2 t. \tag{12}$$

b) Tensor-based decoding (multivariate): In the proposed scheme, multivariate decoding decomposes into low-dimensional interpolations along the coding variables, with total complexity

$$C_{\text{ten}} \sim PM\Big((2s-1)^2 + \sum_{n=2}^{N} u_n^2\Big). \tag{13}$$

c) Decoding speedup: The resulting speedup is

$$S_{\text{decode}} = \frac{(2s-1)^2 t}{(2s-1)^2 + \sum_{n=2}^{N} u_n^2} \approx t \tag{14}$$

for balanced partitions.

d) Latency interpretation: In distributed systems, decoding at the fusion node often lies on the critical path. Thus, reducing decoding complexity directly reduces end-to-end latency. The proposed multivariate scheme achieves substantial latency reduction without affecting other system metrics.

G. System-Level Insights

The comparison between the two approaches reveals several important system-level observations:

- **Decoding efficiency:** The proposed tensor-based scheme reduces decoding complexity by approximately a factor t, significantly improving latency at the fusion node.
- **Memory locality:** By operating directly on tensor sub-tensors, the tensor-based approach avoids the explicit data reshuffling associated with tensor unfolding and can improve memory locality.
- **Parallelism:** Multivariate decoding decomposes into multiple smaller interpolation problems, enabling parallel implementation.
- **No penalty in other metrics:** Recovery threshold, computation, memory, and communication remain identical to the matrix-based baseline.

Overall, the proposed tensor-aware coding framework introduces a new tradeoff dimension: it enables substantial reductions in decoding complexity while preserving all other performance guarantees.

TABLE I
COMPARISON OF MATRIX-POLYDOT AND TENSOR-POLYDOT SCHEMES

Metric	**Matrix-PolyDot**	**Tensor-PolyDot (Proposed)**
Data representation	Unfolded matrix $\mathcal{A}_{(1)}$	Native tensor $\mathcal{A}$
Encoding type	Univariate polynomial	Multivariate polynomial
Recovery threshold	$(2s-1)t$	$(2s-1)t$
Worker computation	$P\frac{I_1}{s}\frac{M}{t}$	Same
Worker memory	$P\frac{I_1}{s} + \frac{I_1}{s}\frac{M}{t} + P\frac{M}{t}$	Same
Communication (total)	$(2s-1)PM$	$(2s-1)PM$
Decoding type	Univariate interpolation	Multivariate interpolation
Decoding complexity	$PM(2s-1)^2 t$	$PM\big((2s-1)^2 + \sum u_n^2\big)$
Decoding latency	High	Reduced ($\approx t\times$)
Memory locality	Poor	Good
Parallel decoding	Limited	High

H. Comparison of Coded Schemes

Table I summarizes the key differences between the matrix-based and tensor-based coded computation schemes.

IV. NUMERICAL EVALUATION

We evaluate the proposed tensor-PolyDot scheme and compare it with the matrix-PolyDot baseline in terms of decoding complexity, runtime, and system-level tradeoffs. Unless otherwise stated, we consider a third-order tensor with

$$I_1 = I_2 = I_3 = 512, \qquad P = 256,$$

and fix the mode-1 partition parameter to $s = 8$. The non-shared modes are partitioned using balanced configurations

$$u_2 = u_3 = u, \qquad t = u_2 u_3 = u^2.$$

We limit the partition factor to $t \leq 128$, which corresponds to practical system configurations with moderate numbers of workers and memory constraints.

A. Baseline Configuration

We first consider a representative setting with $u_2 = u_3 = 4$, yielding $t = 16$ and the recovery threshold

$$K = (2s-1)t = 15 \cdot 16 = 240.$$

The decoding complexities from (12) and (13) are

$$C_{\text{mat}} = PM(2s-1)^2 t = 2.416 \times 10^{11},$$

and

$$C_{\text{ten}} = PM\big((2s-1)^2 + u_2^2 + u_3^2\big) = 1.723 \times 10^{10}.$$

This results in a decoding speedup, from (14), of

$$S_{\text{decode}} = \frac{C_{\text{mat}}}{C_{\text{ten}}} \approx 14.0 \times .$$

This example highlights that the proposed tensor-based scheme significantly reduces decoding complexity while preserving identical recovery threshold, computation, memory, and communication cost.

B. Scaling with Tensor Partitioning

We now examine the impact of the partition factor t.

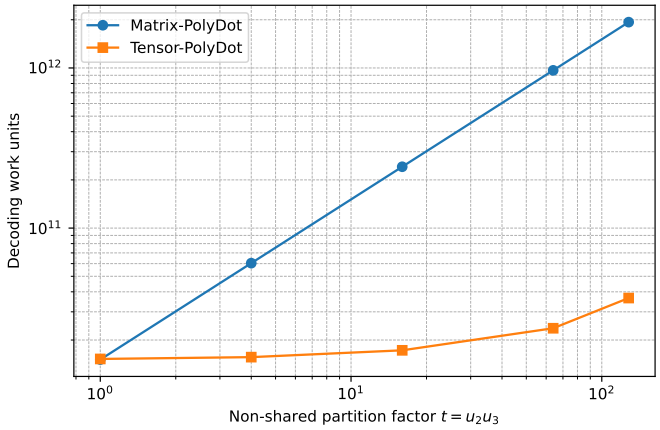

Fig. 1. Decoding complexity versus non-shared partition factor t. Tensor-PolyDot significantly reduces fusion-node decoding complexity compared with matrix-PolyDot.

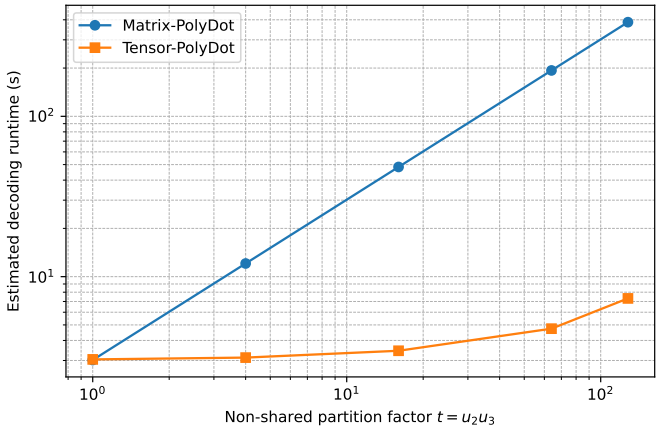

Fig. 3. Estimated decoding runtime versus non-shared partition factor t. Runtime is obtained from decoding complexity assuming an effective CPU throughput of 5×10^9 operations/s.

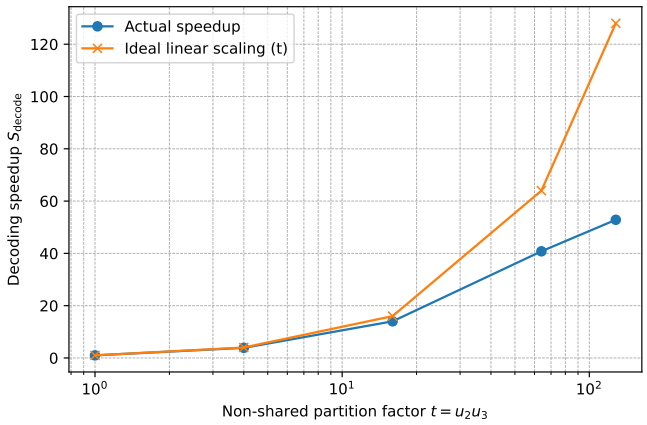

Fig. 2. Decoding speedup of tensor-PolyDot over matrix-PolyDot. The achieved speedup approaches the ideal linear scaling with t for balanced partitions.

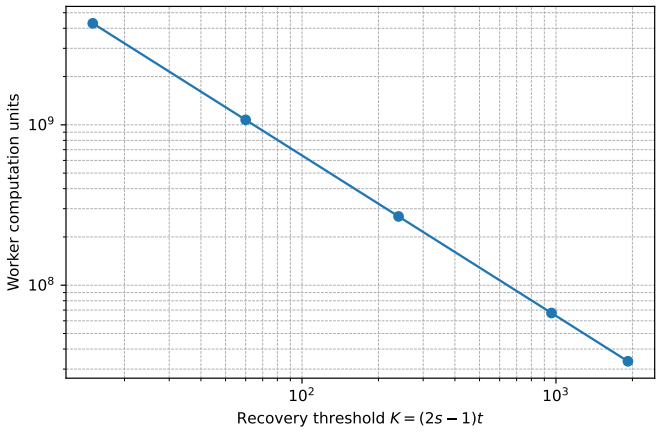

Fig. 4. Tradeoff between recovery threshold and per-worker computation. Larger t reduces worker computation but increases the number of required worker responses.

a) Decoding complexity: Fig. 1 shows the decoding complexity as a function of t. The matrix-PolyDot scheme scales linearly with t, since it requires interpolation of a polynomial of increasing degree. In contrast, tensor-PolyDot grows much more slowly, as decoding is decomposed into multiple lower-dimensional interpolation problems.

b) Decoding speedup: Fig. 2 illustrates the decoding speedup in (14), which increases nearly linearly with t for balanced partitions. For $t = 16$, the speedup reaches approximately $14\times$, close to the ideal value $t = 16$.

C. Runtime Evaluation and Latency

To assess practical implications, we translate decoding complexity into runtime using a simple computational model. Specifically, we map the number of decoding operations to execution time assuming a fixed effective interpolation throughput of 5×10^9 operations per second, which captures the memory-bound nature of polynomial interpolation.

a) Estimated runtime: Fig. 3 shows the estimated decoding runtime as a function of t. The matrix-based scheme exhibits rapidly increasing latency as t grows, whereas tensor-PolyDot maintains significantly lower runtime due to its multivariate decoding structure.

b) Latency implications: The results confirm that decoding constitutes a dominant component of end-to-end latency in coded distributed computation systems. By reducing decoding complexity, the proposed tensor-aware scheme effectively alleviates this bottleneck. For moderate partition sizes (e.g., $t = 16$), the latency reduction approaches an order of magnitude.

D. Threshold–Computation Tradeoff

Finally, we examine the tradeoff between recovery threshold and worker computation.

Increasing t reduces per-worker computation, since each worker processes smaller blocks. However, it also increases the recovery threshold $K = (2s-1)t$, requiring more worker

responses. Fig. 4 illustrates this tradeoff. As t increases, the system shifts from computation-limited to communication/latency-limited regimes. Therefore, the choice of t should balance available worker resources, memory constraints, and latency requirements.

E. Discussion

The results reveal three key insights. First, tensor-PolyDot preserves the same recovery threshold and communication cost as matrix-PolyDot for fixed (s, t). Second, the primary gain arises from decoding: multivariate interpolation replaces a single high-degree problem with multiple smaller ones. Third, this reduction directly translates into lower latency, making the proposed approach particularly effective in systems where the fusion node is the performance bottleneck.

Overall, the proposed tensor-aware coding framework improves end-to-end efficiency without imposing additional worker-side or communication overhead, highlighting the benefit of exploiting tensor structure in distributed signal processing systems.

V. Conclusion

In this work, we proposed a tensor-aware coded computation framework for distributed mode-1 tensor–matrix multiplication. By operating directly on tensor subtensors rather than relying on matrix unfolding, the proposed scheme preserves the inherent multi-dimensional structure of the data and enables a multivariate coding strategy.

We showed that, for a fixed partitioning configuration, the proposed tensor-PolyDot scheme achieves the same recovery threshold, communication cost, and worker-side computation and memory requirements as conventional matrix-based PolyDot approaches. Despite these identical system-level guarantees, the proposed framework significantly reduces decoding complexity at the fusion node by replacing high-degree univariate interpolation with structured multivariate interpolation. This reduction translates directly into lower decoding latency, which is critical in distributed signal processing systems where the fusion node often lies on the critical path.

Our numerical results demonstrate that the decoding complexity can be reduced by a factor proportional to the tensor partitioning factor, yielding substantial speedups, including up to an order-of-magnitude improvement in practical settings. In addition, the tensor-based approach improves memory locality and enables parallel decoding across tensor modes, making it better aligned with modern hardware architectures.

Overall, the proposed framework introduces a new design dimension in coded distributed computing: it enables significant reductions in decoding complexity and latency without incurring any penalty in recovery threshold, communication, or worker-side resources. These properties make it particularly relevant to latency- and resource-constrained aeronautical and aerospace signal processing systems, including airborne sensing, avionics, and satellite-based processing.

References

[1] M. Mozaffari, W. Saad, M. Bennis, and M. Debbah, "A tutorial on UAVs for wireless networks: Applications, challenges, and open problems," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2334–2360, 2019.

[2] N. D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos, "Tensor decomposition for signal processing and machine learning," *IEEE Transactions on Signal Processing*, vol. 65, no. 13, pp. 3551–3582, 2017.

[3] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM review*, vol. 51, no. 3, pp. 455–500, 2009.

[4] A. Cichocki, D. P. Mandic, L. De Lathauwer, G. Zhou, Q. Zhao, C. F. Caiafa, and A. H. Phan, "Tensor decompositions for signal processing applications," *IEEE Signal Processing Magazine*, vol. 32, no. 2, pp. 145–163, 2015.

[5] B. Khaleghi, A. Khamis, F. O. Karray, and S. N. Razavi, "Multisensor data fusion: A review of the state-of-the-art," *Information Fusion*, vol. 14, no. 1, pp. 28–44, 2013.

[6] K. Lee, M. Lam, R. Pedarsani, and K. Ramchandran, "Speeding up distributed machine learning using codes," *IEEE Transactions on Information Theory*, vol. 64, no. 3, pp. 1514–1529, 2018.

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

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

[9] M. Soleymani, R. E. Ali, H. Mahdavifar, and A. S. Avestimehr, "List-decodable coded computing: Breaking the adversarial toleration barrier," *IEEE Journal on Selected Areas in Information Theory*, vol. 2, no. 3, pp. 867–878, 2021.

[10] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Polynomial codes: an optimal design for high-dimensional coded matrix multiplication," in *Proceedings, Advances in Neural Information Processing Systems (NeurIPS)*, Long Beach, CA, USA, Dec. 2017, pp. 4403–4413.

[11] S. Dutta, V. R. Cadambe, and P. Grover, "Short-Dot: Computing large linear transforms distributedly using coded short dot products," in *Proceedings, Advances in Neural Information Processing Systems (NeurIPS)*, vol. 29, Barcelona, Spain, Dec. 2016.

[12] S. Dutta, Z. Bai, H. Jeong, T. M. Low, and P. Grover, "A unified coded deep neural network training strategy based on generalized PolyDot codes," in *Proceedings, IEEE International Symposium on Information Theory (ISIT)*, Vail, CO, Jun. 2018, pp. 1585–1589.

[13] M. R. D. Salehi, A. Tanha, and D. Malak, "Structured polynomial codes," in *Recent results, IEEE International Symposium on Information Theory (ISIT)*, Athens, Greece, Jul. 2024.

[14] A. Tanha, "Structured distributed computation under communication and computation constraints," Ph.D. dissertation, Sorbonne Université, Paris, France, 2026. [Online]. Available: https://theses.fr/s384673

[15] A. Khalesi, A. Tanha, D. Malak, and P. Elia, "Non-linearly separable distributed computing: A sparse tensor factorization approach," in *Proceedings, IEEE International Symposium on Information Theory (ISIT)*, Guangzhou, China, Jun. 2026.

[16] A. Tanha, M. R. D. Salehi, M. Dutta, and D. Malak, "Cooperative coded matrix multiplication in secrecy-constrained vehicular networks," in *IEEE 103rd Vehicular Technology Conference (VTC)*, Nice, France, Jun. 2026, to appear.

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