IMPROVED LATENCY-COMMUNICATION TRADE-OFF FOR MAP-SHUFFLE-REDUCE SYSTEMS WITH STRAGGLERS

Jingjing Zhang and Osvaldo Simeone

Dept. of Informatics, King's College London, UK

ABSTRACT

In a distributed computing system operating according to the map-shuffle-reduce framework, coding data prior to storage can be useful both to reduce the latency caused by straggling servers and to decrease the inter-server communication load in the shuffle phase. In prior work, a concatenated coding scheme was proposed for a matrix multiplication task. In this scheme, the outer Maximum Distance Separable (MDS) code is leveraged to correct erasures caused by stragglers, while the inner repetition code is used to improve the communication efficiency in the shuffle phase by means of coded multicasting. In this work, it is demonstrated that it is possible to leverage the redundancy created by repetition coding in order to increase the rate of the outer MDS code and hence to increase the multicasting opportunities in the shuffle phase. As a result, the proposed approach is shown to improve over the best known latency-communication overhead trade-off.

Index Terms— Distributed computing, map-shuffle-reduce, coded multicasting, stragglers, coding.

1. INTRODUCTION

Consider a distributed computing system, in which K servers are tasked with the computation of the matrix product $\mathbf{Y} = \mathbf{AX}$, where data matrix $\mathbf{X}$ is available at all servers, while the matrix $\mathbf{A}$ can be partially stored at each server. In particular, each server can store information about $\mathbf{A}$ up to a fraction $\mu \leq 1$ of its size. As a result, servers need to collaborate in order to compute the output $\mathbf{Y}$ by communicating over a shared multicast channel. Furthermore, servers are typically subject to random computing times, and hence measures should be taken in order to ensure correct distributed computation even in the presence of a given number of straggling servers [1]. The straggling issue has been addressed in master-slave architecture where there is no inter-server communication [2–7].

Another standard framework to implement this operation is *map-shuffle-reduce* [8, 9]. Accordingly, in the *map phase*, the servers compute Intermediate Values (IVs) that depend on the available information about $\mathbf{A}$ and $\mathbf{X}$. Generally, only a subset of q servers is able to complete this phase within a desired latency. In the *shuffle phase*, functions of the IVs are exchanged among the q non-straggling servers. Finally, in the *reduce phase*, the non-straggling servers can collectively produce all the columns in $\mathbf{Y}$, with each server producing a given subset.

The performance of the system is characterized by a trade-off between computational latency — the time elapsed during the map phase — and communication overhead — the amount of information exchanged during the shuffle phase [10]. In fact, waiting for a longer time for more servers to compute their map operations reduces the need for communication of IVs during the shuffle phase. This trade-off depends on the available storage capacity μ at the servers, which limits the capability of the servers to compute IVs and to withstand erasures due to stragglers [11]. A concatenated coding scheme was proposed in [11] for the described matrix multiplication task (see Fig. 1).

In this paper, it is demonstrated that the redundancy of the repetition code in the scheme proposed in [11] can be used not only to accelerate communications in the shuffle phase, but also to correct erasures caused by the straggling servers. This approach is shown to improve the latency-communication trade-off derived in [11].

Notation: For $a \in \mathbb{N}^+, b \in \mathbb{Z}$, we define $\binom{a}{b} = 0$ when $a < b$ or $b < 0$, and we define $\binom{a}{0} = 1$. For $K, P \in \mathbb{N}^+$ with $K \leq P$, we define the set $[P] \triangleq \{1, 2, \cdots, P\}$, and the set $[K : P] \triangleq \{K, K+1, \cdots, P\}$. $|\mathcal{A}|$ represents the cardinality of set $\mathcal{A}$. We also have $0/0 = 0$. Matrices and vectors will be denoted by upper-case and lower-case bold font, respectively.

2. SYSTEM MODEL AND BACKGROUND

2.1. System Model

Consider a distributed implementation of the matrix multiplication task described by the equality

$$\mathbf{Y} = \mathbf{AX}, \tag{1}$$

with the task-specific matrix $\mathbf{A} \in \mathbb{F}_{2^T}^{m \times n}$ and the input data matrix $\mathbf{X} \in \mathbb{F}_{2^T}^{n \times N}$, where each element of matrices $\mathbf{A}$ and $\mathbf{X}$ consists of T bits, and we have the parameters $T, m, n, N \in$

Jingjing Zhang and Osvaldo Simeone have received funding from the European Research Council (ERC) under the European Union's Horizon 2020 Research and Innovation Programme (Grant Agreement No. 725731).

$\mathbb{N}^+$. We use $\boldsymbol{x}_i \in \mathbb{F}_{2^T}^n$ and $\boldsymbol{y}_i \in \mathbb{F}_{2^T}^m, i \in [N]$ to denote each column vector of input matrix $\mathbf{X}$ and output matrix $\mathbf{Y}$, respectively. Hence, the matrix product (1) corresponds to the N linear equations $\boldsymbol{y}_i = \mathbf{A}\boldsymbol{x}_i$, for $i \in [N]$.

There are K distributed servers, each having a storage of size μmnT bits, with $\mu \in [1/K, 1]$. Hence, each server k, with $k \in [K]$, can store a number of bits equal to a fraction μ of the size of matrix $\mathbf{A}$. The lower bound $1/K$ ensures that the entire matrix can be stored across all servers. Specifically, each server is assumed to store up to $m\mu$ row vectors selected from the rows $\mathcal{C} = \{\boldsymbol{c}_i\}_{i=1}^{m'}$ of the linearly encoded matrix

$$\mathbf{C} = [\boldsymbol{c}_1^T, \cdots, \boldsymbol{c}_{m'}^T]^T = \mathbf{G}\mathbf{A}, \tag{2}$$

where we have defined the encoding matrix $\mathbf{G} \in \mathbb{F}_{2^T}^{m' \times m}$, with integer $m' \geq m$. The rows stored by server k are described by the set $\mathcal{C}_k \subseteq \mathcal{C}$, with $|\mathcal{C}_k| \leq m\mu$. Furthermore, each server is assumed to know the entire data matrix $\mathbf{X}$, and they can communicate to each other over multicast links.

Map phase: Each server $k \in [K]$ computes the products

$$\mathcal{I}_k = \{\boldsymbol{c}\mathbf{X} \in \mathbb{F}_{2^T}^{1 \times N} : \boldsymbol{c} \in \mathcal{C}_k\}, \tag{3}$$

of size $m\mu$ for all stored encoded rows $\boldsymbol{c} \in \mathcal{C}_k$. The contents of set $\mathcal{I}_k$ are referred to as the IVs available at server k using the map-reduce terminology. We define $D(q)$ as the average time required for the first q servers to complete their computations. The set $\mathcal{Q} \in [K]$, with $|\mathcal{Q}| = q$, of q servers is arbitrary and function $D(q)$ is determined by the distribution of the random computation times of the servers. As a specific example, if each server requires a time distributed as a shifted exponential with minimum value μN and average $2\mu N$, i.e., with cumulative distribution function $F(t) = 1 - e^{-(t/(\mu N)-1)}$, for all $t \geq \mu N$, the computation latency $D(q)$ is derived as [11]

$$D(q) = \mu N\left(1 + \sum_{j=K-q+1}^{K} \frac{1}{j}\right). \tag{4}$$

Shuffle phase: After the average time $D(q)$, all servers in the non-straggling set $\mathcal{Q}$ coordinate by assigning each server k in $\mathcal{Q}$ a subset of the vectors $\{\boldsymbol{y}_i\}_{i=1}^N$ to be computed. The indices of the vectors "reduced" at server k are described by the set $\mathcal{R}_k \subseteq [N]$, with $\bigcup_{k \in \mathcal{Q}} \mathcal{R}_k = [N]$. To enable each server k to reconstruct the vectors $\{\boldsymbol{y}_i : i \in \mathcal{R}_k\}$, any functions of the computed IVs (3) can be exchanged among the q servers in $\mathcal{Q}$ during the shuffle phase. We use $\mathcal{M}_k$ to denote the sets of functions of the IVs $\mathcal{I}_k$ that each server k multicasts to all other servers in $\mathcal{Q}$ during the shuffle phase.

Reduce phase: With the received data $\{\mathcal{M}_{k'} : k' \in \mathcal{Q}\backslash\{k\}\}$ multicast by the other servers and with the locally computed IVs $\mathcal{I}_k$, each server k in $\mathcal{Q}$ computes the assigned vectors $\{\boldsymbol{y}_i : i \in \mathcal{R}_k\}$ in the reduce phase.

Performance Criteria: For a value of the number q, of non-straggling servers, the total number of bits exchanged among the servers during the shuffle phase is $\sum_{k \in \mathcal{Q}} |\mathcal{M}_k|$. With normalization by the number mT of bits per output column, we define the communication load as

$$L(q) = \frac{\sum_{k \in \mathcal{Q}} |\mathcal{M}_k|}{mT}. \tag{5}$$

Given a computation latency function $D(q)$, e.g., (4), a pair (D, L) is said to be achievable if there exists a map, shuffling and reduce policy, that ensures the correct computation of matrix product (1) across q non-straggling servers with computation latency $D \leq D(q)$ and communication load $L \leq L(q)$ for some q and sufficiently large m, N, T. Finally, we define the optimal latency-load trade-off curve as

$$L^*(D) = \inf\{L : (D, L) \text{ is achievable}\}. \tag{6}$$

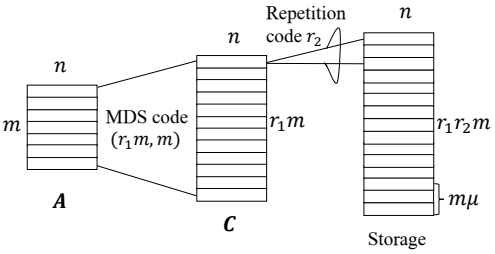

Fig. 1. Encoding scheme used in the map phase.

2.2. Background

In [11], a map-shuffling-reduce coding scheme is introduced that is based on the concatenation of two codes: an MDS code of redundancy r_1 and a repetition code of redundancy r_2, as shown in Fig. 1. The MDS code is used in order to enable decoding from the output of an arbitrary set of q non-straggling servers. To this end, each server needs to store m/q distinct MDS-coded data points, which is made possible by using a $(K \times m/q, m)$ MDS code with redundancy

$$r_1 = K/q. \tag{7}$$

Note that this requires that the fractional cache capacity satisfies $\mu \geq 1/q$. If $\mu > 1/q$, a repetition code of redundancy

$$r_2 = q\mu \tag{8}$$

is used for the purpose of reducing the inter-server communication load during the shuffle phase. This reduction is obtained by leveraging coded multicasting based on the available side information at the servers [8].

Overall, since each server can store at most $m\mu$ distinct encoded rows by the storage constraint, the total storage redundancy across all the K servers with respect to the m rows of matrix $\mathbf{A}$ is given as $Km\mu/m = K\mu$. The map code in Fig. 1 splits this redundancy between the above two codes as

$$K\mu = r_1 r_2, \tag{9}$$

where r_1 and r_2 are selected in [11] as (7) and (8), respectively.

3. IMPROVED CONCATENATED CODING

In this section, we propose a policy that is based on the idea of leveraging the overall (rm, m) concatenated code with redundancy $r = r_1 r_2$ in Fig. 1 for the purpose of correcting erasures due to stragglers. In essence, the repetition code of rate r_2 is used not only as a bandwidth-reducing code, but it also contributes to the reduction of latency in the map phase. This allows us to establish a set of feasible choices of r_1 and r_2 that contain the selection on (7)-(8) as special case.

As a result, the proposed approach allows an MDS code with redundancy $r_1 \leq K/q$ to encode matrix $\mathbf{A}$ and a repetition code with storage redundancy $r_2 \geq q\mu$, while still satisfying the storage constrain $r_1 r_2 \leq K\mu$ (cf. (9)), and the achievability requirement. Hence, as compared to the scheme with the redundancy choice of (7)-(8) in [11], the proposed scheme can potentially operate with a lower rate r_1 and a higher rate r_2 and is potentially able to further speed up the shuffle phase, reducing the communication load $L(q)$.

To proceed, we first provide a characterization of the achievable communication load $L(q)$ for the proposed policy; then, we present an illustrative example; and, finally, we describe the corresponding general scheme.

3.1. Main Result

We first identify sufficient conditions on the pair (r_1, r_2) to yield a feasible policy in the next proposition.

Proposition 1. *For storage capacity $\mu \in [1/K, 1]$ and number of non-straggling servers $q \in [\lceil 1/\mu \rceil : K]$, sufficient conditions for rates (r_1, r_2) to yield a feasible policy are*

$$qr_1 \in [q : K],\ r_2 \in [\lfloor q\mu \rfloor : \lfloor K\mu \rfloor], \tag{10a}$$

$$r_1 r_2 \leq K\mu,\ \textit{and} \tag{10b}$$

$$\binom{K}{r_2} - \binom{K-q}{r_2} \geq \frac{1}{r_1}\binom{K}{r_2}. \tag{10c}$$

Proof. The proof is presented in [12]. □

Remark 1. *Condition* (10a) *defines the domains of redundancies r_1 and r_2. Condition* (10b) *impose the storage capacity constraint (cf.* (9)*), while condition* (10c) *ensures the feasibility of the reconstruction requirement of data matrix* $\mathbf{A}$*. It can be verified that the choice* (7)-(8) *in [11] satisfies all conditions* (10) *(see [12]). Furthermore, when $K - q < \lfloor K\mu \rfloor$, a feasible choice is $(r_1 = 1, r_2 = \lfloor K\mu \rfloor)$. With this choice, the MDS code can be avoided, as shown in the example of Section 4.*

Using a feasible redundancy pair (r_1, r_2) satisfying (10), the followed communication load is achievable.

Proposition 2. *For a matrix multiplication task executed by K distributed servers, each having a fractional storage size $\mu \in [1/K, 1]$, the following communication load is achievable in the presence of $K - q$ straggling servers with $q \in [\lceil 1/\mu \rceil : K]$*

$$\min_{(r_1, r_2)} \left(L(q) = N \sum_{j=s_q}^{s_{max}} \frac{B_j}{j} + \frac{N\left(1 - \frac{r_1 r_2}{K} - \sum_{j=s_q}^{s_{max}} B_j\right)}{s_q - 1} \right) \tag{11}$$

where pair (r_1, r_2) is constrained to satisfy the feasible condition (10) *and we have defined*

$$B_j \triangleq \frac{\binom{q-1}{j}\binom{K-q}{r_2-j}}{\frac{1}{r_1}\binom{K}{r_2}},\ s_q \triangleq \inf\left\{ s : \sum_{j=s}^{s_{max}} B_j \leq 1 - \frac{r_1 r_2}{K} \right\} \tag{12}$$

$$s_{max} \triangleq \min\{q - 1, r_2\}. \tag{13}$$

Proof. The strategy follows the approach in [11], but it relies on a more general choice of values for the rate pair (r_1, r_2) satisfying (10). Details are presented in [12]. □

Remark 2. *Since the solution* (7) *and* (8) *of [11] is always feasible for the constraints* (10)*, it follows that the achievable communication load $L(q)$ in* (11) *is always no larger than that in [11]. This is because the load* (11) *with r_1 and r_2 in* (7)-(8) *is no larger than the load in [11, eq. (9)]. To demonstrate that the improvement can be strict, beside the example given in Section 4, we provide next a numerical example.*

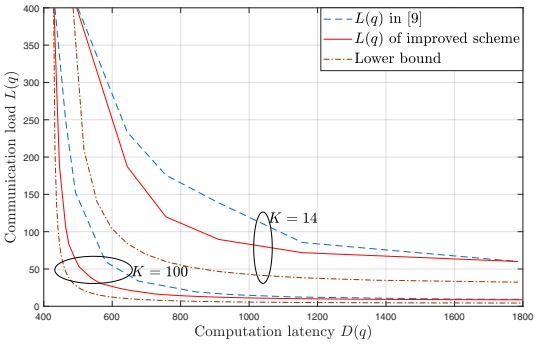

Fig. 2. Achievable loads $L(q)$ as a function of the computation latency $D(q)$ for the proposed scheme and for the scheme in [11], along with lower bound in [11], with $N = 840, \mu = 1/2$ and different values of K.

A comparison of the achievable communication loads of the proposed scheme and of the scheme in [11], as well as the lower bound in [11], can be found in Fig. 2, where we apply the computation latency $D(q)$ modeled as in (4) and we have $N = 840, \mu = 1/2$, and different values of K. For each value of K, at the two end points of computation latency $D(q)$, obtained with $q = \lceil 1/\mu \rceil$ and $q = K$, respectively, the two achievable communication loads coincide. The proposed scheme is seen to bring a positive reduction in communication load, as compared to the algorithm in [11] for intermediate values of the latency $D(q)$, that is, when there are a moderate number of straggling servers. Furthermore, as the total number K of servers increases, it is observed that the reduction is

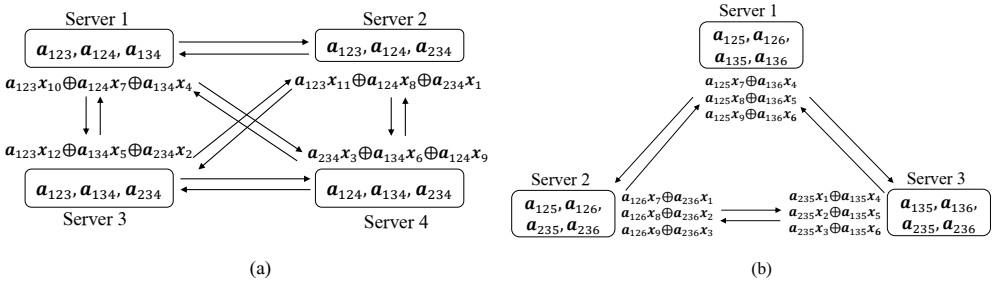

Fig. 3. Illustration of two phases of shuffling for the example in Section 4: (a) Coded shuffling among servers 1, 2, 3 and 4 in the first phase ($i = 3$); (b) Coded shuffling among servers 1, 2, 3 in the second phase ($i = 2$).

more significant, and the gap between the achievable load of the proposed scheme and the lower bound becomes smaller.

4. EXAMPLE AND DISCUSSION

Consider $K = 6$ servers, with $q = 4$ non-stragglers, storage capacity $\mu = 1/2$, as well as the parameters $m = 20$, $N = 12$ and sufficiently large T [13]. For this example, the scheme in [11] chooses $r_1 = 3/2$ and $r_2 = 2$ according to (7)-(8). As shown below, the proposed scheme instead can operate with $r_1 = 1$ and $r_2 = K\mu = 3$. Accordingly, the MDS code is not used and the scheme only relies on the repetition code of rate r_2 to correct erasures.

Map phase: Without using an MDS code to encode the rows of matrix $\mathbf{A}$, i.e., with $r_1 = 1$, we have $\mathbf{C} = \mathbf{A}$ in (2). Each row of matrix $\mathbf{A}$ is then replicated $r_2 = 3$ times, so that each server k stores $|\mathcal{C}_k| = \mu m = 10$ uncoded rows of $\mathbf{A}$. This is done by storing each row in a subset $\mathcal{K} \subseteq [K]$ of three servers, with $|\mathcal{K}| = r_2$. We write $\boldsymbol{a}_{\mathcal{K}}$ for the row that is stored at all servers in set $\mathcal{K}$ and we have $\mathcal{C}_k = \{\boldsymbol{a}_{\mathcal{K}} : k \in \mathcal{K}\}$ for the set of rows stored at server k. WLOG, we assume that servers 1, 2, 3, 4 are the first $q = 4$ servers that complete their computations, i.e., $\mathcal{Q} = \{1, 2, 3, 4\}$. We recall that each server k computes the $|\mathcal{C}_k| = 10$ products in the set $\mathcal{I}_k$ in (3) in the map phase. In the reduce phase, each server k is assigned to output $N/q = 3$ consecutive vectors, i.e., $\mathcal{R}_k = \{\boldsymbol{y}_{3(k-1)+i} = \mathbf{A}\boldsymbol{x}_{3(k-1)+i} : i \in [3]\}$.

Shuffle phase. To this end, each server needs to obtain a set of IVs through multicast transmissions. Take server 1 for example. To reduce vector $\boldsymbol{y}_1 = \mathbf{A}\boldsymbol{x}_1 \in \mathcal{R}_1$, server 1 can use the IVs $\{\boldsymbol{a}\boldsymbol{x}_1 : \boldsymbol{a} \in \mathcal{C}_1\}$ in $\mathcal{I}_1$. Hence, it needs ten extra IVs $\{\boldsymbol{a}\boldsymbol{x}_1 : \boldsymbol{a} \in \mathcal{C}\backslash\mathcal{C}_1\}$, each of which has been computed by at least one of the servers 2, 3, 4. This is because each row $\boldsymbol{a} \notin \mathcal{C}_1$ is stored at $r_2 = 3$ servers that do not include server 1. The same holds for vectors $\boldsymbol{y}_2$ and $\boldsymbol{y}_3$, and thus server 1 needs 30 IVs in total from servers 2, 3, 4. Similarly, each of the other three servers requires 30 IVs. All these required IVs are exchanged in the shuffle phase.

We operate separately by multicasting messages within groups of $i + 1$ servers in three different phases, with $i = 3, 2, 1$, and carried out in this order. This follows the same approach as in [11], with the caveat that here we can benefit from a larger multicasting gain (see Remark 3).

Accordingly, in the first phase, labeled as $i = 3$ and illustrated in Fig. 3(a), the servers share the needed IVs that are available at subsets of three of the four servers in $\mathcal{Q}$. By construction, one such message exists at each of the four disjoint subsets of three servers in $\mathcal{Q}$. To this end, we perform coded shuffling among the four servers by sending a multicasting message from one server to the other three. In each transmission, each of the receiving server can recover one desired IVs. At the end of this phase, each server can obtain three IVs.

In the second phase, for $i = 2$, we deliver the needed IVs that are available at subsets of two of the four server in $\mathcal{Q}$. By construction, there are 72 such IVs, with six available at each of the four subsets of three servers in $\mathcal{Q}$. The four groups of three servers operate in the same way by sending a set of multicasting messages from one server to the other two. Take for example the subset $\{1, 2, 3\}$ illustrated in Fig. 3(b). Server 2 sends the three messages $\{\boldsymbol{a}_{126}\boldsymbol{x}_{6+i} \oplus \boldsymbol{a}_{236}\boldsymbol{x}_i\}_{i=1}^3$, from which server 1 and 3 obtain $\{\boldsymbol{a}_{236}\boldsymbol{x}_i, i \in [3]\}$ and $\{\boldsymbol{a}_{126}\boldsymbol{x}_{6+i}\}_{i=1}^3$, respectively, by canceling their own interference with side information. Server 1 and 3 can act similarly. As a result, for any subset of 3 servers, each server can obtain 6 needed IVs. After this phase, each server can recover 18 IVs in total.

Finally, each server k still needs 9 IVs $\{\boldsymbol{a}_{i56}\boldsymbol{x}_{3(k-1)+j} : i \in [4]\backslash\{k\}, j \in [3]\}$, each of which is computed by only one of the servers in $\mathcal{Q}$. Hence, in the last phase, for $i = 1$, the overall 36 IVs are shared by means of unicast transmission.

Overall, $4 + 36 + 36 = 76$ coded IVs are communicated. This yields a communication load of $L(q = 4) = 76/m = 3.8$, which is smaller than that of $L(q = 4) = 4.2$ in [11].

Remark 3. *This example shows that the MDS code can be avoided when q is sufficient large. This is because, thanks to the repetition code, the entire matrix* $\mathbf{A}$*, and hence the product* (1)*, can be recovered by combining the information available at the non-straggling servers when we store uncoded rows from matrix* $\mathbf{A}$*. As compared to the policy in [11], a higher storage redundancy* $r_2 = K\mu = 3$ *is obtained for the repetition code. As a result, the multicasting gain* $K\mu = 3$ *can be reaped in the shuffle phase, while the maximum gain for [11] is limited to 2. Hence, the proposed scheme achieves a lower communication load.*

5. REFERENCES

[1] J. Dean and L. A. Barroso, "The tail at scale," *Communications of the ACM*, vol. 56, no. 2, pp. 74–80, Feb 2013.

[2] Rashish Tandon, Qi Lei, Alexandros G. Dimakis, and Nikos Karampatziakis, "Gradient coding: Avoiding stragglers in distributed learning," in *Proceedings of the 34th International Conference on Machine Learning*, Aug 2017, vol. 70, pp. 3368–3376.

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

[4] S. Dutta, M. Fahim, F. Haddadpour, H. Jeong, V. Cadambe, and P. Grover, "On the optimal recovery threshold of coded matrix multiplication," *CoRR*, vol. abs/1801.10292, 2018.

[5] Q. Yu, N. Raviv, J. So, and A. S. Avestimehr, "Lagrange coded computing: Optimal design for resiliency, security and privacy," *CoRR*, vol. abs/1806.00939, 2018.

[6] S. Dutta, V. Cadambe, and P. Grover, "Short-dot: Computing large linear transforms distributedly using coded short dot products," in *Advances in Neural Information Processing Systems 29*, pp. 2100–2108. 2016.

[7] Ankur Mallick, Malhar Chaudhari, and Gauri Joshi, "Rateless codes for near-perfect load balancing in distributed matrix-vector multiplication," .

[8] S. Li, M. A. Maddah-Ali, and A. S. Avestimehr, "Coded mapreduce," in *Proc. Allerton Conf. Communication, Control and Computing*, Sept 2015, pp. 964–971.

[9] S. Li, M. A. Maddah-Ali, Q. Yu, and A. S. Avestimehr, "A fundamental tradeoff between computation and communication in distributed computing," *IEEE Trans. Inf. Theory*, vol. 64, no. 1, pp. 109–128, Jan 2018.

[10] H. B. McMahan, E. Moore, D. Ramage, and B. A. y Arcas, "Federated learning of deep networks using model averaging," *CoRR*, vol. abs/1602.05629, 2016.

[11] S. Li, M. A. Maddah-Ali, and A. S. Avestimehr, "A unified coding framework for distributed computing with straggling servers," in *IEEE Globecom Workshops (GC Workshop)*, Dec 2016, pp. 1–6.

[12] J. Zhang and O. Simeone, "Improved latency-communication trade-off for map-shuffle-reduce systems with stragglers," *CoRR*, vol. abs/1808.06583, 2018.

[13] H. Lin and D. J. Costello, *Error control coding fundamentals and applications*, Prentice Hall, New York, 2004.