FAST AND EFFICIENT DISTRIBUTED MATRIX-VECTOR MULTIPLICATION USING RATELESS FOUNTAIN CODES

Ankur Mallick
Carnegie Mellon Univ. (CMU)
amallic1@andrew.cmu.edu

Malhar Chaudhari[*]
Oracle Corporation
malharchaudhari@gmail.com

Gauri Joshi
Carnegie Mellon Univ. (CMU)
gaurij@andrew.cmu.edu

ABSTRACT

We propose a *rateless fountain coding* strategy to alleviate the problem of straggling nodes – computing nodes that unpredictably slowdown or fail – in distributed matrix-vector multiplication. Our algorithm generates linear combinations of the m rows of the matrix, and assigns them to different worker nodes, which then perform row-vector products with the encoded rows. The original matrix-vector product can be decoded as soon as slightly more than m row-vector products are collectively completed by the nodes. This strategy enables fast nodes to steal work from slow nodes, without requiring the knowledge of node speeds. Compared to recently proposed fixed-rate erasure coding strategies which ignore partial work done by straggling nodes, rateless codes have a significantly lower overall delay, and a smaller computational overhead.

***Index Terms*—** Distributed Computing, Straggler Mitigation, Erasure Codes

1. INTRODUCTION

Matrix-vector multiplications form the core of a plethora of scientific computing and machine learning applications that include solving partial differential equations [1], forward and back propagation in neural networks [2], computing the PageRank of graphs [3] etc. In this age of Big Data, most of these applications involve multiplying extremely large matrices and vectors and the computations cannot be performed efficiently on a single machine and are instead distributed across multiple computation nodes. Individual nodes (the *workers*) perform their respective tasks in parallel while a central node (the *master*) aggregates the output of all the workers to complete the computation. Unfortunately, such approaches are often bottlenecked by a few slow or unresponsive workers, called stragglers [4], that delay the entire computation as the master needs to wait for all workers to complete their assigned tasks.

In the past, the problem of stragglers has been addressed by replicating tasks at individual workers [5, 6, 7], and waiting for any one copy to finish. The observation that replication is a special case of the more general erasure coding framework wherein stragglers can be modeled as *erasures* has led to the employment of Maximum Distance Separable (MDS) codes [8, 9, 10, 11] to speed up the computation of matrix vector products in a distributed setting. In this framework codes are employed to add redundancy so that only a subset of nodes are required to complete the tasks assigned to them.

In this work we use rateless fountain codes [12, 13] for distributed multiplication of a $m \times n$ matrix $\mathbf{A}$ with a $n \times 1$ vector $\mathbf{x}$. The rateless coded matrix-vector multiplication algorithm generates coded linear combinations of the m rows of matrix $\mathbf{A}$ and distributes them across p worker nodes. Each node also receives a copy of $\mathbf{x}$. The master node needs to wait for any $m_d = m(1+\epsilon)$ row-vector products to be completed across *all* the nodes, where ϵ is a small overhead ($\epsilon \to 0$ as $m \to \infty$). Using rateless codes has the following key benefits.

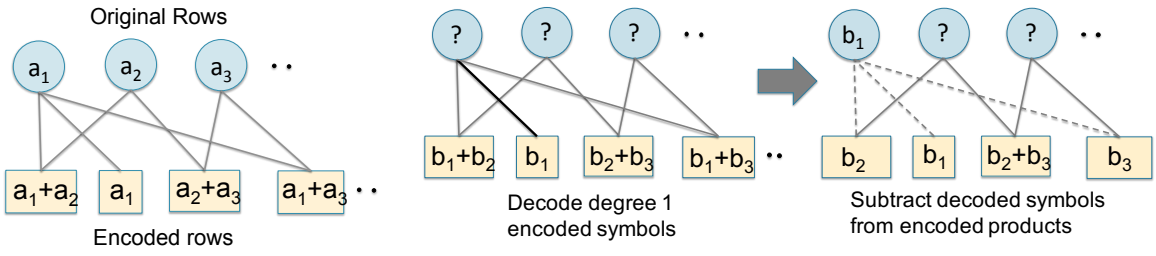
Low Latency via Seamless Load-Balancing. A key drawback of replication or fixed-rate coding strategies is that they rely on using the results of a fast subset of worker nodes, and completely ignore partial computations performed by slow or straggling nodes. On the other hand our rateless coding strategy achieves near *ideal* load balancing by utilizing the computations performed by *all* nodes while ensuring that faster nodes complete more tasks than slower nodes. This also provides robustness to node failures since available nodes can make up for computations lost due to failed nodes.

Negligible Redundant Computation. Rateless Coding performs $m_d = m(1+\epsilon)$ row-vector product computations on average, where, the overhead ϵ goes to zero as the number of matrix rows m increases. In the case of MDS coding or replication, if there is no straggling, the worker nodes collectively perform a large amount of redundant computation.

Low Decoding Complexity. Another benefit of using rateless codes is the extremely fast decoding of $\mathcal{O}(m \ln m)$ which allows our approach to scale efficiently even for very large m.

While the use of LT codes for matrix-vector multiplication has been recently proposed in [14, 15], these works do not utilize partial work done by stragglers, which is the key novel contribution of our work. Due to this our approach achieves the aforementioned benefits of near optimal straggler tolerance, negligible overhead of redundant computation, as well as robustness to node failures.

[*]Malhar Chaudhari performed the work while at CMU



(a) Encoding Graph (b) Decoding Graph

Fig. 1: (a) Bipartite graph representation of the encoding of the rows $\mathbf{a}_1, \mathbf{a}_2, \ldots \mathbf{a}_m$ of matrix $\mathbf{A}$. Each encoded row is the sum of d rows of $\mathbf{A}$ chosen uniformly at random, where d is drawn from the Robust Soliton degree distribution [12]. (b) In each step of the iterative decoding process, a single degree one encoded symbol is decoded directly, and is subtracted from all sums in which it participates.

2. PROBLEM FORMULATION

Consider the problem of multiplying a $m \times n$ matrix $\mathbf{A}$ with a $n \times 1$ vector $\mathbf{x}$ using p worker nodes and a master node. The worker nodes can only communicate with the master, and cannot directly communicate with other workers. The goal is to compute the result $\mathbf{b} = \mathbf{Ax}$ in a distributed fashion and mitigate the effect of unpredictable node slowdown or straggling. The rows of matrix $\mathbf{A}$ are encoded using an erasure code to give the $m_e \times n$ encoded matrix $\mathbf{A_e}$, where $m_e = \alpha m \geq m$. Matrix $\mathbf{A_e}$ is split along its rows to give p submatrices $\mathbf{A}_{\mathbf{e},1}, \ldots, \mathbf{A}_{\mathbf{e},p}$ of equal size such that worker i stores submatrix $\mathbf{A}_{\mathbf{e},i}$. To compute the matrix-vector product $\mathbf{b} = \mathbf{Ax}$, the vector $\mathbf{x}$ is communicated to the workers such that Worker i is tasked with computing the product $\mathbf{A}_{\mathbf{e},i}\mathbf{x}$.

To complete the assigned task, each worker needs to compute a sequence of row vector products of the form $\mathbf{a}_{\mathbf{e},j}\mathbf{x}$ where $\mathbf{a}_{\mathbf{e},j}$ is the j^{th} row of $\mathbf{A_e}$. The time taken by a worker node to finish computing one or more row-vector products may be random due to variability in the node speed or the amount of computation assigned to it. The master node aggregates the computations of all, or a subset of, the workers into the vector $\mathbf{b_e}$, which is then decoded to give the final result $\mathbf{b} = \mathbf{Ax}$. If $\mathbf{b_e}$ is not decodable, the master waits until more row-vector products are completed by the workers.

2.1. Performance Criteria

We use the following metrics to compare different distributed matrix-vector multiplication schemes via theoretical analysis and associated simulations (Section 4).

Definition 1 (Latency (T))**.** *The latency T is the time required by the system to complete enough number of computations so that $\mathbf{b} = \mathbf{Ax}$ can be successfully decoded from the worker computations aggregated in $\mathbf{b_e}$.*

Definition 2 (Computations (C))**.** *The number of computations C is defined as the total number of row-vector products $\mathbf{a}_{\mathbf{e},j}\mathbf{x}$ performed collectively by worker nodes until the vector $\mathbf{b} = \mathbf{Ax}$ is decoded.*

For any strategy, we have $C \geq m$ where m is the number of rows of $\mathbf{A}$ or the number of elements in $\mathbf{b}$.

2.2. Benchmarks for Comparison

We compare the performance of the proposed rateless coded strategy with two benchmarks: the replication, and the MDS-coded strategies.

The $r-$Replication Strategy. In this approach $\mathbf{A}$ is split along its rows into p/r submatrices $\mathbf{A}_1, \ldots, \mathbf{A}_{p/r}$, with rm/p rows each (assume that p/r divides m). Each submatrix is multiplied with $\mathbf{x}$ in parallel on r distinct worker nodes. The master collects the results from only the fastest of the r nodes that have been assigned the task of computing the product $\mathbf{A}_i\mathbf{x}$, for all i and discards the rest. The computed products are aggregated into the $m \times 1$ vector $\mathbf{b}$. Setting $r = 1$ corresponds to the naive or *uncoded* strategy where each submatrix-vector product is computed at a single worker node. Increasing the number of replicas provides greater straggler tolerance at the cost of more redundant computations. Systems like Mapreduce [16] and Spark [17] often use $r = 2$ i.e. each computation is assigned to 2 different worker nodes for added reliability and straggler tolerance.

The (p, k) MDS-Coded Strategy. The MDS-Coded appraoch for straggler mitigation [11, 18] involves pre-multiplying $\mathbf{A}$ at the master with a suitable encoding matrix $\mathbf{F}$ denoting the MDS code. Encoding $\mathbf{A}$ using a (p, k) MDS code involves splitting it along its rows into k matrices $\mathbf{A}_1, \ldots, \mathbf{A}_k$, each having m/k rows. The MDS code adds $p - k$ redundant matrices $\mathbf{A}_{k+1}, \ldots, \mathbf{A}_p$ which are independent linear combinations of the matrices $\mathbf{A}_1, \ldots, \mathbf{A}_k$. Worker i computes the product of matrix $\mathbf{A}_i$ with vector $\mathbf{x}$. The master can recover the product $\mathbf{b} = \mathbf{Ax}$ if any k workers complete their task. Thus the system is robust to $p - k$ stragglers. However this strategy adds a significant computation overhead. When none of the nodes are slow, the system performs mp/k row-vector products, whereas in the uncoded case it only performs m row-vector products.

3. PROPOSED RATELESS CODING STRATEGY

We now propose our rateless coding strategy which uses LT codes [12] to mitigate the effect of stragglers in computing the matrix-vector product $\mathbf{b} = \mathbf{A}\mathbf{x}$ in the master-worker framework described in Section 2.

Luby Transform (LT) codes proposed in [12] are a class of erasure codes that can be used to generate a limitless number of encoded symbols from a finite set of source symbols. We apply LT codes to matrix-vector multiplication by treating the m rows of the matrix $\mathbf{A}$ as source symbols. Each encoded symbol is the sum of d source symbols chosen uniformly at random from the matrix rows. Thus if $\mathcal{S}_d \subseteq \{1, 2, \dots m\}$ is the set of d row indices, the corresponding encoded row is $\mathbf{a_e} = \sum_{i \in \mathcal{S}_d} \mathbf{a}_i$. The number of original rows in each encoded row, or the degree d, is chosen according to the Robust Soliton degree distribution the details of which are described in Appendix A of [19] and in Luby's original work [12].

The $m \times n$ matrix $\mathbf{A}$ is encoded to generate an $m_e \times n$ encoded matrix $\mathbf{A_e}$ where $m_e = \alpha m$. Once the αm rows of the encoded matrix $\mathbf{A_e}$ are generated, they are distributed equally among the p worker nodes. To multiply $\mathbf{A}$ with a vector $\mathbf{x}$, the master sends $\mathbf{x}$ to the workers. Each worker multiplies $\mathbf{x}$ with each row of $\mathbf{A_e}$ stored in its memory and returns the product (a scalar) to the master. The master collects row-vector products of the form $\mathbf{a_{e},}_j\mathbf{x}$ (elements of $\mathbf{b_e}$) from workers until it has enough elements to be able to recover $\mathbf{b}$.

To decode the desired matrix vector product $\mathbf{b} = \mathbf{A}\mathbf{x}$ from a subset of M' symbols of $\mathbf{b_e}$ we use the the iterative peeling decoder described in [12, 13]. If $\mathbf{b} = [b_1, b_2, \dots b_m]$, the decoder may receive symbols $b_1 + b_2 + b_3$, $b_2 + b_4$, b_3, b_4, and so on. Decoding is performed in an iterative fashion. In each iteration, the decoder finds a degree one encoded symbol, covers the corresponding source symbol, and subtracts the symbol from all other encoded symbols connected to that source symbols (see Figure 1b). Once the master receives M' elements of $\mathbf{b_e}$ (M' is the number of symbols required for successful decoding), it sends a *done* signal to all workers nodes to stop their local computation.

Since the encoding uses a random bipartite graph, the number of symbols required to decode the m source symbols successfully is a random variable M'. For the Robust Soliton distribution, [12] gives a high probability bound on M'.

Lemma 1 (Theorems 12 and 17 in [12])**.** *The original set of m source symbols can be recovered from a set of any $M' = m + \mathcal{O}(\sqrt{m}\ln^2(m/\delta))$ with probability at least $1 - \delta$.*

In our analysis we denote $m_d = \mathbb{E}[M']$. Using the theoretical guarantees for LT codes (see Appendix A of [19]) we can show that $m_d = m(1 + \epsilon)$, where $\epsilon \to 0$ as $m \to \infty$. In practice one can choose a small enough value of δ and compute the corresponding value of M' such that decoding is susccessful with a high probability.

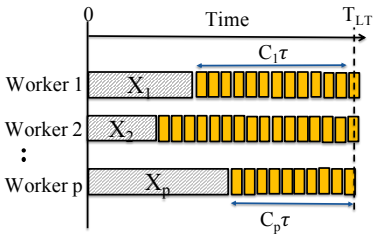


Fig. 2: Worker i has a random exponential initial delay X_i, after which it completes row-vector product tasks (denoted by the small rectangles), taking time τ per task. The latency T_{LT} is the time to complete M' tasks in total.

4. THEORETICAL ANALYSIS

Our main theoretical results involve comparing the proposed rateless coding strategy with the MDS and replication coding strategies in terms of latency and computations. The proofs of the theoretical results presented here are contained in Appendix B of the full version [19]. We assume that worker i performs B_i computations in time Y_i where

$$Y_i = X_i + \tau B_i, \quad \text{for all } i = 1, \dots, p \tag{1}$$

where X_i is a random variable that includes the network latency, initial setup time, and other random components, and τ is a constant shift which is the time taken by any worker to perform a single computation (row-vector multiplication). When X_i is exponentially distributed with rate μ, the time taken by worker i to perform b computations is distributed as $\Pr(Y_i \leq t) = 1 - \exp(-\mu(t - \tau b)$. While this follows the shifted exponential delay model used in [**?**, 9], the key difference is that the shift is parameterized by the number of computations at each worker. We believe this is a more realistic model as it captures the effect of increasing the amount of computations on the delay – if a worker is assigned more computations, there is larger delay. Figure 2 illustrates the latency of the LT coded strategy, T_{LT} under this delay model.

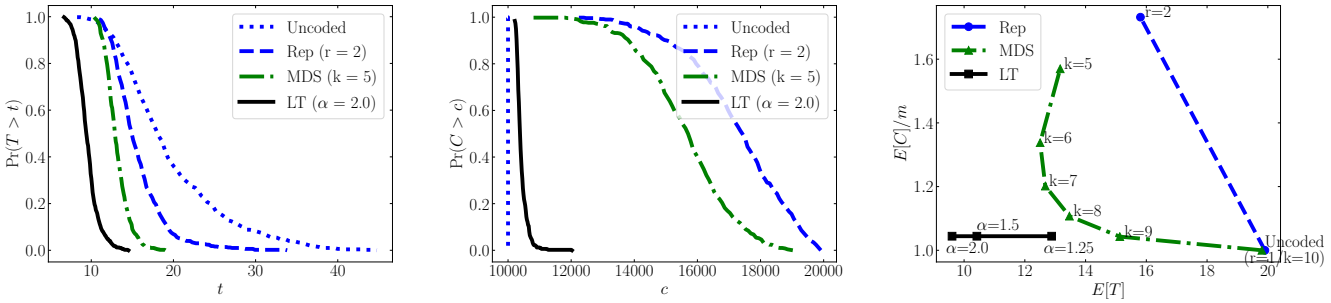
Theorem 1 (Latency of the Rateless Coded Strategy)**.** *For large m_e i.e. $\alpha = m_e/m \to \infty$, the expected latency for the LT-coded case with p workers and $X_i \sim \exp(\mu)$ for all workers $i = 1, \dots, p$, is bounded as.*

$$\mathbb{E}[T_{LT}] \leq \frac{\tau m_d}{p} + \frac{1}{\mu} + \tau, \tag{2}$$

$$\mathbb{E}[T_{LT}] \geq \frac{\tau m_d}{p} + \frac{1}{p\mu}, \tag{3}$$

where $m_d = m(1 + \epsilon)$ is the expected number of symbols necessary for successful decoding.

Remark 1. While we need $\alpha \to \infty$ for the above results to strictly hold, we observe empirically (Figure 3a) that LT codes exhibit superior latency performance over the benchmark strategies even for $\alpha = 2.0$.



(a) Latency Tail (b) Computation Tail (c) Avg. Computations v/s latency

Fig. 3: The tail probability of the latency is the highest for the replication schemes. MDS codes perform better in terms of latency but they perform a large number of redundant computations. The latency tail of LT codes is the minimum among all the schemes. Moreover the LT coded scheme performs significantly fewer redundant computations than MDS Codes or replication. All results were obtained from 500 Monte-Carlo Simulations with number of matrix rows, $m = 10000$, number of worker nodes $p = 10$ and delay model parameters $\mu = 0.2, \tau = 0.005$.

Theorem 2 (Latency of Replication and MDS Coding)**.** *The expected latency for the $r-$Replication and the (p, k) MDS-coded strategies with $X_i \sim \exp(\mu)$ for all workers $i = 1, \ldots, p$ is*

$$\mathbb{E}[T_{rep}] = \frac{\tau m r}{p} + \frac{1}{\mu} H_{p/r} \simeq \frac{\tau m r}{p} + \frac{1}{\mu} \log \frac{p}{r} \tag{4}$$

$$\mathbb{E}[T_{MDS}] = \frac{\tau m}{k} + \frac{1}{\mu}(H_p - H_{p-k}) \simeq \frac{\tau m}{k} + \frac{1}{\mu} \log \frac{p}{p-k}, \tag{5}$$

where $H_j = \sum_{v=1}^{j} 1/v$, the j^{th} Harmonic number.

Remark 2. Observe that in (4) and (5) above, adding redundancy (increasing r and reducing k respectively) leads to an increase in the first term (more computation at each node) and decrease in the second term (less delay due to stragglers). Thus, straggler mitigation comes at the cost of additional computation at the workers which might even lead to an increase in latency. This is seen in Figure 3c where the latency actually increases on adding redundancy for the MDS-coded case.

Remark 3. One of the main advantages of the rateless coded strategy is that the number of computations performed by all the workers asymptotically (as $m \to \infty$) approaches the minimum number of computations (m) required to recover a $m-$dimensional matrix-vector product (lemma 1). On the other hand, the following theorems show that the total computations performed by *all* workers (fast and slow) in the replication and MDS-coded schemes is much larger than m.

Theorem 3 (Tail of Computations for MDS Coding)**.** *The tail of the number of computations of the MDS coded strategy, C_{MDS}, with p workers and $X_i \sim \exp(\mu)$ is bounded as*

$$\Pr(C_{MDS} \geq \frac{mp}{k} - C_0) \geq 1 - \exp\left(-\mu\theta_{MDS}\right) \tag{6}$$

$$\theta_{MDS} = \frac{\tau C_0}{(p-k)^2} - \frac{\tau}{p-k} \tag{7}$$

Theorem 4 (Tail of Computations for Replication)**.** *The tail of the number of computations of the replication strategy, C_{rep}, with p workers and $X_i \sim \exp(\mu)$ is bounded as*

$$\Pr(C_{rep} \geq mr - C_0) \geq 1 - \sum_{i=0}^{p/r-1} \frac{1}{i!} \exp(-\mu\theta_{Rep})(\mu\theta_{Rep})^i \tag{8}$$

$$\theta_{rep} = \frac{\tau C_0}{(r-1)^2} - \frac{\tau p}{r(r-1)} \tag{9}$$

Remark 4. While the benefits of using partial work from all workers can be obtained by using any random linear code on the rows of $\mathbf{A}$, the key strength of LT codes is their low decoding complexity of $\mathcal{O}(m \ln m)$. Using an (m_e, m) MDS code over the rows of $\mathbf{A}$ then the decoding complexity would be $\mathcal{O}(m^3)$ which is unacceptable for large m.

5. CONCLUDING REMARKS

Due to the massive size of matrices arising in modern data-driven applications, computations such as matrix-vector multiplication need to be parallelized across multiple nodes. In this paper we propose an erasure coding strategy based on *rateless fountain codes* to overcome bottlenecks caused by slow or straggling nodes. For a matrix with m rows, our strategy requires the nodes to collectively finish slightly more than m row-vector products, and thus can seamlessly adapt to varying node speeds and achieve near-perfect load balancing. Moreover, it has a small overhead of redundant computations (asymptotically zero), and low decoding complexity. Thus, it strikes a better latency-computation trade-off than the uncoded and fixed-rate erasure coding strategies. The ideas of this paper can potentially be extended to any random linear network code including other rateless codes such as Raptor Codes [20] and other low-complexity random linear codes [21].

6. REFERENCES

[1] William F Ames, *Numerical Methods for Partial Differential Equations*, Academic Press, 2014.

[2] William Dally, "High-performance hardware for machine learning," *NIPS Tutorial*, 2015.

[3] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd, "The pagerank citation ranking: Bringing order to the web.," Tech. Rep., Stanford InfoLab, 1999.

[4] Jeffrey Dean and Luiz André Barroso, "The tail at scale," *Communications of the ACM*, vol. 56, no. 2, pp. 74–80, 2013.

[5] Ganesh Ananthanarayanan, Srikanth Kandula, Albert G Greenberg, Ion Stoica, Yi Lu, Bikas Saha, and Edward Harris, "Reining in the outliers in map-reduce clusters using mantri.," in *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2010, vol. 10, p. 24.

[6] Ganesh Ananthanarayanan, Ali Ghodsi, Scott Shenker, and Ion Stoica, "Effective straggler mitigation: Attack of the clones.," in *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2013, vol. 13, pp. 185–198.

[7] Da Wang, Gauri Joshi, and Gregory Wornell, "Using straggler replication to reduce latency in large-scale parallel computing," *ACM SIGMETRICS Performance Evaluation Review*, vol. 43, no. 3, pp. 7–11, 2015.

[8] Gauri Joshi, Yanpei Liu, and Emina Soljanin, "On the delay-storage trade-off in content download from coded distributed storage systems," *IEEE Journal on Selected Areas of Communications*, vol. 32, no. 5, pp. 989–997, May 2014.

[9] Sanghamitra Dutta, Viveck Cadambe, and Pulkit Grover, "Short-dot: Computing large linear transforms distributedly using coded short dot products," in *Advances In Neural Information Processing Systems*, 2016, pp. 2100–2108.

[10] Gauri Joshi, Emina Soljanin, and Gregory Wornell, "Efficient redundancy techniques for latency reduction in cloud systems," *ACM Transactions on Modeling and Performance Evaluation of Computing Systems*, vol. 2, no. 12, may 2017.

[11] Kangwook Lee, Maximilian Lam, Ramtin Pedarsani, Dimitris Papailiopoulos, and Kannan Ramchandran, "Speeding up distributed machine learning using codes," *IEEE Transactions on Information Theory*, 2017.

[12] Michael Luby, "LT Codes," in *Foundations of Computer Science, 2002. Proceedings. The 43rd Annual IEEE Symposium on*. IEEE, 2002, pp. 271–280.

[13] Amin Shokrollahi, "Raptor codes," *IEEE transactions on information theory*, vol. 52, no. 6, pp. 2551–2567, 2006.

[14] Albin Severinson, Alexandre Graell i Amat, and Eirik Rosnes, "Block-diagonal and lt codes for distributed computing with straggling servers," *arXiv preprint arXiv:1712.08230*, dec 2017.

[15] Sinong Wang, Jiashang Liu, and Ness Shroff, "Coded sparse matrix multiplication," *arXiv preprint arXiv:1802.03430*, 2018.

[16] Jeffrey Dean and Sanjay Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[17] Matei Zaharia, Mosharaf Chowdhury, Michael J Franklin, Scott Shenker, and Ion Stoica, "Spark: Cluster computing with working sets.," *HotCloud*, vol. 10, no. 10-10, pp. 95, 2010.

[18] Songze Li, Mohammad Ali Maddah-Ali, and A Salman Avestimehr, "A unified coding framework for distributed computing with straggling servers," in *IEEE Global Communications Conference (GLOBECOM) Workshops*. IEEE, 2016, pp. 1–6.

[19] Ankur Mallick, Malhar Chaudhari, and Gauri Joshi, "Rateless codes for near-perfect load balancing in distributed matrix-vector multiplication," *arXiv preprint arXiv:1804.10331*, 2018.

[20] Amin Shokrollahi, Michael Luby, et al., "Raptor codes," *Foundations and trends in communications and information theory*, vol. 6, no. 3–4, pp. 213–322, 2011.

[21] Gauri Joshi and Emina Soljanin, "Round-robin overlapping generations coding for fast content download," in *IEEE International Symposium on Information Theory (ISIT)*, July 2013, pp. 2740–2744.