Sequential addition of coded sub-tasks for straggler mitigation

Ajay Badita
Indian Institute of Science
Bengaluru, KA 560012, India
ajaybadita@iisc.ac.in

Parimal Parag
Indian Institute of Science
Bengaluru, KA 560012, India
parimal@iisc.ac.in

Vaneet Aggarwal
Purdue University
West Lafayette, IN 47907, USA
vaneet@purdue.edu

***Abstract*—Straggler mitigation can be achieved by redundant computation. In MDS redundancy method, a task is divided into k sub-tasks which are encoded to n coded sub-tasks, such that a task is completed if any k coded sub-tasks are completed. Two important metrics of interest are task completion time, and server utilization cost which is the aggregate completed work by all servers in this duration. We consider a proactive straggler mitigation strategy where n_0 out of n coded sub-tasks are started at time 0 while the remaining $n-n_0$ coded sub-tasks are launched when $\ell_0 \leqslant \min(n_0, k)$ of the initial ones finish. The coded sub-tasks are halted when k of them finish. For this flexible forking strategy with multiple parameters, we analyze the mean of two performance metrics for the proposed forking strategy when the random service completion time at each server is independent and distributed identically to a shifted exponential. Our analysis demonstrates that the regime of $n_0 < k$ leads to higher mean service completion time and no change in mean server utilization cost as compared to no forking ($n_0 = n$), and is thus not a regime of interest. For $n_0 \geqslant k$, we find that there is a tradeoff between the two performance metrics and leads to decrease in mean server utilization cost at the expense of mean service completion time and an efficient choice of the parameters is helpful.**

***Index Terms*—Straggler mitigation, distributed computing, completion time, scheduling, forking points.**

I. Introduction

Distributed computing uses multiple distributed servers to process the job. With the use of erasure coding, a job can be divided into $n$coded sub-tasks, such that the job is complete when any k of them are finished. Erasure coding is a more general form of redundancy than simple replication, where $k = 1$. Such flexibility of finishing any k out of n coded sub-tasks has been shown in the prior works on distributed computing [1], [2]. The slowest tasks that determine the job execution time are called stragglers.

Assuming that each server is working on a unique coded sub-task, we consider the following important question. When should the n coded sub-tasks be started? One option is to start all coded sub-tasks at time 0, corresponding to the job request time. This leads to using all n servers until the first k of them have finished, resulting in low service completion time at the cost of a high server utilization. On the other hand, we may start k servers at time 0. This would help avoid the excess server utilization cost for the remaining $n-k$ servers, while it is *unclear* how the server utilization cost or the service completion time is affected. A more flexible approach is to start with $n_0 < n$ coded sub-tasks at time 0. When $\ell_0 < k$ of them are finished, we launch the remaining $n_1 = n - n_0$ servers and this launching point is called *forking point*. In the first example of starting all n servers, we have $n_0 = n$, $\ell_0 = k$, $n_1 = 0$. In the second example of starting with k, we have $n_0 = \ell_0 = k$. Thus, the proposed approach affords a flexible framework for launching the coded sub-tasks. It is not *apriori* clear as to how should these parameters be chosen so that both the metrics are optimized. This paper aims to find the affect of n_0, n_1, and ℓ_0 on the two metrics - the service completion time, and the server utilization cost.

A. Related Work

Given the unpredictable nature of the nodes in distributed computing systems, coding theoretic techniques have been used to achieve high-quality algorithmic results in the face of uncertainty. Coding-theoretic approaches have been shown to provide a tradeoff between latency and cost in distributed storage systems [4]. It was shown in [5] that MDS codes are the latency-minimizing code among a class of symmetric codes for distributed systems. Coding theoretic techniques have been provided for mitigating stragglers in matrix multiplication [6]–[8]. The authors of [1], [9] consider the problem of computing gradients in a distributed system, and propose a novel coded computation scheme tailored for computing a sum of functions. While most of the works focus on the application of coded computation to linear operations, coding has also been found useful in distributed computing frameworks involving nonlinear operations [10]. Efficient coding theoretic techniques to reduce the communication cost in the process of transferring the results of mappers to reducers have been studied in [6], [11]–[14]. These works demonstrate the improvement of service completion times with the use of coding. However, this line of work does not take the server utilization cost into account. We will consider this cost to determine efficient launching times of the different coded sub-tasks.

This work is supported in part by the Science and Engineering Research Board under Grant DSTO-1677, the Department of Telecommunications, Government of India, under Grant DOTC-0001, the VAJRA fellowship, the National Science Foundation under Grant CNS-1618335, and CISCO. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies.

One of the key approaches to mitigate the effect of stragglers is to either re-launch a certain task if it is delayed, or preemptively assign each task to multiple nodes and move on with the copy that completes first. Speculative execution have been studied in [15], which acts after the tasks have already slowed down. In a proactive mitigation approach, one can launch redundant copies of a task hoping that at least one of them will finish in a timely manner. The authors of [16] perform cloning to mitigate the effect of stragglers. The authors of [17] analyzed the latency and cost for replication-based strategies for straggler mitigation. A machine learning approach for predicting and avoiding these stragglers has been studied in [18]. Recently, coding-theory-inspired approaches have been applied to mitigate the effect of straggling as mentioned earlier. Single fork analysis with coding has been studied in [2], where k coded sub-tasks are started at $t = 0$. Further, after a fixed deterministic time Δ, additional $n - k$ coded sub-tasks are started. Our work differs from [2] since

(i) we allow for general number of starting coded sub-tasks,
(ii) the start time of new coded sub-tasks is random and based on the completion time of certain number of coded sub-tasks rather than a fixed constant, and
(iii) our framework allows for an optimization of different parameters to provide a tradeoff between service utilization cost and service completion time.

B. Contributions

We characterize the means of the service completion time and the server utilization cost, for a single (n, k)-MDS coded job with single forking. The MDS coding implies that the job is fragmented into k sub-tasks and encoded into n coded sub-tasks, where completion of any k coded sub-tasks finishes the job. The single forking implies that the job is started with n_0 coded sub-tasks at the job request time, and another n_1 coded sub-tasks are started on completion of $\ell_0 < k$ out of initial n_0 coded sub-tasks. Further, the random execution time of each coded sub-task is assumed to be independent and identically distributed (*i.i.d.*) as a shifted exponential distribution, which is shown as a decent approximation of task completion times on compute clusters [2], [19].

We compute the two performance metrics for the choice of system parameters n_0, n_1, and ℓ_0, and demonstrate the quantitative tradeoff between these two metrics. For comparison, we consider the no forking case, when $n_0 = n$. We find there is no advantage to choose $n_0 < k$ for either of the metrics as compared to no forking case. This is because the service utilization cost does not change with the value of n_0 when $n_0 < k$ while the service completion time increases as n_0 decreases. Thus, one should not perform forking with $n_0 < k$, and thus the only regime of interest is $n_0 \geqslant k$.

In this regime $n_0 \geqslant k$, we make the following observations. Keeping parameters ℓ_0 and n fixed, we observe that the server utilization cost is not monotone in the initial number of tasks n_0, whereas the mean service completion time decreases with n_0 as expected. For a fixed n_0 and n, increasing the fork task threshold ℓ_0 increases the service completion time while decreases the server utilization cost. Thus, there is a tradeoff in the two metrics and efficient choice of parameters can be decided by the system designer based on the weighted combination of the two metrics.

II. System Model

In this section, we describe the different components of the system model in detail. We consider a distributed compute system with n identical servers.

A. Coding Model

We assume that each compute job can be divided into k sub-tasks, which are encoded into n coded sub-tasks and sent to n distinct servers. We assume the jobs to be MDS coded, which implies that the coded sub-task completion at any k out of these n distinct servers results in the completion of the original job.

B. Single-Fork Scheduling

We assume a single-fork scheduling, where a job starts at n_0 parallel servers at time $t_0 = 0$, and adds $n_1 = n - n_0$ servers at a random time instant t_1 corresponding to service completion time of the ℓ_0th coded sub-task out of n_0 initial servers. The total service completion time is given by t_2 when the remaining coded sub-tasks at $\ell_1 = k - \ell_0$ servers are completed. Since we can't have more service completions than the number of servers in service, we have $\ell_0 \leqslant n_0$ and $\ell_0 + \ell_1 = k \leqslant n$.

C. Service Model

We assume the cost of server utilization to be λ per unit time. Each server $i \in [n] \triangleq \{1, \dots, n\}$ has an independent and identically distributed (*i.i.d.*) random service time T_i with distribution function F for each scheduled coded sub-task on this server. Recent works [4], [19]–[21] suggest that a shifted exponential distribution is a good fit for modeling the service time distribution in distributed computation networks. It is suggested that the service time for each computation of coded sub-task can be modeled by two aggregate components: a constant server start-time and a random memoryless component. These studies along with the goal of analytical tractability influenced us to assume the service time distribution for each coded sub-task to be a shifted exponential with rate μ and shift c, such that the complementary distribution function $\bar{F} = 1 - F$ can be written

$$\bar{F}(x) \triangleq P\{T_1 > x\} = \mathbb{1}_{\{x \in [0,c]\}} + e^{-\mu(x-c)} \mathbb{1}_{\{x \geqslant c\}}. \quad (1)$$

We see that $T_i' \triangleq T_i - c$ are *i.i.d.* random variables distributed exponentially with rate μ. We denote the jth order statistic of $(T_1', \dots, T_n')$ by X_j^n.

Remark 1. The jth order statistic of $(T_1, \dots, T_n)$ is $c + X_j^n$.

Remark 2. Let $(X_1, \dots, X_n)$ be n *i.i.d.* random variables with common distribution function F, and we denote the jth order statistics of this collection by X_j^n. Then the distribution of X_j^n is given by $P\left\{X_j^n \leqslant x\right\} = \sum_{i=j}^{n} \binom{n}{i} F(x)^i \bar{F}(x)^{n-i}$.

Remark 3. Denoting $X_0^n = 0$, from the memoryless property of T_i', we observe the following equality in joint distribution of two vectors

$$(X_j^n - X_{j-1}^n : j \in [n]) = \left(\frac{T_j'}{n-j+1} : j \in [n]\right).$$

D. *Performance Metrics*

The service completion time for k coded sub-tasks is denoted by t_2 and the server utilization cost by W. We denote the service completion time of rth coded sub-task in ith stage $[t_i, t_{i+1})$ by $t_{i,r}$ where $i \in \{0, 1\}$. Since each stage consists of ℓ_i service completions, we have $r \in \{0, \dots, \ell_i\}$ such that $t_{i,0} = t_i$ and $t_{i,\ell_i} = t_{i+1,0} = t_{i+1}$.

Assuming that a server is discarded after its coded sub-task completion, we can write the utilization cost in this case as the time-integral of number of servers that are ON during the service completion $[0, t_2]$, multiplied by the server utilization cost per unit time

$$W = \lambda \sum_{i=0}^{1} \Big[\sum_{r=0}^{\ell_i - 1} (t_{i,r+1} - t_{i,r}) \Big(\sum_{j=0}^{i} (n_j - \ell_j) + \ell_i - r \Big) \Big]. \quad (2)$$

The total service completion time $S = t_2$ can be written as the following telescopic sum

$$S = \sum_{i=0}^{1} \left[\sum_{r=1}^{\ell_i} (t_{i,r} - t_{i,r-1}) \right]. \quad (3)$$

We are interested in the optimal trade-off between the mean service completion time $\mathbb{E}[t_2]$ and the mean server utilization cost $\mathbb{E}[W]$ for k coded sub-tasks scheduled in two stages over these n servers. To this end, we will first analytically compute the mean service completion time and the mean server utilization cost.

We note that the problem is important even when there are stochastic arrivals since this procedure of forking can be used for any arriving job. The exact queueing analysis for coded-jobs with forking is not straightforward to extend and remains open, while the analysis in this paper provide insights on how to efficiently fork a job in lightly-loaded scenarios. This scenario arises in the case of low arrival rates so that the queues are empty with high probability, and hence the system can be modeled as an $M/G/1$ queue where the service time of a job is the completion time computed in this work. Thus, one can achieve a tradeoff between the two performance metrics for lightly loaded queueing systems.

III. Analysis of the Performance Metrics

Recall that we have two contiguous stages. The time interval $[t_0, t_1)$ corresponds to the stage 0, and the interval $[t_1, t_2]$ corresponds to the stage 1. In stage 0, we switch on n_0 initial servers at instant $t_0 = 0$. This stage is completed at the single forking point denoted by the instant t_1, when ℓ_0 coded sub-tasks out of n_0 are completed. At the beginning of stage 1, additional $n_1 = n - n_0$ servers are switched on, each working on a unique coded sub-task. The job is completed at the end of this second stage, when remaining $k - \ell_0$ coded sub-tasks are completed. The kth service completion time is denoted by t_2. The server utilization cost can be written as sum of the server utilization cost in each of the two stages as

$$W = W_0 + W_1.$$

We will separately analyze these two stages in the following subsections.

A. *Stage 0 Analysis*

To compute mean duration of the stage 0, and the mean server utilization in this stage, we need to compute the mean of the interval $[t_{0,r-1}, t_{0,r})$ for each $r \in [\ell_0]$.

Lemma 4. *The mean time between two coded sub-task completions in the single forking scheme for* i.i.d. *shifted exponential coded sub-task completion times in stage* 0 *is*

$$\mathbb{E}[t_{0,r} - t_{0,r-1}] = \begin{cases} c + \frac{1}{\mu n_0}, & r = 1, \\ \frac{1}{\mu(n_0 - r + 1)}, & r \in \{2, \dots, \ell_0\}. \end{cases} \quad (4)$$

Proof: Since $t_{0,r}$ is the completion time of first r coded sub-tasks out of n_0 parallel coded sub-tasks, we have $t_{0,r} = c + X_r^{n_0}$ from Remark 1. Hence, for each $r \in [\ell_0]$, we have

$$t_{0,r} - t_{0,r-1} = (c + X_r^{n_0}) - (c + X_{r-1}^{n_0}).$$

The coded sub-tasks are initiated at time $t_{0,0} = t_0 = 0$ and hence the first coded sub-task is completed at $t_{0,1} - t_{0,0} = c + X_1^{n_0}$.

From Remark 3, we can write the following equality in distribution

$$t_{0,r} - t_{0,r-1} = \begin{cases} c + \frac{T_1'}{n_0}, & r = 1, \\ \frac{T_r'}{(n_0 - r + 1)}, & r \in \{2, \dots, \ell_0\}, \end{cases}$$

where $(T_1', \dots, T_n')$ are *i.i.d.* exponentially distributed random variables with rate μ. Taking expectations on both sides, we get the result. ∎

Corollary 5. *Consider single-forking with* i.i.d. *shifted exponential coded sub-task completion times and initial number of servers n_0 in stage* 0*. The mean forking time is given by*

$$\mathbb{E}[t_1] = c + \sum_{r=1}^{\ell_0} \frac{1}{\mu(n_0 - r + 1)}. \quad (5)$$

The mean server utilization cost in stage 0 *is given by*

$$\mathbb{E}[W_0] = \frac{\lambda}{\mu}(\ell_0 + \mu n_0 c). \quad (6)$$

Proof: We can write the completion time t_1 of ℓ_0th coded sub-task out of n_0 in parallel, as a telescopic sum of length of coded sub-task completions given in (3). Taking expectations on both sides, the mean forking point $\mathbb{E}[t_1]$ follows from the the linearity of expectations and the mean length of each coded sub-task completion (4).

Taking expectation of the server utilization cost in (2), the mean server utilization cost $\mathbb{E}[W_0]$ in stage 0 follows from the linearity of expectations and the mean length of each coded sub-task completion (4). ∎

B. *Stage* 1 *Analysis*

To compute the mean duration of the stage 1, and the mean server utilization in this stage, we need to compute the mean of the interval $[t_{1,r-1}, t_{1,r})$ for each $r \in [\ell_1]$. The difficulty in this computation is that additional n_1 servers that start working on coded sub-tasks at the single forking-time t_1, have an initial start-up time of c due to the shifted exponential service distribution. Hence, none of these additional n_1 servers can complete service before time $t_1 + c$. Whereas, some of the $n_0 - \ell_0$ servers with unfinished coded sub-tasks from stage 0 can finish their coded sub-task in this time-interval $(t_1, t_1 + c]$. In general, the number of coded sub-task completions in the interval $(t_1, t_1 + c]$ is a random variable, which we denote by $N(t_1, t_1 + c) \in \{0, \dots, n_0 - \ell_0\}$.

To be able to compute the mean length of the stage 1, and compute the mean server utilization in this stage, we first need to compute the probability mass function of this discrete valued random variable $N(t_1, t_1 + c)$. We denote the event of $j - \ell_0$ coded sub-task completions in this interval $(t_1, t_1 + c]$ for any $\ell_0 \leqslant j \leqslant n_0$ by

$$E_{j-\ell_0} \triangleq \left\{N(t_1, t_1 + c) = j - \ell_0, t_1 = c + X^{n_0}_{\ell_0}\right\}.$$

Lemma 6. *The probability distribution of the number of coded sub-task completions $N(t_1, t_1 + c)$ in the interval $(t_1, t_1 + c]$ for $\ell_0 \leqslant j \leqslant n_0$ is given by $p_{j-\ell_0} \triangleq P(E_{j-\ell_0})$ where*

$$p_{j-\ell_0} = \binom{n_0 - \ell_0}{j - \ell_0}(1 - e^{-\mu c})^{j-\ell_0} e^{-(n_0-j)\mu c}. \qquad (7)$$

Proof: Let the number of service completions until time $t_1 + c$ be $j \in \{\ell_0, \dots, n_0\}$. From Remark 1, we can write the event of $j - \ell_0$ service completions in the interval $(t_1, t_1 + c]$ as

$$\left\{X^{n_0}_j - X^{n_0}_{\ell_0} \leqslant c\right\} \cap \left\{X^{n_0}_{j+1} - X^{n_0}_{\ell_0} \leqslant c\right\}^c.$$

From the definition of order statistics for continuous random variables, we have $X^{n_0}_j < X^{n_0}_{j+1}$. This implies that the intersection of events $\left\{X^{n_0}_j - X^{n_0}_{\ell_0} \leqslant c\right\}$ and $\left\{X^{n_0}_{j+1} - X^{n_0}_{\ell_0} \leqslant c\right\}$ is $\left\{X^{n_0}_{j+1} - X^{n_0}_{\ell_0} \leqslant c\right\}$.

Therefore, from the disjointness of complementary events and probability axiom for summation of disjoint events, it follows

$$p_{j-\ell_0} = P\left\{X^{n_0}_j - X^{n_0}_{\ell_0} \leqslant c\right\} - P\left\{X^{n_0}_{j+1} - X^{n_0}_{\ell_0} \leqslant c\right\}.$$

From Remark 3, we can write the above as

$$p_{j-\ell_0} = P\left\{X^{n_0-\ell_0}_{j-\ell_0} \leqslant c\right\} - P\left\{X^{n_0-\ell_0}_{j+1-\ell_0} \leqslant c\right\}.$$

From Remark 2 for exponentially distributed random variables with rate μ, we get the required form of $p_{j-\ell_0}$. ■

We next provide a definition that will be used in the analysis.

Definition 7. We denote the Pochhammer function $(a)_n \triangleq \frac{\Gamma(a+n)}{\Gamma(a)}$ to define the z-transform of hypergeometric series as

$${}_pF_q(z) \triangleq {}_pF_q\left[\begin{matrix} a_1, \dots, a_p \\ b_1, \dots, b_q \end{matrix}; z\right] = \sum_{n=0}^{\infty} \frac{\prod_{i=1}^{p}(a_i)_n z^n}{\prod_{j=1}^{q}(b_j)_n (n)!}. \qquad (8)$$

Because generalizations of the above series also exist [22], this series is referred to here as the hypergeometric series rather than as the generalized hypergeometric series.

Recall that in the duration $(t_1, t_1 + c)$, there are $n_0 - \ell_0$ parallel servers in their memoryless phase, since the additional n_1 servers started at time t_1 are still in their start-up phase. In this duration, the event of $j - \ell_0 \in \{0, 1, \dots, n_0 - \ell_0\}$ coded sub-task completions is $E_{j-\ell_0}$.

Let $\{s_1, s_2, \dots, s_{n_0-\ell_0}\}$ be the coded sub-task completion times in stage 1 after the forking time t_1, which in definition correspond to $\{t_{1,1} = t_1 + s_1, t_{1,2} = t_1 + s_2, \dots, t_{1,n_0-\ell_0} = t_1 + s_{n_0-\ell_0}\}$. In stage 1, the coded sub-task completions numbered $r \in [j - \ell_0]$ are finished only by the $n_0 - \ell_0$ servers within time $t_1 + c$, since none of the n_1 servers started at forking point t_1 are able to finish even a single coded sub-task with in the time $t_1 + c$, whereas the coded sub-task completions numbered $r \in \{j - \ell_0 + 1, \dots, k - \ell_0\}$ are finished by $n - j$ servers which include subset of combination of both left over initial servers and all forked servers.

We next find mean of rth completion time in stage 1 conditioned on the event $E_{j-\ell_0}$.

Lemma 8. *For any $r \in [j - \ell_0]$ and $\alpha = 1 - e^{-c\mu}$, we have*

$$\mathbb{E}\left[s_r | E_{j-\ell_0}\right] = \begin{cases} {}_3F_2\left(\begin{smallmatrix} 1,1,r+1 \\ 2,j-\ell_0+2 \end{smallmatrix}; \alpha\right) \frac{r\alpha}{\mu(j-\ell_0+1)}, & r < j - \ell_0, \\ c\Big[1 - \alpha^{-r} + \sum_{i=1}^{r} \frac{\alpha^{i-r}}{ic\mu}\Big], & r = j - \ell_0. \end{cases}$$

Proof: The detailed proof is provided in Appendix B. ■

In stage 1, for $1 \leqslant r \leqslant j - \ell_0$, we have

$$t_{1,r} - t_{1,r-1} = (X^{n_0-\ell_0}_r - X^{n_0-\ell_0}_{r-1})\mathbb{1}_{E_{j-\ell_0}}.$$

For $j - \ell_0 + 2 \leqslant r \leqslant k - \ell_0$, the difference $t_{1,r} - t_{1,r-1}$ is equal to

$$(X^{n-j}_{r-j+\ell_0} - X^{n-j}_{r-j+\ell_0-1})\mathbb{1}_{E_{j-\ell_0}}. \qquad (9)$$

When $r = j - \ell_0 + 1$, we write the time difference between rth and $(r-1)$th coded sub-task completion instants as

$$t_{1,r} - t_{1,r-1} = t_{1,r} - (t_1 + c) + (t_1 + c) - t_{1,r-1}.$$

For $r = j - \ell_0 + 1$, we have $t_{1,r-1} \leqslant t_1 + c < t_{1,r}$. In the disjoint intervals $[t_{1,r-1}, t_1 + c)$ and $[t_1 + c, t_{1,r})$, there are $n_0 - j$ and $n - j$ *i.i.d.* exponentially distributed parallel servers respectively. Since the age and excess service times of exponential random variables are independent at any constant time, we have independence of $t_{1,r} - (t_1 + c)$ and $(t_1 + c) - t_{1,r-1}$ for $r = j - \ell_0 + 1$.

Conditioned on the event $E_{j-\ell_0}$ of $j - \ell_0$ coded sub-task completions in the interval $(t_1, t_1 + c]$, the conditional mean of inter-coded sub-task completion time in stage 1 is

$$\mathbb{E}\left[(t_{1,r} - t_{1,r-1})|E_{j-\ell_0}\right] = \mathbb{E}[(s_r - s_{r-1})(\mathbb{1}_{\{j-\ell_0>r-1\}} + \mathbb{1}_{\{j-\ell_0=r-1\}} + \mathbb{1}_{\{j-\ell_0<r-1\}})|E_{j-\ell_0}].$$

Lemma 9. *For any $r \in [k-\ell_0]$, and $\alpha = 1 - e^{-c\mu}$ the conditional mean $\mathbb{E}\left[(t_{1,r} - t_{1,r-1}) | E_{j-\ell_0}\right]$ equals*

$$\begin{cases} {}_2F_1\Big(\begin{smallmatrix}1,r\\ j-\ell_0+2\end{smallmatrix};\alpha\Big)\frac{r\alpha}{\mu(j-\ell_0+1)}, & r < j-\ell_0+1,\\ c\Big[\frac{1}{\alpha^{(r-1)}} - \sum\limits_{i=1}^{r-1}\frac{\alpha^{i-r+1}}{ic\mu}\Big] + \frac{1}{\mu(n-j)}, & r = j-\ell_0+1,\\ \frac{1}{\mu(n-\ell_0-r+1)}, & r > j-\ell_0+1. \end{cases} \quad (10)$$

Proof: Recall that, we have $n_0-\ell_0$ parallel servers in their memoryless phase working on individual coded sub-tasks in the interval $(t_1, t_1+c]$. In this duration, $N(t_1, t_1+c)$ coded sub-tasks are completed and additional n_1 parallel servers start their memoryless phase at time t_1+c.

We first consider the case when $r-1 > N(t_1, t_1+c) = j-\ell_0$. This implies that $t_{1,r-1} > t_1+c$ and there are $n-\ell_0-r+1$ parallel servers in their memoryless phase working on remaining coded sub-tasks. From Remark 3, the following equality holds in distribution

$$t_{1,r} - t_{1,r-1} = \frac{T'_r}{n-\ell_0-r+1}.$$

Recall that $E_{j-\ell_0} \in \sigma(T'_1,\dots,T'_{j-\ell_0+1})$, and since $(T'_i : i \in \mathbb{N})$ is an *i.i.d.* sequence, it follows that $t_{1,r}-t_{1,r-1}$ is independent of the event $E_{j-\ell_0}$ for $r > j-\ell_0+1$ and hence $\mathbb{E}\left[t_{1,r}-t_{1,r-1}|E_{j-\ell_0}\right] = \mathbb{E}\left[t_{1,r}-t_{1,r-1}\right]$. The result follows from the fact that $\mathbb{E}\left[T'_i\right] = \frac{1}{\mu}$.

We next consider the case when $r-1 = N(t_1, t_1+c) = j-\ell_0$. By definition of $N(t_1, t_1+c)$, we have $t_{1,r-1} \leqslant t_1+c < t_{1,r}$. In the disjoint intervals $(t_{1,r-1}, t_1+c]$ and $(t_1+c, t_{1,r}]$, there are n_0-j and $n-j$ *i.i.d.* exponentially distributed parallel servers respectively. Therefore, writing $t_{1,r}-t_{1,r-1}$ as $(t_{1,r}-(t_1+c))+((t_1+c)-t_{1,r-1})$, and using Remark 3, we compute the conditional mean of the first part as

$$\mathbb{E}\left[t_{1,r}-(t_1+c) \;\middle|\; E_{j-\ell_0}\right] = \mathbb{E}\left[\frac{T'_{r+1}}{n-j}\right] = \frac{1}{\mu(n-j)}.$$

By using the fact $t_{1,r-1} = t_1 + s_{r-1}$, we can write the conditional mean of the second part as $\mathbb{E}\left[t_1+c-t_{1,r-1} \;\middle|\; E_{j-\ell_0}\right] = c - \mathbb{E}\left[s_{r-1} \;\middle|\; E_{j-\ell_0}\right]$, where $\mathbb{E}\left[s_{r-1} \;\middle|\; E_{j-\ell_0}\right]$ is given by Lemma 8. Summing these two parts, we get the conditional expectation for $r = j-\ell_0+1$.

For the case when $r \in [j-\ell_0]$, the result follows from Lemma 8 and the fact $t_{1,r} = t_1 + s_r$. ■

We next compute the unconditional mean of inter-coded sub-task completion time $\mathbb{E}\left[(t_{1,r}-t_{1,r-1})\right]$ by averaging out the conditional mean $\mathbb{E}\left[(t_{1,r}-t_{1,r-1})|E_{j-\ell_0}\right]$ over all possible values of j. We denote $m = j-\ell_0$ for convenience.

Corollary 10. *For each $r \in [k-\ell_0]$, by considering all possible values of m from the set $\{0,1,\dots,n-\ell_0\}$, the mean inter-service completion time in stage 1, is*

$$\mathbb{E}\left[t_{1,r}-t_{1,r-1}\right] = \sum_{m:m+1<r} p_m \frac{1}{\mu(n-\ell_0-r+1)}$$
$$+ \sum_{m:m+1=r} p_m \Big[c\Big[\frac{1}{\alpha^{(r-1)}} - \sum_{i=1}^{r-1}\frac{\alpha^{i-r+1}}{ic\mu}\Big] + \frac{1}{\mu(n-j)}\Big]$$
$$+ \sum_{m:m+1>r} {}_2F_1\Big(\begin{matrix}1,r\\ m+2\end{matrix};\alpha\Big)\frac{r\alpha}{\mu(m+1)}p_m.$$

Proof: The result follows by using Lemma 9 and from the tower property of nested expectations

$$\mathbb{E}\left[(t_{1,r}-t_{1,r-1})\right] = \mathbb{E}\left[\mathbb{E}\left[(t_{1,r}-t_{1,r-1})|E_{j-\ell_0}\right]\right],$$

and the fact that $N(t_1, t_1+c) \in \{0,\dots,n_0-\ell_0\}$ and p_m is defined in (7), as the probability of the number of service completions $N(t_1, t_1+c)$ in the interval $(t_1, t_1+c]$ being $m = j-\ell_0$ where t_1 is the time of ℓ_0 completions of initial n_0 coded sub-tasks. ■

We have all the necessary results to compute the means of service completion time and server utilization cost. Next, we consider two possibilities for the initial number of servers n_0: when $n_0 < k$ and otherwise. Note that when $n_0 < k$, then $t_2 > t_1 + c$ almost surely, since k coded sub-tasks can never be finished by initial n_0 servers.

1) **Case $n_0 < k$:**

In this case, the initial n_0 servers are always less than required number of coded sub-tasks k. There are $\ell_0 \leqslant n_0 < k$ completed coded sub-tasks at time instant t_1 and hence we need additional servers to be switched on at this instant t_1 to finish the remaining $k-\ell_0$ coded sub-tasks. The additional number of coded sub-task completions in the duration $(t_1, t_1+c]$ is denoted by $N(t_1, t_1+c) = j-\ell_0 \in \{0,1,\dots,n_0-\ell_0\}$. These $j-\ell_0$ coded sub-tasks are completed only by the remaining $n_0-\ell_0$ servers out of initial n_0 servers. Since $n_0 < k$, the service completion time $t_2 > t_1+c$ and the remaining $k-j$ coded sub-tasks are completed by $n-j$ parallel servers working on individual coded sub-tasks. From the memoryless property of exponential random variables, the excess service of each of these $n-j$ parallel servers is exponentially distributed with rate μ.

We now compute the mean service completion time and mean server utilization cost for $n_0 < k$ case.

Theorem 11. *For the single forking case with n total servers for k sub-tasks and initial number of servers $n_0 < k$, the mean server utilization cost is*

$$\mathbb{E}\left[W\right] = \lambda nc + \frac{\lambda k}{\mu}, \quad (11)$$

and the mean service completion time is

$$\mathbb{E}\left[t_2\right] = c + \mathbb{E}\left[t_1\right] + \frac{1}{\mu}\sum_{j=\ell_0}^{n_0} p_{j-\ell_0}\sum_{i=j}^{k-1}\frac{1}{(n-i)}, \quad (12)$$

where $\mathbb{E}\left[t_1\right]$ is given in (5) *and $p_{j-\ell_0}$ is given in* (7)*.*

Proof: The detailed proof is provided in Appendix C. ■

From the Theorem 11, we observe that the mean server utilization cost remains same for all values of initial number of servers $n_0 < k$ and forking threshold ℓ_0. We further observe that the mean service completion time decreases as we increase the number of initial servers $n_0 < k$. Hence, it follows that for the case when $n_0 < k$, the optimal number of initial servers is

$n_0^* = k-1$ at time $t=0$. Further, since increasing ℓ_0 increases the mean service completion time for any n_0 and the mean service utilization cost does not depend on ℓ_0, it follows that $\ell_0^* = 1$ is the best choice for $n_0 < k$. Thus, the joint best choices for (n_0^*, ℓ_0^*) in this regime are $(k-1, 1)$.

In addition, we note that if $n_0 = n$ and all the n coded sub-tasks are started at $t = 0$, the mean service utilization cost can be easily shown to be $\lambda nc + \frac{\lambda k}{\mu}$ which is the same as that for all $n_0 < k$. Thus, as compared to no forking ($n_0 = n$), the single forking with $n_0 < k$ has the same mean server utilization cost while it has higher mean service completion time. Thus, this regime doesn't provide any tradeoff point between service completion time and server utilization cost which is worse than no-forking, and hence the only region of interest for a system designer is $n_0 \geqslant k$.

2) **Case $n_0 \geqslant k$:**

In this case, the initial number of servers n_0 is always greater than the required number of coded sub-tasks k, and hence the number of completed coded sub-tasks ℓ_0 at the forking point t_1 are in $\{0, 1, \dots, k\}$. There are three different possibilities for completing k coded sub-tasks. First possibility is $\ell_0 = k$, when all the required k coded sub-tasks are finished on initial n_0 servers without any forking. In this case, $t_2 = t_1$. For the next two possibilities, $\ell_0 < k$ and hence forking is needed.

Second possibility is $\ell_0 < k$ and $\ell_0 + N(t_1, t_1 + c) = j \leqslant k-1$, where $j - \ell_0$ service completions occur in the duration $[t_1, t_1 + c)$ and $\ell_0 \leqslant j \leqslant k-1$. This implies that even though $n_0 > k$, the total coded sub-tasks finished until instant $t_1 + c$ are still less than k and remaining $k - j > 0$ coded sub-tasks among the required k are completed only after $t_1 + c$, when $n - j$ parallel servers are in their memoryless phase. In this case, $t_2 = t_1 + c + X_{k-j}^{n-j}$ for $N(t_1, t_1 + c) = j - \ell_0 \in \{0, \dots, k - \ell_0 - 1\}$.

Third possibility is when $\ell_0 < k$ and $\ell_0 + N(t_1, t_1 + c) \geqslant k$. That is, even though the coded sub-tasks are forked on additional n_1 servers at time t_1, the job is completed at k out of n_0 initial servers before the constant start-up time of these additional $n_1$servers is finished. This happens when $s_{k-\ell_0} \leqslant c$ and in this case, $t_2 = t_1 + s_{k-\ell_0}$ for $N(t_1, t_1 + c) \geqslant k - \ell_0$. Recall that $s_{k-\ell_0}$ is the $(k - \ell_0)$th service completion in stage 1 after t_1. Summarizing all the results, we write the service completion time in the case $n_0 \geqslant k$ and $N(t_1, t_1 + c) = j - \ell_0$ as

$$t_2 = t_1 + s_{k-\ell_0} \mathbb{1}_{\{\ell_0 < k \leqslant j\}} + (c + X_{k-j}^{n-j}) \mathbb{1}_{\{\ell_0 \leqslant j < k\}}.$$

For $n_0 \geqslant k$, the mean service completion time and the mean server utilization cost are given in the following theorem.

Theorem 12. *In single forking scheme, for $n_0 \geqslant k$ case, the mean service completion time $\mathbb{E}[t_2]$ is*

$$\mathbb{E}[t_1] + \Big[\sum_{r=1}^{k-\ell_0} \mathbb{E}[t_{1,r} - t_{1,r-1}] \Big] \mathbb{1}_{\{\ell_0 < k\}} \quad (13)$$

and the mean server utilization cost $\mathbb{E}[W]$ is

$$\mathbb{E}[W_0] + \lambda \sum_{r=1}^{k-\ell_0} (n - \ell_0 - r + 1) \mathbb{E}[t_{1,r} - t_{1,r-1}] \mathbb{1}_{\{\ell_0 < k\}}. \quad (14)$$

Where $\mathbb{E}[t_{1,r} - t_{1,r-1}]$ in the above expressions is given by Corollary 10.

Proof: In Corollary 5, we have already computed the mean completion time $\mathbb{E}[t_1]$ of stage 0, and the mean server utilization cost $\mathbb{E}[W_0]$ in stage 0. Recall that since completion of any k coded sub-tasks suffice for the job completion, the forking threshold $\ell_0 \leqslant k$.

We first consider the case when $\ell_0 = k$. In this case, we do not need to add any further servers because all the required tasks are already finished in stage 0 itself. Hence, there is no need of forking in this case, and the mean service completion time is given by $\mathbb{E}[t_1]$ and the mean server utilization cost is given by $\mathbb{E}[W_0]$.

We next consider the case when $\ell_0 < k$. In this case, the job completion occurs necessarily in stage 1. Thus, we need to compute $\mathbb{E}[t_2 - t_1]$ and $\mathbb{E}[W_1]$ in order to evaluate the mean service completion time $\mathbb{E}[t_2]$ and the mean server utilization cost $\mathbb{E}[W_0 + W_1]$. The duration of stage 1 can be written as a telescopic sum of inter service times

$$t_2 - t_1 = \sum_{r=1}^{k-\ell_0-1} (t_{1,r} - t_{1,r-1}).$$

Further for $\ell_0 < k$, the number of servers that are active in stage 1 after $(r-1)$th service completions are $n - \ell_0 - r + 1$ and the associated cost incurred in the interval $[t_{1,r-1}, t_{1,r})$ is $\lambda(t_{1,r} - t_{1,r-1})(n - \ell_0 - r + 1)$. Therefore, we can write the server utilization cost in stage 1 as

$$W_1 = \lambda \sum_{r=1}^{k-\ell_0-1} (n - \ell_0 - r + 1)(t_{1,r} - t_{1,r-1}).$$

The result follows from taking mean of the duration $t_2 - t_1$ and server utilization cost W_1, from the linearity of expectations, and considering both possible cases. ■

We observe that when $n_0 \geqslant k$, the mean service utilization cost depends on the initial number of servers n_0 as well as the total number of servers n, unlike the case $n_0 < k$ where this cost depends only on the total number of servers n. In the following section, we numerically investigate the tradeoff between the two performance metrics, which allows us the proper choice of system parameters to work in a specified regime.

IV. Numerical Studies

For numerical evaluation of mean service completion time and mean server cost utilization for single forking systems, we choose the following system parameters. We select the sub-task fragmentation of a single job as $k = 12$, and a maximum redundancy factor of $n/k = 2$. That is, we choose the total number of servers $n = 24$. We take the server utilization cost rate to be $\lambda = 1$. Coded-task completion time at each server was chosen to be an *i.i.d.* random variable having a shifted exponential distribution. For numerical studies in this section, we choose the shift parameter $c = 1$ and the exponential rate $\mu = 0.5$. We compare the two cases $n_0 < k$ and $n_0 \geqslant k$ with no-forking case $n_0 = n$, where all the available servers are used as initial servers.

We first study the impact of number of initial servers n_0 on the mean service completion time, when $n_0 < k$. To this end, we plot the mean service completion time as a function of fork-task threshold $\ell_0 \in [n_0]$ in Figure 1, for different values of initial servers $n_0 \in \{3, 5, 7, 9, 11\}$ such that $n_0 < k$. The analytical results in Theorem 11 are substantiated, by observing that the mean service completion time $\mathbb{E}[S]$ increases with increase in fork-task threshold ℓ_0 and decreases with increase in initial number of servers n_0. Since the mean service utilization cost in $n_0 < k$ is constant for any choice of n_0 and ℓ_0, it is not depicted.

From Figure 1, we infer that as compared to the no forking case of $n_0 = n$, the case of single forking with $n_0 < k$ has higher service completion time for the same server utilization cost. Thus, the regime $n_0 < k$ is not interesting for practical applications.

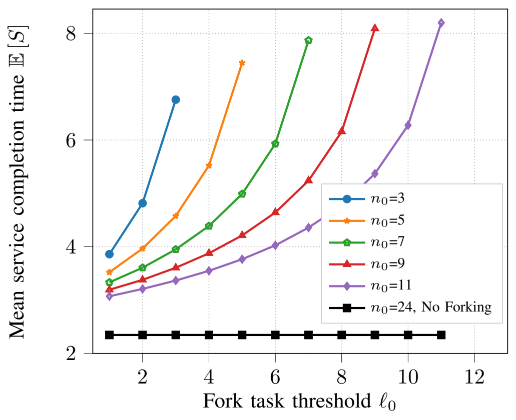

Figure 1. For the setting $n_0 < k$, this graph displays the mean service completion time $\mathbb{E}[S]$ as a function of fork task threshold ℓ_0 for single forking when the total number of servers $n = 24$, the required of coded sub-task completions $k = 12$, and different initial servers $n_0 \in \{3, 5, 7, 9, 11\}$. The single coded sub-task execution time at servers are taken as *i.i.d.* shifted exponential distribution with shift $c = 1$ and rate $\mu = 0.5$.

We next study the case when $n_0 \geqslant k$. To this end, we plot the mean service completion time in Figure 2 and mean server utilization in Figure 3, both as a function of fork-task threshold $\ell_0 \in [k]$, for different values of initial servers $n_0 \in \{12, 14, 16, 18, 20\}$. The analytical results in Theorem 12 are substantiated by observing that the mean service completion time $\mathbb{E}[S]$ increases with increase in fork-task threshold ℓ_0 and decreases with increase in initial number of servers n_0. Further, the mean server utilization cost $\mathbb{E}[W]$ decreases with increase in fork-task threshold ℓ_0. Thus, there is a tradeoff between the two performance measures as a function of fork-task threshold ℓ_0. The tradeoff between the two performance metrics of interest is plotted in Figure 4, which suggests that the number of initial servers n_0 and the forking threshold ℓ_0 affords a true tradeoff between these metrics.

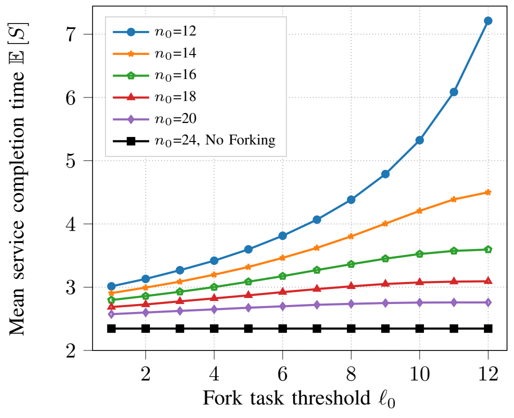

Figure 2. For the setting $n_0 \geqslant k$, this graph displays the mean service completion time $\mathbb{E}[S]$ as a function of fork task threshold ℓ_0 for single forking with the total number of servers $n = 24$, the total needed coded sub-tasks $k = 12$, and different numbers of initial servers $n_0 \in \{12, 14, 16, 18, 20\}$. The single coded sub-task execution time at servers are assumed to be *i.i.d.* shifted exponential distribution with shift $c = 1$ and rate $\mu = 0.5$.

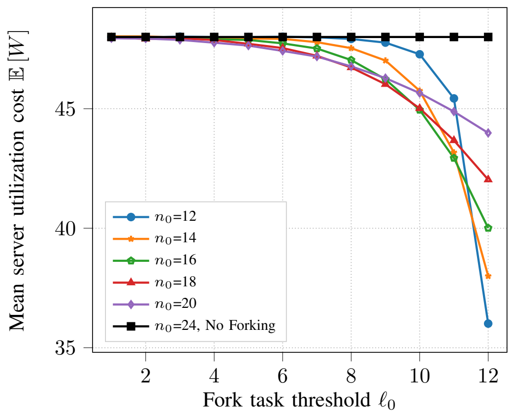

Figure 3. For the setting $n_0 \geqslant k$, this graph displays the mean server utilization cost $\mathbb{E}[W]$ as a function of fork task threshold ℓ_0 for single forking with the total number of servers $n = 24$, the total needed coded sub-tasks $k = 12$, and different numbers of initial servers $n_0 \in \{12, 14, 16, 18, 20\}$. The single coded sub-task execution time at servers are assumed to be *i.i.d.* shifted exponential distribution with shift $c = 1$ and rate $\mu = 0.5$.

It is interesting to observe the behavior of mean server utilization cost as a function of initial number of servers n_0 in Figure 3. We note that for each fork-task threshold ℓ_0, there exists an optimal number of initial servers n_0 that minimizes the server utilization cost. We further observe in Figure 4 that for $n_0 = 20$, the mean service completion time increases only 17.635% while the mean server utilization cost can be decreased 8.3617% by an appropriate choice of ℓ_0 as compared to choosing no forking case of $n_0 = n$. However, a value of ℓ_0 cannot be chosen for $n_0 = 20$ that reduces the mean server utilization cost beyond 8.3617%. In order to have further reduction in mean server utilization cost, we can choose n_0 to 18 which helps to decrease mean server completion time by 12.43% at an expense of 31.888% increase in mean service completion time as compared to the no forking case $n_0 = n$. The intermediate points on the curve of $n_0 = 18$ further provide tradeoff points that can be chosen based on the desired combination of the two measures as required by the system designer. The choice of $n_0 = 12$ further helps decrease the mean server utilization cost by 24.976% by having 207.49% times increase in the mean service completion time as compared to the no forking case $n_0 = n$. Thus, we see that appropriate choice of n_0 and ℓ_0 provide tradeoff points that help minimizing the mean server utilization cost at the expense of the mean service completion time.

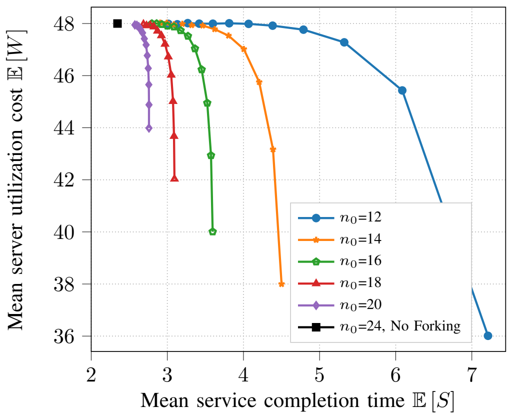

Figure 4. For the setting $n_0 \geqslant k$, we have plotted the mean server utilization cost $\mathbb{E}[W]$ as a function of the mean service completion time $\mathbb{E}[S]$ by varying fork task threshold $\ell_0 \in [n_0]$ in single forking. The total number of servers considered are $n = 24$, the total coded sub-task needed are $k = 12$. The single coded sub-task execution time at servers are assumed to be *i.i.d.* shifted exponential distribution with shift $c = 1$ and rate $\mu = 0.5$. We have plotted the same curve for different values of initial servers $n_0 \in \{12, 14, 16, 18, 20\}$. For each curve, ℓ_0 increasing from left to right.

V. Conclusions and Future Work

We study the single-forking for a single job that can be divided into k sub-tasks to be computed over n servers, in two stages. We assume that k sub-tasks can be coded into n computation coded sub-tasks using (n, k)-MDS coding, such that completion of any k coded sub-tasks lead to completion of the entire job. We assume that only n_0 out of n servers are started at time 0. After completion of ℓ_0 out of n_0 servers, remaining $n - n_0$ servers are initiated. Using the shifted exponential service times of the servers, we derive expressions for two performance metrics:

(i) the mean service completion time which indicates the mean time when k servers have finished execution, and
(ii) the mean service utilization cost which indicates the aggregate mean of times each server is busy processing the coded sub-tasks.

We show that the mean service completion time and the mean server utilization cost are respectively increasing and decreasing function of the fork-task threshold ℓ_0. We also note that though the mean service completion is decreasing function of the number of initial servers n_0, there exists an optimal number of initial servers n_0 for each fork-task threshold ℓ_0. For $n_0 \geqslant k$, we find that there is a tradeoff between the two performance metrics and leads to decrease in mean server utilization cost at the expense of mean service completion time and an efficient choice of the parameters is helpful.

Generalization of this study to multi-forking points with coded sub-tasks remains an important future direction. We believe that the framework provided in this article can be utilized to quantify the performance gains of multi-forked coded jobs.

Appendix A
Auxiliary Result

Remark 13. For positive integers p, q and positive reals c, μ, we have the following identity in terms of the hypergeometric series ${}_pF_q$ defined in Definition 7

$$\int_0^c xe^{-\mu x}\frac{(1-e^{-\mu x})^q(e^{-\mu x}-e^{-\mu c})^{p-q}}{(1-e^{-\mu c})^{p+2}}dx$$
$$= \left(\frac{1}{(p+2)\mu^2\binom{p+1}{q+1}}\right){}_3F_2\left[\begin{matrix}1,1,q+2\\2,p+3\end{matrix};1-e^{-\mu c}\right].$$

Remark 14. For positive integer m and positive reals c, μ, we have the following identity $m\mu\int_0^c xe^{-\mu x}(1-e^{-\mu x})^{m-1}dx = c(1-e^{-\mu c})^m - c + \sum_{i=1}^m \frac{(1-e^{-\mu c})^i}{i\mu}$. Using the definition of hypergeometric series in (8), it can be verified that expression in Remark 13 simplifies to the above expression for $p = q = m - 1$.

Appendix B
Proof of Lemma 8

We denote $m = j - \ell_0$ for convenience. Let $N(t_1, t_1 + c) = m$, then $t_1 + s_1, \dots, t_1 + s_m$ are the coded sub-task completion times of the first m servers out of $n_0 - \ell_0$ parallel servers in their memoryless phase in the duration $[t_1, t_1 + c)$. In the duration $[t_{1,r-1}, t_{1,r})$ for $r \in [m]$, there are $n_0 - \ell_0 - r + 1$ parallel servers in their memoryless phase, and hence the inter-service completion times $(t_{1,r} - t_{1,r-1} : r \in [m])$ are independent and distributed exponentially with parameter $\mu_r \triangleq (n_0 - \ell_0 - r + 1)\mu$. Denoting $s_0 = 0$, we have $s_r - s_{r-1} = t_{1,r} - t_{1,r-1}$ for each $r \in [m]$. From the definition of μ_r's and p_m, the independence of $s_r - s_{r-1}$, and rearrangement of terms we can write the conditional joint density of vector $s = (s_1, s_2, \dots, s_m)$ given event E_m as

$$f_{s_1,\dots,s_m|E_m} = \prod_{i=1}^m \frac{i\mu e^{-\mu s_i}}{1-e^{-c\mu}}. \tag{15}$$

From the definition of the task completion times, the possible values of the vector $s = (s_1, \dots, s_m)$ satisfy the constraint $0 < s_1 < \dots < s_m < c$. That is, we can write the set of possible values for vector s as A_m, where A_m is a vector of increasing co-ordinates bounded between $(0, c)$, and can be written as

$$A_m \triangleq \{s \in \mathbb{R}^m : 0 < s_1 < \dots < s_m < c\}.$$

This constraint couples the set of achievable values for the vector s, and hence even though the conditional density has a product form, the random variables $(s_1,\dots,s_m)$ are not conditionally independent given the event E_m.

To compute the conditional expectation $\mathbb{E}\left[s_r|E_m\right]$, we find the conditional marginal density of s_r given the event E_m. To this end, we integrate the conditional joint density of vector s over variables without s_r. In terms of $s_r \in (0,c)$, we can write the region of integration as the following intersection of regions,

$$A_m^{-r} = \cap_{i<r}\left\{0 < s_i < s_{i+1}\right\} \cap_{i>r}\left\{s_{i-1} < s_i < c\right\}.$$

Using the conditional density of vector s defined in (15) in the above equation, and denoting $\alpha \triangleq 1-e^{-c\mu}$ and $\alpha_r \triangleq 1-e^{-\mu s_r}$ for clarity of presentation, we can compute the conditional marginal density function [?]

$$f_{s_r|E_m} = \frac{m\mu(1-\alpha_r)}{\alpha^m}\binom{m-1}{r-1}(\alpha_r)^{r-1}(\alpha-\alpha_r)^{m-r}. \quad (16)$$

The conditional mean $\mathbb{E}\left[s_r|E_m\right] = \int_0^c f_{s_r|E_m} ds_r$ is obtained by integrating the conditional marginal density in (16), over $s_r \in (0,c)$. For $r \in [m-1]$, the result follows from the integral identity of Remark 13 for $x = s_r, q = r-1, p = m-1$ and $\alpha = 1-e^{-\mu c}$. Similarly, the result for $r = j-\ell_0$ follows from Remark 14 for $x = s_m$ and $m = j-\ell_0$.

Appendix C
Proof of Theorem 11

We have already computed $\mathbb{E}\left[W_0\right]$ and $\mathbb{E}\left[t_1\right]$ in Corollary 5. Recall the event $E_{j-\ell_0} = \{N(t_1,t_1+c) = j-\ell_0\}$ for $j \in \{\ell_0,\dots,n_0\}$, and we are considering the case $n_0 < k \leqslant n$. We first compute the mean service completion time, and it suffices to show that the mean duration of stage 1 is given by $\mathbb{E}\left[t_2-t_1\right] = c + \frac{1}{\mu}\sum_{j=\ell_0}^{n_0} p_{j-\ell_0}\sum_{i=j}^{k-1}\frac{1}{n-i}$. To see this, we first observe that since $n_0 < k$, service for k coded sub-tasks can't be completed in the duration $[t_1, t_1+c)$. Conditioned on the event $E_{j-\ell_0}$, there are $n-j$ parallel servers in their memoryless phase at instant t_1+c and there are $k-j$ remaining coded sub-tasks. Hence, we can write $t_2 = t_1+c+\sum_{j=\ell_0}^{n_0} X_{k-j}^{n-j}\mathbb{1}_{\left\{E_{j-\ell_0}\right\}}$. Result follows by taking expectations on both sides, realizing that $p_{j-\ell_0} = \mathbb{E}\left[\mathbb{1}_{\left\{E_{j-\ell_0}\right\}}\right]$, and the fact that $\mathbb{E}\left[X_{k-j}^{n-j}\right] = \frac{1}{\mu}\sum_{i=j}^{k-1}\frac{1}{n-i}$ from Remark 3.

We next compute the mean server utilization cost, and it suffices to show that the mean server utilization cost in stage 1 is given by $\mathbb{E}\left[W_1\right] = \lambda n_1 c + \frac{\lambda}{\mu}(k-\ell_0)$. To this end, we recall that the server utilization cost in stage 1 can be written as $W_1 = \lambda\sum_{r=0}^{k-\ell_0-1}(t_{1,r+1}-t_{1,r})(n-\ell_0-r)$.

Step 1: Expansion of server utilization cost. Using the definition of $s_r = t_{1,r}-t_1, s_0 = 0, n = n_0+n_1$, and denoting $m = j-\ell_0 \in \{0,\dots,n_0-\ell_0\}$, we can re-write the server utilization cost in stage 1 in terms of the indicator to the events $E_m = \{N(t_1,t_1+c) = m\}$, as

$$\begin{aligned}W_1 = \lambda n_1 c + \lambda\sum_{m=0}^{n_0-\ell_0}\mathbb{1}_{\{E_m\}}\Big[\sum_{r=0}^{m-1}(s_{r+1}-s_r)(n_0-\ell_0-r)\\ + (c-s_m)(n_0-j) + (s_{m+1}-c)(n-j)\\ + \sum_{r=m+1}^{k-\ell_0-1}(s_{r+1}-s_r)(n-\ell_0-r)\Big].\end{aligned}$$

Step 2: Rearrangement and expectation. With this substitution and re-arrangement of the second and the third term in the above expression, we get $\sum_{r=0}^{m-1}(s_{r+1}-s_r)(n_0-\ell_0-r)+(c-s_m)(n_0-j) = c(n_0-j)+\sum_{r=1}^{m}s_r$. Therefore, we can write the mean server utilization cost in stage 1 in terms of the conditional means $\mathbb{E}\left[s_r|E_m\right]$ as

$$\begin{aligned}\mathbb{E}\left[W_1\right] = \lambda c n_1 + \lambda\sum_{m=0}^{n_0-\ell_0}p_m\Big(c(n_0-j)+\sum_{r=1}^{m}\mathbb{E}\left[s_r|E_m\right]\\ + (n-j)\mathbb{E}\left[(s_{m+1}-c)|E_m\right]\\ + \sum_{r=m+1}^{k-\ell_0-1}(n-\ell_0-r)\mathbb{E}\left[(s_{r+1}-s_r)|E_m\right]\Big).\end{aligned}$$

Step 3: Conditional mean of inter-service completion times. Since there are $n-\ell_0-r$ parallel servers working in their memoryless phase after t_1+c, it follows from Remark 3 that $\mathbb{E}\left[s_{m+1}-c|E_m\right] = \frac{1}{\mu(n-j)}$ and $\mathbb{E}\left[(s_{r+1}-s_r)|E_m\right] = \frac{1}{\mu(n-\ell_0-r)}$ for $r > m$. Therefore, we have

$$\begin{aligned}\frac{1}{\lambda}\mathbb{E}\left[W_1\right] = cn_1 + c(n_0-\ell_0) + \frac{k-\ell_0}{\mu}\\ + \sum_{m=0}^{n_0-\ell_0}p_m\Big[\mathbb{E}\left[\sum_{r=1}^{m}s_r|E_m\right] - (c+\frac{1}{\mu})(m)\Big].\end{aligned}$$

Step 4: Conditional mean of summation of completion times. From the conditional joint distribution of the vector $s = (s_1,\dots,s_m)$ defined in (15), we can find the moment generating function of sum $Y \triangleq \sum_{r=1}^{m}s_r$ conditioned on the event E_m as

$$\Phi_{Y|E_m}(\theta) = \mathbb{E}\left[e^{-\theta Y}|E_m\right] = \Big(\frac{\mu(1-e^{-(\theta+\mu)c})}{\alpha(\theta+\mu)}\Big)^m.$$

Since the conditional expectation $\mathbb{E}\left[Y|E_m\right]$ is equal to $-\frac{d}{d\theta}\Phi_{Y|E_m}(\theta)|_{\theta=0}$, we obtain $\mathbb{E}\left[\sum_{r=1}^{m}s_r|E_m\right] = m\Big(c+\frac{1}{\mu}\Big)-\frac{mc}{\alpha}$. By using this result, we have

$$\frac{1}{\lambda}\mathbb{E}\left[W_1\right] = cn_1 + c(n_0-\ell_0) + \frac{k-\ell_0}{\mu} + \sum_{m=0}^{n_0-\ell_0}p_m\frac{mc}{\alpha}.$$

Step 5: Mean number of completions in $[\mathbf{t_1}, \mathbf{t_1}+\mathbf{c})$. From Lemma 6, we see that $N(t_1,t_1+c)$ is a binomial random variable with parameters $(n_0-\ell_0,\alpha)$ for $\alpha = 1-e^{-\mu c}$. Thus, $\mathbb{E}\left[N(t_1,t_1+c)\right] = \sum_{m=0}^{n_0-\ell_0}mp_m = (n_0-\ell_0)\alpha$ and the result $\mathbb{E}\left[W_1\right] = \lambda n_1 c + \frac{\lambda}{\mu}(k-\ell_0)$ follows.

References

[1] R. Tandon, Q. Lei, A. G. Dimakis, and N. Karampatziakis, "Gradient coding: Avoiding stragglers in distributed learning," in *Int. Conf. Mach. Learning (ICML)*, 2017, pp. 3368–3376.

[2] M. F. Aktas, P. Peng, and E. Soljanin, "Straggler mitigation by delayed relaunch of tasks," *ACM SIGMETRICS Perf. Eval. Review*, vol. 45, no. 2, pp. 224–231, 2018.

[3] Y. Xiang, T. Lan, V. Aggarwal, and Y. F. R. Chen, "Joint latency and cost optimization for erasure-coded data center storage," *ACM SIGMETRICS Perf. Eval. Review*, vol. 42, no. 2, pp. 3–14, Sep. 2014.

[4] Y. Xiang, T. Lan, V. Aggarwal, and Y. Chen, "Joint latency and cost optimization for erasure-coded data center storage," *IEEE/ACM Trans. Netw.*, vol. 24, no. 4, pp. 2443–2457, Aug 2016.

[5] A. Badita, P. Parag, and J.-F. Chamberland, "Latency analysis for distributed coded storage systems," *IEEE Trans. Inf. Theory*, vol. 65, no. 8, pp. 4683–4698, Aug 2019.

[6] 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, 2018.

[7] S. Dutta, V. Cadambe, and P. Grover, "Short-dot: Computing large linear transforms distributedly using coded short dot products," in *Adv. Neural Inf. Process. Sys.*, 2016, pp. 2100–2108.

[8] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Straggler mitigation in distributed matrix multiplication: Fundamental limits and optimal coding," in *IEEE Inter. Symp. Inf. Theory (ISIT)*, June 2018, pp. 2022–2026.

[9] M. Ye and E. A. Abbe, "Communication-computation efficient gradient coding," in *Int. Conf. Mach. Learning (ICML)*. International Machine Learning Society (IMLS), 2018, p. 9716p.

[10] K. Lee, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Coded computation for multicore setups," in *IEEE Inter. Symp. Inf. Theory (ISIT)*, 2017, pp. 2413–2417.

[11] 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, 2018.

[12] K. Wan, D. Tuninetti, M. Ji, and P. Piantanida, "Fundamental limits of distributed data shuffling," in *Allerton Conf. Commun., Control, Comp. (Allerton)*, 2018, pp. 662–669.

[13] M. A. Attia and R. Tandon, "Near optimal coded data shuffling for distributed learning," *arXiv preprint arXiv:1801.01875*, 2018.

[14] L. Song, C. Fragouli, and T. Zhao, "A pliable index coding approach to data shuffling," in *IEEE Inter. Symp. Inf. Theory (ISIT)*, 2017, pp. 2558–2562.

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

[16] G. Ananthanarayanan, A. Ghodsi, S. Shenker, and I. Stoica, "Effective straggler mitigation: Attack of the clones." in *USENIX Symp. Net. Sys. Desgn. Impl. (NSDI)*, vol. 13, 2013, pp. 185–198.

[17] D. Wang, G. Joshi, and G. Wornell, "Using straggler replication to reduce latency in large-scale parallel computing," *ACM SIGMETRICS Perf. Eval. Review*, vol. 43, no. 3, pp. 7–11, 2015.

[18] N. J. Yadwadkar, B. Hariharan, J. E. Gonzalez, and R. Katz, "Multi-task learning for straggler avoiding predictive job scheduling," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 3692–3728, 2016.

[19] R. Bitar, P. Parag, and S. El Rouayheb, "Minimizing latency for secure distributed computing," in *IEEE Inter. Symp. Inf. Theory (ISIT)*, Aachen, Germany, 2017, pp. 2900–2904.

[20] 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.

[21] A. O. Al-Abbasi and V. Aggarwal, "Video streaming in distributed erasure-coded storage systems: Stall duration analysis," *IEEE/ACM Trans. Netw.*, vol. 26, no. 4, pp. 1921–1932, 2018.

[22] G. Gasper, M. Rahman, and G. George, *Basic hypergeometric series*. Cambridge university press, 2004, vol. 96.