Optimal Server Selection for Straggler Mitigation

Ajay Badita, *Student Member, IEEE*, Parimal Parag, *Member, IEEE*, and Vaneet Aggarwal, *Senior Member, IEEE*

***Abstract*— The performance of large-scale distributed compute systems is adversely impacted by stragglers when the execution time of a job is uncertain. To manage stragglers, we consider a multi-fork approach for job scheduling, where additional parallel servers are added at forking instants. In terms of the forking instants and the number of additional servers, we compute the job completion time and the cost of server utilization when the task processing times are assumed to have a shifted exponential distribution. We use this study to provide insights into the scheduling design of the forking instants and the associated number of additional servers to be started. Numerical results demonstrate orders of magnitude improvement in cost in the regime of low completion times as compared to the prior works.**

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

I. Introduction

LARGE scale computing jobs require multi-stage computation, where computation per stage is performed in parallel over a large number of servers. The execution time of a task on a machine has stochastic variations due to many contributing factors such as co-hosting, virtualization, hardware and network variations [1]. A slow server can delay the onset of next stage computation, and we call it a *straggling* server. One of the key challenges in cloud computing is the problem of straggling servers, which can significantly increase the job completion time [2]–[4]. Straggler mitigation is a particularly important problem, considering this the organizations such as VMWare and Amazon have spent substantial effort optimizing the operation of virtualization technologies for massive-scale systems [2]. This paper aims to find efficient scheduling mechanisms for straggler mitigation by analyzing how the replication of straggling tasks affects the mean service completion time and the mean server utilization cost of computing resources.

The idea of replicating tasks in parallel computing has been adopted at a large scale via the speculative execution in both Hadoop MapReduce [1], and Apache Spark [5]. The use of redundancy to reduce mean service completion time has also attracted attention in other contexts such as cloud storage and networking [6], [7]. These works focus on the queuing aspects at the storage servers. Replication is a special case of general redundancy mechanism and is considered in this paper. Replication is also referred to as forking in popular scheduling parlance. Replicating a job on multiple servers affords us the parallelism gains, while it comes at the cost of server utilization. We consider a dynamic replication strategy, where an unfinished task is sequentially forked over multiple servers at certain forking times. We thus provide an efficient tradeoff between the mean service completion time and the mean utilization cost of computing resources.

Recently, the authors of [8] provided a framework for analyzing straggling tasks for a computing job. The authors of [8] considered executing K jobs (or tasks), where one copy for each job was started at time $t = 0$. They had a single forking point at the instant of job completion of a fraction $(1 - p)$ of all K jobs. At this forking point, each of the remaining pK incomplete jobs is replicated r times. Two variants, where the original tasks were killed or kept at the forking point were considered. In this setting, the mean service completion time and the mean server utilization cost of computing resources per job were computed in the limit as $K \to \infty$, where the execution time follows either a shifted exponential or a Pareto distribution. The analysis assumes a single forking point, corresponding to the time where multiple replicas are run for an unfinished job.

In contrast, we provide a multi-fork analysis of the computing jobs, with a selection of number of servers for replication at each forking point. Specifically, we assume K jobs, all starting at $t_0 = 0$ and an identical sequence of m forking points for each job, denoted by t_i for $i \in [m] \triangleq \{1, \ldots, m\}$. We initialize each task on n_0 parallel servers at instant $t_0 = 0$. At each forking point t_i, we start additional n_i replicas for each unfinished job. If a job is unfinished for any time $t \in [t_i, t_{i+1})$, then it has $N_i = \sum_{j=0}^{i} n_j$ active replicas. This procedure is illustrated in Fig. 1, where we plot the time-evolution of number of active replicas for a single unfinished task. With multiple forking points, the mean service completion time and average server utilization cost are evaluated where the server execution times are assumed

Manuscript received April 25, 2019; revised November 9, 2019; accepted January 22, 2020; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor K. Jagannathan. Date of publication February 28, 2020; date of current version April 16, 2020. This work was supported in part by the Science and Engineering Research Board under Grant DSTO-1677, in part by the Department of Telecommunications, Government of India, under Grant DOTC-0001, in part by the Robert Bosch Center for Cyber-Physical Systems, in part by the Centre for Networked Intelligence (a Cisco CSR initiative) of the Indian Institute of Science, Bengaluru, in part by the VAJRA Fellowship, in part by the National Science Foundation under Grant CNS-1618335, and in part by Cisco. *(Corresponding author: Vaneet Aggarwal.)*

Ajay Badita and Parimal Parag are with the Department of Electrical and Communications Engineering, Indian Institute of Science, Bengaluru 560012, India (e-mail: ajaybadita@iisc.ac.in; parimal@iisc.ac.in).

Vaneet Aggarwal is with the School of Industrial Engineering and the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907 USA (e-mail: vaneet@purdue.edu).

This article has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the authors.

Digital Object Identifier 10.1109/TNET.2020.2973224

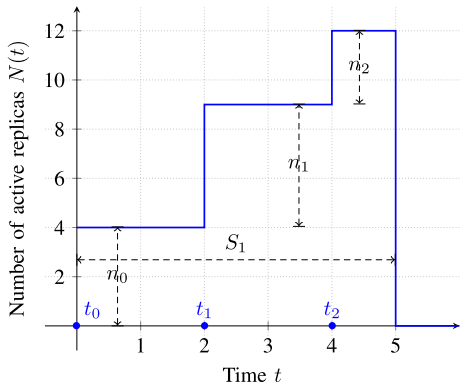

Fig. 1. We illustrate the two-forking for a single task with total number of servers $N = 12$, by plotting the time-evolution of number of active replicas $N(t)$. We consider the example when the sequence of number of forked servers is $(n_0, n_1, n_2) = (4, 5, 3)$, the sequence of forking times is $(t_0, t_1, t_2) = (0, 2, 4)$, and the service completion time is $S_1 = 5$. For this case, the server utilization cost $W = n_0 S_1 + n_1(S_1 - t_1) + n_2(S_1 - t_2)$.

to be *i.i.d.* following a shifted exponential distribution, and the forking points are separated by at least the shift of the distribution.

The results of single forking point analysis show that starting with multiple copies per job at time $t_0 = 0$ can perform much better than starting with a single copy per job as proposed in [8], when the forking time is below a certain threshold. Numerical evaluations show orders of magnitude improvement in the average server utilization cost for a fixed service completion time. The proposed framework thus shows that the single forking point strategies used in the literature may be significantly suboptimal, and one must judiciously select the number of servers to run at each forking time. Further, having more forking points help achieve a better tradeoff between the mean service completion time and the mean server utilization cost.

A. Related Work

It has been observed that task execution times have significant variability, partly due to resource sharing by multiple jobs [9]. The slowest tasks that determine the job execution time are known as "stragglers". One of the key approaches to mitigate the effect of stragglers is to either re-launch a delayed task, or pre-emptively assigning each such task to multiple servers and taking the result of first completing server per task and canceling the same completed task at remaining servers. It is known that cancellation overhead can reduce the parallelism gains afforded by the additional servers [10]. However, for simplicity of analysis and to obtain insight into optimistic performance gains, we assumed idealized assumption of negligible cancellation overhead.

Speculative execution have been studied in [11], which acts after the tasks have already slowed down. Proactive approaches launch redundant copies of a task in a hope that at least one of them will finish in a timely manner. The authors of [12] perform cloning to mitigate the effect of stragglers. The authors of [8] 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 [13].

The problem of analyzing the completion of replicated parallel tasks is equivalent to having multiple redundant requests. The authors of [14] present an analysis of redundant requests where each job enters the queue at multiple servers. Service time completion can be generalized to finding mean waiting time of a stream of arriving redundant requests, and has been studied in the context of distributed storage. We note that the queueing studies for streaming arrival of requests exist only for fixed redundancy per request, and are difficult to characterize analytically even for this case. This implies that each job is forked to the identical number of servers, and job is completed by joining identical number of service completions. Tight numerical bounds are provided in [6], analytical bounds are presented in [7], [15]–[17], analytical approximations appear in [18], exact analysis for small systems in [19], exact analysis for random independent scheduling for asymptotically large number of servers in [20], and an exact analysis of tail index for Pareto-distributed file sizes in [21].

Even though we are not considering the streaming arrival of requests, our setting is a generalization of the fixed redundancy scheduling approach studied in the above-mentioned works, since the number of parallel servers available to each task is a time-varying function in our problem setting.

B. Main Contributions

Our main contribution is the design of a multi-forking straggler mitigation policy that can efficiently trade-off mean service completion time and mean server utilization cost, by sequentially starting a number of replicas at forking points. The key contributions are summarized below.

1) We analytically compute the mean service completion time and mean server utilization cost for any finite number of forking points when the completion time of each job on any server is independent and identically distributed according to a shifted exponential distribution with shift c and rate μ, and the inter-forking times $t_i - t_{i-1} \geqslant c$ for each $i \in [m] \triangleq \{1, 2, \ldots, m\}$.
2) For a single forking point, the mean service completion time and mean server utilization cost are analytically computed for all values of forking instants t_1, initial number of replicas n_0, and additional replicas n_1. We demonstrate that for single forking point t_1, having initial number of replicas $n_0 = 1$ is sub-optimal since both the performance metrics decrease with initial number of replicas $n_0 \leqslant n_0^*$, where the inflection point $n_0^* \geqslant 1$ when the forking point $t_1 \leqslant t_1^*$.
3) Numerical results for multi-forking show orders of magnitude improvement in the tradeoff between the two metrics when compared to the baseline case of single-forking with single replica initialization of [8].
4) We performed numerical studies for single and multi-forking when the job execution times are assumed to have heavy-tailed distributions such as Pareto and Weibull. We also studied single and multi-forking on a real compute cluster. We verified that the insights derived from the analytical studies for the shifted exponential distribution continue to hold in all three cases.

C. Organization

The rest of the paper is organized as follows. Section II describes the model used in the paper. Section III provides the analytical results, where the mean service completion time and the mean server utilization are characterized for multiple forking points, with single forking being a special case. Section IV explores further properties with single forking point. Sections V and VI provide a tradeoff between the mean service completion time and the mean server utilization for single and multiple forking points, respectively. We also compare our approach with that in [8]. Section VII provides the experimental results on a real compute cluster, Intel DevCloud. Section VIII concludes the paper, with directions for future work.

II. System Model

We consider a distributed computation system with K jobs and KN identical servers, with the cost of server utilization λ per unit time. Each server $n \in [KN] \triangleq \{1, \ldots, KN\}$ has an independent and identically distributed (*i.i.d.*) random service time T_n with distribution function F for each scheduled job on this server. Uncertainty in execution time at various servers due to independent background processes, motivates our assumption of independently random execution time at each server. Identical distribution at each server is motivated primarily by analytical tractability, and the fact that we expect similar randomness at each identical server in a homogeneous cloud. Thus, following the existing literature [6]–[8], [10], [18], [21], we adopted this commonly-used assumption for analysis.

It has been shown in [7], [17], [22], [23] that shifted exponential well models the service time distribution in distributed computation networks. That is, it suggests that service time for each computation task can be modeled by aggregation of two components: a constant overhead and a random exponentially distributed component. Motivated by these studies together with the goal of analytical tractability, we assume the service time distribution to be a shifted exponential with rate μ and shift c, such that the complementary distribution function $\bar{F} = 1 - F$ can be written

$$\bar{F}(x) \triangleq P\{t_0 > x\} = \begin{cases} 1, & x \in [0, c], \\ e^{-\mu(x-c)}, & x \geqslant c. \end{cases} \tag{1}$$

We assume that KN servers are partitioned into K disjoint sets of N servers, where each set of N servers can be utilized by a single job. The service completion time for job $k \in [K]$ sequentially scheduled over N servers is denoted by S_k and its server utilization cost is denoted by W_k. Then the service completion time for all K jobs (also known as the makespan of the jobs) is the maximum of service completion times of all K jobs, and is denoted by

$$S = \max_{k \in [K]} S_k. \tag{2}$$

Similarly, the average server utilization cost for K independent jobs is defined as the average of server utilization cost for all K jobs, and is denoted by

$$W = \frac{1}{K} \sum_{k \in [K]} W_k. \tag{3}$$

We are interested in the optimal trade-off between mean service completion time $\mathbb{E}S$ and mean server utilization cost $\mathbb{E}W$ for K jobs over these KN servers. We will see that starting all the servers initially minimizes the mean service completion time, whereas it leads to maximum server utilization cost. Hence, we adopt an identical sequential policy for each of the K jobs. A job $k \in [K]$ starts with n_0 parallel servers at time $t_0 = 0$, and sequentially adds n_i servers at instant $t_i > t_{i-1}$ until we utilize all the N servers. We let m denote the number of sequential addition of servers such that $n_0 + \cdots + n_m = N$.

That is, we are considering K parallel jobs, where each job is replicated on N servers sequentially. Sequential addition of servers is motivated by the fact that service times are random and there is a cost associated with the on-time of each server. Hence, we should commission additional service only when absolutely necessary. For analytical tractability, we have further assumed K parallel jobs to be uncoupled and we add extra servers in an identical fashion for each unfinished job at the same forking times. One can couple the K jobs, by adding additional servers performing coded version of the tasks, such that any K task completions suffice [7], [15], [29]. However, this can incur encoding and decoding delay of the computational tasks [10], and requires mixing of K sub-tasks which may not always be desirable.

We will consider the general case of $m \geqslant 1$, and find the mean service completion time and the mean server utilization cost for the case when the inter server addition interval $t_i - t_{i-1} \geqslant c$. Next, we will consider the specific case of single forking when $m = 1$ and $t_i - t_{i-1} > 0$.

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. Even though the exact queueing analysis for multi-forking with stochastic arrivals remains open, we provide insights on sequential scheduling of K initial jobs assigned to total N servers each. In particular, the results in this paper can provide an understanding of how many servers to use at each forking time to optimize the mean service completion time $\mathbb{E}S$ and the mean server utilization cost $\mathbb{E}W$.

III. Analysis

We observe that service completion time S_k for each job $k \in [K]$ is independent due to independence of server completion times. Further, since we employ the identical forking strategy for each job, the service completion time S_k for each job $k \in [K]$ has an identical distribution as well. From the *i.i.d.* service completion times for individual job, it follows from (2) that $F_S(x) = F_{S_1}^K(x)$. From the positivity of service completions times, we have

$$\mathbb{E}S = \int_{\mathbb{R}_+} \bar{F}_S(x) dx = \int_{\mathbb{R}_+} (1 - (1 - \bar{F}_{S_1}(x))^K) dx. \tag{4}$$

From the similar arguments, we can conclude that the server utilization costs $(W_k : k \in [K])$ are *i.i.d.*, and from the

linearity of expectations, we have

$$\mathbb{E}W = \mathbb{E}W_1. \tag{5}$$

It follows that we should first find the complementary distribution of service completion time $F_{S_1}(x)$ and the mean server utilization cost $\mathbb{E}W_1$ for any single task.

A. Single Task

At instant t_i, we switch on n_i servers that continue being utilized until the service completion time S_1 for a single task. Hence, the total cost of server utilization in terms of service completion times S_1 for single task is

$$W_1 = \lambda \sum_{i=0}^{m} n_i (S_1 - t_i)_+, \tag{6}$$

where $(x)_+ \triangleq \max\{x, 0\}$.

Let the time-interval $I_i \triangleq [t_i, t_{i+1})$ and we define $t_{m+1} = \infty$. Clearly, the disjoint intervals I_i partition the positive reals and any $t \in \mathbb{R}_+$ belongs to a unique interval I_i for some $i \in [m]_0 = \{0, 1, \ldots, m\}$. Let $t \in I_i$, then we have n_ℓ servers switched on at time t_ℓ for $l \leqslant i$. The event that the service completion time is longer than duration t is identical to the event that none of the servers started before this time t have finished until this time t. Let $T_{\ell,p}$ denote the service completion time for the pth server started at time t_ℓ, then for time $t \in I_i$ we can write

$$\begin{aligned} P\{S_1 > t\} &= P \bigcap_{\ell=0}^{i} \{ \min_{p \in [n_\ell]} (T_{\ell,p} + t_\ell) > t \} \\ &= P \bigcap_{\ell=0}^{i} \bigcap_{p \in [n_\ell]} \{T_{\ell,p} > t - t_\ell\}. \end{aligned}$$

From the *i.i.d.* service completion time for all servers, we can write the complementary distribution function of service completion time S_1 as

$$\bar{F}_{S_1}(t) = \prod_{\ell=0}^{i} \bar{F}(t - t_\ell)^{n_\ell}, \quad t \in I_i. \tag{7}$$

For a single task, we have $N_i \triangleq \sum_{\ell=0}^{i} n_\ell$ servers working in parallel during the interval $[t_i, t_{i+1})$. If the task is unfinished until time t_i, then n_ℓ servers switched on at instant $t_\ell < t_i$ have been working on this task since then. Hence the server utilization until time t_i is denoted by

$$\tau_i \triangleq \sum_{\ell=0}^{i} n_\ell (t_i - t_\ell). \tag{8}$$

Shifted exponential distribution of server completion time T_n defined in (1), is akin to a constant start-up time c for the server after which the random service time $T_n - c$ is distributed exponentially with rate μ. Hence, the servers switched on at time instant t_i only begin the random part of the service at time $t_i + c$. Accordingly, we define shifted intervals $\tilde{I}_i \triangleq [t_i + c, t_{i+1} + c) = c + I_i$ where N_i servers are working in parallel. In the following, we use the notation $[m]_0 = \{0, 1, \ldots, m\}$.

Lemma 1: Consider a single task being served by N servers started sequentially at times $(t_j : j \in [m]_0)$ in batches of $(n_j : j \in [m]_0)$. When the job completion time for each server has an i.i.d. *shifted exponential distribution as defined in* (1)*, then the complementary distribution of service completion time for a single task is given by*

$$\bar{F}_{S_1}(t) = e^{(-\mu N_i (t - t_i - c) - \mu \tau_i)}, \quad t \in \tilde{I}_i. \tag{9}$$

Proof: Let $t \in \tilde{I}_i$, then from the definition of service completion time, we can write

$$P\{S_1 > t\} = P \bigcap_{\ell=0}^{i} \bigcap_{p=1}^{n_\ell} \{T_{\ell,p} > c + (t - t_\ell - c)\}.$$

Since the job completion time at each server is *i.i.d.* with the common shifted exponential distribution defined in (1), we get

$$P\{S_1 > t\} = \exp(-\mu \sum_{\ell=0}^{i} n_\ell (t - t_\ell - c)), \quad t \in \tilde{I}_i.$$

The result follows from the definition of τ_i from equation (8), and the definition of aggregate number of forked servers $N_i = \sum_{\ell=0}^{i} n_\ell$ at ith forking time t_i. □

Lemma 2: Consider a single task being served by N servers started sequentially at times $(t_j : j \in [m]_0)$ in batches of $(n_j : j \in [m]_0)$. When the job completion time for each server has an i.i.d. *shifted exponential distribution as defined in* (1)*, then the mean server utilization cost is given by*

$$\mathbb{E}W_1 = \lambda \sum_{i=0}^{m} n_i \int_{t_i}^{t_i + c} \bar{F}_{S_1}(t) dt + \lambda \sum_{i=0}^{m} N_i \int_{\tilde{I}_i} \bar{F}_{S_1}(t) dt. \tag{10}$$

Proof: From the equation (6) for the service utilization cost for a single task, the linearity of expectations, and positivity of random variables $(S_1 - t_i)_+$, we can write the mean server utilization cost as

$$\mathbb{E}W_1 = \lambda \sum_{i=0}^{m} n_i \mathbb{E}(S_1 - t_i)_+ = \lambda \sum_{i=0}^{m} n_i \int_{t_i}^{\infty} \bar{F}_{S_1}(t) dt.$$

We can write the integral over $[t_i, \infty)$ as the sum of integrals over its partition $\{[t_i, t_i + c), \tilde{I}_i, \tilde{I}_{i+1}, \ldots, \tilde{I}_m\}$. Exchanging summations over indices $i \in [m]_0$ and $j \geqslant i$, we get the result. □

For general m, there is no straightforward way to evaluate the integral $\int_{t_i}^{t_i + c} \bar{F}_{S_1}(t) dt$ when $t_{i+1} - t_i \in (0, c)$. This is because the integration has to account for the servers started between t_i and $t_i + c$, which makes the integral evaluation cumbersome. For simplicity, we stick with the case when $t_{i+1} - t_i \geqslant c$ for all $i \in [m]_0$. The results for $\mathbb{E}S$ and $\mathbb{E}W$ in this case will be provided in Corollary 2. However for the single forking case when $m = 1$, we will derive the results when $t_1 - t_0 > 0$ and not necessarily larger than c in Section III-C.

B. Parallel Tasks

Next, we find the mean of service completion time and the mean of server utilization cost for K parallel tasks on

N servers each, using the complementary service distribution $\bar{F}_{S_1}$ for a single task, defined in (9). Formally, we describe our setup below.

Problem 1: Consider K parallel tasks, where each single task is being served by N servers starting in batches of $(n_j : j \in [m]_0)$, sequentially at times $(t_j : j \in [m]_0)$ such that the total number of servers is N and timing thresholds are at least c distance apart. That is, we have the following constraints, $t_0 = 0, t_{m+1} = \infty$, and

$$\sum_{j=0}^{m} n_j = N, \qquad t_{j+1} - t_j \geqslant c,\ j \in [m]_0.$$

When the job completion time for each server has an *i.i.d.* shifted exponential distribution as defined in (1), find the mean of the service completion time to finish all K parallel tasks and the mean of the server utilization cost.

The time evolution of number of active replicas for a single task S_k is illustrated in Fig. 1. When a task is completed from any replica, the number of active replicas for that task becomes zero. The overall service completion time S of the K tasks is the maximum of the completion of each of the K tasks, i.e. $S = \max_{k\in[K]} S_k$. We need the following Lemma to evaluate the mean service completion time.

Lemma 3: We can write the following integrals for complementary distribution of service completion times. For $i \in [m]_0$, we have

$$\int_{t\in\tilde{I}_i} \bar{F}_S(t)dt = -\frac{1}{N_i\mu}\sum_{k=1}^{K}\binom{K}{k}\frac{(-1)^k}{k}\left(e^{-k\mu\tau_i} - e^{-k\mu\tau_{i+1}}\right). \tag{11}$$

For $1 \leqslant i \leqslant m$, we can write

$$\int_{t_i}^{t_i+c} \bar{F}_S(t)dt = -\sum_{k=1}^{K}\binom{K}{k}\frac{(-e^{-\mu\tau_i})^k}{kN_{i-1}\mu}\left(e^{k\mu N_{i-1}c} - 1\right), \tag{12}$$

where the total number of active servers in interval $\tilde{I}_i$ is $N_i = \sum_{\ell=0}^{i} n_\ell$ and server utilization until time $t_i + c$ is $\tau_i = \sum_{\ell=0}^{i} n_\ell(t_i - t_\ell)$.

Proof: From the fact that $F_S(x) = F_{S_1}^K(x)$ and the binomial expansion of $(1-x)^K$, we can write

$$\bar{F}_S(t) = 1 - (1 - \bar{F}_{S_1}(t))^K = -\sum_{k=1}^{K}\binom{K}{k}(-1)^k\bar{F}_{S_1}^k(t).$$

Using the definition of single task service distribution in (9) and definitions of N_i and τ_i, we can integrate $\bar{F}_{S_1}^k(t)$ over interval $\tilde{I}_i$, to get

$$\int_{t\in\tilde{I}_i} \bar{F}_{S_1}^k(t)dt = \frac{1}{kN_i\mu}(e^{-k\mu\tau_i} - e^{-k\mu\tau_{i+1}}).$$

To integrate $\bar{F}_{S_1}^k(t)$ over the interval $[t_i, t_i+c)$, we notice that $[t_i, t_i+c) \subseteq \tilde{I}_{i-1}$ since $t_{i-1}+c \leqslant t_i$ by hypothesis. Therefore, we can write

$$\int_{t_i}^{t_i+c} \bar{F}_{S_1}^k(t)dt = \frac{1}{kN_{i-1}\mu}(e^{-k\mu(\tau_i - N_{i-1}c)} - e^{-k\mu\tau_i}).$$

The result follows from combining the above expressions. □

Corollary 1: We can futher simplify the above integrals for complementary distribution of service completion times of Lemma 3. For integers $0 \leqslant i \in m$, we have

$$\int_{t\in\tilde{I}_i} \bar{F}_S(t)dt = \frac{1}{N_i\mu}\sum_{k=1}^{K}\frac{1}{k}\left((1-e^{-\mu\tau_{i+1}})^k - (1-e^{-\mu\tau_i})^k\right). \tag{13}$$

For $1 \leqslant i \leqslant m$, we can write

$$\int_{t_i}^{t_i+c} \bar{F}_S(t)dt = \frac{(e^{\mu N_{i-1}c} - 1)}{N_{i-1}\mu}\sum_{k=1}^{K}\frac{1}{k}(1 - (1 - e^{-\mu\tau_i})^k). \tag{14}$$

Proof: We define the following integrals as a function of number of tasks

$$h_1(K) = \int_{t\in\tilde{I}_i} \bar{F}_S(t)dt, \qquad h_2(K) = \int_{t=t_i}^{t_i+c} \bar{F}_S(t)dt.$$

We next observe the following identity for binomial coefficients

$$\frac{1}{k}\binom{K}{k} = \frac{1}{k}\binom{K-1}{k} + \frac{1}{K}\binom{K}{k},\ k \in [K].$$

Multiplying with a geometric term in k and summing over all $k \in [K]$, we get

$$-\sum_{k=1}^{K}\binom{K}{k}\frac{\alpha^k}{k} = -\sum_{k=1}^{K-1}\binom{K-1}{k}\frac{\alpha^k}{k} + \frac{1-(1+\alpha)^K}{K}.$$

Hence, we conclude that

$$h_2(K) = h_2(K-1) + \frac{(1-e^{-\mu\tau_i})^K - (1-e^{-\mu(\tau_i - N_{i-1}c)})^K}{KN_{i-1}\mu},$$

$$h_1(K) = h_1(K-1) + \frac{(1-e^{-\mu\tau_{i+1}})^K - (1-e^{-\mu\tau_i})^K}{KN_i\mu}.$$

The results follow by taking the summation of $h_1(k)$ and $h_2(k)$ over $k \in [K]$ with initial conditions $h_1(0) = h_2(0) = 0$. □

Now, we have all the necessary results to compute the means of service completion time and cost server utilization for K parallel tasks.

Theorem 1: For the Problem 1, the mean service completion time is

$$\mathbb{E}S = c + \frac{1}{\mu}\sum_{k=1}^{K}\frac{1}{k}\left(\frac{1}{N_m} + \sum_{i=1}^{m}\frac{n_i}{N_iN_{i-1}}(1-e^{-\mu\tau_i})^k\right), \tag{15}$$

and the mean server utilization cost is

$$\mathbb{E}W_1 = \lambda c n_0 + \frac{\lambda}{\mu} + \frac{\lambda}{\mu}\sum_{i=1}^{m} n_i e^{-\mu\tau_i}\left(\frac{e^{\mu N_{i-1}c} - 1}{N_{i-1}}\right). \tag{16}$$

Proof: We will first find the mean server utilization cost for single task. From (10), we have

$$\frac{1}{\lambda}\mathbb{E}W_1 = n_0\int_{t_0}^{t_0+c}\bar{F}_{S_1}(t)dt + \sum_{i=1}^{m} n_i\int_{t_i}^{t_i+c}\bar{F}_{S_1}(t)dt + \sum_{i=0}^{m} N_i\int_{\tilde{I}_i}\bar{F}_{S_1}(t)dt.$$

First, we notice that $\int_{t_0}^{t_0+c} \bar{F}_{S_1}(t)dt = c$ since $t_0 = 0$ and there is initial startup delay of c for all shifted exponential job completion times. Taking $K = 1$, and substituting equation (11) for integers $0 \leqslant i \leqslant m$ and equation (12) for integer $1 \leqslant i \leqslant m$, in the above equation, we get

$$\frac{1}{\lambda}\mathbb{E}W_1 = n_0 c + \frac{1}{\mu}\sum_{i=1}^{m} \frac{n_i e^{-\mu\tau_i}}{N_{i-1}}(e^{\mu N_{i-1}c} - 1) + \frac{1}{\mu}\sum_{i=0}^{m}(e^{-\mu\tau_i} - e^{-\mu\tau_{i+1}}).$$

The result for mean server utilization cost follows from the telescopic sum and the fact that $\tau_0 = 0, \tau_{m+1} = \infty$.

To compute the mean of service completion time S, we use its positivity to write $\mathbb{E}S = \int_{\mathbb{R}_+} \bar{F}_S(t)dt$. By writing the integral over positive reals, as the sum of integrals over the partition $\{[0, t_0 + c), \tilde{I}_0, \tilde{I}_1, \ldots, \tilde{I}_m\}$, we get

$$\mathbb{E}S = \int_0^{t_0+c} \bar{F}_S(t)dt + \sum_{i=0}^{m}\int_{\tilde{I}_i} \bar{F}_S(t)dt.$$

Substituting the fact that $t_0 = 0, \tau_0 = 0, \tau_{m+1} = \infty$, $\int_0^c \bar{F}_{S_1}(t) = c$, and equation (13) in the above equation, followed by exchanging summations over indices k and i, we get the result. □

As a special case of Theorem 1, we can obtain the mean service completion time and the mean server utilization cost for a single task, as is given in the following corollary.

Corollary 2: For a single task served by N servers with multiple forks, the mean service completion time is

$$\mathbb{E}S = c + \frac{1}{N\mu} + \frac{1}{\mu}\sum_{i=1}^{m} \frac{n_i}{N_i N_{i-1}}(1 - e^{-\mu\tau_i}), \qquad (17)$$

and the mean server utilization cost for single task is

$$\mathbb{E}W = \lambda c n_0 + \frac{\lambda}{\mu} + \frac{\lambda}{\mu}\sum_{i=1}^{m} n_i e^{-\mu\tau_i}\left(\frac{e^{\mu N_{i-1}c} - 1}{N_{i-1}}\right). \quad (18)$$

We show that making the forking instants smaller and increasing number of servers at any forking instant can reduce the service completion time, irrespective of the common service time distribution.

Proposition 1: For K parallel tasks, each forked sequentially on N identical servers with random i.i.d. *execution times with the common distribution function F, the following statements are true.*

- *(i) Consider two increasing sequences of forking times $t = (t_0, \ldots, t_m)$ and $t' = (t'_0, \ldots, t'_m)$ each with identical sequence of forked replicas such that $t'_i \geqslant t_i$ at each stage $0 \leqslant i \leqslant m$. Then $\mathbb{E}S^{(t)} \leqslant \mathbb{E}S^{(t')}$.*
- *(ii) Consider sequences of forked replicas $n = (n_0, \ldots, n_m)$ and $n' = (n'_0, \ldots, n'_m)$ with identical sequence of forking instants $t = (t_0, \ldots, t_m)$ such that $n'_j \leqslant n_j$ for stages $0 \leqslant j \leqslant m$. Then $\mathbb{E}S^{(n)} \leqslant \mathbb{E}S^{(n')}$.*

Proof: The detailed proof is given in Appendix A, which uses stochastic dominance. □

Second condition in the above theorem is very strict in that for a fixed forking time sequence t, the two forked replica sequence is such that the number of forked replicas at each forking time are always larger for one sequence. We would like the theorem to hold for the following weaker condition: for a fixed forking time sequence t and the two server sequences n, n' such that the cumulative number of server sequences $N \leqslant N'$ are point-wise ordered. Notice that, in this case we would have to use specific properties of the service-time distribution at each server, and it links the forking instant sequence and the server sequence. In the following result, we will show that the result could be refined for the shifted exponential distribution.

Theorem 2: Let there be K parallel tasks, each forked sequentially on N identical servers with random i.i.d. *execution times with the common distribution function F being the shifted exponential as defined in* (1)*. Consider sequences of forked replicas $n = (n_0, \ldots, n_m)$ and $n' = (n'_0, \ldots, n'_m)$ with identical sequence of forking instants $t = (t_0, \ldots, t_m)$ such that for each stage $0 \leqslant i \leqslant m$,*

$$\sum_{j=0}^{i} n'_j \leqslant \sum_{j=0}^{i} n_j, \text{ and } \sum_{j=0}^{i} n'_j t_j \geqslant \sum_{j=0}^{i} n_j t_j.$$

Then $\mathbb{E}S^{(n)} \leqslant \mathbb{E}S^{(n')}$.

Proof: Following the arguments in Theorem 1, it suffices to show the monotonicity of the complementary distribution function of service times for single task. It follows from the theorem hypothesis that $N_i = \sum_{\ell=0}^{i} n_\ell \geqslant \sum_{\ell=0}^{i} n'_\ell = N'_i$ and $\tau'_i = \sum_{\ell=0}^{i} n'_\ell(t_i - t_\ell) \leqslant \sum_{\ell=0}^{i} n_\ell(t_i - t_\ell) = \tau_i$ for all stages $i \in [m]_0$. Therefore, for any time $u \in \tilde{I}_i$,

$$\bar{F}_{S_1^{(n)}}(u) = e^{-\mu N_i(u - t_i - c) - \mu\tau_i} \leqslant e^{-\mu N'_i(u - t_i - c) - \mu\tau'_i} = \bar{F}_{S_1^{(n')}}(u).$$

Hence, the result follows. □

Remark 1: For single-fork case starting with forking points $0 = t_0 < t_1$, the condition $\sum_{j=0}^{i} n'_j t_j \geqslant \sum_{j=0}^{i} n_j t_j$ in Theorem 2 reduces to $n'_1 \geqslant n_1$. Hence, if both the systems have identical number of servers, i.e. $n_0 + n_1 = n'_0 + n'_1$, then $n'_0 \leqslant n_0$, and both the theorem conditions hold.

C. Single Forking Parallel Tasks

We consider the single forking case for K parallel tasks when $m = 1$ and $t_1 > 0$. Formally, we define the problem below.

Problem 2: Consider K parallel tasks, where each single task is being served by N servers starting in two batches of (n_0, n_1), sequentially at times $(0, t_1)$ such that the total number of servers is $N = n_0 + n_1$. When the job completion time for each server has an *i.i.d.* shifted exponential distribution as defined in (1), find the mean of the service completion time to finish all K parallel tasks and the mean of the server utilization cost.

Since $n_1 = N - n_0$, we have only two variables n_0 and t_1 in this case. Further, we have $t_2 = \infty$ and we can write $\tau = n_0 t_1$. For the ease of further analysis, we would define following normalized constants. We define the amount of work done by all servers $N_1 = N$ in parallel each having independent random execution time distributed exponentially with rate μ

in the shift-interval c as $\alpha \triangleq c\mu N$. We denote the normalized forking time by $u \triangleq t_1/c$ and the initial fraction of servers by $x \triangleq n_0/N$.

Theorem 3: For the Problem 2, the scaled mean service completion time is

$$\frac{1}{c}\mathbb{E}S = 1 + \frac{1}{\alpha}\sum_{k=1}^{K}\frac{1}{k}\left(1 + \frac{1-x}{x}(1-e^{-\alpha x u})^k\right), \quad (19)$$

and the scaled and shifted mean server utilization cost $\frac{\mu}{\lambda}\mathbb{E}W_1 - (1+\alpha)$ *for single task equals*

$$\begin{cases} \alpha(1-x)\left(\dfrac{e^{-\alpha x(u-1)} - e^{-\alpha x u)}}{\alpha x} - 1\right), & u \geqslant 1, \\ \alpha(1-x)\left(\dfrac{(1-e^{-\alpha x u})}{\alpha x} - u\right), & u \leqslant 1. \end{cases} \quad (20)$$

Proof: The result for the mean service completion time $\mathbb{E}S$ can be obtained by substituting $m = 1$ in the equation (15). To compute the mean server utilization cost for $t_1 \geqslant c$, we substitute $m = 1$ in the equation (16). For $t_1 < c$, we need to evaluate the integral $\int_{t_1}^{t_1+c} \bar{F}_{S_1}^k(t)dt$. In this case, we have

$$\int_{t_1}^{t_1+c} \bar{F}_{S_1}(t)dt = \int_{t_1}^{c} \bar{F}_{S_1}(t)dt + \int_{c}^{t_1+c} \bar{F}_{S_1}(t)dt.$$

Since $\bar{F}_{S_1}(t) = 1$ for $t \leqslant c$ due to initial startup delay c, and there are n_0 parallel independent servers working at the exponential rate μ in the interval $[t_1, t_1 + c)$, we have

$$\int_{t_1}^{t_1+c} \bar{F}_{S_1}(t)dt = c - t_1 + \frac{1}{n_0\mu}(1 - e^{-\mu n_0\ t_1}).$$

The result follows from aggregating both the cases. □

IV. Optimal Single Forking

We have the expression for mean of service completion and server utilization for single forking case in Theorem 3. We study the impact of forking time and initial number of servers on these two performance metrics.

Proposition 2: Consider the single forking for K parallel tasks, each forked sequentially over N parallel servers, each forked task having i.i.d. *random service times with the common shifted exponential distribution with shift c and rate μ.*

The partial derivative of the mean service completion time with respect to normalized forking time u is

$$\frac{\partial \mathbb{E}S}{\partial u} = c(1-x)(1-(1-e^{-\alpha x u})^K).$$

The partial derivative of the mean service completion time with respect to the initial fraction of servers x is

$$\frac{\partial \mathbb{E}S}{\partial x} = -\frac{c}{\alpha x^2}\sum_{k=1}^{K}\frac{1}{k}(1-e^{-\alpha x u})^k + \frac{u}{x}\frac{\partial \mathbb{E}S}{\partial u}.$$

The partial derivative of the mean server utilization cost with respect to the normalized forking time u is

$$\frac{\partial \mathbb{E}W_1}{\partial u} = \begin{cases} -\dfrac{\lambda}{\mu}\alpha(1-x)e^{-\alpha x u}(e^{\alpha x} - 1), & u \geqslant 1, \\ -\dfrac{\lambda}{\mu}\alpha(1-x)(1-e^{-\alpha x u}), & u \leqslant 1. \end{cases}$$

The scaled partial derivative $\frac{\mu}{\alpha\lambda}\frac{\partial \mathbb{E}W_1}{\partial x}$ *of the mean server utilization cost with respect to the initial fraction of servers x equals*

$$\begin{cases} 1 - e^{-\alpha x u}\left[(\dfrac{1}{x}-1)((u-1)e^{\alpha x} - u) + \dfrac{(e^{\alpha x}-1)}{\alpha x^2}\right], & u \geqslant 1 \\ u + (\dfrac{1}{x}-1)ue^{-\alpha x u} - \dfrac{1}{\alpha x^2}(1-e^{-\alpha x u}), & u \leqslant 1. \end{cases}$$

Proof: Results follow by taking partial derivatives of mean server utilization and mean service completion task with respect to normalized forking time u and initial fraction of servers x. □

Even though the initial fraction of servers x lie in the set $\{\frac{1}{N}, \ldots, 1\}$, we approximate it by a real number $x \in [\frac{1}{N}, 1]$ to get insight on the dependence of the above two performance metrics on this fraction.

Theorem 4: The following statements are true for the single forking problem.

(i) The mean service completion time is an increasing function of forking time t_1.

(ii) The mean service completion time is a decreasing function of initial fraction x.

(iii) The mean server utilization cost is a decreasing function of forking time t_1.

(iv) There exists a unique optimal initial fraction of servers $x^* \in [\frac{1}{N}, 1]$ that minimizes the mean server utilization cost. For normalized forking time $u \geqslant v_3$, the optimal initial fraction is $x^* = 1/N$. For normalized forking time $u < v_3$, the optimal initial fraction is the unique solution to the following implicit equation, where $e^{\alpha x u}$ equals

$$\begin{cases} (\dfrac{1}{x}-1)((u-1)e^{\alpha x} - u) + \dfrac{(e^{\alpha x}-1)}{\alpha x^2}, & u \in [1, v_3), \\ -(\dfrac{1}{x}-1) + \dfrac{1}{\alpha x^2 u}(e^{\alpha x u} - 1), & u < v_3 \wedge 1, \end{cases} \quad (21)$$

where the normalized forking point threshold v_3 is the unique solution to the implicit equation, where $\frac{c\mu}{N}e^{c\mu v_3} + \frac{(N-1)c\mu}{N}$ *equals*

$$\begin{cases} \left(\dfrac{c\mu(N-1)(v_3-1)}{N} + 1\right)(e^{c\mu} - 1), \\ (1 - \dfrac{c\mu}{N})\dfrac{(e^{c\mu}-1)}{c\mu} > 1, \\ \dfrac{1}{v_3}(e^{c\mu v_3} - 1), \\ (1 - \dfrac{c\mu}{N})\dfrac{(e^{c\mu}-1)}{c\mu} \leqslant 1. \end{cases}$$

Proof: The proof is provided in Appendix B. □

V. Numerical Studies: Single Forking

We numerically evaluate the behavior of mean service completion time and mean server cost utilization for single forking below, with total number of servers $N = 12$ for each of the $K = 10$ parallel tasks, taking $\lambda = 1$. We have analytically studied the case when service time at each server is an *i.i.d.* random variable having a shifted exponential distribution, and we present the numerical studies for the single forking case with shifted exponential distribution. We note that the insights

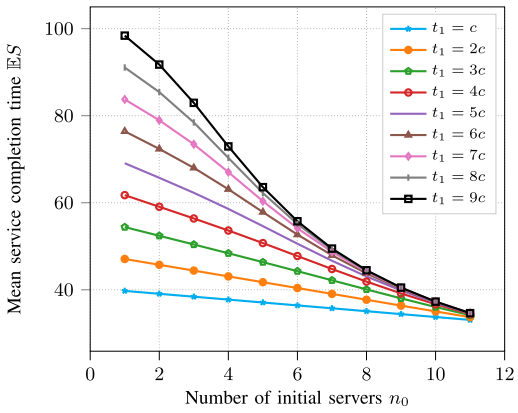

Fig. 2. Mean service completion time $\mathbb{E}S$ as a function of initial number of servers $n_0 \in \{1, \ldots, 11\}$ for single forking of $K = 10$ parallel tasks at different forking times $t_1 \in \{c, 2c, \ldots, 9c\}$.

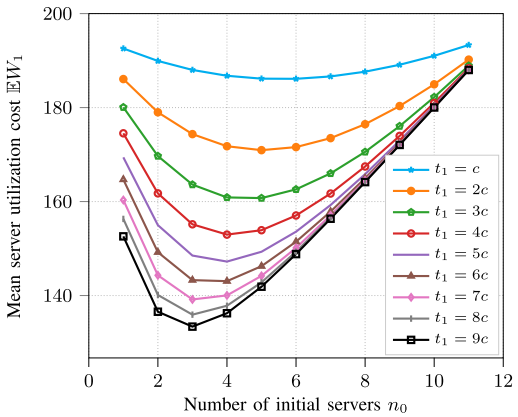

Fig. 3. Mean server utilization cost $\mathbb{E}W$ as a function of initial number of servers $n_0 \in \{1, \ldots, 11\}$ for single forking of $K = 10$ parallel tasks at different forking times $t_1 \in \{c, 2c, \ldots, 9c\}$.

obtained from this study hold for heavy-tailed distributions such as Pareto and Weibull distribution as well, and the supporting numerical results are presented in Appendix F-A and Appendix F-B.

We have taken the job completion times at each server to be an *i.i.d.* random variable having a shifted exponential distribution with the shift parameter $c = 8$ and the exponential rate $\mu = 0.01$. From the discussion in Appendix E-C, we observe that $c\mu = 0.08 < N - 2 = 10 < x'$, and hence $cv_3 \geqslant 1$. Specifically, we can numerically compute the forking time threshold $v_3 \approx \frac{y}{c\mu} \approx 47$ where $e^y = 1 + 11y$ is satisfied by $y \approx 3.741$. That is, for any forking point $t_1 \leqslant 47c$, we can have optimal number of initial servers $n_0^* \geqslant 1$. For the given system parameters, we plot the mean service completion time in Fig. 2 and mean server utilization in Fig. 3 as a function of initial servers $n_0 \in \{1, \ldots, 11\}$ for values of forking times in $t_1 \in \{c, 2c, \cdots, 9c\}$. We corroborate the analytical results obtained in Theorem 4, by observing that mean service completion time increases and the mean server utilization cost decreases with increase in the forking time t_1. We further observe that the optimal number of initial servers $n_0^* \geqslant 1$ for mean server utilization cost for different values of forking time t_1. In addition, we notice the decrease in the mean service completion time as the number of initial server n_0 increases.

These results point to an interesting tradeoff between the two metrics. First observation is that forking time gives a true tradeoff between these two metrics. Second and more interesting observation is that there exist a minimum number of initial servers for each forking time, until which point we can decrease both the mean service completion time and the mean server utilization cost. This also points to the sub-optimality of single-forking with unit server in [8].

The authors of [8] considered a single fork analysis where at $t = 0$, one copy of the task is started and when pn jobs are complete, each unfinished job is replicated r times. The analysis considered two possibilities, one where the currently running job is kept running at the forking point and second where it is killed. It was shown that keeping the currently running job performed better for both mean service completion time and mean server utilization cost, when the service distribution is shifted exponential. We compare our results with the baseline results obtained in the [8, Theorem 2] for the case when the straggler job is kept running at the forking point. We restate the above-mentioned Theorem, adapted to our notation, for easy reference.

Lemma 4: [8, Theorem 2] Consider K parallel computing tasks, each started on a single server each, i.e. $t_0 = 0, n_0 = 1$. If r replicas of each unfinished task are started, after $(1-p)K$ tasks are completed, and the execution time of each task is assumed to be i.i.d. *ShiftedExp(c, μ), then the mean service completion time and the mean server utilization cost metrics for $K \to \infty$, are*

$$\mathbb{E}S = \frac{2r+1}{r+1}c + \frac{1}{(r+1)\mu}(\ln K - r \ln p + \gamma_{EM})$$
$$\mathbb{E}W = c + \frac{1}{\mu} + pc + pr\frac{(1-e^{-\mu c})}{\mu},$$

where $\gamma_{EM} \approx 0.577$ is the Euler-Mascheroni constant.

Though our model is quite different than the one studied here, we will make broad comparisons. We let $n_0 = 1$ for this model and let t_1 to be the mean time to finish $(1-p)K$ tasks with K parallel servers working at rate μ. Then

$$\mu K(t_1 - c) \approx K(1-p).$$

Further, at instant t_1, we have $n_1 = N - 1 = rp$ new servers being started per job. Therefore, we can take the forking point to be $t_1 = c + \frac{(1-p)}{\mu}$ and the total number of servers to be $N = 1 + rp$. Given total number of available servers N and forking time t_1, we can compute the fraction of completed tasks $p = 1 - \mu(t_1 - c)$ and the number of replicas $r = \frac{(N-1)}{p}$. In Fig. 4, we have plotted the mean of service completion times with respect to mean server utilization cost when $\lambda = 1$ for the single forking proposed in [8] as the baseline curve and our proposed single forking varying the initial number of servers $n_0 \in \{1, \ldots, 11\}$, when the forking time $t_1 \in \{2c, 4c, 6c, 8c\}$. We see that our trade-off curves are well inside the baseline curve. Specifically, we observe significant reduction in the mean server utilization cost for optimal server initialization when compared to single-server initialization of [8], for the identical mean service completion time in both the cases.

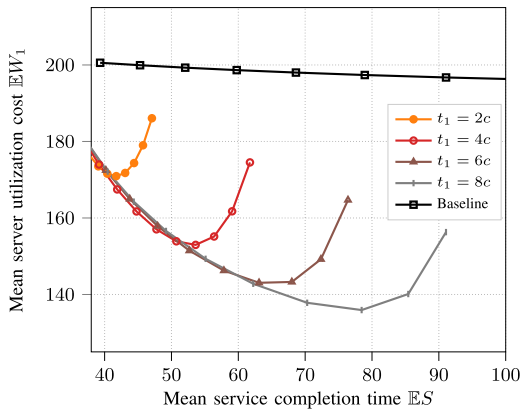

Fig. 4. Mean server utilization cost $\mathbb{E}W$ as a function of mean service completion time $\mathbb{E}S$ when we vary the number of initial servers $n_0 \in \{1, \ldots, 11\}$ for single forking of $K = 10$ parallel tasks at different forking times $t_1 \in \{2c, 4c, 6c, 8c\}$. The service distribution of each replica is assumed to be *i.i.d.* shifted exponential with shift parameter $c = 8$ and exponential rate $\mu = .01$. We have plotted the same curve for initial servers $n_0 = 1$ for single forking for $K = 10$ parallel tasks for these different values of forking times t_1, for the baseline (r, p) model where $r = \frac{(N-1)}{p}$ and $p = 1 - \mu(t_1 - c)$.

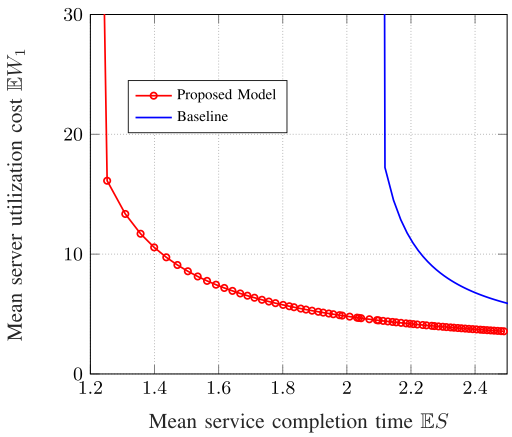

Fig. 5. Tradeoff between mean service completion time and mean server utilization cost, obtained by changing the value of β. The service distribution of each replica is assumed to be *i.i.d.* shifted exponential with shift parameter $c = 1$ and exponential rate $\mu = 1$.

VI. Numerical Studies: Multiple Forking

In a multi-forking scenario the free variables are number of forked servers $(n_0, \ldots, n_{m-1}, n_m)$ under the constraint of finite number of servers N per task, i.e. $\sum_{i=0}^{m} n_i = N$, and the forking instants $(t_0 = 0, t_1, \ldots, t_m)$. It is a multi-dimensional optimization problem and not easy to evaluate. The single forking results in Section IV leads us to believe that even for the general case of multiple forking points with *i.i.d.* execution times having shifted exponential distribution, there should be a tradeoff between the two metrics of mean server utilization cost $\mathbb{E}W$ and the mean service completion time $\mathbb{E}S$. We attempt two approaches to understand this tradeoff.

A. Joint Cost for Large N

To explore this tradeoff, we formulate the joint optimization in terms of a tradeoff parameter β as

$$\begin{aligned} MP: \quad & \min \mathbb{E}S + \beta \mathbb{E}W \\ & \text{such that } (15), (16), t_0 = 0 \\ & \text{variables } n_0, \cdots n_m, t_0, \cdots, t_m \end{aligned} \tag{22}$$

We note from Fig. 4 that based on the value of β, the tradeoff point chosen will be different. Thus, finding the forking instants and the number of servers added at each forking point, are important. For the optimization problem, we chose the total number of servers N, to be unbounded. For $(n_0, \ldots, n_m)$ an integer sequence, the above problem is a mixed-integer programming problem, and known to be hard. As such, we relax the integer constraints and allow n_i to be real valued, in which case the problem reduces to a linear programming problem and can be solved using interior point algorithm [24]. We round off the values of n_i to nearest integers to get a heuristic integral solution.

For this multi-objective optimization defined in (22), changing the value of β provides a tradeoff between the two metrics. For numerically solving the multi-objective optimization, we take the parameters of shifted exponential distribution as shift $c = 1$ and service rate $\mu = 1$, the server utilization cost per unit time $\lambda = 1$, the number of parallel tasks $K = 25$, and the number of forking points $m = 4$. We depict the tradeoff between mean service completion time and mean server utilization cost for the proposed heuristic algorithm in Fig. 5.

We compare the performance of multi-forking obtained by the proposed heuristic algorithm to the baseline single-forking approach proposed in [8]. We can compute the linear cost of the optimization problem in (22) for any tradeoff parameter β, by obtaining the mean service completion time and the mean server utilization cost from Lemma 4. Fig. 5 shows a significant improvement of the proposed model as compared to that in [8] for the tradeoff between the two metrics. For a service completion time lower than 2, there is significant reduction in the mean server utilization cost, thus showing the huge savings that the multi-forking can provide. The performance gains are due to two factors, initializing the task on multiple servers at time $t = 0$ and multi-forking. We observed that the improvement due to multi-forking was small in this setting and the corresponding tradeoff curve for single forking looks very similar, and hence we do not provide the tradeoff curve for single-forking in this setting.

We note that the lowest mean service completion point in Fig. 5 corresponds to starting large number of servers at $t = 0$ since having large number of servers at $t = 0$ achieves the lowest completion time. However, if the number of total servers is bounded by a number N as is the case in our single forking analysis, the points on the very left in the mean service completion time may not be achievable. In other words, the curve will get truncated on the left side with an upper bound on N.

B. Comparison With Optimal Single Forking

In general, finding the optimal forking points and the corresponding number of servers to be forked at each forking point, is not an easy task. In the following, we compare

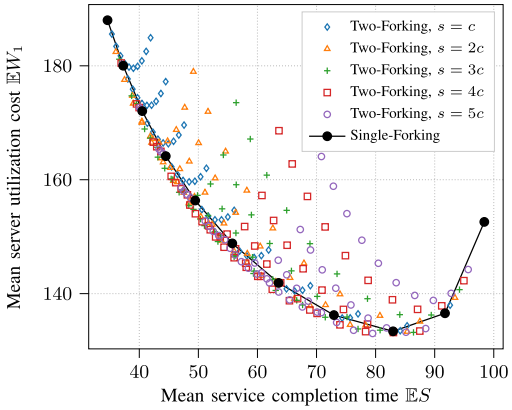

Fig. 6. This figure illustrates achievable points with shifted exponential execution times for mean server utilization cost and mean service completion time for two-forking with different values of forked servers m_0, m_1 and $N - m_0 - m_1$ at forking points $t_0 = 0$, $s \in \{c, 2c, 3c, 4c, 5c\}$ and $t = 9c$, respectively. For comparison, we also plot the tradeoff points of single-forking at forking time $t = 9c$ varying the number of initial servers n_0.

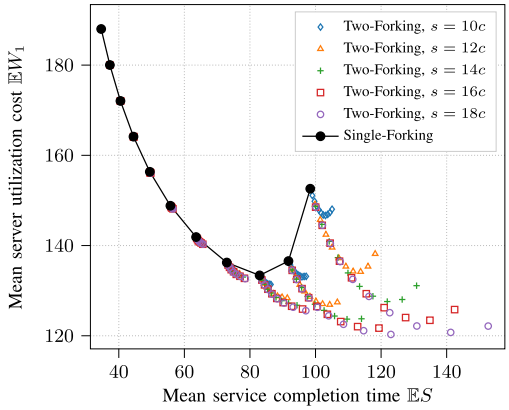

Fig. 7. This figure illustrates achievable points with shifted exponential execution times for mean server utilization cost and mean service completion time for two-forking with different values of forked servers m_0, m_1 and $N - m_0 - m_1$ at forking points $t_0 = 0$, $t = 9c$ and $s \in \{10c, 12c, \ldots, 18c\}$, respectively. For comparison, we also plot the tradeoff points for single-forking at forking time $t = 9c$ varying the number of initial servers n_0.

optimal single forking to sub-optimal two-forking to quantify potential gains of multi-forking for *i.i.d.* shifted exponential execution times. We assume the system parameters to be $c = 8, \mu = 0.01, K = 10, N = 12, \lambda = 1$. We consider two different setups for the comparison, depending on the location of the other forking with respect to t. The single forking can be thought of as two-forking with zero forked servers at this other forking point.

As a first case, we take the other forking point $s < t$. In this case, the single forking can be thought of as a two-forking sequence $((0, n_0), (s, 0), (t, N - n_0))$. For all possible values of $0 \leqslant m_0, m_1$ such that $m_0 + m_1 \leqslant N$, we consider two-forking sequences $((0, m_0), (s, m_1), (t, N - m_0 - m_1))$. We plot the tradeoff curve between mean service completion time and mean server utilization cost for the single and two-forking sequences in Fig. 6 for the values of forking points $t = 9c$ and $s \in \{c, 2c, 3c, 4c, 5c\}$, varying the number of forked servers $n_0 \in [N]$ in single-forking case and m_0, m_1 in two-forking case. We observe that for *some* feasible choice of forked servers m_0, m_1 and forking point $s < t$, the two-forking system achieves better tradeoff points as compared to the single-forking system.

For the other case, we take the second forking point $s > t$. In this case, the single forking can be thought of as a two-forking sequence $((0, n_0), (t, N - n_0), (s, 0))$. For all possible values of $0 \leqslant m_0, m_1$ such that $m_0 + m_1 \leqslant N$, we consider two-forking sequences $((0, m_0), (t, m_1), (s, N - m_0 - m_1))$. We plot the tradeoff curve between mean service completion time and mean server utilization cost for the single and two-forking sequences in Fig. 7 for the values of forking points $t = 9c$ and $s \in \{10c, 12c, \ldots, 18c\}$, varying the number of forked servers $n_0 \in [N]$ in single-forking case and m_0, m_1 in two-forking case. We observe that for *any* choice of forked servers m_0, m_1 and forking point $s > t$, the two-forking system achieves better tradeoff points as compared to the single-forking system.

Looking closely, we observe that setting the other forking point $s < t$ in two-forking can achieve better tradeoff points for the mean service completion time below a threshold. In contrast, setting the other forking point $s > t$ helps two-forking achieve significantly better tradeoff points when the mean service completion time is above that threshold. We also remark that at this threshold, the mean server utilization cost is minimum for single-forking. Hence, two-forking can further reduce the mean server utilization cost when compared to single-forking. Thus, an investigation of optimal forking points and the number of forked-servers at the different forking points is an important future research direction. We observe that the insights obtained for the shifted exponential distribution continue to hold for heavy-tailed distributions such as Pareto and Weibull as well, and the supporting numerical results are presented in Appendix G-A and Appendix G-B.

VII. Experiments on Intel DevCloud Servers

Intel DevCloud is a cloud computing service made available by Intel [25] for several profiles of researchers, students and professional engineers.[1] Intel DevCloud is a compute cluster, consisting of multiple servers called compute nodes, storage servers, and the login node. Each node has Intel Xeon processor of the Skylake architecture (Intel Xeon Scalable Processors family), an Intel Xeon Gold 6128 CPU, on-platform memory of 192 GB and a Gigabit Ethernet interconnect. To maximize the utilization of the compute cluster, one can submit jobs either by running Jupyter Notebook session on one of the compute nodes or by accessing the login node using an SSH client in a text-based terminal to a job queue dedicated to the authenticating account. For best performance, we created new environments with core Python 3 using Intel Distribution for Python. When a job is submitted to the queue, the scheduler picks the first available compute node for that job.

A. Setup

In our experiment, we reserved one node per job and submitted multiple single-node jobs at forking points by launching a distributed-memory parallel job explicitly requesting multiple

[1]The authors would like to thank Intel for giving us access to the cluster for this project.

compute nodes, which correspond to the servers on which the job is forked. This ensures that all the forked jobs start at the same forking time on the compute nodes to which the jobs are forked. Single node jobs are submitted through a job script file using the *qsub* command. We submitted a parallel job using the command *mpirun*, which launches the single node job at multiple nodes, by creating MPI program using Message Passing Interface (MPI) library called Intel MPI which is installed on all nodes. From here after, in this section, we refer *parallel job* to *replicated single-node jobs* which is *requesting multiple compute nodes at once*.

B. Objective

In this experimental set up, we have K jobs. For both single-forking and two-forking, we take each of the K jobs to be an identical algebraic computation with approximate mean completion time of 600 seconds. As a test-case, each algebraic job is taken as the repeated addition of two numbers in a loop, that runs 6×10^9 times. This section aims at answering the following questions.

1) Given K jobs, KN servers, and a forking mechanism, is it possible to get a tradeoff between the avearge server utilization cost and average service completion time on real cloud setup?
2) Are the tradeoff curves for this practical setup qualitatively similar to the one predicted by the analytical study for random execution times modeled to be distributed as a shifted exponential?

C. Experiment

To cater to this requirement, we initialized a parallel job requesting n_0 compute nodes at time $t_0 = 0$ for each of the K jobs. In the single-forking experiment, at time t_1 seconds, we initialized a parallel job requesting n_1 compute nodes for each of the unfinished jobs and waited for the completion. Similarly, in the two-forking experiment, at times t_1, t_2 seconds, we again initialized parallel jobs requesting n_1, n_2 compute nodes, respectively, for each of the unfinished jobs and waited for the completion. As soon as one of the replica of a ith job is finished we logged that time stamp S_i into a log file and killed the other replicas of that particular ith job immediately using the *qdel* command.

Using the observed job completion times (S_i) and forking time and server sequences, we compute the two performance metrics: the server utilization cost and the service completion time, by using the equations (6), (3), and (2) for each run $j \in [J]$ for $J = 1 \times 10^4$ runs. In addition, we also computed the empirical distribution of job completion times, which is plotted in Fig. 8.

D. Evaluations

We evaluate the single forking setup on Intel DevCloud with $K = 10$ parallel tasks and with each task being replicated on total number of $N = 12$ servers. We ran this experiment $J = 1 \times 10^4$ times for the given system parameters. For the kth task in jth run, we denote the service completion time and the server utilization cost by $S_k^{(j)}$ and $W_k^{(j)}$ respectively.

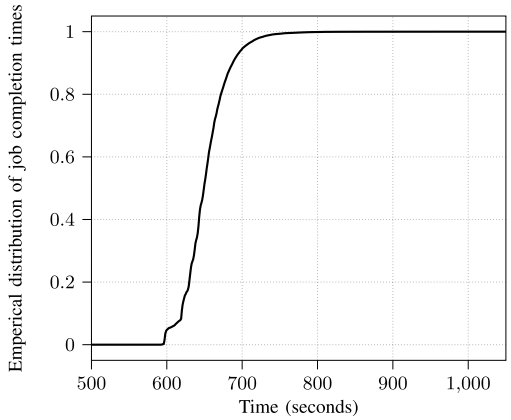

Fig. 8. This figure illustrates the empirical distribution of job completion times that are collected during the Intel DevCloud experiments. The job here is a algebraic computation of addition of two numbers, repeated 6×10^9 times.

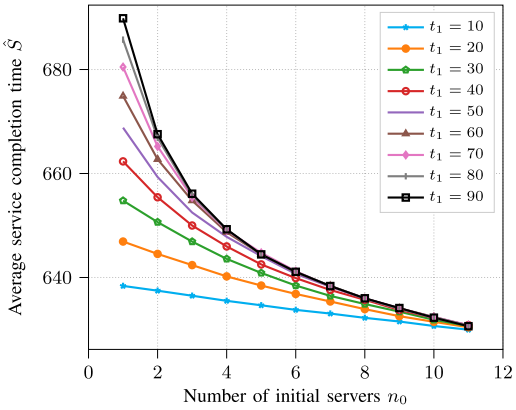

Fig. 9. Empirical average of service completion time $\hat{S}$ as a function of initial number of servers $n_0 \in \{1, \ldots, 11\}$ for single forking of $K = 10$ parallel tasks at different forking times $t_1 \in \{10, 20, \ldots, 90\}$ when jobs are executed on Intel DevCloud.

Hence, we computed the empirical average of service completion time and server utilization costs as

$$\hat{S} \triangleq \frac{1}{J} \sum_{j=1}^{J} \max_{k \in [K]} S_k^{(j)}, \quad \hat{W} \triangleq \frac{1}{J} \sum_{j=1}^{J} \frac{1}{K} \sum_{k=1}^{K} W_k^{(j)}. \tag{23}$$

We plot the empirical average of service completion time $\hat{S}$ in Fig. 9 and empirical average of server utilization cost $\hat{W}$ in Fig. 10 as a function of initial servers $n_0 \in \{1, \ldots, 11\}$ for values of forking times $t_1 \in \{10, 20, \cdots, 90\}$ seconds. In Fig. 11, we plot the empirical average of service completion times with respect to empirical average of server utilization cost when $\lambda = 1$ for the single forking varying the initial number of servers $n_0 \in \{1, \ldots, 11\}$ for the forking times $t_1 \in \{10, 20, \cdots, 90\}$ seconds. We also evaluate the two-forking setup on Intel DevCloud with the same parameters. Given the first forking point at t, the second forking point at $s > t$, two-forking sequences $((0, m_0), (t, m_1), (s, N - m_0 - m_1))$, the tradeoff is plotted in Fig. 12.

E. Results

From Fig. 8, we observe that the empirical distribution of the job execution times at each node has characteristics of a shifted exponential distribution. The empirical distribution has

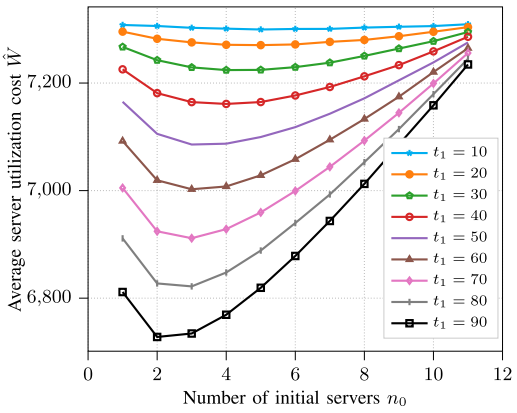

Fig. 10. Empirical and task average of server utilization cost $\hat{W}$ as a function of initial number of servers $n_0 \in \{1, \ldots, 11\}$ for single forking of $K = 10$ parallel tasks at different forking times $t_1 \in \{10, 20, \ldots, 90\}$ when jobs are executed on Intel DevCloud.

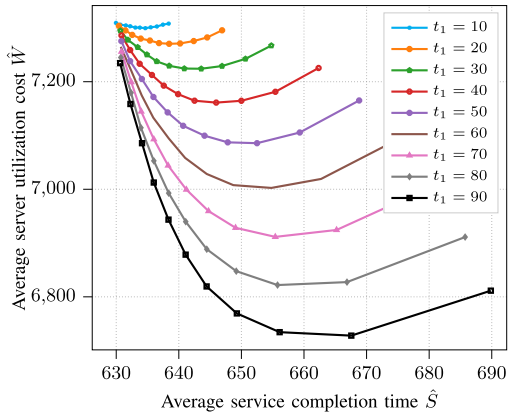

Fig. 11. Empirical and task average of server utilization cost $\hat{W}$ as a function of empirical average of service completion time $\hat{S}$ by varying the number of initial servers $n_0 \in \{1, \ldots, 11\}$ for single forking of $K = 10$ parallel tasks at different forking times t_1 when jobs are executed on Intel DevCloud.

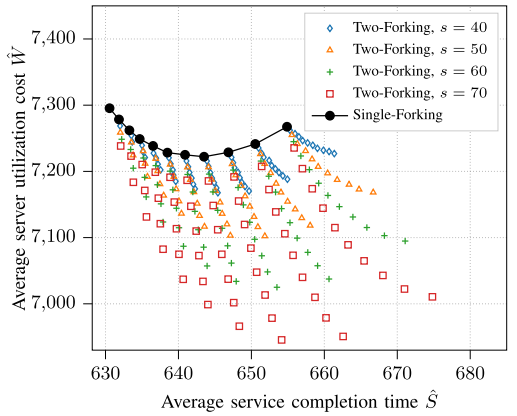

Fig. 12. This figure illustrates achievable points on Intel DevCloud cluster for empirical and task averaged server utilization cost and empirical average of service completion time for two-forking with different values of forked servers m_0, m_1 and $N - m_0 - m_1$ at forking points $t_0 = 0$, $t = 30$ and $s \in \{40, 50, 60, 70\}$, respectively. For comparison, we also plot the tradeoff points for single-forking at forking time $t = 30$ varying the number of initial servers n_0.

a distinct constant shift corresponding to the start delay, and the random part of the job execution time doesn't have long tails.

We substantiate the analytical results obtained in Theorem 4 for single-forking, by observing that the mean service completion time $\mathbb{E}S$ decreases with increase in initial number of servers n_0. Further, the tradeoff suggests that the number of initial servers n_0 is an important consideration for an efficient system design. The insights obtained in this experiment for two-forking are identical to those obtained from the shifted exponential service distribution.

VIII. Conclusions and Future Work

This paper considers a multi-fork analysis for running cloud computing jobs. We analytically computed the mean service completion time and the mean server utilization cost for multiple computation jobs, when the job execution time at each server is assumed to be *i.i.d.* with a shifted exponential distribution. We show that having multiple forking points for speculative execution of jobs provide significantly improved tradeoff between the two performance metrics. As a special case, we also show that starting with a single server in speculative execution of tasks is sub-optimal. This paper considers replication as a strategy for speculative execution.

We empirically verified that the insights derived from the shifted exponential distribution continue to hold when the job execution times at individual servers have heavy-tailed distributions such as Pareto and Weibull. We also conducted this study on a real compute cluster, and verified that the empirical distribution of the job execution time has a constant shift and light tails. This implies that a shifted exponential distribution capture the service time well in real compute clusters. As a result, the insights derived from the analytical study continue to hold on the studied compute cluster as well.

Recently, coding-theory-inspired approaches have been applied to mitigate the effect of stragglers [26]–[28]. Single fork analysis with coding has been studied in [27], [29]. In [27] k tasks of a job are started at $t = 0$ whereas in [29] the authors considered starting multiple jobs at $t = 0$ for a better tradeoff. Considering multi-fork analysis with such general coding flexibilities remains an important future direction. We hope that the framework provided in this article can be utilized to quantify the performance gains of multi-forked coded replicas.

Further, this work considers the performance metrics for a single job system. Analysis of overall completion time of jobs when there is a sequence of job arrivals is an open problem. When the system load is low, the request queue has a single job with high probability, and our work provides insights for the queueing system in this regime.

Acknowledgment

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.

References

[1] D. Cheng, J. Rao, Y. Guo, and X. Zhou, "Improving mapreduce performance in heterogeneous environments with adaptive task tuning," in *Proc. 15th Int. Middleware Conf. (Middleware)*. Bordeaux, France: ACM, 2014, pp. 97–108.

[2] P. Garraghan, X. Ouyang, R. Yang, D. Mckee, and J. Xu, "Straggler root-cause and impact analysis for massive-scale virtualized cloud datacenters," *IEEE Trans. Services Comput.*, vol. 12, no. 1, pp. 91–104, Jan. 2019.

[3] X. Ouyang, P. Garraghan, C. Wang, P. Townend, and J. Xu, "An approach for modeling and ranking node-level stragglers in cloud datacenters," in *Proc. IEEE Int. Conf. Services Comput. (SCC)*, San Francisco, CA, USA, Jun. 2016, pp. 673–680.

[4] Y. Guo, J. Rao, C. Jiang, and X. Zhou, "Moving hadoop into the cloud with flexible slot management and speculative execution," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 3, pp. 798–812, Mar. 2017.

[5] G. Vernik, M. Factor, E. K. Kolodner, E. Ofer, P. Michiardi, and F. Pace, "Stocator: A high performance object store connector for spark," in *Proc. 10th ACM Int. Syst. Storage Conf. (SYSTOR)*. Haifa, Israel: ACM, 2017, p. 27.

[6] N. B. Shah, K. Lee, and K. Ramchandran, "When do redundant requests reduce latency?" *IEEE Trans. Commun.*, vol. 64, no. 2, pp. 715–722, Feb. 2016.

[7] Y. Xiang, T. Lan, V. Aggarwal, and Y.-F.-R. 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.

[8] D. Wang, G. Joshi, and G. W. Wornell, "Efficient straggler replication in large-scale parallel computing," *ACM Trans. Model. Perform. Eval. Comput. Syst.*, vol. 4, no. 2, pp. 1–23, Apr. 2019.

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

[10] K. Lee, R. Pedarsani, and K. Ramchandran, "On scheduling redundant requests with cancellation overheads," *IEEE/ACM Trans. Netw.*, vol. 25, no. 2, pp. 1279–1290, Apr. 2017.

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

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

[13] 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.

[14] K. Gardner, S. Zbarsky, S. Doroudi, M. Harchol-Balter, and E. Hyytia, "Reducing latency via redundant requests: Exact analysis," *ACM SIGMETRICS Perf. Eval. Rev.*, vol. 43, no. 1, pp. 347–360, 2015.

[15] G. Joshi, Y. Liu, and E. Soljanin, "On the delay-storage trade-off in content download from coded distributed storage systems," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 5, pp. 989–997, May 2014.

[16] Y. Xiang, T. Lan, V. Aggarwal, and Y. F. R. Chen, "Joint latency and cost optimization for erasurecoded data center storage," *SIGMETRICS Perform. Eval. Rev.*, vol. 42, no. 2, pp. 3–14, Sep. 2014.

[17] 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, Aug. 2018.

[18] 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.

[19] K. Gardner, S. Zbarsky, S. Doroudi, M. Harchol-Balter, E. Hyytiä, and A. Scheller-Wolf, "Queueing with redundant requests: Exact analysis," *Queueing Syst*, vol. 83, nos. 3–4, pp. 227–259, Aug. 2016.

[20] W. Wang, M. Harchol-Balter, H. Jiang, A. Scheller-Wolf, and R. Srikant, "Delay asymptotics and bounds for multitask parallel jobs," *Queueing Syst*, vol. 91, nos. 3–4, pp. 207–239, Apr. 2019.

[21] A. O. Al-Abbasi, V. Aggarwal, and T. Lan, "TTLoC: Taming tail latency for erasure-coded cloud storage systems," *IEEE Trans. Netw. Serv. Manage.*, vol. 16, no. 4, pp. 1609–1623, Dec. 2019.

[22] 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, Mar. 2018.

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

[24] R. H. Byrd, M. E. Hribar, and J. Nocedal, "An interior point algorithm for large-scale nonlinear programming," *SIAM J. Optim.*, vol. 9, no. 4, pp. 877–900, Jan. 1999.

[25] *Intel Devcloud*. Accessed: Oct. 10, 2019. [Online]. Available: https://software.intel.com/en-us/devcloud

[26] R. Tandon, Q. Lei, A. G. Dimakis, and N. Karampatziakis, "Gradient coding: Avoiding stragglers in distributed learning," in *Proc. Int. Conf. Mach. Learn. (ICML)*, vol. 70, 2017, pp. 3368–3376.

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

[28] S. Sasi, V. Lalitha, V. Aggarwal, and B. Sundar Rajan, "Straggler mitigation with tiered gradient codes," 2019, *arXiv:1909.02516*. [Online]. Available: http://arxiv.org/abs/1909.02516

[29] A. Badita, P. Parag, and V. Aggarwal, "Sequential addition of coded tasks for straggler mitigation," in *Proc. IEEE Int. Conf. Comp. Commun. (INFOCOM)*, to be published.

[30] R. M. Corless, G. H. Gonnet, D. E. G. Hare, D. J. Jeffrey, and D. E. Knuth, "On the LambertW function," *Adv. Comput. Math.*, vol. 5, no. 1, pp. 329–359, Dec. 1996.

[31] D. A. Barry, J.-Y. Parlange, G. C. Sander, and M. Sivaplan, "A class of exact solutions for Richards' equation," *J. Hydrol.*, vol. 142, nos. 1–4, pp. 29–46, Feb. 1993.

[32] D. A. Barry, J.-Y. Parlange, L. Li, H. Prommer, C. J. Cunningham, and F. Stagnitti, "Analytical approximations for real values of the Lambert W-function," *Math. Comput. Simul.*, vol. 53, nos. 1–2, pp. 95–103, Aug. 2000.

Ajay Badita (Student Member, IEEE) received the B.Tech. degree from JNTU, Kakinada, in 2011, and the M.Tech. degree from NIT, Rourkela, in 2015, both in electronics and communication engineering. He is currently pursuing the Ph.D. degree with the ECE Department, Indian Institute of Science. His research interests include delay-sensitive communication, and computing and storage in distributed systems.

Parimal Parag (Member, IEEE) received the M.Tech. and B.Tech. degrees from IIT Madras in 2004, and the Ph.D. degree from Texas A&M University in 2011, all in electrical engineering. Prior to that, he was a Senior System Engineer (R&D) at ASSIA Inc., Redwood City, CA, USA, from 2011 to 2014. He is currently an Assistant Professor with the ECE Department, Indian Institute of Science. He was a coauthor of the 2018 IEEE ISIT Student Best Paper. He was a recipient of the 2017 Early Career Award from the Science and Engineering Research Board.

Vaneet Aggarwal (Senior Member, IEEE) received the B.Tech. degree from the IIT Kanpur, India, in 2005, and the M.A. and Ph.D. degrees from Princeton University, Princeton, NJ, USA, in 2007 and 2010, respectively, all in electrical engineering.

He was a Senior Member of Technical Staff Research with AT&T Labs-Research, NJ, USA, from 2010 to 2014, an Adjunct Assistant Professor with Columbia University, NY, USA, from 2013 to 2014, and an VAJRA Adjunct Professor with Indian Institute of Science (IISc), Bengaluru, India, from 2018 to 2019. He has been an Associate Professor with Purdue University, West Lafayette, IN, USA, since January 2015. His current research interests include communications and networking, cloud computing, and machine learning. He was a recipient of the Princeton University's Porter Ogden Jacobus Honorific Fellowship in 2009, the 2017 Jack Neubauer Memorial Award recognized as the Best Systems Paper published in the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, and the 2018 INFOCOM Workshop Best Paper Award. He is serving on the editorial boards for the IEEE TRANSACTIONS ON COMMUNICATIONS, the IEEE TRANSACTIONS ON GREEN COMMUNICATIONS AND NETWORKING, and the IEEE/ACM TRANSACTIONS ON NETWORKING.