Queues with Redundancy: Latency-Cost Analysis

Gauri Joshi
EECS Dept., MIT
Cambridge MA, USA
gauri@mit.edu

Emina Soljanin
Bell Labs Alcatel-Lucent
Murray Hill NJ, USA
emina@bell-labs.com

Gregory Wornell
EECS Dept., MIT
Cambridge, MA, USA
gww@mit.edu

1. INTRODUCTION

A major advantage of cloud computing and storage is the large-scale sharing of resources, which provides scalability and flexibility. But resource-sharing causes variability in the latency experienced by the user, due to several factors such as virtualization, server outages, network congestion etc . This problem is further aggravated when a job consists of several parallel tasks, because the task run on the slowest machine becomes the latency bottleneck.

A promising method to reduce latency is to assign a task to multiple machines and wait for the earliest to finish. Similarly, in cloud storage systems requests to download the content can be assigned to multiple replicas, such that it is sufficient to download any one replica. Although studied actively in systems in the past few years, there is little work on rigorous analysis of how redundancy affects latency. The effect of redundancy in queueing systems was first analyzed only recently in [2,3,6], assuming exponential service time. General service time distribution, in particular the effect of its tail, is considered in [7,8].

This work analyzes the trade-off between latency and the cost of computing resources in queues with redundancy, without assuming exponential service time. We study a generalized fork-join queueing model where finishing any k out of n tasks is sufficient to complete a job. The redundant tasks can be canceled when any k tasks finish, or earlier, when any k tasks start service. For the $k = 1$ case, we get an elegant latency and cost analysis by identifying equivalences between systems without and with early redundancy cancellation to $M/G/1$ and $M/G/n$ queues respectively. For general k, we derive bounds on the latency and cost. Please see [4] for an extended version of this work.

2. PROBLEM SETUP

Consider a distributed system with n statistically identical servers. Jobs arrive according to a rate λ Poisson process. The scheduler forks each incoming job into n tasks, and assigns them respectively to first-come first-serve queues at the n servers. The n tasks are designed such that completion of any k tasks is sufficient to complete the job. The case $k = 1$ corresponds to running replicas of a job on multiple machines. General k arise in approximate computing, or in content download from coded distributed storage.

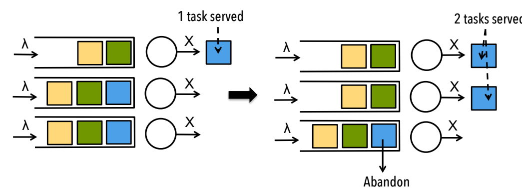

Fig. 1: The $(3, 2)$ fork-join system. When any 2 tasks of a job finish, the third task abandons its queue.

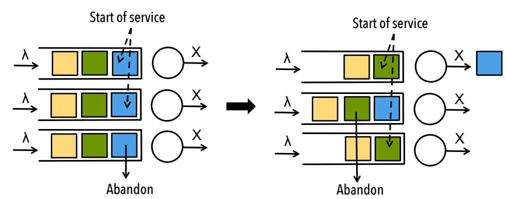

Fig. 2: The $(3, 2)$ fork-early-cancel system. When any 2 tasks of a job start service, the third abandons its queue.

The time taken to serve a task, is modeled by the random variable X, with distribution F_X, and is assumed to be i.i.d. across requests and servers. Dependence across servers due to the job size can be modeled by adding a constant proportional to average job size to service time X.

When any k out of the n tasks of a job are served, the scheduler immediately cancels the remaining $n-k$ redundant tasks, as illustrated in Fig. 1. We refer to this system as the (n, k) fork-join system, defined formally as follows.

Definition 1 ((n, k) fork-join system). *A job is forked into n tasks that join first-come first-serve queues at the n servers. When any k tasks finish service, all other tasks are canceled and abandon their queues immediately.* [1]

Instead of waiting for k tasks to finish, we could cancel the redundant tasks as soon as k tasks start service. This variant, called the (n, k) fork-early-cancel system is formally defined as follows.

Definition 2 ((n, k) fork-early-cancel system). *A job is forked into n tasks that join queues at the n servers. When any k tasks start service, all redundant tasks are canceled immediately. If more than k tasks start service simultaneously, we retain any k chosen uniformly at random. The job is complete when these k tasks finish.*

This work was supported in part by NSF under Grant No. CCF-1319828, AFOSR under Grant No. FA9550-11-1-0183, and a Schlumberger Faculty for the Future Fellowship.

[1] The (n, k) fork-join system is a generalization of the well known fork-join queue, which corresponds to the $k = n$ case.

Table 1: Summary of Results on Latency-Cost Analysis. We get exact analysis for $k = 1$, and bounds for general k.

	Replicated System ($k = 1$)		General k	
	$(n, 1)$ **fork-join**	$(n, 1)$ **fork-early-cancel**	(n, k) **fork-join**	(n, k) **fork-early-cancel**
Latency $\mathbb{E}[T]$	Thm. 1, using $\equiv$ to $M/G/1$ queue	Thm. 2, using $\equiv$ to $M/G/n$ queue	Bounds in Thm. 3	Upper Bound (generalizing [5])
Cost $\mathbb{E}[C]$	$n\mathbb{E}[X_{1:n}]$, where $X_{1:n} \triangleq \min(X_1, \ldots, X_n)$	$\mathbb{E}[X]$, where X is the task service time	Bounds in Thm. 4 (Tight for $k = 1, n$)	$k\mathbb{E}[X]$

Fig. 1 and Fig. 2 illustrate the (n, k) fork-join and fork-early-cancel systems respectively for $n = 3$ and $k = 2$. Early cancellation of redundant tasks can save computing cost, but could result in higher latency because of loss of diversity. In this work we develop insights into when early cancellation is better. We now define the latency and computing cost metrics, and analyse their trade-off afterwards.

Definition 3 (Latency). *The latency $\mathbb{E}[T]$ is defined as the expected time from the instant when a job arrives, until any k of its tasks are served.*

Definition 4 (Computing Cost). *The computing cost $\mathbb{E}[C]$ is the expected total time spent serving the tasks of a job, not including the waiting time in queue.*

We now express the service capacity of the system in terms of $\mathbb{E}[C]$, the average total server time utilized per job.

Claim 1 (Service Capacity in terms of $\mathbb{E}[C]$). *For a system of n servers, and task assignment symmetric across the servers, the maximum λ for which $\mathbb{E}[T] < \infty$ is*

$$\lambda_{max} = \frac{n}{\mathbb{E}[C]}. \tag{1}$$

Thus $\mathbb{E}[C]$ can be used to compare systems in the high λ regime. We will illustrate this new technique in Fig. 4, comparing the system with and without early cancellation.

Table 1 summarizes the key results of the latency-cost analysis presented in Sections 3 and 4 below. We use the notation $X_{i:n}$ to denote the i^{th} smallest of i.i.d. random variables $X_1, \ldots X_n$, with distribution F_X. All proofs are omitted here and can be found in the extended version [4].

3. REPLICATED SYSTEM $(k = 1)$

Observing that the $(n, 1)$ fork-join system is equivalent to an $M/G/1$ queue, and the $(n, 1)$ fork-early-cancel system is equivalent to an $M/G/n$ queue will help us derive the latency and the cost of these systems.

Theorem 1. *The latency and computing cost of an $(n, 1)$ fork-join system is given by*

$$\mathbb{E}[T] = \mathbb{E}[X_{1:n}] + \frac{\lambda \mathbb{E}[X_{1:n}^2]}{2(1 - \lambda \mathbb{E}[X_{1:n}])}, \tag{2}$$

$$\mathbb{E}[C] = n\mathbb{E}[X_{1:n}]. \tag{3}$$

To prove Thm. 1 we identify that in the $(n, 1)$ fork-join system, all tasks of a job start service simultaneously. Thus, it is equivalent to an $M/G/1$ queue with service time $X_{1:n}$, whose latency is given by the Pollaczek-Khinchine formula (2). Fig. 3 shows the latency-cost trade-off when the service time $X = \Delta + \text{Exp}(\mu)$, a shifted exponential with $\mu = 0.5$, and $\lambda = 0.25$. As n increases along each curve, $\mathbb{E}[T]$ decreases and $\mathbb{E}[C] = n\mathbb{E}[X_{1:n}]$ increases. Only when X is a pure exponential ($\Delta = 0$), we can reduce latency without any additional cost. Also, the system reduces to an $M/M/1$ queue and the latency $T \sim \text{Exp}(\mu - \lambda)$. In [4], we show that if the tail distribution $\bar{F}_X$ is log-convex (e.g. hyperexponential), then latency and cost reduce simultaneously with n.

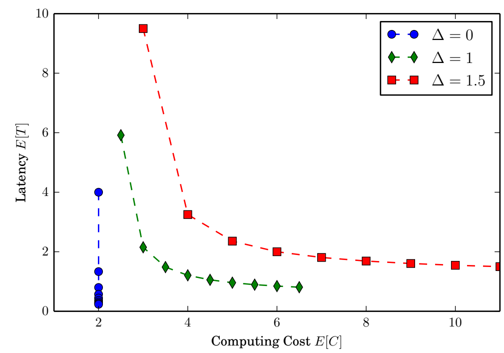

Fig. 3: For $X \sim \Delta + \text{Exp}(\mu)$, $\mu = 0.5$, and $\lambda = 0.25$, latency decreases and cost increases, as n increases along each curve. But for $\Delta = 0$ latency reduces at no additional cost.

Theorem 2. *The latency and cost of the $(n, 1)$ fork- early cancel system are given by*

$$\mathbb{E}[T] = \mathbb{E}\left[T^{M/G/n}\right] \approx \mathbb{E}[X] + \frac{\mathbb{E}[X^2]}{2\mathbb{E}[X]^2}\mathbb{E}\left[W^{M/M/n}\right], \tag{4}$$

$$\mathbb{E}[C] = \mathbb{E}[X], \tag{5}$$

where $\mathbb{E}\left[W^{M/M/n}\right]$ is the expected waiting time in an $M/M/n$ queueing system with service time $X \sim F_X$.

To prove Thm. 2, we identify that in the $(n, 1)$ fork-early-cancel system, one task of each job joins the shortest queue available, and the other tasks are canceled before they begin service. Thus, it is equivalent to an $M/G/n$ system whose latency is given by the well-known approximation (4). Since the cost is $\mathbb{E}[C] = \mathbb{E}[X]$ which is independent of n, there is no latency-cost trade-off similar to Fig. 3.

In Fig. 4, we compare the $(4, 1)$ system with and without early cancellation by plotting latency vs. λ. The service time $X \sim 2 + \text{Exp}(0.5)$, a shifted exponential. Early cancellation gives lower latency in the high λ regime. This can be inferred from Claim 1, since $\mathbb{E}[C]$ with early cancellation ($\mathbb{E}[X]$) is smaller than that without ($n\mathbb{E}[X_{1:n}]$), when X is shifted exponential. If we plot latency vs. Δ, the constant part of the service time, we observe that early cancellation gives lower latency for higher Δ ('less random' X).

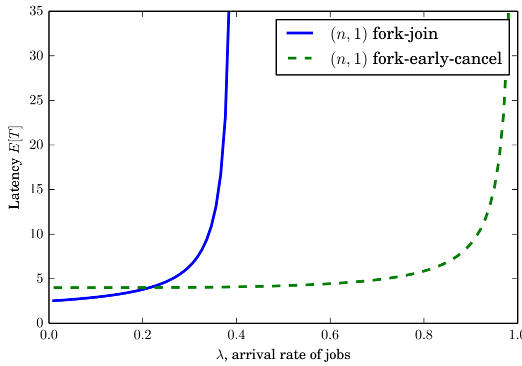

Fig. 4: Comparison of $\mathbb{E}[T]$ of the $(4,1)$ system with and without early cancellation, vs. λ for $X \sim 2+\text{Exp}(0.5)$. Early cancellation is gives higher service capacity (by Claim 1).

4. GENERAL CASE: $1 \leq k \leq n$

In the traditional fork-join queue ($k = n$ case in Def. 1 with exponential service time), an exact expression for latency can be found only for $n = 2$ [1]. Only bounds are known for general k and n [3, 5]. We present the first latency and cost bounds for general F_X.

THEOREM 3. *The latency* $\mathbb{E}[T]$ *of the* (n,k) *fork-join system is bounded as*

$$\mathbb{E}[T] \leq \mathbb{E}[X_{k:n}] + \frac{\lambda \mathbb{E}[X_{k:n}^2]}{2(1 - \lambda \mathbb{E}[X_{k:n}])}, \quad (6)$$

$$\mathbb{E}[T] \geq \mathbb{E}[X_{k:n}] + \frac{\lambda \mathbb{E}[X_{1:n}^2]}{2(1 - \lambda \mathbb{E}[X_{1:n}])}. \quad (7)$$

To get (6), we use the split-merge system, in which no two jobs are served simultaneously. In (7), we use the waiting time of the $(n,1)$ fork-join system to lower bound that of the (n,k) system. Fig. 5 shows the latency bounds and simulation values vs. k for $n = 10$, $\lambda = 0.5$, and X following the Pareto distribution with $x_m = 0.5$ and $\alpha = 2.5$. For $k = n$, we can get a tighter bound than (6) by generalizing the approach used in [5]. The same approach can be used to upper bound $\mathbb{E}[T]$ of the (n,k) fork-early-cancel system.

THEOREM 4. *The computing cost* $\mathbb{E}[C]$ *of the* (n,k) *fork-join system is bounded as*

$$\mathbb{E}[C] \leq (k-1)\mathbb{E}[X] + (n-k+1)\mathbb{E}[X_{1:n-k+1}], \quad (8)$$

$$\mathbb{E}[C] \geq \sum_{i=1}^{k} \mathbb{E}[X_{i:n}] + (n-k)\mathbb{E}[X_{1:n-k+1}]. \quad (9)$$

The key idea for proving Thm. 4 is our observation that for each job, some $n-k+1$ of its tasks start service simultaneously, which allowed us to analyze them separately. The bounds are tight for $k = 1$ and $k = n$ as seen in Fig. 6.

For the (n,k) fork-early-cancel system, since exactly k tasks start and finish service, it follows that $\mathbb{E}[C] = k\mathbb{E}[X]$.

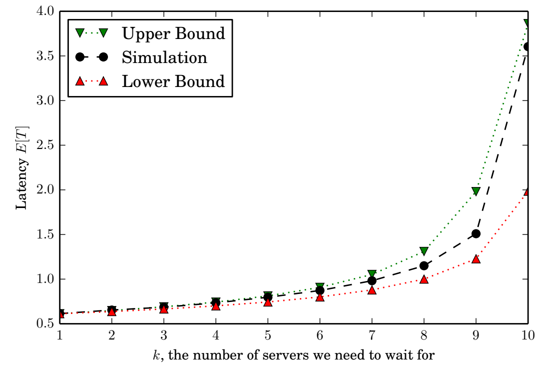

Fig. 5: Bounds $\mathbb{E}[T]$ vs. k, with service time $X = \text{Pareto}(0.5, 2.5)$ with $n = 10$, and $\lambda = 0.5$. The $k = n$ upper bound is determined by generalizing the approach in [5].

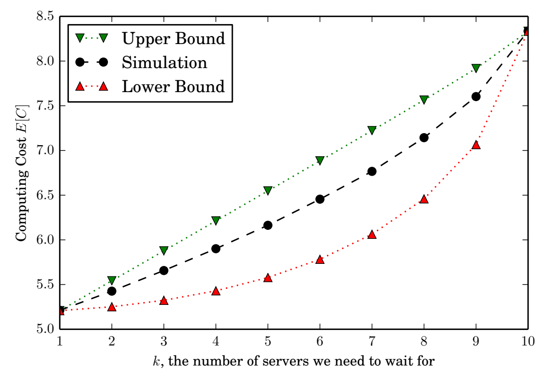

Fig. 6: Bounds on $\mathbb{E}[C]$ vs. k, for the same parameters as Fig. 5. The bounds are tight for $k = 1$ and $k = n$.

5. REFERENCES

[1] FLATTO, L., AND HAHN, S. Two parallel queues created by arrivals with two demands I. *SIAM Journal on Applied Mathematics 44*, 5 (1984), 1041–1053.

[2] JOSHI, G., LIU, Y., AND SOLJANIN, E. Coding for fast content download. *Allerton Conference on Communication, Control and Computing* (Oct. 2012), 326–333.

[3] JOSHI, G., LIU, Y., AND SOLJANIN, E. On the Delay-storage Trade-off in Content Download from Coded Distributed Storage. *IEEE Journal on Selected Areas on Communications* (May 2014).

[4] JOSHI, G., SOLJANIN, E., AND WORNELL, G. Efficient redundancy techniques for latency reduction in cloud systems. *arXiv [cs.dc]* (Aug. 2015).

[5] NELSON, R., AND TANTAWI, A. Approximate analysis of fork/join synchronization in parallel queues. 739–743.

[6] SHAH, N., LEE, K., AND RAMACHANDRAN, K. The mds queue: Analyzing the latency performance of erasure codes. *IEEE International Symposium on Information Theory* (July 2014).

[7] SHAH, N., LEE, K., AND RAMCHANDRAN, K. When do redundant requests reduce latency? In *Allerton Conference on Communication, Control and Computing* (Oct. 2013), pp. 731–738.

[8] WANG, D., JOSHI, G., AND WORNELL, G. Using straggler replication to reduce latency in large-scale parallel computing (extended version). *arXiv:1503.03128 [cs.dc]* (Mar. 2015).