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 $\lambda$ 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

---

1 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.

Copyright is held by author/owner(s).
Table 1: Summary of Results on Latency-Cost Analysis. We get exact analysis for $k = 1$, and bounds for general $k$.

<table>
<thead>
<tr>
<th>Latency $E[T]$</th>
<th>Cost $E[C]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replicated System ($k = 1$)</td>
<td></td>
</tr>
<tr>
<td>(n,1) fork-join</td>
<td>$E[X_{1:n}]$, where $X_{1:n} \triangleq \min(X_1, \ldots, X_n)$</td>
</tr>
<tr>
<td>(n,1) fork-early-cancel</td>
<td>$E[X]$, where $X$ is the task service time</td>
</tr>
<tr>
<td>General $k$</td>
<td></td>
</tr>
<tr>
<td>(n,k) fork-join</td>
<td>Bounds in Thm. 3</td>
</tr>
<tr>
<td>(n,k) fork-early-cancel</td>
<td>Upper Bound (generalizing [5])</td>
</tr>
</tbody>
</table>

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 $E[T]$ 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 $E[C]$ is the expected total time spent serving the tasks of a job, not including the waiting time in queue.

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.

and $\lambda = 0.25$. As $n$ increases along each curve, $E[T]$ decreases and $E[C] = nE[X_{1:n}]$ 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 $F_X$ is log-convex (e.g. hyperexponential), then latency and cost reduce simultaneously with $n$.

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

$$E[T] = E[X] + E[X^2] \frac{1}{2\lambda E[X_{1:n}]}$$

(4)

$$E[C] = nE[X_{1:n}]$$

(5)

where $E[X_{1:n}]$ is the expected waiting time in an $M/M/1$ 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 $M/G/1$ queue 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 $\mu = 0.5$, shifted exponential. Early cancellation gives lower latency in the high $\lambda$ regime. This can be inferred from Claim 1, since $E[C]$ with early cancellation ($E[X]$) is smaller than that without ($nE[X_{1:n}]$), when $X$ is shifted exponential. If we plot latency vs. $\Delta$, the constant part of the service time, we observe that early cancellation gives lower latency for higher $\Delta$ (‘less random’ $X$).
Fig. 4: Comparison of $\mathbb{E}[T]$ of the $(4,1)$ system with and without early cancellation, vs. $\lambda$ 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

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

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

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 $x_{\alpha} = 0.5$ 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

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

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

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 $\mathbb{E}[C] = k\mathbb{E}[X]$.

5. REFERENCES


