Towards Performance Modeling of Speculative Execution for Cloud Applications

Tommi Nylander, Johan Ruuskanen, Karl-Erik Årzén, Martina Maggio
{tommi,johan.ruuskanen,karlerik,martina}@control.lth.se
Lund University

ABSTRACT

Interesting approaches to counteract performance variability within cloud datacenters include sending multiple request clones, either immediately or after a specified waiting time. In this paper we present a performance model of cloud applications that utilize the latter concept, known as speculative execution. We study the popular Join-Shortest-Queue load-balancing strategy under the processor sharing queuing discipline. Utilizing the *near-synchronized service* property of this setting, we model speculative execution using a simplified synchronized service model. Our model is approximate, but accurate enough to be useful even for high utilization scenarios. Furthermore, the model is valid for any, possibly empirical, inter-arrival and service time distributions. We present preliminary simulation results, showing the promise of our proposed model.

CCS CONCEPTS

• **Computer systems organization → Cloud computing**; **Redundancy**; *Reliability*.

KEYWORDS

Cloning, Speculative Execution, Cloud Computing, Datacenters

ACM Reference Format:
Tommi Nylander, Johan Ruuskanen, Karl-Erik Årzén, Martina Maggio. 2020. Towards Performance Modeling of Speculative Execution for Cloud Applications. In *ACM/SPEC International Conference on Performance Engineering Companion (ICPE '20 Companion), April 20–24, 2020, Edmonton, AB, Canada.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3375555.3384379

1 INTRODUCTION

Speculative execution is a popular method employed by cloud providers as a tool for increasing predictability of the execution time of jobs [3, 7]. Redundancy is introduced by launching copies of tasks that have been running for an unusually long time. The general idea is that the unpredictability of task execution times due to effects such as resource contention or network queues, can be mitigated by identifying slow running instances and launching copies that will hopefully complete before the original.

A closely related topic which has recently received increased attention from researchers is cloning. As explained by Ganesh et al. [2], cloning can be seen as a special case of speculative execution with no speculation time, i.e. all clones are sent immediately. We refer to [6] for more related work on this topic.

Much of the research on speculative execution have been done considering the case of straggler mitigation in distributed computing using big data frameworks such as MapReduce [4]. Here a job is split into several tasks, and is not considered completed until all, or a subset, of tasks have been completed. The ultimate response time of a job is thus highly sensitive to slow running tasks. Modeling and analysis of such systems often either assume that each server can only take a single job at a time [8], or that the introduction of redundancy does not affect the service times of other jobs. An exception for the latter is the recent contribution of Aktas et al. [1], which shortly considers the effect of redundancy on the response time distribution of tasks.

In this work we instead consider the case of replicated cloud applications subject to independent user requests, and seek to model the behaviour of such systems under speculative execution from a queuing model perspective. Our approach is dependent on two key concepts from [6], summarized in the following paragraph.

Near-Synchronized Service. The concept of *synchronized service* was introduced in [6] to simplify modeling of request cloning. Its full definition is given in [6], but in short, cloning under synchronized service implies that the clone that completes first is the one that receives the shortest service time. For the processor sharing (PS) queuing discipline, synchronized service implies that all n request clones $r^c_{1:n}$ of an original request r^o experience *identical* processor shares across all n servers. As synchronized service is very difficult to achieve in practice, the concept of *near-synchronized* service was further introduced in [6] to model scenarios that include imperfections such as arrival and cancellation delays. Additionally, it was shown empirically that the widely used load-balancing strategy Join-Shortest-Queue (JSQ) provides near-synchronized service for all request clones $r^c_{1:n}$, when using PS as queuing discipline. This property is very interesting as it allows for accurate approximate performance modeling of JSQ cloud applications subject to cloning, by using a simplified synchronized service model.

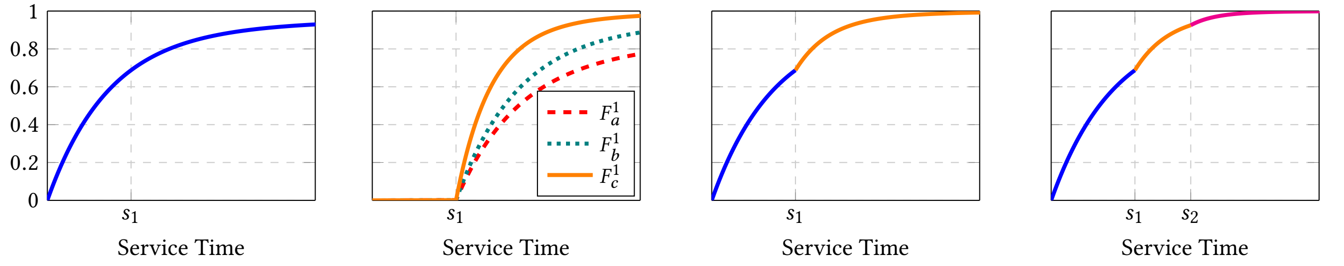
Contributions. Using the near-synchronization property of JSQ under PS, we (i) derive a novel performance model for replicated cloud applications subject to speculative execution; (ii) assuming Poisson arrivals, use existing results from queuing theory to obtain an approximate yet accurate expression for the average response time; and (iii) empirically demonstrate the potential of our model through simulations.

2 MODEL

We consider performance modeling of a cloud application replicated over m homogeneous servers, modeled using the PS queuing

ICPE '20 Companion, April 20–24, 2020, Edmonton, AB, Canada

ACM ISBN 978-1-4503-7109-4/20/04…$15.00
https://doi.org/10.1145/3375555.3384379



(a) Original service time CDF F_r^0 **(b) Three intermediate CDFs** **(c) Resulting service time CDF F_r^1** **(d) Resulting service time CDF F_r^2**

Figure 1: Speculative cloning for an example scenario $\mathcal{S}_2 = \{s_1, s_2\}$. From F_r^0 at s_1, three intermediate CDFs are formed using equations (2)-(4). Then F_c^1 is added to F_r^0 at s_1 according to Eq. (1) to form F_r^1. The procedure is then repeated at s_2 to form F_r^2.

discipline with service rate μ. User requests r_o with rate λ enter at the load balancer, leading to a system utilization $\rho = \lambda/(m\mu)$. The requests are dispatched to the servers using the JSQ strategy, that always chooses the least occupied server. We do not assume any specific distributions, however, for simplicity we require the service times to be independent and identically distributed (i.i.d.) across all m servers. When a specified amount of *service time* has been processed for an original request r_o, a speculative clone r_s^i is dispatched to a unique server, again using JSQ. This server system under JSQ and PS was shown in [6] to provide near-synchronized service, while our modeling approach is performed assuming synchronized service. Our derived performance metrics, including utilization ρ and average response time T, are thus approximate.

Define s_i as the service time when the speculative clone r_s^i is dispatched to the server system and $\mathcal{S}_n = \{s_1, s_2, \ldots, s_n\}$ as the ordered set of the service times of all speculative cloning instances with $s_{i-1} \leq s_i$. Denote by $F(x)$ the cumulative distribution function (CDF), and F_r^0 as the original service time CDF. Using Theorem 2 in [6], the following iterative formula (1) can be used to determine the resulting service time CDF F_r^n for the speculation scenario $\mathcal{S}_n$:

$$F_r^i(x) = \begin{cases} F_r^0(x), & x \leq s_1 \\ F_r^0(s_1) + (1 - F_r^0(s_1)) \cdot F_c^1(x), & s_1 < x \leq s_2 \\ \quad\vdots & \\ F_r^{i-1}(s_i) + (1 - F_r^{i-1}(s_i)) \cdot F_c^i(x), & s_i < x \end{cases} \tag{1}$$

with the intermediate CDF $F_c^i(x)$ determined as

$$F_a^i(x) = F_r^{i-1}(x|s_i < x) \tag{2}$$

$$F_b^i(x) = F_r^0(x - s_i) \tag{3}$$

$$F_c^i(x) = 1 - (1 - F_a^i(x)) \cdot (1 - F_b^i(x)). \tag{4}$$

The algorithm is visualized in Figure 1 for an example scenario $\mathcal{S}_2$. From (1), we can calculate the new average service rate $\mu(\mathcal{S}_n)$ for a scenario using n speculative clones as

$$\mu(\mathcal{S}_n) = \left(\int_0^\infty (1 - F_r^n(x))dx \right)^{-1}. \tag{5}$$

We define the *service factor* $f_\mu(\mathcal{S}_n)$ as the normalized increase of $\mu(\mathcal{S}_n)$ compared to the original $\mu(\mathcal{S}_0) = \mu$:

$$f_\mu(\mathcal{S}_n) = \frac{\mu(\mathcal{S}_n)}{\mu}. \tag{6}$$

To model the changes to the server system load, we need to consider the amount of speculative clones sent for each original request r_o and the time they spend in the system. We define the *speculation factor* f_p^i for a speculative clone at time s_i as the probability $f_p^i = 1 - F_r^i(s_i)$ that the clone is sent. Furthermore, we define the *sojourn factor* f_s^i for a speculative clone sent at time s_i as its time spent in the system compared to the original request r_o

$$f_s^i = \frac{\int_{s_i}^\infty (1 - F_r^n(x|s_i < x))dx}{\int_0^\infty (1 - F_r^n(x))dx}. \tag{7}$$

Now we define the *arrival factor* $f_\lambda(\mathcal{S}_n)$ for the total contributions to system load from all n speculative clones as

$$f_\lambda(\mathcal{S}_n) = 1 + \sum_{i=1}^{n} f_p^i \cdot f_s^i. \tag{8}$$

Finally, the *load factor* $f_\rho(\mathcal{S}_n)$ can then be defined as

$$f_\rho(\mathcal{S}_n) = \frac{f_\lambda(\mathcal{S}_n)}{f_\mu(\mathcal{S}_n)}. \tag{9}$$

$f_\rho(\mathcal{S}_n) > 1$ thus means that speculative cloning under scenario $\mathcal{S}_n$ results in an increase of the original system load $\rho(\mathcal{S}_0) = \rho$, whereas $f_\rho(\mathcal{S}_n) < 1$ represents a decrease. Also, we can define the modeled utilization of scenario $\mathcal{S}_n$ as

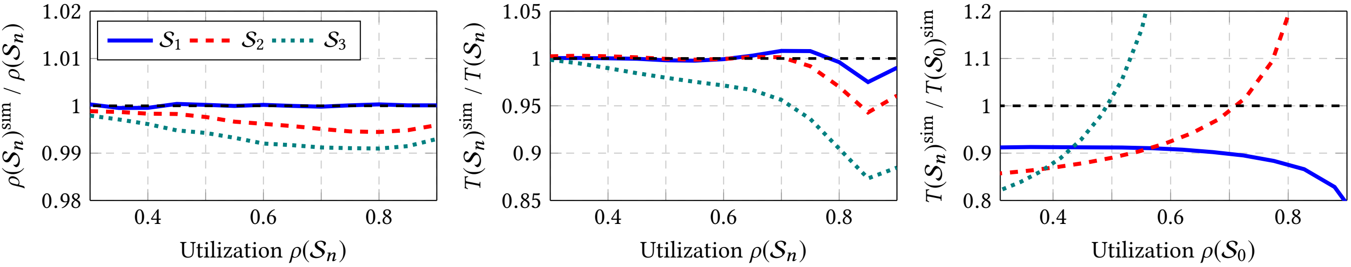
$$\rho(\mathcal{S}_n) = f_\rho(\mathcal{S}_n) \cdot \rho. \tag{10}$$

Equation (10) is very useful as it allows us to determine stability for the scenario $\mathcal{S}_n$ by studying if $\rho(\mathcal{S}_n) < 1$. Note that the arrival factor $f_\lambda(\mathcal{S}_n) > 1$ does not imply an increase in the arrival rate of original requests r_o, i.e. $\lambda(\mathcal{S}_n) = \lambda(\mathcal{S}_0) = \lambda$ for all n. Instead, it represents the contributions to the system load from all speculative clones. We model this as a decrease in the number of available servers $m(\mathcal{S}_0) = m$ as

$$m(\mathcal{S}_n) = \frac{m}{f_\lambda(\mathcal{S}_n)}. \tag{11}$$

As a result, $m(\mathcal{S}_n) \in \mathbf{R}^+$ is defined as a positive real number. Note that for non-speculative cloning (with all clones sent at $s_i = 0$), $f_\lambda(\mathcal{S}_n)$ and $m(\mathcal{S}_n)$ assume integer values. The clone-to-clusters model in [6], which divides n servers into m clusters thus fits as a special case in our speculative execution model.

To be able to get explicit response time measures, we need to assume Poisson arrival rates for $\lambda(\mathcal{S}_n)$. This allows us to utilize the



(a) Simulated vs modeled $\rho(\mathcal{S}_n)$.

(b) Simulated vs modeled $T(\mathcal{S}_n)$.

(c) Speculation scenarios $\mathcal{S}_n$ vs $\mathcal{S}_0$.

Figure 2: Simulation results using 20 repeated runs with 10^6 requests. Confidence intervals are tight and left out for readability.

very accurate (within 2-3%) approximate response time model for JSQ under PS from [5]. It provides average response times $T(\mathcal{S}_n)$ from the inputs (i) arrival rate $\lambda(\mathcal{S}_n)$; (ii) service rate $\mu(\mathcal{S}_n)$; and (iii) number of servers $m(\mathcal{S}_n)$.

Using a simplified synchronized service approach, we are thus able to approximately model utilization, stability and average response times for a replicated cloud application under a speculation scenario $\mathcal{S}_n$, assuming a JSQ+PS setup. The model accuracy is a potential issue that is examined in the next section. Another drawback with our approach is that it might be complicated to implement triggering of speculative clones at processed service times s_i as these can be cumbersome to keep track of in a real system.

3 EVALUATION

We evaluate our model using a discrete-event simulator, based on the `cloning-simulator` from [6] but extended with support for speculative execution. We use Poisson arrivals and our service times are distributed as Pareto (Type 1, shape=2.1, scale=0.5). We simulate using $m = 10$ servers under system loads $\rho(\mathcal{S}_n)$ from 0.3 to 0.9 and consider three different speculation scenarios: (i) $\mathcal{S}_1 = \{1.5\}$; (ii) $\mathcal{S}_2 = \{0.7, 1.0\}$; and (iii) $\mathcal{S}_3 = \{0.3, 0.6, 0.9\}$ (all units in seconds).

Figure 2 shows our preliminary results. In Figure 2a, the simulated system utilization $\rho(\mathcal{S}_n)^{\text{sim}}$ is normalized against our modeled $\rho(\mathcal{S}_n)$. The results are very close to 1 for all scenarios and loads, which points towards that our model is very accurate at predicting utilization and stability. Figure 2b shows the results of the simulated average response times $T(\mathcal{S}_n)^{\text{sim}}$ normalized against our modeled $T(\mathcal{S}_n)$. As can be seen, the accuracy of our model is very high for low to medium loads for all three speculation scenarios. However, for higher loads our model accuracy is worse (but still reasonable) for the more complicated scenarios. A probable explanation is that the service is further away from synchronization here. The final Figure 2c shows the results of the simulated average response times $T(\mathcal{S}_n)^{\text{sim}}$ for the speculation scenarios normalized against $T(\mathcal{S}_0)^{\text{sim}}$, where no speculation is present. A value below 1 indicates that the speculation scenarios are beneficial, and as can be seen all three scenarios perform well for low loads. Scenario $\mathcal{S}_1$ distinguishes itself from the the other two by actually outperforming the no speculation case at all system loads. The reason is that its load factor $f_\rho(\mathcal{S}_1)$ is below 1, i.e. it always *decreases* the system load. This is very interesting as it can be shown, using techniques from [6], that standard cloning (all $s_i = 0$) under this particular Pareto distribution is only beneficial for low loads. Speculative execution thus has the potential to be more useful than cloning under high loads.

4 CONCLUSION

We have presented a novel model of a replicated cloud application subject to speculative execution, that looks promising in our preliminary evaluation. We plan to expand our evaluation to be more general, and to use our model to find optimal speculation configurations $\mathcal{S}_n^{\star}$, providing the shortest response times. A possible approach could be to search for the configurations that minimize the system utilization, in order to provide performance enhancements even for server systems under high load.

ACKNOWLEDGMENTS

This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation, by the Nordforsk Nordic Hub on Industrial IoT (HI2OT), and by the ELLIIT Excellence Center at Lund University.

REFERENCES

[1] M. F. Aktaş and E. Soljanin. 2019. Straggler Mitigation at Scale. *IEEE/ACM Transactions on Networking* 27, 6 (Dec 2019), 2266–2279.

[2] Ganesh Ananthanarayanan, Ali Ghodsi, Scott Shenker, and Ion Stoica. 2013. Effective Straggler Mitigation: Attack of the Clones. In *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation (nsdi'13)*. USENIX Association, 185–198.

[3] Jeffrey Dean and Luiz André Barroso. 2013. The tail at scale. *Commun. ACM* 56, 2 (feb 2013), 74.

[4] Jeffrey Dean and Sanjay Ghemawat. 2008. MapReduce: Simplified Data Processing on Large Clusters. *Commun. ACM* 51, 1 (Jan. 2008), 107–113.

[5] Varun Gupta, Mor Harchol Balter, Karl Sigman, and Ward Whitt. 2007. Analysis of join-the-shortest-queue routing for web server farms. *Performance Evaluation* 64, 9-12 (oct 2007), 1062–1081.

[6] Tommi Nylander, Johan Ruuskanen, Karl-Erik Årzén, and Martina Maggio. 2020. Modeling of Request Cloning in Cloud Server Systems using Processor Sharing. In *Proceedings of the 2020 ACM/SPEC International Conference on Performance Engineering (ICPE '20), April 20–24, 2020, Edmonton, AB, Canada.*

[7] Xiaoqi Ren, Ganesh Ananthanarayanan, Adam Wierman, and Minlan Yu. 2015. Hopper: Decentralized Speculation-aware Cluster Scheduling at Scale. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication - SIGCOMM '15.*

[8] H. Xu and W. C. Lau. 2017. Optimization for Speculative Execution in Big Data Processing Clusters. *IEEE Transactions on Parallel and Distributed Systems* 28, 2 (Feb 2017), 530–545.