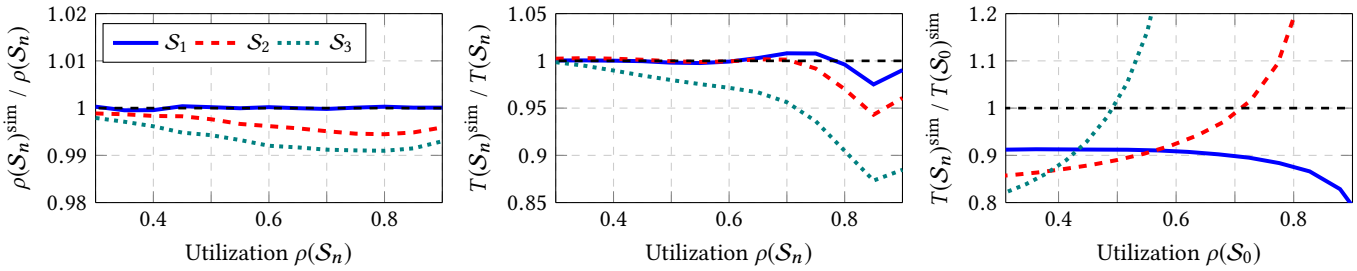

(a) Simulated vs modeled $\rho(\mathcal{S}_n)$. **(b) Simulated vs modeled $T(\mathcal{S}_n)$.** **(c) Speculation scenarios $\mathcal{S}_n$ vs $\mathcal{S}_0$.**

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.