The Tail at Scale: How to Predict It?

Minh Nguyen, Zhongwei Li, Feng Duan, Hao Che, Yu Lei, Hong Jiang
Department of Computer Science and Engineering
The University of Texas at Arlington

Abstract

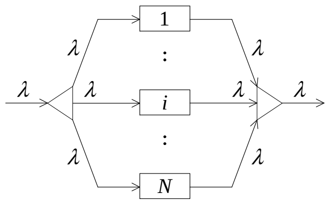
Scale-out applications have emerged as the dominant Internet services today. A request in a scale-out workload generally involves task partitioning and merging with barrier synchronization, making it difficult to predict the request tail latency to meet stringent tail Service Level Objectives (SLOs). In this paper, we find that the request tail latency can be faithfully predicted, in the high load region, by a prediction model using only the mean and variance of the task response time as input. The prediction errors for the 99th percentile request latency are found to be consistently within 10% at the load of 90% for both model and measurement-based testing cases. Consequently, the work in this paper establishes an important link between the request tail SLOs and the low order task statistics in a high load region, where the resource provisioning is desired. Finally, we discuss how the prediction model may facilitate highly scalable, tail-constrained resource provisioning for scale-out workloads.

1 Introduction

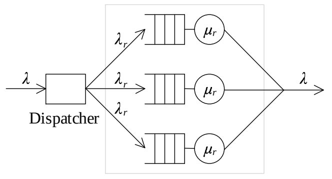
Scale-out, online-data-intensive (OLDI) workloads, such as web searching and social networking, provide user-facing services that involve a large number of servers for parallel processing, while requiring sub-second request responsiveness under high incoming request rates. The system running these workloads usually operates under stringent SLOs, such as imposing a tight tail constraint on high percentile request response time, e.g., 99th or 99.9th-percentile, to satisfy as many user requests as possible [7, 17].

However, imposing tight tail SLO for OLDI workloads makes resource provisioning in datacenter a challenging task. Due to the lack of good understanding of request tail behaviors, the current practice is to overprovision datacenter resources to meet SLO, at the cost of low resource utilization, e.g., less than 50% CPU and memory utilizations [5, 8, 22]. Although resource provisioning proposals with tail SLOs in mind exist, they generally do not incorporate tail SLOs explicitly as design constraints and rely on empirical data to verify whether the design meets tail SLOs or not. For example, the resource provisioning problem is formulated as the minimization of the variance of data flow path latency, as a way to indirectly curtail the tail latency [12]; the target tail latency SLOs are tracked using online dynamic feedback-loop-control-based schedulers [9, 25]; and employing job priority and a rate limiting technique based on the network calculus theory [27]. The root cause of the status quo is due to the lack of a link between system-level request tail SLOs and the subsystem-level task performance requirements. The key difficulty lies in the fact that a scale-out workload may involve *task partitioning* and *merging* as well as *task queuing*. Each request in the request flow with average request rate λ involves tasks to be queued at and processed by up to several thousands of task subsystems in parallel and then all the task results are merged and returned, as depicted in Fig. 1a. Here a task subsystem may involve multiple replicated servers for task-level fault tolerance and load balancing, e.g., Fig. 1b, where $\lambda_r = \lambda/3$ in the case of load balancing. Notable examples are Web search engines [4] and social networking [20]. In this case, the request response time is determined by the slowest task [7]. As the system scales out, the probability that the request response time may hit the tail task latency quickly increases [7]. To date, no general results are available that can predict the tail request response time at scale.

This lack of understanding of request tail behaviors is further exacerbated by the various task scheduling and tail-cutting techniques being used in task processing. In particular, as an effective tail-cutting technique, replicated servers in each subsystem are being used to allow redundant task issues to more than one replicated server to be processed, with the earliest result returned



(a) A system with subsystems as black boxes.



(b) A subsystem with one dispatcher and three replicated servers.

Figure 1: The task partitioning and merging model.

and rest removed [7, 24]. Although some analytic results are available on redundant task issues [10, 21, 26], they either address only a single replicated server subsystem with exponential task service time distribution only [10] or parallel request load balancing without task partitioning [21, 26]. The task partitioning-and-merging part of a scale-out workload generally lies in the critical path for request processing and constitutes a major part of request processing time and hardware cost, e.g., more than two-third of the total processing time and 90% hardware cost for a Web search engine [13]. Hence, it is of paramount importance to establish a link between the system-level request tail SLOs and the subsystem-level task performance requirements to facilitate explicitly tail-constrained resource provisioning at scale.

This paper aims at tackling the above challenge. It makes the following two major contributions. First, by treating each subsystem as a black box, we find that the tail behavior of a task mapped to a subsystem can be captured by a generalized exponential distribution function in the high load region, which uses the mean and variance of the task response time as input. This black-box solution allows the request distribution function and thus any given request tail SLOs to be explicitly expressed as a function of the means and variances of the individual task response times as the system scales out. Hence, in the case of homogeneous subsystems for parallel task processing, the request tail SLO is only dependent on the mean and variance of the task response time for one task mapped to any given subsystem. Second, we discuss how the proposed request tail prediction mechanism may be used to facilitate highly scalable, explicitly tail-constrained resource provisioning using homogeneous virtual machines (VMs) in a cloud environment.

The remainder of the paper is organized as follows. Section 2 presents the prediction model, simulation results, and analyses. Section 3 discusses how the proposed model may facilitate tail-constraint resource provisioning. Finally, Section 4 concludes the paper.

2 Tail Latency Prediction Model

2.1 Basic Ideas

A system serving scale-out, OLDI workloads commonly involves a large number of task subsystems for parallel processing. The diversity in the actual implementation of subsystems makes it extremely difficult to predict the task performance, let alone the request performance, in general. However, since the ultimate goal of this research is to be able to design request scheduling algorithms that can meet stringent tail SLOs by proof of design, while achieving high resource utilization, we are interested in the peak-load resource provisioning in a high load region, e.g., 90% or higher. In this region, it is possible to predict the task performance for a task mapped to a wide range of subsystems using a simple prediction model, as we now explain.

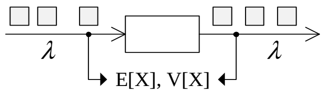


Figure 2: A subsystem as a black box.

There is a large body of research results in the context of queuing performance in high load regions (e.g., see [23] and the references therein). In particular, a classic result, known as the central limit theorem for heavy traffic queuing systems [14,15], states that for a G/G/m (here m is the number of servers) queue under heavy traffic load, the waiting time distribution could be approximated by an exponential distribution. Clearly, this theorem applies to the response time distribution as well, since the response time distribution converges to the waiting time distribution as the traffic load increases. The intuition behind this approximation is that in the high load region, the long queuing effect helps effectively smooth out service time fluctuations (i.e., the law of large numbers), which causes the waiting time or response time to converge to a distribution closely surrounding its mean value, i.e., the short-tailed exponential distribution, regardless of the actual arrival process and service time distribution. Inspired by this result, in this paper, we treat any task subsystem, e.g., the one in Fig. 1b, as a black box, given in Fig. 2. We further postulate that for a task mapped to a black box subsystem and in the high load region, the task response time distribution $F(x)$ for any arrival process can be approximated as a generalized exponential distribution function [11], as follows,

$$F_{ge}(x) = \begin{cases} (1-e^{-\mu x})^{\alpha} & x > 0, \\ 0 & \text{otherwise}, \end{cases} \quad (1)$$

where μ and α are the scale and shape parameter, respectively. The mean and variance of the task response time are given by [11]

$$E[X] = \frac{1}{\mu}[\psi(\alpha+1)-\psi(1)], \quad (2)$$

$$V[X] = \frac{1}{\mu^2}[\psi'(1)-\psi'(\alpha+1)], \quad (3)$$

where $\psi(.)$ and its derivatives are the digamma and polygamma functions.

From Eqs. (2) and (3), it is clear that the distribution in Eq. (1) is completely determined by the mean and variance of the task response time. The rationale behind the use of this distribution, instead of the exponential distribution, is that it can capture both heavy-tailed and short-tailed task behaviors depending on the parameter settings and meanwhile, it degenerates to the exponential distribution at $\alpha = 1$ and $E[X] = 1/\mu$. As we shall see in the following subsection, this distribution significantly outperforms the exponential distribution in terms of tail latency predictive power for all the cases studied.

The implication of the above black box approximation is significant. It allows not only the task performance of a task mapped to a diverse range of subsystems to be captured by a unified distribution function, but also the request response time distribution and hence the tail SLO for the entire task-partitioning-merging system to be derived. To see why this is the case, one notes that with all the task subsystems in Fig. 1a being viewed as black boxes, one effectively transforms the task-partitioning-merging problem into a split-and-merge model [16] whose distribution function can be expressed as follows, assuming the task response times for tasks mapped to different subsystems are independent random variables,

$$F_{(N)}(x) = \begin{cases} \prod_{i=1}^{N}(1-e^{-\mu_i x})^{\alpha_i} & x > 0, \\ 0 & \text{otherwise}, \end{cases} \quad (4)$$

Now assume that the parallel subsystems are homogeneous, the distribution function can be further simplified as,

$$F_{(N)}(x) = \begin{cases} (1-e^{-\mu x})^{N\alpha} & x > 0, \\ 0 & \text{otherwise}, \end{cases} \quad (5)$$

With Eq. (5), it can be easily shown that the p-th percentile request response time x_p can be written as,

$$x_p = -\frac{1}{\mu}\log\left(1-\left(\frac{p}{100}\right)^{\frac{1}{N\alpha}}\right) \quad (6)$$

Since x_p is a function of μ and α, which in turn, are functions of $E[X]$ and $V[X]$ of the task response time (according to Eqs. (2) and (3)), a link between any given tail SLO in terms of x_p and p, and $E[X]$ and $V[X]$ is established. The implication of this result is significant. On one hand, with any given tail SLO, the resulting $E[X]$ and $V[X]$ can serve as the task response time budgets for highly scalable, distributed task-level resource provisioning. On the other hand, with given measured task response time statistics in terms of $E[X]$ and $V[X]$, whether the system meets the target tail SLO or not can be accurately predicted. In the following two subsections, we test the performance of this prediction model at the subsystem and system levels, separately.

2.2 Subsystem Tail Latency Prediction

In this section, we test the accuracy of the proposed prediction model against a wide range of subsystems including pure model-based subsystems, hybrid measurement-and-model-based subsystems, as well as a pure measurement-based subsystem.

For pure model-based and hybrid subsystems, we consider a typical subsystem setup given in Fig. 1b. It includes a dispatcher and three replicated servers. A task arriving at the subsystem is distributed to server replicas by a dispatcher based on a predetermined policy. Each server replica is modeled as an M/G/1 queuing system. Namely, for all the cases studied, the task arrival process is modeled as a Poisson process, which is considered a good model for scale-out workloads [19]. Both model-based and measurement-based service time distribution functions are considered, including the following,

– Empirical distribution measured from a Google search test leaf node provided in [18], which has a mean service time of 4.22ms, a coefficient of variance (CV) of 1.12, and the largest tail value of 276.6ms;

– A heavy-tailed truncated Pareto distribution [2] with the same mean service time, i.e., 4.22ms, and a CV of 1.2, resulting in the corresponding parameters: the shape $\alpha = 2.0119$, the lower bound $L = 2.14$ms, and the upper bound $H = 276.6$ms, which is set at the same maximum value of the empirical distribution above.

– Weibull distribution [6] also with the same mean service time and a CV of 1.5, resulting in the corresponding parameters: the shape parameter $\alpha = 0.6848 < 1$, i.e., a heavy-tailed distribution [6], and the scale parameter $\beta = 3.2630$.

We consider two task dispatching policies. The first policy is a popular one, known as the Round-Robin (RR) policy. In this policy, the dispatcher will send tasks to different server replicas in an RR fashion. The second policy is still RR, but it also allows redundant-task issue, a well-known tail-cutting technique [7, 24]. This policy allows one or more replications of a task to be sent to different server replicas in the subsystem. The replications may be sent in predetermined intervals to avoid overloading the server replicas. In our experiments, at most one task replication can be issued, provided that the original one does not finish within 10ms, which is around the 95th-percentile of the empirical distribution above.

For model-based and hybrid subsystems, the simulated tail task response time is compared against the tail response time predicted by the proposed prediction model, i.e., Eq. (1), which uses the simulated mean and variance of the task response time as input.

For the pure measurement-based subsystem, we implemented a Solr search engine [1] subsystem using a cluster of three Amazon EC2 m3.medium instances, each responsible for the same sample shard of the Wikipedia index. The dispatching policy is, again, RR. In this experiment, multiple client threads issue tasks to the servers in a stop-and-wait fashion, i.e., a client sends a task and then waits for the response before sending the next one. The random combination of the task flows from all the clients mimics a random arrival process. We focus on the cases when the number of clients is large enough to put a heavy load on the servers. Without knowing the inner-working of the VMs, we simply treat them as black boxes and the testing is solely based on the measured task response time statistics.

For the experiments on both pure model-based and hybrid subsystems, Fig. 3 presents the prediction errors for both the exponential and generalized exponential distributions at the load of 90%. First, we note that the generalized exponential distribution significantly outperforms the exponential distribution for all the cases studied. Second, the prediction errors for the generalized exponential distribution are consistently within 10% across the entire 95-99.9th percentile response time range, even for the RR case without tail cutting. These results confirm our postulation that the generalized exponential distribution function could accurately predict the task tail performance in the high load region.

Now we further test the performance of the generalized exponential distribution for the aforementioned measurement-based subsystem. The relative errors of the predicted task tail latencies against the measured ones are given in Table 1. In this experiment, as the number of clients increases, the aggregate task throughput increases and then levels off as the number of clients reaches 40, indicating that the subsystem is under heavy load condition. As one can see, the prediction errors reduce to less than 10% for all cases as the number of clients reaches 40, consistent with the performance data for the model-based and hybrid cases.

Table 1: The prediction errors for the measurement-based subsystem.

	Percentiles		
#clients	**95th**	**99th**	**99.9th**
20	-11.305	7.911	24.216
30	-3.233	5.295	13.429
40	-1.718	5.452	2.974
50	0.703	2.015	-1.381

2.3 System Tail Latency Prediction

In this section, we evaluate the accuracy of our generalized exponential distribution model as the system scales out. We consider the task-partitioning-merging system in Fig. 1a with $N = 10, 100, 500,$ and 1000 nodes for all the previously studied model-based and hybrid subsystems. Fig. 4 presents the prediction errors at different load levels for the 99th percentile request response times. Again, for all the cases studied, the errors are within 10% at the load of 90%. Even at the load of 80%, the prediction errors are with 10% and 20% for the cases with and without tail cutting, respectively.

As a work in progress, the testing of the proposed prediction model for a complete Solr-based search engine on Amazon EC2 is underway. The above testing results at both subsystem and system levels give us the confidence to expect that the results from this testing case will also be fairly accurate.

3 Facilitating Resource Provisioning

In this section, we discuss how the above prediction model may be used to facilitate highly scalable, explicitly tail-constrained resource provisioning. For ease of discussion, we use the following example scenario as a guide throughout the discussion. Assume that a content service provider wants to outsource its OLDI scale-out services to a cloud service provider. With a given size of parallel searchable database D (e.g., an entire index as in a Web search engine) and monetary budget C, the content service provider wants to know whether or not the service to be deployed may sustain R requests per second, while meeting the tail SLO, i.e., the p-th-percentile request response time of L ms.

An ad hoc approach is to immediately deploy the service to a certain scale and at runtime, scale out/up or down the system dynamically in a pay-as-you-go manner to meet the performance targets or monetary budget. Without an initial estimation, however, such approaches run the risk of either over budgeting or failing to meet SLO and/or targeted request throughput performance. Moreover, using the pay-as-you-go service for dynamic resource provisioning is generally much more

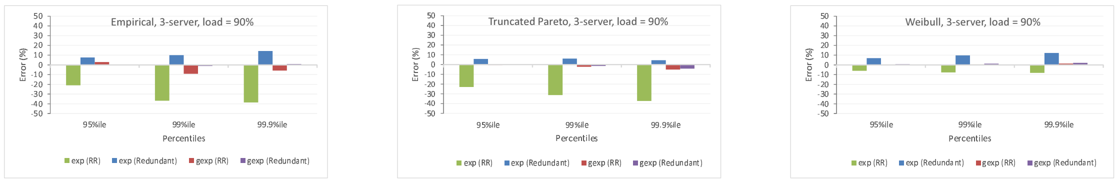


Figure 3: The prediction errors for both model-based and hybrid subsystems with the Round-Robin (RR) and redundant-task-issue policies.

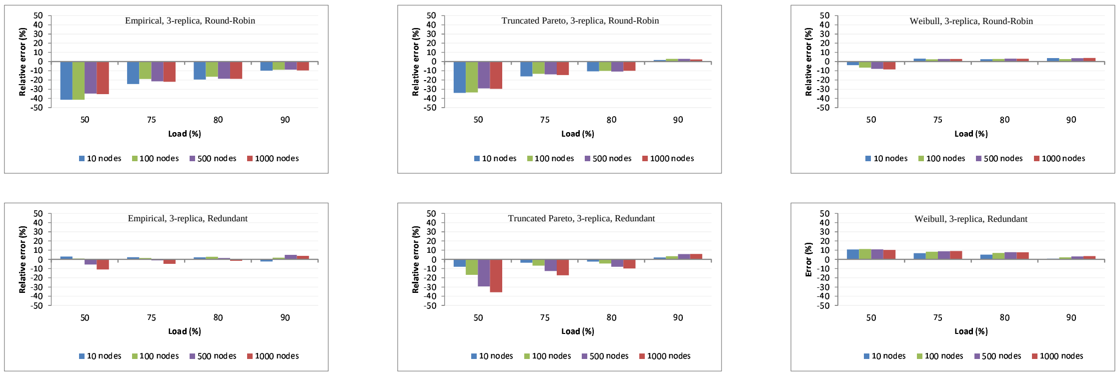


Figure 4: The prediction errors for the system composed of both model-based and hybrid subsystems with the Round-Robin (upper three plots) and redundant-task-issue (lower three plots) policies.

expensive than static resource reservation for resource planning [3]. Given the sheer size of the system to be deployed, it is of paramount importance to develop an offline, highly scalable resource provisioning approach that can provide a quick initial assessment of whether the performance targets and monetary budget can be met or not.

The idea of our approach is sketched by the following tail-constrained resource provisioning procedure, in the context of the above example scenario:

– For a desired type of VMs with, e.g., given CPU speed, memory size, and pricing model, build a replicated server cluster subsystem in the cloud using m (two to three) VMs by replicating a portion of the total database, i.e., D/N, to all the VM replicas, where N, an integer value, may be selected in such a way that D/N can fit comfortably in the memory in each VM;

– Measure the mean and variance of task response time in the cluster running a task scheduling policy, at desired task rate $\lambda = R$;

– Find the parameters of the generalized exponential distribution in Eq. (1) by plugging in the measured mean and variance task latency into Eqs. (2) and (3), respectively;

– Estimate the p-th-percentile request response time x_p based on Eq. (6);

– Finally, x_p is compared against L and the total cost for running N VM clusters with m each is compared against the associated budget C, to see if both the tail SLO and monetary budget are met. If both are met, a feasible tail-constrained resource provisioning is found. Otherwise, the performance targets and/or budget are revised and then rerun the procedure. Note that if x_p is found well below L, one may consider reducing N and/or m and see if it is still below L. This iterative testing can help minimize the cost.

4 Conclusions

This paper proposed a simple prediction model to predict the tail SLOs for scale-out applications involving task-partitioning-merging. The required inputs to the prediction model are only the mean and variance of task response time for a task mapped to a subsystem. The experimental results showed that the prediction model yields accurate prediction with errors consistently within 10% at the server loads of 90% or higher, providing a much needed prediction tool to facilitate tail-constrained resource provisioning for scale-out applications.

This is a work in progress. A full-fledged testing of the proposed prediction model in Amazon EC2 cloud is currently underway.

5 Acknowledgments

We would like to thank anonymous reviewers for their helpful comments and suggestions on improving the presentation of the paper.

References

[1] Solr. `http://lucene.apache.org/solr/`.

[2] Aban, I. B., Meerschaert, M. M., and Panorska, A. K. Parameter Estimation for the Truncated Pareto Distribution. *Journal of the American Statistical Association 101*, 473 (Mar. 2006), 270–277.

[3] Barroso, L. A., Clidaras, J., and Hölzle, U. The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines, Second edition. *Synthesis Lectures on Computer Architecture 8*, 3 (July 2013), 1–154.

[4] Barroso, L. A., Dean, J., and Hölzle, U. Web Search for a Planet: The Google Cluster Architecture. *IEEE Micro 23*, 2 (Mar. 2003), 22–28.

[5] Barroso, L. A., and Hölzle, U. The Case for Energy-proportional Computing. *Computer 40*, 12 (2007), 33–37.

[6] Bolch, G., Greiner, S., de Meer, H., and Trivedi, K. S. *Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications*. Wiley-Interscience, 2006.

[7] Dean, J., and Barroso, L. A. The Tail at Scale. *Communications of the ACM 56*, 2 (Feb. 2013), 74–80.

[8] Delimitrou, C., and Kozyrakis, C. Quasar: resource-efficient and QoS-aware cluster management. *ACM SIGARCH Computer Architecture News 42*, 1 (apr 2014), 127–144.

[9] Ferguson, A. D., Bodik, P., Boutin, E., and Fonseca, R. Jockey : Guaranteed Job Latency in Data Parallel Clusters. In *7th ACM European Conference on Computer Systems* (2012), pp. 99–112.

[10] Gardner, K., Zbarsky, S., Doroudi, S., Harchol-Balter, M., Hyytiä, E., and Scheller-wolf, A. Reducing Latency via Redundant Requests: Exact Analysis. In *Proceedings of the ACM SIGMETRICS/international conference on Measurement and modeling of computer systems - SIGMETRICS '15* (2015).

[11] Gupta, R. D., and Kundu, D. Generalized exponential distributions. *Australian & New Zealand Journal of Statistics 41*, 2 (1999), 173–188.

[12] Jalaparti, V., Bodik, P., Kandula, S., Menache, I., Rybalkin, M., and Yan, C. Speeding Up Distributed Request-Response Workflows. In *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM* (New York, NY, USA, 2013), SIGCOMM '13, ACM, pp. 219–230.

[13] Jeon, M., Kim, S., Hwang, S.-w., He, Y., Elnikety, S., Cox, A. L., and Rixner, S. Predictive Parallelization: Taming Tail Latencies in Web Search. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval - SIGIR '14* (2014), pp. 253–262.

[14] Kingman, J. F. C., and Atiyah, M. F. The single server queue in heavy traffic. *Proceedings of the Cambridge Philosophical Society 57* (1961), 902–904.

[15] Köllerström, J. Heavy Traffic Theory for Queues with Several Servers. I. *Journal of Applied Probability 11*, 3 (1974), 544–552.

[16] Lebrecht, A., and Knottenbelt, W. J. Response Time Approximations in Fork-Join Queues. In *23rd Annual UK Performance Engineering Workshop (UKPEW)* (2007).

[17] Lo, D., Cheng, L., Govindaraju, R., Barroso, L. A., and Kozyrakis, C. Towards energy proportionality for large-scale latency-critical workloads. In *Proceeding of the 41st Annual International Symposium on Computer Architecture* (Piscataway, NJ, USA, 2014), ISCA '14, IEEE Press, pp. 301–312.

[18] Meisner, D., Junjie, W., and Wenisch, T. F. BigHouse: A simulation infrastructure for data center systems. In *Performance Analysis of Systems and Software (ISPASS), 2012 IEEE International Symposium on* (2012), pp. 35–45.

[19] Meisner, D., Sadler, C. M., Andre, L. B., Weber, W.-D., and Wenisch, T. F. Power Management of Online Data-Intensive Services. In *Proceedings of the 38th annual International Symposium on Computer Architecture* (2011), ISCA '11, pp. 319–330.

[20] Nishtala, R., Fugal, H., Grimm, S., Kwiatkowski, M., Lee, H., Li, H. C., McElroy, R., Paleczny, M., Peek, D., Saab, P., Stafford, D., Tung, T., and Venkataramani, V. Scaling Memcache at Facebook. In *Proceedings of the 10th USENIX conference on Networked Systems Design and Implementation - NSDI'13* (Apr. 2013), USENIX Association, pp. 385–398.

[21] Qiu, Z., and Perez, J. F. Evaluating the Effectiveness of Replication for Tail-Tolerance. In *2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing* (may 2015), IEEE, pp. 443–452.

[22] Reiss, C., Tumanov, A., Ganger, G. R., Katz, R. H., and Kozuch, M. A. Heterogeneity and dynamicity of clouds at scale. In *Proceedings of the Third ACM Symposium on Cloud Computing - SoCC '12* (oct 2012), pp. 1–13.

[23] Sani, S., and Daman, O. A. Mathematical Modeling in Heavy Traffic Queuing Systems. *American Journal of Operations Research 4* (2014), 340–350.

[24] Vulimiri, A., Sherry, J., Berkeley, U. C., Godfrey, P. B., Ratnasamy, S., and Shenker, S. Low Latency via Redundancy. In *CoNEXT'13* (2013), pp. 283–294.

[25] Wang, A., Venkataraman, S., Alspaugh, S., Katz, R., and Stoica, I. Cake: Enabling High-level SLOs on Shared Storage Systems. In *Proceedings of the Third ACM Symposium on Cloud Computing - SoCC'12* (2012), pp. 14:1–14:14.

[26] Wang, D., Joshi, G., and Wornell, G. Efficient Task Replication for Fast Response Times in Parallel Computation. *arXiv:1404.1328* (Apr. 2014), 1–20.

[27] Zhu, T., Tumanov, A., Kozuch, M. a., Harchol-Balter, M., and Ganger, G. R. PriorityMeister: Tail Latency QoS for Shared Networked Storage. pp. 1–14.