Tail Latency Prediction for Datacenter Applications in Consolidated Environments

Sami Alesawi[1,2], Minh Nguyen[1], Hao Che[1], Akshit Singhal[1]
[1]Department of Computer Science and Engineering, University of Texas at Arlington, Arlington, Texas, USA
[2]Faculty of Computing and Information Technology, King Abdulaziz University, Rabigh, Saudi Arabia
Email: salesawi@kau.edu.sa, mqnguyen@mavs.uta.edu, hche@cse.uta.edu, akshit.singhal2@mavs.uta.edu

***Abstract*—Consolidating applications is a practical necessity in today's datacenters to reduce cost and improve resource utilization. However, resource sharing among different applications may result in high latency in responses to user requests. Due to the lack of a performance model for tail latency of Fork-Join structures, which underlay the workflows of lots of datacenter applications, the current practice is to overprovision resource in an attempt to satisfy as many user requests as possible. However, this practice leads to low resource utilization. Therefore, it is of importance to have a performance model that can accurately predict tail latency in such an environment, especially at high load regions, where resource provisioning is desired at most. In this paper, we propose an analytical solution for the prediction of tail latency of a target application in a consolidated environment where it is mixed with other background applications. The proposed model is validated against simulation through extensive case studies. The experimental results show the effectiveness of the proposed model in tail latency prediction at high load region, yielding all the prediction errors well within 10% at the load of 75% or higher, making the model a valuable tool for resource provisioning and supporting scheduling decisions in datacenter clusters to guarantee user satisfactions.**

***Index Terms*—tail latency, Fork-Join queuing networks, consolidated datacenters, resource provisioning.**

I. Introduction

Parallel computing has become predominant in datacenters nowadays due to ever-increasing amount of data to be processed in datacenter applications. Such applications usually need to be split into smaller tasks that are executed concurrently on hundreds or thousands machines for faster response time. While parallel computing can improve responsiveness and scalability, it makes effective job scheduling and resource provisioning extremely challenging. For example, the run-time variabilities of distributed task execution times, especially in the presence of synchronization barriers, can result in highly variable job completion times.

Similarly, consolidating applications in datacenters becomes a necessity to reduce cost and improve the return on investment by increasing the utilization and allowing resource sharing among different applications [1]. Unfortunately, this comes at a price of poor user experience and high delays in processing user requests. Therefore, maintaining a good user experience with guaranteed low latency in response time has become the current trend in datacenters [2], [3]. This newly arising demand for providing users with guaranteed service level objectives (SLOs) provoked lots of researchers to engineer many novel solutions to this challenging problem [4]–[6]. However, due to the lack of a performance model for the problem, a common practice today for datacenter applications is to overprovision resources to ensure that a job can finish within the allocated time slot or meet a predefined SLO in terms of, e.g., tail latency. Consequently, to meet the performance targets for applications, today's datacenters typically run at 10%–50% of their capacities [7], [8]. This may result in high costs from both user's and service provider's points of view.

Fork-Join structures are basic building blocks that underlay the workflows of many datacenter applications, e.g., web search, machine-translation, and social networking [2]. In these structures, an incoming job spawns multiple tasks which are processed by multiple processing nodes in the system. The job is considered to be completed when all of its tasks are finished and all the partial results are merged, which is called *barrier synchronization*. Therefore, the slowest task determines the response time of such a job. This processing pattern exactly follows the classical Fork-Join queuing network (FJQN) model, which is notoriously hard to solve in queuing theory [9]. Most of previous works on FJQNs in the literature attempt to find the approximation for job mean response time [10]–[13] and its bounds [14], [15]. With the emergence of user-facing latency-sensitive applications, tail latency has drawn more attentions recently. Several recent works [16]–[19] attempted to provide analytical solutions for the tail latency in FJQNs. However, they primarily considered only a flow of jobs from a single application. The work in [19] included the approximation for the tail latency of consolidated applications but based on a black-box approach, which approximates the task response time distributions using the measured means and variances of task response times on the Fork nodes.

To the best of our knowledge, no previous work attempts to provide an analytical solution for FJQNs with consolidated workloads. This is the primary motivation for this research work. In this paper, we propose a closed-form solution, i.e., *a white-box approach*, to the approximation of the tail latency of a given target application in FJQNs with *a mixture of applications*, each following a different service time distribution. We will elaborate more on the effect of consolidated workloads on the accuracy of the prediction model through detailed scenarios, considering different distributions for different applications and different percentages of target application against the background applications. This was not adequately covered in [19] where only a trace-driven simulated result based on black-box measurements was presented as a shrunken section in the published work. For all the cases studied, the validation against simulation results shows that the proposed model yields all the prediction errors well within 10% at the load of 75% or higher. This indicates the effectiveness of the proposed model in predicting tail latency for a target application in a consolidated environment at high load regions, where resource provisioning is most desirable.

The rest of the paper is organized as follows. Section II presents the proposed prediction model for the tail latency of a target application in a mixture of consolidated applications. Section III shows experimental results for different scenarios of the consolidated workloads. Section IV reviews related work. Finally, Section V concludes the paper and discusses future work.

II. The Proposed Model

In this section, we present our proposed analytical solution to the approximation of the tail latency of a target application in a consolidated environment, i.e., a mixture of applications sharing common datacenter resources. We shall derive a closed-form approximate solution for the pth percentile of a target (or tagged) application in the mixture.

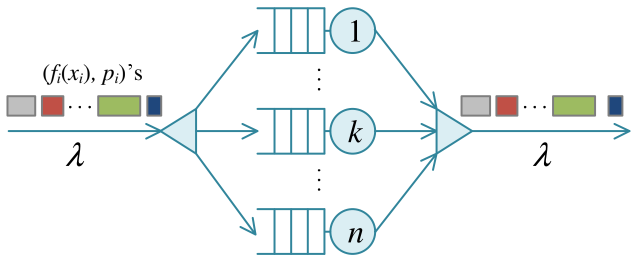


Fig. 1: A Fork-Join model with each node as an M/G/1 queue.

Consider a system running a mixture of applications $A_1, A_2, \ldots, A_m$ with corresponding weights $p_1, p_2, \ldots, p_m$ in the mixture as in Fig. 1. The applications flow into the system with an average arrival rate of λ.

Assume that service times of application A_i follow distribution $f_i(x)$, i.e., $X_i \sim f_i(x)$, whose mean and variance are μ_i and σ_i^2, respectively.

Let X be a random variable representing service times on each node in the system. For a long runtime and load balanced between nodes, the effective service time distribution puts on each node can be viewed as a mixture of individual distributions with their corresponding weights, i.e.,

$$X \sim f(x) = \sum_{i=1}^{m} p_i \cdot f_i(x). \tag{1}$$

The kth moment of X can be written as

$$\begin{aligned}\mu^{(k)} = \mathbb{E}[X^k] &= \sum_{i=1}^{m} p_i \cdot \mathbb{E}[X_i^k] \\ &= \sum_{i=1}^{m} p_i \cdot \mu_i^{(k)} = \mathbb{E}[\mathbb{E}[X^k|\mu_i]],\end{aligned} \tag{2}$$

where $\mu_i^{(k)}$ is the kth moment of X_i.

According to the law of total variance, the variance of X is given by

$$\begin{aligned}Var(X) &= \mathbb{E}[Var(X|\mu_i)] + Var(\mathbb{E}[X|\mu_i]) \\ &= \sum_{i=1}^{m} p_i \cdot \sigma_i^2 + \sum_{i=1}^{m} p_i \cdot \mu_i^2 - \left(\sum_{i=1}^{m} p_i \cdot \mu_i\right)^2\end{aligned} \tag{3}$$

In this paper, we assume that the applications arrive at the system following Poisson process, which is a reasonably acceptable model for datacenter applications in practice [20], and their tasks are randomly distributed to the Fork nodes. Therefore, each Fork node can be viewed as an M/G/1 queue system, i.e., a Poisson arrival process with a general service time distribution and one service center.

The first and second moments of the task waiting time at each Fork node, i.e., an M/G/1 queuing system, are given as follows [21],

$$\mathbb{E}[W] = \frac{\lambda \mathbb{E}[X^2]}{2(1-\rho)} = \frac{\rho \mathbb{E}[X]}{1-\rho}\left(\frac{1+C_X^2}{2}\right), \tag{4}$$

$$\mathbb{E}[W^2] = 2\mathbb{E}[W]^2 + \frac{\lambda \mathbb{E}[X^3]}{3(1-\rho)}, \tag{5}$$

where $\mathbb{E}[X^k]$ is the kth moment of the service time of the mixture given by Eq. (2); $C_X^2 = Var(X)/\mathbb{E}[X]^2$ is the squared coefficient of variation with $Var(X)$ being the variance given by Eq. (3); and $\rho = \lambda\mathbb{E}[X]$ is the utilization or load on each Fork node at average arrival rate λ.

Therefore, the mean and variance of response times of the target task at each Fork node can be written as,

$$\mathbb{E}[T] = \mathbb{E}[W] + \mathbb{E}[X_t], \tag{6}$$

$$\begin{aligned}Var(T) &= Var(W) + Var(X_t), \\ &= (\mathbb{E}[W^2] - \mathbb{E}[W]^2) + (\mathbb{E}[X_t^2] - \mathbb{E}[X_t]^2),\end{aligned} \tag{7}$$

where $\mathbb{E}[X_t]$ and $\mathbb{E}[X_t^2]$ are the first and second moments of the target task service time; and $\mathbb{E}[W]$ and $\mathbb{E}[W^2]$ are given in Eqs. (4) and (5).

It was proven that the waiting time distribution for G/G/m queue systems at heavy traffic conditions converges to an exponential distribution [22], [23]. Inspired by this result, the work in [18] postulated that the response time distribution can be approximated by a generalized exponential distribution as in Eq. (8), which outperforms the exponential distribution in term of tail latency prediction at high load regions for a wide range of service time distributions, including light-tailed, heavy-tailed, and empirical distributions,

$$F_T(x) = (1 - e^{-x/\beta})^{\alpha}, \quad x > 0,\ \alpha > 0,\ \beta > 0, \tag{8}$$

where α and β are shape and scale parameters, respectively.

Similarly, in this paper, the response time distribution of the target task on each Fork node is also approximated by the generalized exponential distribution as in Eq. (8), whose mean and variance are given by [24],

$$\begin{aligned} \mathbb{E}[T] &= \beta[\psi(\alpha+1)-\psi(1)], && (9) \\ Var(T) &= \beta^2[\psi'(1)-\psi'(\alpha+1)], && (10) \end{aligned}$$

where $\psi(.)$ and its derivative are the digamma and polygamma functions. Given service time distribution $f_i(x_i)$[1] and weight p_i $(i = 1, 2, \ldots, m)$ for each application, one can find parameters α and β for the target task by plugging the calculated mean and variance from Eqs. (6) and (7) into Eqs. (9) and (10), respectively, and solving this system of equations.

Assume that the jobs from the target application is forked to k nodes $(1 \le k \le N)$ with corresponding target task response time distributions $F_{T_j}(x)$'s $(j = 1, 2, \ldots, k)$, as in [19], the system response time distribution for the target application can be approximated as,

$$\begin{aligned} F(x) &= P(\max_{1\le j\le k} T_j \le x) \\ &\approx \prod_{j=1}^{k} F_{T_j}(x) = \prod_{j=1}^{k} (1-e^{-x/\beta_j})^{\alpha_j}, \qquad (11) \end{aligned}$$

and the pth percentile can be written as,

$$x_p = F^{-1}(p/100). \qquad (12)$$

In case all the Fork nodes are homogeneous, the target response time distribution can be simplified as,

$$F(x) = (1-e^{-x/\beta})^{k\alpha} \qquad (13)$$

from which the pth percentile can be derived as,

$$x_p = -\beta \log\left(1-(\frac{p}{100})^{\frac{1}{k\alpha}}\right). \qquad (14)$$

Eqs. (12) and (14) show that the pth percentile, i.e., tail latency, of the target application can be expressed as a function of α_j's and β_j's and in turn a function of means and variances of the target task response times, according to Eqs. (9) and (10). Clearly, this establishes a link between job-level SLO, i.e., the job tail latency requirement, and task budgets.

III. Experiments and Results

In this section, we validate the predicted tail latencies, e.g., the 99th percentile of response times, for a target application in a system with a mixture of applications against those from the simulation. The accuracy of the prediction is measured by the relative error between the predicted value from the proposed model, t_{pred}, and the simulated one, t_{sim},

$$err = 100 \cdot \frac{t_{\text{pred}} - t_{\text{sim}}}{t_{\text{sim}}}$$

[1]Indeed, the approximate solution in this paper requires only the first three moments of the applications' tasks, not the entire task distributions.

To illustrate the effectiveness of the proposed model, here we consider some typical scenarios with two classes of applications, a target application, which needs to be kept track, and a background application, which represents the remaining applications running in the system. The validation is performed under different settings for the target and background applications, including simple cases with the same type of service time distribution with different parameters and complicated cases with two different distributions. We incorporate both light-tailed and heavy-tailed distributions in the scenarios to represent different applications. In particular, we consider the following distributions:

– A light-tailed Exponential distribution (Exp) whose CDF is defined as [25]

$$F_{X_i}(x) = 1 - e^{-\frac{x}{\mu}} \quad x \ge 0, \qquad (15)$$

where μ is the mean service time $(\mu > 0)$,

– A heavy-tailed truncated Pareto distribution (TPa) [25] whose CDF is given by

$$F_{X_i}(x) = \frac{1-(L/x)^\alpha}{1-(L/H)^\alpha}, \qquad (16)$$

where α is a shape parameter; L and H are lower and upper bounds, respectively.

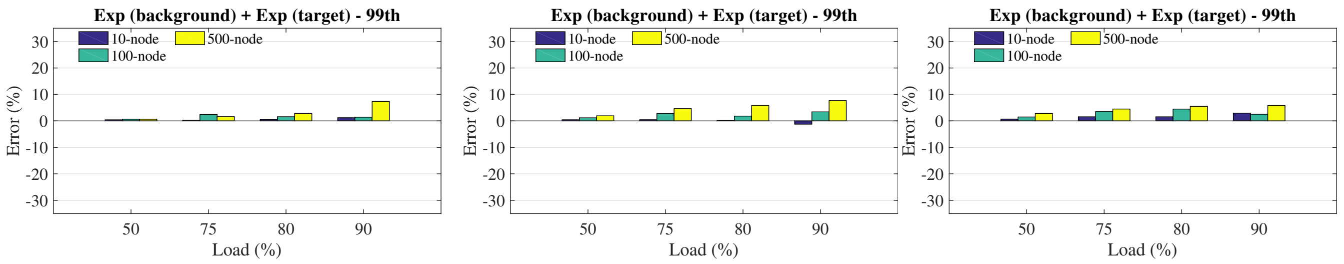
We report the results for the systems with different numbers of Fork nodes, i.e., $N = 10$, 100, and 500, and different weights for the target application in the mixture, i.e., 10%, 50%, and 90%, assuming that the tasks spawned from all the incoming jobs are randomly dispatched to N nodes in the system, i.e., $k = N$. Specifically, we consider three scenarios:

– **Scenario 1**–The same distribution (Fig. 2): Exponential distribution for both target and background applications with different mean service times, $\mu_{\text{tg}} = 13.78ms$ for the target and $\mu_{\text{bg}} = 4.22ms$ for the background.

– **Scenario 2**–The same distribution (Fig. 3): truncated Pareto distribution for both target and background applications with the same coefficient of variation $CV = 1.2$ and different mean service times, $\mu_{\text{tg}} = 4.22ms$ for the target and $\mu_{\text{bg}} = 15.0ms$ for the background, which results in the distribution parameters $\alpha_{\text{tg}} = 2.0119, L_{\text{tg}} = 2.14ms, H_{\text{tg}} = 276.63ms$ for the target and $\alpha_{\text{bg}} = 2.0119, L_{\text{bg}} = 7.75ms, H_{\text{bg}} = 276.63ms$ for the background.

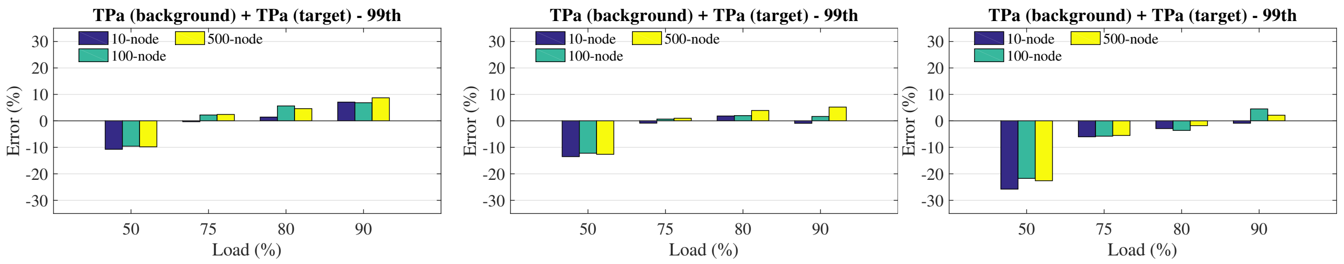
– **Scenario 3**–Different distributions (Fig. 4): Exponential distribution with mean service time $\mu_{\text{tg}} = 4.22ms$ for the target application and truncated Pareto distribution with coefficient of variation $CV = 1.2$ and mean service times $\mu_{\text{bg}} = 15.0ms$ for the background application.

Figs. 2–4 show the prediction errors for all the cases studied with different load regions, i.e., 50%, 75%, 80%, and 90%. The results show that the prediction model is able to yield quite accurate predictions for the 99th percentiles of job response times, with most of the errors within 10% at the load of 75% or higher. This makes the proposed solution a powerful tool for resource provisioning in datacenters with consolidated applications at high load regions, where precise resource provisioning is most desirable.



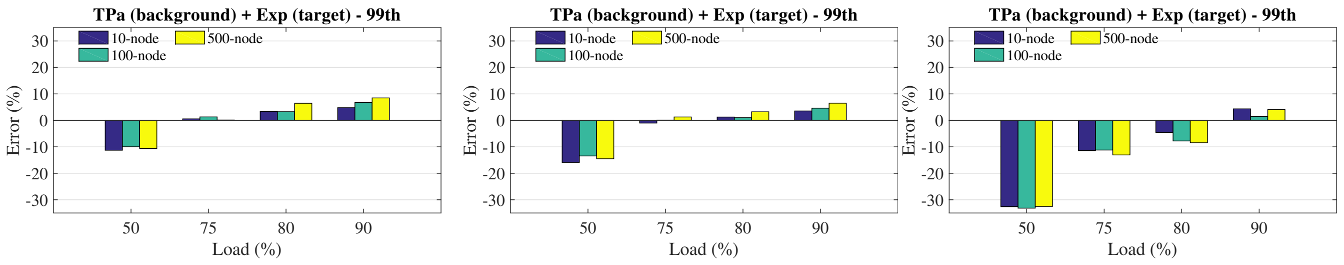
(a) Target = 10% (b) Target = 50% (c) Target = 90%

Fig. 2: Prediction errors for the cases of different exponential distributions for both background and target tasks.



(a) Target = 10% (b) Target = 50% (c) Target = 90%

Fig. 3: Prediction errors for the cases of different truncated Pareto distributions for both background and target tasks.



(a) Target = 10% (b) Target = 50% (c) Target = 90%

Fig. 4: Prediction errors for the cases of truncated Pareto distribution for background tasks and exponential distribution for target tasks.

IV. Related Work

Fork-Join structures underlay many datacenter applications, in which the workflows are usually handled by a large number of nodes and the partial results from those nodes are then merged. Fork-Join structures are traditionally modeled by Fork-Join queuing networks (FJQNs) [9]. FJQN models have been studied extensively in the literature. To date, the exact solution exists for a two-node network only [10], [26]. Most of previous research efforts mainly focus on the approximation of job mean response time [10]–[12] and its bounds [14], [15], [17]. Some works attempt to find the approximation of response time distribution for the networks applying simple queuing models for each Fork node, e.g., M/M/1 [27] or M/M/k [28], i.e., Poisson arrival process and exponential service times with one or more servers. For general service time distribution, the hybrid approach, which combines analysis and simulation, are usually used to derive the approximation for job mean response time [10], [13].

The approximation of tail latency for homogeneous FJQNs with phase-type service time distributions is introduced in a recent work [16], which is based on the analytical results from single-node and two-node networks. Unfortunately, the computational complexity of this approach renders it inapplicable to online resource provisioning. Also, it only covers a limited design space and thus cannot be used to facilitate resource provisioning in practice. In [18], [19], the authors propose to use a black-box approach for the approximation of tail latency at high load regions, in which each Fork node was treated as a black-box. The approach requires only measured means and variances of response times at the Fork nodes as input.

To our knowledge, no previous research effort provides analytical solutions for the approximation of the tail latency in FJQNs with consolidated workloads. The work in [19] does consider consolidated workloads but using the black-box approach, i.e., approximating task response time distributions based on the measured means and variances of response times on the Fork nodes. Here we present an analytical solution, i.e., a white-box approach, for the prediction of the tail latency in a consolidated environment and elaborate more on the consolidated impact on the accuracy of the prediction model. The proposed solution could be of great help for future research works that target the consolidated applications in FJQNs.

V. Conclusions and Future Work

In this paper, we presented an analytical solution accompanied by several case studies for the prediction of tail latency regarding the consolidated applications in datacenter environments. Provided the results of three scenarios with different load characteristics, the prediction model has proven to be reliable, yielding all the errors well within the acceptable window of accuracy, 10% at the load of 75% or higher, for all the cases studied. These encouraging results suggest that the proposed model could be used to translate user demands into task budgets which in turn would help a scheduler make better scheduling decisions to satisfy user requirements. We look forward to extending this work for more sophisticated cases that include multiple processing stages, which is commonly in many production environments [29].

VI. Acknowledgments

This work was partially supported by NSF grants XPS-1629625 and SHF-1704504. The authors would like to thank the anonymous reviewers for their valuable feedback and helpful suggestions. The first author would also like to thank the Faculty of Computing and Information Technology, King Abdulaziz University, Rabigh for the continuous help and support.

References

[1] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: research problems in data center networks," *ACM SIGCOMM computer communication review*, vol. 39, no. 1, pp. 68–73, 2008.

[2] J. Dean and L. A. Barroso, "The tail at scale," *Communications of the ACM*, vol. 56, no. 2, pp. 74–80, 2013.

[3] W. Chen, J. Rao, and X. Zhou, "Preemptive, low latency datacenter scheduling via lightweight virtualization," in *2017 USENIX Annual Technical Conference (USENIX ATC 17)*, 2017, pp. 251–263.

[4] V. Jalaparti, P. Bodik, S. Kandula, I. Menache, M. Rybalkin, and C. Yan, "Speeding up distributed request-response workflows," in *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4. ACM, 2013, pp. 219–230.

[5] K. Karanasos, S. Rao, C. Curino, C. Douglas, K. Chaliparambil, G. M. Fumarola, S. Heddaya, R. Ramakrishnan, and S. Sakalanaga, "Mercury: Hybrid centralized and distributed scheduling in large shared clusters." in *USENIX Annual Technical Conference*, 2015, pp. 485–497.

[6] K. Ousterhout, P. Wendell, M. Zaharia, and I. Stoica, "Sparrow: distributed, low latency scheduling," in *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*. ACM, 2013, pp. 69–84.

[7] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, and M. A. Kozuch, "Heterogeneity and Dynamicity of Clouds at Scale," in *SoCC '12*, 2012, pp. 7:1–7:13.

[8] C. Delimitrou and C. Kozyrakis, "Quasar: Resource-Efficient and QoS-aware Cluster Management," in *ASPLOS '14*, 2014, pp. 127–144.

[9] A. Thomasian, "Analysis of Fork/Join and Related Queueing Systems," *ACM Computing Surveys*, vol. 47, no. 2, pp. 1–71, 2014.

[10] R. Nelson and A. N. Tantawi, "Approximate Analysis of Fork/Join Synchronization in Parallel Queues," *IEEE Transactions on Computers*, vol. 37, no. 6, pp. 739–743, 1988.

[11] E. Varki, "Response time analysis of parallel computer and storage systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 12, no. 11, pp. 1146–1161, 2001.

[12] F. Alomari and D. A. Menasce, "Efficient Response Time Approximations for Multiclass Fork and Join Queues in Open and Closed Queuing Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 6, pp. 1437–1446, 2014.

[13] R. J. Chen, "A hybrid solution of fork/join synchronization in parallel queues," *IEEE Transactions on Parallel and Distributed Systems*, vol. 12, no. 8, pp. 829–845, 2001.

[14] S. Balsamo, L. Donatiello, and N. M. Van Dijk, "Bound performance models of heterogeneous parallel processing systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 9, no. 10, pp. 1041–1056, 1998.

[15] R. J. Chen, "An Upper Bound Solution for Homogeneous Fork/Join Queuing Systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 5, pp. 874–878, 2011.

[16] Z. Qiu, J. F. Pérez, and P. G. Harrison, "Beyond the Mean in Fork-Join Queues: Efficient Approximation for Response-Time Tails," *Performance Evaluation*, vol. 91, pp. 99–116, 2015.

[17] A. Rizk, F. Poloczek, and F. Ciucu, "Computable Bounds in Fork-Join Queueing Systems," in *SIGMETRICS '15*, 2015, pp. 335–346.

[18] M. Nguyen, Z. Li, F. Duan, H. Che, and H. Jiang, "The tail at scale: how to predict it?" in *8th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 16)*. USENIX Association, 2016.

[19] M. Nguyen, S. Alesawi, N. Li, H. Che, and H. Jiang, "Forktail: a black-box fork-join tail latency prediction model for user-facing datacenter workloads," in *Proceedings of the 27th International Symposium on High-Performance Parallel and Distributed Computing*. ACM, 2018, pp. 206–217.

[20] D. Meisner, C. M. Sadler, A. L. Barroso, W. D. Weber, and T. F. Wenisch, "Power Management of Online Data-Intensive Services," in *Proceedings of the 38th annual International Symposium on Computer Architecture*, ser. ISCA '11. ACM, 2011, pp. 319–330.

[21] L. Kleinrock, *Queueing Systems, Vol. 1: Theory*. Wiley-Interscience, 1975.

[22] J. Kingman, "The single server queue in heavy traffic," in *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 57, no. 04. Cambridge Univ Press, 1961, pp. 902–904.

[23] J. Köllerström, "Heavy traffic theory for queues with several servers. i," *Journal of Applied Probability*, vol. 11, no. 03, pp. 544–552, 1974.

[24] R. D. Gupta and D. Kundu, "Theory & methods: Generalized exponential distributions," *Australian & New Zealand Journal of Statistics*, vol. 41, no. 2, pp. 173–188, 1999.

[25] M. Harchol-Balter, *Performance Modeling and Design of Computer Systems: Queueing Theory in Action*, 1st ed. Cambridge University Press, 2013.

[26] L. Flatto and S. Hahn, "Two Parallel Queues Created by Arrivals with Two Demands I," *SIAM Journal on Applied Mathematics*, vol. 44, no. 5, pp. 1041–1053, 1984.

[27] S. Balsamo and I. Mura, "Approximate response time distribution in Fork and Join systems," in *SIGMETRICS '95/PERFORMANCE '95*, 1995, pp. 305–306.

[28] S. S. Ko and R. F. Serfozo, "Response Times in M/M/s Fork-Join Networks," *Advances in Applied Probability*, vol. 36, no. 3, pp. 854–871, 2004.

[29] E. Boutin, J. Ekanayake, W. Lin, B. Shi, J. Zhou, Z. Qian, M. Wu, and L. Zhou, "Apollo: Scalable and coordinated scheduling for cloud-scale computing." in *OSDI*, vol. 14, 2014, pp. 285–300.