ForkMV: Mean-and-Variance Estimation of Fork-Join Queuing Networks for Datacenter Applications

*

Prathyusha Enganti
Computer Science and Engineering
University of Texas at Arlington
Arlington(TX),USA
prathyusha.enganti@mavs.uta.edu

Todd Rosenkrantz
Computer Science and Engineering
University of Texas at Arlington
Arlington(TX),USA
stoddard.rosenkrantz@mavs.uta.edu

Lin Sun
Computer Science and Engineering
University of Texas at Arlington
Arlington(TX),USA
lxs5171@mavs.uta.edu

Zhijun Wang
Computer Science and Engineering
University of Texas at Arlington
Arlington(TX),USA
zhijun.wang@uta.edu

Hao Che
Computer Science and Engineering
University of Texas at Arlington
Arlington(TX),USA
hche@cse.uta.edu

Hong Jiang
Computer Science and Engineering
University of Texas at Arlington
Arlington(TX),USA
hong.jiang@uta.edu

2022 IEEE International Conference on Networking, Architecture and Storage (NAS) | 978-1-6654-5408-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/NAS55553.2022.9925531

***Abstract*—The Fork-Join structure underlays many distributed computing applications in data centers. In this paper, we develop a technique, called ForkMV, to estimate the mean and variance of request response time for Fork-Join queuing networks (FJQNs) with both short-tailed and long-tailed service time distributions and arbitrary request fanout degrees (i.e., the number of Fork nodes). Specifically, for an FJQN with any given service time distribution of practical interests, ForkMV is able to estimate the mean and variance of request response time, accurate enough to facilitate effective resource allocation for data center applications. The test results indicate that in the entire range of the fanout degrees being tested (i.e., [1, 4000]), ForkMV is able to estimate the mean response time within 5% and 15% and variance response time within 15% and 10% of the simulation results for short-tailed exponential service distribution and long-tailed truncated Pareto distribution, respectively, at the 90% load or higher.**

***Index Terms*—Mean-latency, Mean-latency with bounded variance, Tail latency, Fork-Join queuing networks.**

I. Introduction

Today's data center applications can be broadly classified into two types, i.e., online user-facing and background batch applications [1]. While the former usually needs to meet stringent request tail-latency service level objectives (SLOs), the latter may need to satisfy request mean-latency or mean-latency with a bounded variance SLOs. For both types of applications, a user request or job (in this paper, we use request and job interchangeably) may spawn multiple tasks (in this paper, the exact number of tasks spawned is called the job fanout degree), which are dispatched to different server machines to be queued and processed, and finally the processing results are merged and returned to the user. Notable examples are inverse indexing for web search [2], a user-facing application, and MapReduce-based background batch applications [3]. This request process follows a Fork-Join structure with synchronization barrier (i.e., the slowest task determines the request response time), which can be modeled as Fork-Join queuing network model (FJQN). FJQNs, which find their roots in queuing theory, are popular models for the performance analysis of such workloads. Consequently, to facilitate proper server resource allocation to meet the request tail-latency, mean latency and mean latency with bounded variance SLOs for data center applications, it is important to estimate the tail, mean, and variance of the request latency for an FJQN that underlays the Fork-Join structure and the workload in question [4]–[6]. Due to the structure of the workload and server resource variability, the task service time distribution for the queuing servers may be short or long tailed.

Unfortunately, FJQNs are notoriously difficult to solve, with closed-form solutions available only for an FJQN with M/M/1 queuing servers and fanout degree of two or two Fork nodes. Although approximate solutions are available, the design space they cover falls short of the above requirements. Table I depicts the desired design space to be covered and the notable existing approximate solutions, together with the areas in the design space they cover. As one can see, except for ForkTail [5] and ForkMean [6], all other solutions fail to cover the case of long-tailed distributions. The area that ForkTail covers allows it to apply to user-facing applications with request tail-latency SLOs. Yet, it alone is insufficient to deal with user-facing applications, which call for latency

978-1-6654-5408-7/22/$31.00 ©2022 IEEE

TABLE I
EXPLORING DESIGN SPACE

Solution	Fanout degree	Tail		Performance measure		
		Short	Long	Tail	Mean	Variance
EV [10]	≤100	✓	×	×	✓	×
AM [11]	≤100	✓	×	×	✓	×
VMC [12]	≤100	✓	×	×	✓	×
GV [7]	≤20	✓	×	×	✓	✓
VM [13]	≤20	✓	✓	×	✓	×
NT [9]	≤32	✓	×	×	✓	×
ForkMean [6]	≥100	✓	✓	×	✓	×
EAT [16]	any	✓	×	✓	×	×
ForkTail [5]	any	✓	✓	✓	×	×
ForkMV	any	✓	✓	×	✓	✓

mean and variance estimation as well. ForkMean provides the similar coverage for background batch applications with request mean-latency SLOs, except that it was not designed to fit the case where fanout degrees are smaller than 100. To the best of our knowledge, GV [7] is the only solution available that is capable of estimating latency variance, which however, only works for the case where the fanout degree is small. In other words, the design space covered by the existing solutions are insufficient to facilitate resource allocation for cloud applications in general.

In this paper, we put forward ForkMV, a latency mean and variance estimation solution for FJQNs that fills the void of the design space in Table I, and hence, together with ForkTail, provides a much needed tool to facilitate effective resource allocation for cloud applications. Just like ForkMean [6], ForkMV is derived based on ForkTail [4], [5]. However, ForkMV employs a more elaborate modeling technique to allow accurate estimation of the mean latency at any given fanout degree, including those smaller than 100. Moreover, ForkMV also allows the latency variance at any fanout degree to be estimated with high accuracy. The testing results demonstrate that ForkMV can estimate latency mean and variance consistently within 15% of the simulation results at any load level for FJQN with short-tailed M/M/1 queuing servers and within 10% of the simulation results at 90% load or higher for FJQN with a long-tailed Truncated Pareto distribution.

TABLE II
NOTATION SUMMARY

Symbol	
λ	Arrival rate
n	Number of Tasks
μ	Service rate
$G(t)$	Service time cumulative distribution function (CDF)
α	Shape parameter of Truncated Pareto service time distribution
L	Lower bound of Truncated Pareto service time distribution
H	Upper bound of Truncated Pareto service time distribution
$F_T(t)$	Task response time cumulative distribution function (CDF)
β	Scale parameter of generalized exponential distribution
η	Shape parameter of generalized exponential distribution
E	Mean of task response time
V	Variance of task response time
ρ	Utilization
$F_J(t,n)$	CDF of the job response time
t_p	p^{th}-percentile job tail-latency
Δ_m	Tail latency to Mean latency gap
t_p and t_m	Exact tail latency and mean latency
t_p^a and t_m^a	Analytically estimated tail and mean latency
w	The Right Quantile Weight of $G(t)$
q	Quantile
Δ_s	Tail latency to standard Deviation latency gap
t_s and t_s^a	Exact and analytically estimated standard deviation latency
G_{ge}	Generalized exponential distribution function

II. RELATED WORK

For FJQNs, the exact closed-form solution is only available for a 2-node FJQN with M/M/1 fork queues, which alludes to the complexity in solving FJQNs [8], [9]. This motivated the development of various response time approximation techniques [[10], [11], [12], [7], [13], [9], [6]], most of which can only approximate FJQN with M/M/1 queues. Others focus on getting the response time bounds [14], [15].

Table I covers some major approximation techniques. EV [10] derives response time approximation using mean value analysis (MVA) for FJQNs with exponential service time distributions. AM [11] uses MVA to come up with the response time approximation of heterogeneous multi-class Fork-Join structure for open and closed queuing networks, again for exponential service time distributions only. VMC [12] works with FJQNs with M/M/1 queues and puts forward an algorithm to find optimistic and pessimistic bounds and mean response time approximation. Although designed for exponential service time distribution, VMC was applied to and tested for long tail distributions as well. GV [7] estimates the mean response time of a cloud computing system by modeling virtual machines to act as devices in FJQNs with M/M/1 queues and estimates the mean response time by using a combination of analytical approach with an empirical one. It only studies the case with fanout degrees up to 20. VM [13] provides a heuristic to estimate mean response time for FJQNs with Erlang-2 (E2) and Hyperexponential-2 (H2) service time distributions with Poisson arrivals. The authors study fanout degrees only up to 20. NT [9] uses a scaling approximate technique to come up with mean response time approximation

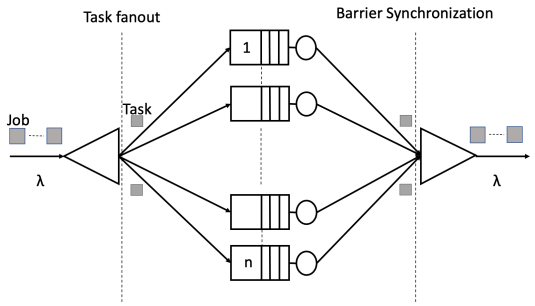


Fig. 1. A M/G/1-based FJQN model

for FJQNs with short-tailed service time distributions. ForkMean [6] can estimate the mean response time for FJQNs with both short and long tailed service time distributions with the estimation errors becoming relatively large at low fanout degrees, especially for short-tailed distributions.

To the best of our knowledge, GV [7] is the only work that provides variance response time approximation along with mean approximation. GV provides a mathematical model and a neural networks model to approximate the variance for fanout degrees no larger than 20. The neural networks model would need one to train at every fanout degree and hence is not practical for higher degrees of fanout. ForkTail [5] and EAT [16] are tail response time estimation solutions. While EAT works for FJQNs with short-tailed service time distributions, ForkTail works for FJQNs with both short and long tailed distributions.

III. ForkMV

This section is organized as follows. Section III-A defines the FJQN models to be studied. Section III-B summarizes the prior results ForkMV is based upon. Section III-C presents ForkMV.

A. Model

We consider M/G/1-based FJQNs as depicted in Fig. 1. Job arrivals follow a Poisson arrival process with mean arrival rate, λ. Note that the Poisson process has been considered to be a good model to characterize the job arrival processes in operational data center clusters [17]. Each job spawns n tasks, which are dispatched to n Fork nodes to be queued and serviced by a single server based on an M/G/1 FIFO queue, where G represents a known service time distribution of practical interests with cumulative distribution function (CDF) denoted as $G(t)$, which may be short, medium, or long tailed. In this paper, we only consider the homogeneous case where queuing servers at all the Fork nodes share the same CDF, $G(t)$, although in principle, ForkMV can be generalized to deal with heterogeneous cases as well. This means that an M/G/1-based FJQN is uniquely determined by $G(t)$. The triangle on the right represents the synchronization barrier, meaning that the job does not complete until its slowest task finishes. In other words, the job response time or latency (in this paper, we use response time and latency interchangeably) is determined by its slowest task.

To cover a sufficiently wide range of FJQNs of practical interests, in this paper, we shall focus on the following two FJQNs:

- Short-tailed exponential service time distribution:

$$G(t) = 1 - e^{-\mu t}, \tag{1}$$

where μ is the mean service rate.

- Long-tailed Truncated Pareto service time distribution:

$$G(t) = \frac{1 - L^{\alpha} t^{-\alpha}}{1 - (\frac{L}{H})^{\alpha}}, \tag{2}$$

where α is the shape parameter and L and H are the lower and upper bounds, respectively. Here, α determines the heaviness of the tail and according to [6], for real data center workloads, $\alpha < 2$. Hence, to cover the workloads of practical interests, in all our case studies, we set $\alpha = 2.0119$, to cover the worst case scenario.

B. Prior Work

This section summarises the results from the prior work [4]–[6] the current solution is based on.

ForkTail [4], [5]: In the design of ForkTail, two approximations are made. The first approximation is to assume that the task response time CDF, $F_T(t)$, can be adequately expressed as a generalized exponential distribution, i.e.,

$$F_T(t) \approx \begin{cases} (1 - e^{-\beta t})^{\eta} & , t > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

Where β and η are the scale and shape parameters, respectively, which satisfy the following equations,

$$E = \frac{1}{\beta}[\Psi(\eta + 1) - \Psi(1)], \tag{4}$$

$$V = \frac{1}{\beta^2}[\Psi'(1) - \Psi'(\eta + 1)], \tag{5}$$

Where Ψ and its derivative(s) are digamma and polygamma functions, respectively, and E and V are the mean and variance of task response time. Hence, the two parameters are uniquely determined by E and V. In other words, E and V uniquely determine $F_T(t)$. Since we assume that the CDF for the task service time, $G(t)$, is a given, according to the Takacs recurrence theorem [20], we have,

$$E = E[s](1 + \frac{\rho}{1 - \rho} * \frac{1 + Cs^2}{2}), \tag{6}$$

and,

$$V = E\left[w\right]^2 + \frac{\lambda E\left[s^3\right]}{3\left(1 - \rho\right)} + E\left[s^2\right] - E\left[s\right]^2, \tag{7}$$

where $E[s^k]$ is the k^{th} moment of service time, which can be easily calculated given $G(t)$, $\rho = \lambda E[s]$ is the queuing server

utilization, Cs is the coefficient of variation of the service time, and $E[w] = \lambda E[s^2] / [2(1-\rho)]$ is the mean of the task waiting time. In other words, with any given FJQN with known $G(t)$, the task response time distribution is known and given by Eq. (3).

The second approximation made in ForkTail is to assume that the task response times for all the tasks belonging to the same job are independent random variables, and hence, according to the extreme value theorem [21], the CDF of the job response time, $F_J(t,n)$, the ordered statistics, can be expressed as follows,

$$F_J(t,n) = F_T(t)^n = \begin{cases} (1-e^{-\beta t})^{\eta n}, & t > 0 \\ 0 & \text{otherwise.} \end{cases} \quad (8)$$

With this CDF, the job mean, variance, and tail latency can be calculated analytically. In particular, the p^{th}-percentile job tail-latency of t_p time units can be written as,

$$t_p = F_J^{-1}(\frac{p}{100}, n) = \frac{-1}{\beta} ln(1 - (\frac{p}{100})(\frac{1}{\eta n})). \quad (9)$$

It turns out that ForkTail above provides sufficiently accurate estimation of the tail latency for resource allocation throughout the entire load range of $[0, 100]\%$ and 80% load or higher for short-tailed and long-tailed $G(t)$'s of practical interests, respectively [4]–[6]. This makes ForkTail a practically useful tool to facilitate effective resource allocation for user-facing applications.

ForkMean [6]: The design of ForkMean is based on two key observations. The first observation is that although directly applying Eq. (8) to estimate the mean latency results in relatively large errors, especially in the case of FJQNs with short-tailed task distributions, the gap between the exact tail-latency-to-mean-latency ratio and its estimated counterpart stays almost unchanged, as the fanout degree n becomes large (e.g., above 100), regardless of the load. Mathematically, it means that the gap, $\Delta_m(\rho, n, G)$, defined as follows,

$$\Delta_m(\rho, n, G) = \frac{t_p}{t_m} - \frac{t_p^a}{t_m^a}, \quad (10)$$

can be viewed as a constant for any given $G(t)$ (or any given FJQN) and ρ ($0 < \rho < 1$) for any $n(> 100)$, where t_p and t_m are the exact tail latency and mean latency, respectively (they can be accurately measured based on long simulation runs) and t_p^a and t_m^a are the tail latency and mean latency estimated analytically using the above ForkTail approximation technique. Fig. 2(a) and 2(b) give two examples to demonstrate that this is indeed the case, respectively. Since the estimated tail-latency t_p^a is accurate, meaning that $t_p \approx t_p^a$ is a good approximation, substituting it into Eq. (10) and rearranging Eq. (10), we have,

$$t_m \approx \frac{t_p^a}{\frac{t_p^a}{t_m^a} + \Delta_m(\rho, n, G)}. \quad (11)$$

In other words, as long as the gap, $\Delta_m(\rho, n(> 100), G)$, can be accurately estimated, t_p^a and t_m^a estimated using the ForkTail approximation technique can still be employed to accurately estimate the job mean latency at $n > 100$. Now, the problem boils down to the one of estimating, $\Delta_m(\rho, n(> 100), G)$. Note that if Δ_m is indeed a direct function of FJQN or G, it will have to be modeled one FJQN at a time. To allow it to be applied to any FJQN or G, it takes us to the second observation.

TABLE III
TABLE FROM [NGUYEN ET AL. 2020]

ρ	Δ
75%	$0.0371w^{-7.517} - 0.0052$
80%	$0.0322w^{-8.008} + 0.0056$
90%	$0.0274w^{-8.654} + 0.0284$

The second observation is that the gap, $\Delta_m(\rho, n(> 100), G)$ is much less of a function of the task service time distribution, $G(t)$, than the Right Quantile Weight of $G(t)$, $w(G)$, or the tail weight for short, i.e., $\Delta_m = \Delta_m(\rho, n, w(G))$, where $w(G)$ is defined as follows [22],

$$w(G) = \frac{G^{-1}(\frac{1+q}{2}) + G^{-1}(1-\frac{q}{2}) - 2G^{-1}(0.75)}{G^{-1}(\frac{1+q}{2}) - G^{-1}(1-\frac{q}{2})}. \quad (12)$$

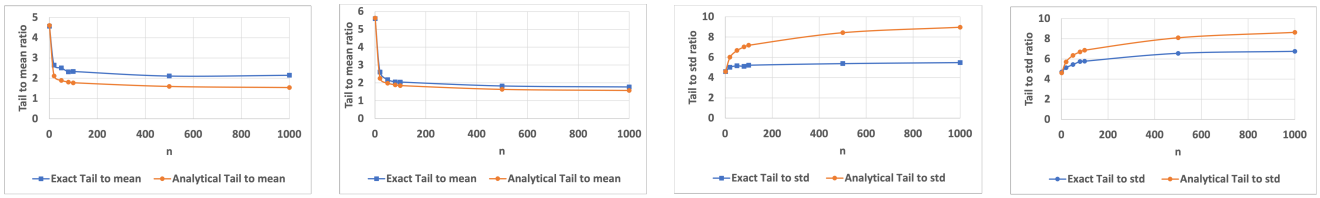
where quantile, q, is set to 0.99 to capture the tail effect. This observation implies that as long as two different FJQNs with distinct task service time distributions, $G_1(t)$ and $G_2(t)$, have the same tail weight value, i.e., $w(G_1) = w(G_2)$, they share the same gap. This makes it possible to use the generalized exponential distribution function in Eq. (3) to generate different task service time distributions by tuning the parameters, β and η, to cover a wide range of possible tail weights. Then for each distribution or w thus obtained and each load, ρ, of interest, use Eq. (10) to estimate Δ_m at an n value greater than 100. This allows $\Delta_m(\rho, n(> 100), w)$ as a function of w to be established for different ρ's. For example, Table III (i.e., Table II in [6] lists the gaps as a function of w at three different loads.

With the above preparation, ForkMean then works as follows. For any FJQN with a given $G(t)$, ForkMean first calculates $w(G)$ by Eq. (12). Then, at a given load of interest, e.g., $\rho = 80\%$, ForkMean uses $w(G)$ thus obtained as input to find Δ_m from Table III. Finally, Substituting Δ_m thus obtained, together with t_p^a and t_m^a estimated analytically by means of F_J in Eq. (8), the ForkTail approximation technique, into Eq. (14) to estimate the job mean latency for $n > 100$.

C. ForkMV: Filling the Void

ForkMV aims to fill the two voids in the design space in Table I: (a) extend ForkMean to cover the entire fanout degree dimension; and (b) develop a solution that can estimate the variance for the entire fanout degree dimension. In what follows, we develop the two, separately.

Mean Estimation: To extend ForkMean to cover $n < 100$, we note from Fig. 2(a) and 2(b) that Δ_m is roughly an increasing function of n but the impact of n diminishes as



(a) (b) (c) (d)

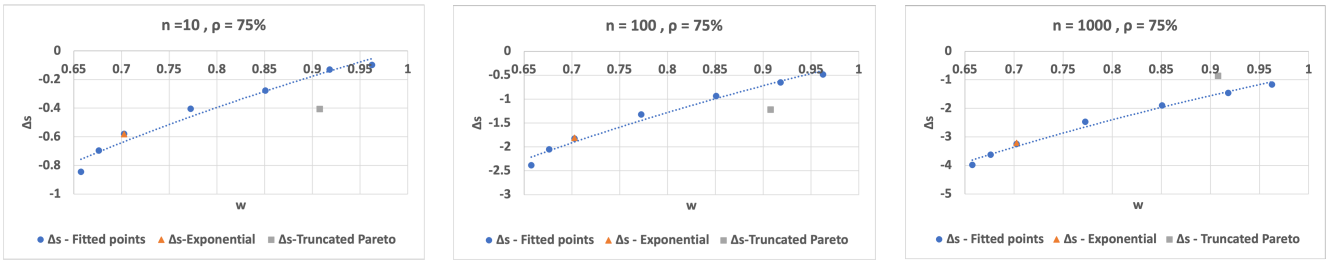
Fig. 2. At ρ = 90%, mean gaps for plots (a) and (b), and standard deviation gaps (abbreviated to std) for plots (c) and (d), for exponential and truncated Pareto distributions, respectively

TABLE IV
MEAN GAP FUNCTION

ρ	Δ_m
75%	$(-0.3577/(nw)) + (0.039w^{-7.34})$
80%	$(-0.3838/(nw)) + (0.04152w^{-7.354})$
90%	$(-0.3646/(nw)) + (0.05124w^{-7.08})$

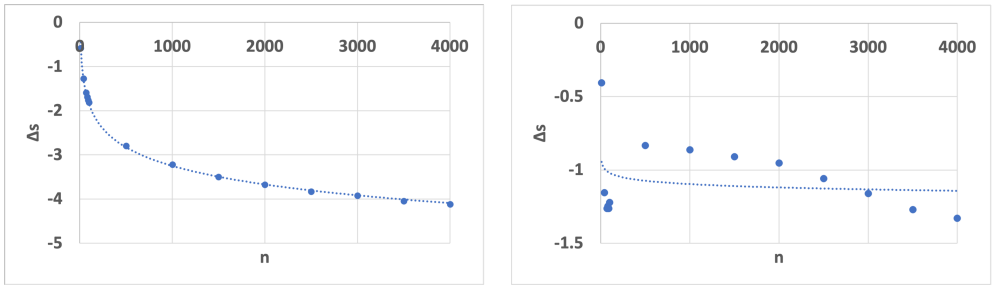
TABLE V
STD GAP FUNCTION

ρ	Δ_s
75%	$0.599 + -0.2394\log(1+(nw))/w^{2.568}$
80%	$0.6009 + -0.2522\log(1+(nw))/w^{2.476}$
90%	$0.6152 + -0.2757 * \log(1+(nw))/w^{2.317}$



(a) (b) (c)

Fig. 3. Standard deviation gap (Δ_s) vs tail weight (w) at $\rho = 75\%$



(a) (b)

Fig. 4. Standard deviation gap (Δ_s) vs fanout degree (n) at ρ = 75% for (a) exponential distribution and (b) truncated Pareto distribution

n reaches 100 and beyond. We also note that Δ_m takes the following function format: $\Delta_m = a(\rho) + b(\rho)w^{c(\rho)}$ for $n > 100$, according to Table III. These observations prompt us to extend this format to include another term that is inversely proportional to n. This term is negative. It helps reduce the gap as n gets smaller and becomes negligibly small as n exceeds 100, so that Δ_m becomes pretty much a constant with respect to n for $n > 100$. We used MATLAB's curve fitting toolbox [18] to perform nonlinear regression and generate a custom fit for the function format above. It turns out that at the loads, $\rho = 75\%$, 80%, and 90%, where resource allocations are desirable, the fitted gap functions in Table IV give smallest sum of squared estimate of errors (SSE) (i.e., 0.2938, 0.3887, and 1.3320, respectively), an indicator of goodness of fit. SSE measures the difference between data points and predicted model.

In summary, ForkMV follows the same steps as ForkMean to estimate the mean latency analytically, except that now it uses Table IV, not Table III.

Variance Estimation:

We find that just like the mean latency estimation, the variance estimation based on the ForkTail approximation is inaccurate. To see if the approach taken for the mean latency estimation can be applied here as well, we define the tail-to-

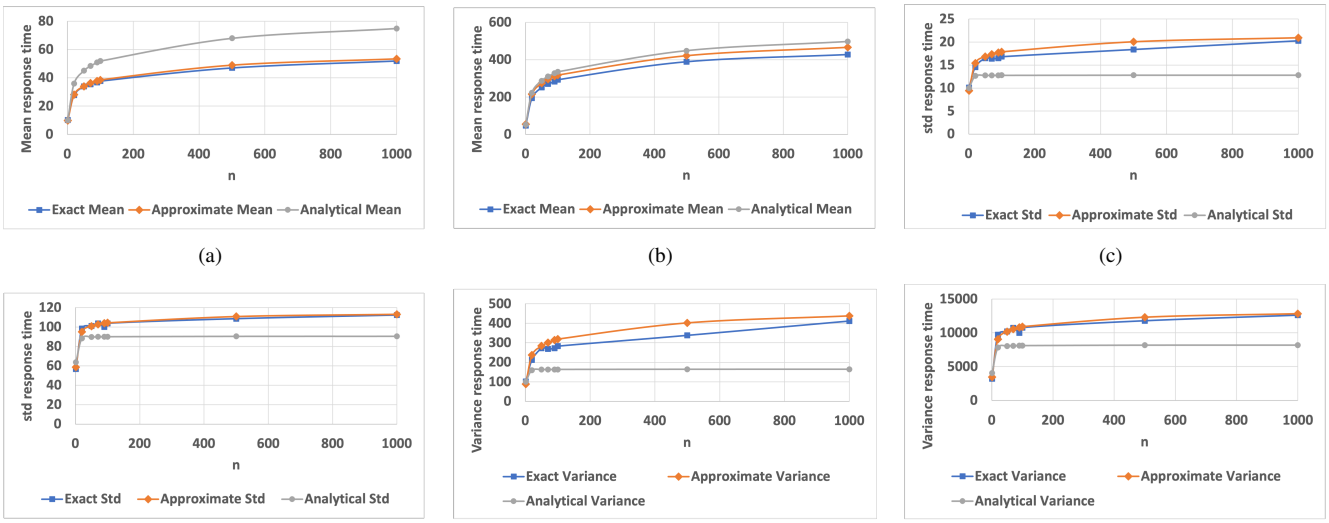


Fig. 5. At ρ = 90% for both exponential and truncated Pareto distributions respectively, (a) and (b) mean response time, (c) and (d) standard deviation response time, (e) and (f) response time variance

standard deviation ratio gap, Δ_s, as follows,

$$\Delta_s(\rho, n, G) = \frac{t_p}{t_s} - \frac{t_p^a}{t_s^a}, \tag{13}$$

where t_s and t_s^a are the exact standard deviation and the estimated standard deviation by means of F_J given in Eq. (8). Note that here we use standard deviation (i.e., the square root of variance) rather than variance itself, because the tail-to-standard-deviation ratio is unitless, whereas the tail-to-variance is not. As seen in Fig. 2(c) and 2(d), Δ_s increases with n, similar to Δ_m. However, unlike Δ_m, it does not converge to a constant as n increases. This means that Δ_s is a strong function of n, which must be explicitly modeled. Again letting $t_p \approx t_p^a$, Eq. (13) can be rewritten as,

$$t_s \approx \frac{t_p^a}{\frac{t_p^a}{t_s^a} + \Delta_s(\rho, n, G)}. \tag{14}$$

Namely, the problem boils down to the one of modeling $\Delta_s(\rho, n, G)$.

Before we proceed further, again, we want to know if the dependence of Δ_s on G can be translated into the dependence of Δ_s on $w(G)$ instead, so that the solution can be applied to all FJQNs, rather than one FJQN at a time. To this end, we draw in Fig. 3 the curves of $\Delta_s(\rho, n, G_{ge})$ as a function of w, at $n = 10, 100, 1000$ and $\rho = 75\%$, generated by G_{ge}, the generalized exponential distribution function given in Eq. (3), at different β and η values. Now, we also plot Δ_s for both exponential and truncated Pareto distributions at their respective tail weights. As one can see, these data points fall near the curves, especially for those corresponding to the exponential distribution. Our experiment also shows that they fall closer to the curves as the load, ρ, further increases (not shown here due to page limitation). This observation implies that $\Delta_s(\rho, n, G) \approx \Delta_s(\rho, n, w(G))$ is a reasonably good approximation.

To model $\Delta_s(\rho, n, w)$, we first make two observations. First, Fig. 3 clearly indicates that Δ_s is a negative function and it increases and approaches zero as w increases. In other words, Δ_s is likely to be inversely proportional to w. Second, as clearly shown in Fig. 4, Δ_s decreases logarithmically fast as n increases. These two observations make it possible for us to find Δ_s with very small SSE values using the curve fitting tool in MATLAB. Specifically, Table V gives the fitted functions for standard deviation gap at three different load levels 75%, 80% and 90% with SSE values of 2.3845, 2.4967 and 2.4826 respectively, as indicators of goodness of fit.

Finally, following the same procedure for the estimation of the mean latency, the standard deviation and hence the variance can be estimated for any FJQN or $G(t)$ analytically.

IV. Performance Evaluation

In this section we use simulation to generate exact measures to evaluate and validate ForkMV. We use exponential service time distribution with $\mu^{-1} = 1\ ms$ to model short tailed distribution and truncated Pareto service time distribution with the upper bound H as $276.6ms$, lower bound L as $2.14ms$, α as 2.0119 and coefficient of variance, CV, as 1.2 to model long tailed distributions.

Fig. 5 gives the mean, standard deviation and variance gap correction using ForkMV. After correcting the gap, the prediction error of mean response time and response time variance between exact measurement and approximation in both exponential and truncated Pareto distribution cases is less than 15% at the high load region, i.e., ρ=90%. This is where the resource provisioning becomes quite challenging [19]. Our test results at lower loads also indicate (not shown here due to page limitation) that while exponential service time distribution gets accurate mean and variance approximation at all load levels, truncated Pareto service time distribution's standard deviation and

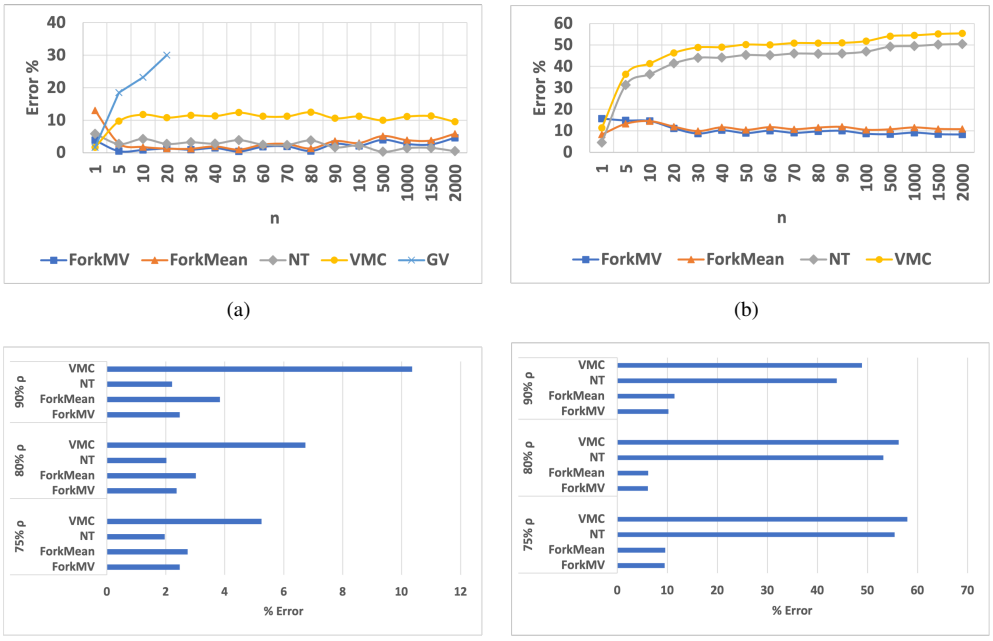


Fig. 6. Mean response time validation,at ρ = 90%,for (a) exponential service time distribution and (b) truncated Pareto service time distribution; (c) and (d) Average prediction error comparison for exponential and truncated Pareto distributions respectively

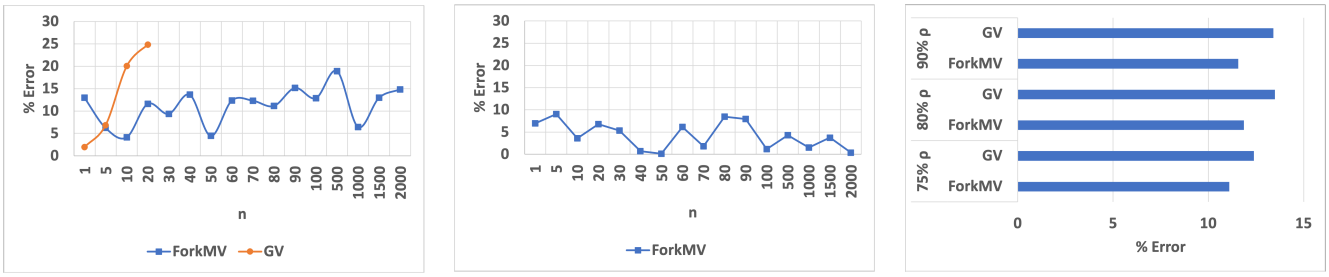


Fig. 7. At ρ = 90%, variance Response time validation for (a) exponential service time distribution, (b) truncated Pareto service time distribution,(c) average prediction error

variance approximation are not as accurately approximated. For example, at loads of $\rho = 75\%$ and 80%, the estimation errors reaches 40% and 30% respectively. This follows the fact that tail approximation of long tailed distributions give increased prediction errors as the load decreases [4]. Since we use the tail approximation to calculate standard deviation latency approximation, such errors are expected. Additionally, we have used α = 2.0119 as a parameter in the service time distribution of long-tailed truncated Pareto, as opposed to $\alpha < 2$, to cover all the practical regions of interest.

Mean Estimation Comparison: Fig. 6(a) gives a detailed comparison of ForkMV with related work, at $\rho = 90\%$, for exponential service time distribution. ForkMV gives best results in the lower fanout region while outperforming ForkMean throughout n. It is on par with NT overall, performing better in the low fanout region and slightly behind NT in the higher fanout region. ForkMV performs much better than VMC and GV. Note that GV is only compared from n=1 to 20, which is the fanout degree used by the authors of the paper. ForkMV for truncated Pareto distribution, as shown in Fig. 6(b), at lower fanout degree (n <10) is not as good as ForkMean but performs better than ForkMean throughout the rest of n. It also gives much better prediction errors than NT and VMC, which are specifically designed for exponential distributions.

Fig. 6(c) and 6(d) depict the mean response time errors, averaged over n=1 to 4000, for exponential and truncated Pareto service time distributions, respectively. GV is omitted from these plots to show consistency of the average value taken. For exponential distribution, error for ForkMV is better than ForkMean and VMC, and falls second to NT, which was specifically designed for FJQN M/M/1 queues. For truncated Pareto distribution, the average prediction error is far less than VMC and NT and while on par with ForkMean but

slightly better at $\rho = 90\%$. Unlike NT, ForkMV however, performs consistently well for both exponential and truncated Pareto distributions.

Variance Estimation Comparison: Fig. 7 gives the comparison of variance approximation at ρ = 90%. A comparison is provided only up to fanout degree of 20 for GV, the largest fan-out degree tested and ForkMV n = 1 to 2000. ForkMV outperforms GV. In fact, ForkMV provides less prediction errors even at higher fanout degrees, as can be seen in Fig. 7(a). Fig. 7(b) shows error for truncated Pareto distribution for ForkMV (note that GV cannot be applied to Pareto distribution), which is $< 10\%$. Fig. 7(c) presents the average prediction error comparison of ForkMV and GV. Note that ForkMV's average is the average of prediction errors over the fanout degree range of 1 to 4000, while GV's average is the average of fanout degree 1 to 20. Even with such high fanout degree range, ForkMV's prediction is more accurate than GV.

V. Conclusions and Future Work

In this paper we proposed ForkMV, an approximation technique to estimate both mean and variance response time for Fork-Join queuing networks (FJQNs) with M/G/1 queues, covering short, medium and long tailed service time distributions. Driven by both short-tailed exponential and long-tailed truncated Pareto distributions, our simulation test results of ForkMV demonstrate that ForkMV gives better prediction in most cases than the state-of-the-art solutions. ForkMV can estimate both mean latency and latency variance with less than 15% errors at the load of 90% or higher, where resource allocation is challenging and matters the most. In our future work, we plan to develop a resource allocation solution for cloud applications in a data center computing architecture, underpinned by ForkMV, together with ForkTail, a tail latency estimation technique [5].

Acknowledgment

This work was supported by the US NSF under Grant No. CCCF SHF-1704504 and CCF SHF-2008835.

References

[1] Awasthi, M., Suri, T., Guz, Z., Shayesteh, A., Ghosh, M., Balakrishnan, V. (2015, January). System-level characterization of datacenter applications. In Proceedings of the 6th ACM/SPEC International Conference on Performance Engineering (pp. 27-38).

[2] Barroso, L. A., Dean, J., Holzle, U. (2003). Web search for a planet: The Google cluster architecture. IEEE micro, 23(2), 22-28.

[3] Dean, J., Ghemawat, S. (2008). MapReduce: simplified data processing on large clusters. Communications of the ACM, 51(1), 107-113.

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

[5] Nguyen, M., Alesawi, S., Li, N., Che, H., Jiang, H. (2018, June). 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 (pp. 206-217).

[6] Nguyen, M., Alesawi, S., Li, N., Che, H., Jiang, H. (2020). A black-box Fork-join latency prediction model for data-intensive applications. IEEE Transactions on Parallel and Distributed Systems, 31(9), 1983-2000.

[7] Gorbunova, A. V., Vishnevsky, V. M. (2020). Estimating the response time of a cloud computing system with the help of neural networks. Advances in Systems Science and Applications, 20(3), 105-112.

[8] Flatto, L., Hahn, S. (1984). Two parallel queues created by arrivals with two demands I. SIAM Journal on Applied Mathematics, 44(5), 1041-1053.

[9] Nelson, R., Tantawi, A. N. (1988). Approximate analysis of fork/join synchronization in parallel queues. IEEE transactions on computers, 37(6), 739-743.

[10] Varki, E. (2001). Response time analysis of parallel computer and storage systems. IEEE Transactions on parallel and distributed systems, 12(11), 1146-1161.

[11] Alomari, F., Menasce, D. A. (2013). Efficient response time approximations for multiclass fork and join queues in open and closed queuing networks. IEEE Transactions on Parallel and Distributed Systems, 25(6), 1437-1446.

[12] Varki, E., Merchant, A., Chen, H. (2008). The M/M/1 fork-join queue with variable sub-tasks. unpublished, available online.

[13] Varma, S., Makowski, A. M. (1994). Interpolation approximations for symmetric fork-join queues. Performance Evaluation, 20(1-3), 245-265.

[14] Balsamo, S., Donatiello, L., Van Dijk, N. M. (1998). Bound performance models of heterogeneous parallel processing systems. IEEE transactions on parallel and distributed systems, 9(10), 1041-1056.

[15] Chen, R. J. (2010). An upper bound solution for homogeneous fork/join queuing systems. IEEE Transactions on Parallel and Distributed Systems, 22(5), 874-878.

[16] Qiu, Z., Pérez, J. F., Harrison, P. G. (2015). Beyond the mean in fork-join queues: Efficient approximation for response-time tails. Performance Evaluation, 91, 99-116.

[17] Meisner, D., Sadler, C. M., Barroso, L. A., Weber, W. D., Wenisch, T. F. (2011, June). Power management of online data-intensive services. In Proceedings of the 38th annual international symposium on Computer architecture (pp. 319-330).

[18] MATLAB. (2019). 9.7.0.1216025 (R2019b). Natick, Massachusetts: The MathWorks Inc.

[19] Delimitrou, C., Kozyrakis, C. (2014). Quasar: Resource-efficient and qos-aware cluster management. ACM SIGPLAN Notices, 49(4), 127-144.

[20] Kleinrock, L. (1975). Theory, volume 1, Queueing systems. Wiley-Interscience.

[21] De Haan, L., Ferreira, A., Ferreira, A. (2006). Extreme value theory: an introduction (Vol. 21). New York: Springer.

[22] Brys, G., Hubert, M., Struyf, A. (2006). Robust measures of tail weight. Computational statistics data analysis, 50(3), 733-759.