Non-Asymptotic Delay Bounds for Multi-Server Systems with Synchronization Constraints

Markus Fidler, *Senior Member, IEEE*, Brenton Walker, and Yuming Jiang

Abstract—Parallel computing has become a standard tool with architectures such as Google MapReduce, Hadoop, and Spark being broadly used in applications such as data processing and machine learning. Common to these systems are a fork operation, where jobs are first divided into tasks that are processed in parallel, and a join operation where completed tasks wait for the other tasks of the job before leaving the system. The synchronization constraint of the join operation makes the analysis of fork-join systems challenging, and few explicit results are known. In this work, we formulate a max-plus server model for parallel systems which allows us to derive performance bounds for a variety of systems in the GI | GI and G | G cases. We contribute end-to-end delay bounds for multi-stage fork-join networks. We perform a detailed comparison of different multi-server configurations, including an analysis of single-queue fork-join systems that achieve a fundamental performance gain. We compare these results to both simulation and a live Spark system.

Index Terms—Parallel computing, MapReduce, Hadoop, spark, performance analysis, stochastic network calculus

1 Introduction

There are many models and many implementations of parallel computing systems. The simplest is a traditional load balancing system where jobs are submitted to the system and dispatched to one of many servers. Other architectures fork jobs into tasks which are processed in parallel by the servers, and the results joined once all tasks complete. This join is a synchronization constraint on the output of the system. In addition to the forking/non-forking behavior of different parallel computing architectures, the location and nature of the queueing of jobs and tasks can have a dramatic effect. Specifically, whether the system has a single centralized queue that dispatches jobs and tasks to servers as they become available, or multiple queues, one at each server.

These two dichotomies, load balancing versus forking and single-queue versus multi-queue, create a matrix of four types of parallel systems, and each is plausibe in different situations. Under load balancing no job's processing is sped up, since each job is assigned in its entirety to a single server. In a forking system jobs may finish faster than if they had to run on a single server, but a job must wait for all of its tasks to finish before departing the system; if one task is delayed then the entire job suffers, and as the number of tasks increases so does the chance that one of them will be delayed.

- *M. Fidler and B. Walker are with the Institute of Communications Technology, Leibniz Universität Hannover, Hannover 30167, Germany. E-mail: {markus.fidler, brenton.walker}@ikt.uni-hannover.de.*
- *Y. Jiang is with the Department of Information Security and Communication Technology, NTNU Trondheim, Trondheim 7491, Norway. E-mail: jiang@ntnu.no.*

Manuscript received 7 Apr. 2017; revised 17 Oct. 2017; accepted 27 Nov. 2017. Date of publication 1 Jan. 2018; date of current version 13 June 2018. (Corresponding author: Markus Fidler.)
Recommended for acceptance by R. Prodan.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPDS.2017.2779872

There are also trade-offs in the queueing dimension. Consider for example the MapReduce model of parallel computation which has become extremely popular, with implmentations such as Google's MapReduce [2], Hadoop, and Apache Spark [3]. A MapReduce program typically operates on a large dataset that is divided into "slices" and distributed amongst the servers. In a map stage a function is applied in parallel to the records in these slices. In a single-queue system, each task is assigned to the first available server. Therefore the task processing a certain slice may not be assigned to the server which holds that slice, requiring slices to be shuffled between servers. A MapReduce system could therefore reasonably be implemented to behave like either a single-queue or multi-queue system. The default schedulers in both Hadoop MRv2/Yarn and Apache Spark have a single task queue, but the InputSplit interface in Hadoop MRv2 makes slice location information available to the scheduler, which would enable it to act as a multi-queue system.

In this paper we use the tools of max-plus network calculus to formulate a general definition of a parallel server and (σ, ρ) burstiness constraints, and derive bounds on the waiting and sojourn time distributions in the GI|GI and G|G settings. We then show that this model applies to the entire matrix of forking/non-forking and single-queue/multi-queue systems, as well as multi-stage fork-join systems, yielding specific performance bounds for each system type. We compare the bounds to simulation and to experiments run on a real Spark cluster. Additionally, we apply our result for the fork-join system to arrival and service process data extracted from publicly available cluster traces. We discover cases of significant correlations in some users' arrival processes, demonstrating the necessity of deriving bounds for the G|G case.

The most intensely studied fork-join model is the multi-queue one, depicted in Fig. 1. Jobs are *forked* into k tasks and queued at k servers. The tasks are processed in order of arrival, and then the results *joined* after all tasks are

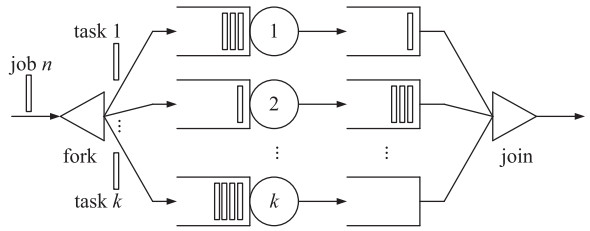


Fig. 1. Multi-queue fork-join system. Each job is composed of k tasks with individual service requirements, that are distributed to k servers (fork). Once all tasks of a job are completed, the job leaves the system (join), i.e., the tasks of a job wait at the join step until all tasks of the job are completed.

complete. Exact results for the fork-join model are known only for a few special cases, such as two parallel $\mathrm{M}\,|\,\mathrm{M}\,|\,1$ queues [4], [5]. For more complex systems, approximation techniques, e.g., [5], [6], [7], [8], [9], [10], [11], [12], and bounds, using stochastic orderings [13], martingales [14], or stochastic burstiness constraints [15], have been explored. For a more elaborate discussion see also [14]. Given the difficulties posed by single-stage systems, few works consider multi-stage networks. A notable exception is [10] where an approximation for closed fork-join networks is developed.

In terms of the diversity of models studied, this paper is similar to [16] which derives mean sojourn times for single-queue fork-join systems using a bulk arrival $M\,|\,M\,|\,k$ multi-server model. In terms of mathematical tools employed, this paper is in the same vein as three recent papers. In [15] the authors consider single-stage fork-join systems with load balancing, general arrivals of the type defined in [17], and deterministic service. They use the concept of service curve, see, e.g., the textbooks [18], [19], [20], to characterize fork-join systems and derive statistical delay bounds. In [14] the authors prove delay bounds for single-stage fork-join systems with renewal and Markov-modulated inter-arrival times and independent and identically distributed (iid) service times. They prove that delays for fork-join systems grow as $\mathcal{O}(\ln k)$ for k parallel servers, as also found in [13]. They also consider blocking systems, where no task of a job can start until the entire previous job has departed, and adapt their analysis to multi-path routing. In [21], the authors evaluate different task assignment policies for parallel server systems with task replication, considering the effects of correlated replicas. Task replication relates to the more general concept of (k, l) fork-join systems [1], where a job is considered completed once l out of k tasks have finished service.

Compared to these previous studies, this paper considers the $\mathrm{G}\,|\,\mathrm{G}$ case, and uses our general formulation of parallel system to derive delay bounds for the full matrix of fork-join/load-balancing and multi-/single-queue models as well as for multi-stage fork-join networks. We validate our results with experiments on a real Spark systems and evaluate our bounds based on empirical trace data from a real cluster.

The rest of this paper is structured as follows. In Section 2 we formulate general models of $\mathrm{G}\,|\,\mathrm{G}$ and $\mathrm{GI}\,|\,\mathrm{GI}$ systems in max-plus system theory. In Section 3 we apply these results to the multi-queue fork-join model, and to multi-stage systems. In Sections 4, 5, and 6 we derive bounds for the rest of the matrix of system types, and compare the results to simulation and experiments. We conclude with Section 7.

2 Max-Plus System Model

We derive a set of results for a general parallel system in max-plus system theory. Max-plus system theory [18], [19], [22], [23], [24], [25], [26] is a branch of the deterministic [18], [19], [27], respectively, stochastic network calculus [18], [20], [28], [29], [30], [31] that can deal specifically with time-stamps of jobs. In comparison to [14], which is focused entirely on waiting and sojourn times of specific systems, the more general max-plus approach enables us to derive bounds for a wide variety of parallel systems as well as multi-stage fork-join networks. This also generalizes the $\mathcal{O}(\ln k)$ scaling from [14] by considering general arrival and service processes. Throughout this work, we consider only the case of homogeneous servers, i.e., all servers have identical service time distribution. Heterogeneous servers can be dealt with in the same way by a notational extension. We show results for heterogeneous servers in [1].

2.1 Notation and Queueing Model

We label jobs in the order of arrival by $n \geq 1$ and let $A(n)$ denote the time of arrival of job n. It follows for $n \geq m \geq 1$ that $A(n) \geq A(m) \geq 0$. For notational convenience, we define $A(0) = 0$. Further, we let $A(m,n) = A(n) - A(m)$ be the time between the arrival of job m and job n for $n \geq m \geq 1$. Hence, $A(n, n+1)$ is the inter-arrival time between job n and job $n+1$ for $n \geq 1$. Similarly, $D(n)$ denotes departure times. To model systems, we adapt the definition of g-server from [18, Definition 6.3.1] using a notion of service process $S(m,n)$ that characterizes the cumulated service time of jobs m to n.

Definition 1 (Max-plus server). *A system with arrivals $A(n)$ and departures $D(n)$ is an $S(m,n)$ server under the max-plus algebra if it holds for all $n \geq 1$ that*

$$D(n) \leq \max_{m \in [1,n]} \{A(m) + S(m,n)\}.$$

It is an exact $S(m,n)$ server if it holds for all $n \geq 1$ that

$$D(n) = \max_{m \in [1,n]} \{A(m) + S(m,n)\}.$$

The following Lemma 1 [18, Example 6.2.3] shows that the general class of $\mathrm{G}\,|\,\mathrm{G}\,|\,1$ systems satisfy the definition of exact server. We phrase the result as a lemma including a proof as it serves as a template for subsequent systems. We use $V(n)$ to denote the time at which job n starts service.

Lemma 1 (Work-conserving system). *Consider a lossless, work-conserving, first-in first-out system and let $L(n)$ denote the service time of job n, where $n \geq 1$. Define for $n \geq m \geq 1$*

$$S(m,n) = \sum_{v=m}^{n} L(v).$$

The system is an exact $S(m,n)$ server.

Proof. As the system is lossless, work-conserving, and serves jobs in first-in first-out order, job $n \geq 2$ starts service at

$$V(n) = \max\{A(n), V(n-1) + L(n-1)\}, \tag{1}$$

and job 1 at $V(1) = A(1)$. By recursive insertion of (1)

$$V(n) = \max_{m\in[1,n]}\left\{A(m) + \sum_{\nu=m}^{n-1} L(\nu)\right\}, \tag{2}$$

for $n \geq 1$. Since $D(n) = V(n) + L(n)$, it follows with (2) that $D(n) = \max_{m\in[1,n]}\{A(m) + \sum_{\nu=m}^{n} L(\nu)\}$, which proves that the work-conserving system is an exact max-plus server. □

For the sojourn time of job $n \geq 1$, defined as $T(n) = D(n) - A(n)$, it follows by insertion of Def. 1 that

$$T(n) = \max_{m\in[1,n]}\{S(m,n) - A(m,n)\}. \tag{3}$$

The waiting time of job $n \geq 1$ is $W(n) = V(n) - A(n)$. As in the case of work-conserving systems in Lemma 1, $V(n) = \max\{A(n), D(n-1)\}$, so we have $W(n) = [D(n-1) - A(n)]^+$, where $[X]^+ = \max\{X, 0\}$ is the non-negative part and $D(0) = 0$ by definition. With Definition 1, it holds that

$$W(n) = \left[\sup_{m\in[1,n-1]}\{S(m,n-1) - A(m,n)\}\right]^+. \tag{4}$$

Here, we use the supremum since for $n = 1$ (4) evaluates to an empty set. For non-negative real numbers the sup of an empty set is zero. While we used the definition of an exact server to derive the sojourn and waiting times, we note that the upper bound specified by the definition of a server is sufficient to provide upper bounds of sojourn and waiting times.

2.2 Statistical Performance Bounds

Next, we derive statistical performance bounds for servers as defined above. Throughout the paper, we generally assume that the arrival and service processes are independent of each other. Considering general arrival and service processes, the server is a G|G|1 queue. The results enable us to generalize recent conclusions on the speed of the tail decay obtained for iid service times, i.e., for a GI service model, in [14].

We consider arrival and service processes that belong to the broad class of (σ,ρ)-constrained processes [18]. The parameters σ and ρ specify an affine bounding function, i.e., intercept and slope, of the logarithm of the moment generating function (MGF), and can be thought of as a stochastic version of the burst and rate parameters of a leaky-bucket regulator. The MGF of a random variable X is defined as $\mathsf{M}_X(\theta) = \mathsf{E}\left[e^{\theta X}\right]$ where θ is a free parameter. MGFs of a variety of relevant sources are known including periodic, regulated, Markov, and fractional Brownian motion processes [32] as well as empirical MGFs obtained from data traces [33]. In the following definition we adapt the (σ,ρ) constraint [18, Definition 7.2.1] to max-plus systems.

Definition 2 ((σ,ρ)-Arrival and Service Envelopes). *An arrival process is (σ_A,ρ_A)-lower constrained if for all $n \geq m \geq 1$ and $\theta > 0$ it holds that*

$$\mathsf{E}\left[e^{-\theta A(m,n)}\right] \leq e^{-\theta(\rho_A(-\theta)(n-m)-\sigma_A(-\theta))}.$$

Similarly, a service process is (σ_S,ρ_S)-upper constrained if for all $n \geq m \geq 1$ and $\theta > 0$ it holds that

$$\mathsf{E}\left[e^{\theta S(m,n)}\right] \leq e^{\theta(\rho_S(\theta)(n-m+1)+\sigma_S(\theta))}.$$

Considering the service times of jobs as in Lemma 1, we also apply Definition 2 to characterize the cumulative service process $L(m,n) = \sum_{\nu=m}^{n} L(\nu)$ by (σ_L,ρ_L).

In the special case of GI arrival processes, $A(m,n) = \sum_{\nu=m}^{n-1} A(\nu,\nu+1)$ has iid inter-arrival times $A(\nu,\nu+1)$. It follows that $\mathsf{E}\left[e^{-\theta A(\nu,\nu+1)}\right] = \mathsf{E}\left[e^{-\theta A(1,2)}\right]$ for $\nu \geq 1$. Next, we use that the MGF of a sum of independent random variables is the product of their individual MGFs, i.e., $\mathsf{E}\left[e^{-\theta A(m,n)}\right] = \mathsf{E}\left[e^{-\theta A(1,2)}\right]^{n-m}$ to derive minimal traffic parameters from Definition 2 as $\sigma_A(-\theta) = 0$ and

$$\rho_A(-\theta) = -\frac{1}{\theta}\ln \mathsf{E}\left[e^{-\theta A(1,2)}\right]. \tag{5}$$

Similarly for GI service processes, $S(m,n)$ is composed of iid service increments $S(\nu,\nu)$, i.e., $S(m,n) = \sum_{\nu=m}^{n} S(\nu,\nu)$, so that it has minimal parameters $\sigma_S(\theta) = 0$ and

$$\rho_S(\theta) = \frac{1}{\theta}\ln \mathsf{E}\left[e^{\theta S(1,1)}\right]. \tag{6}$$

Parameter $\rho_A(-\theta)$ decreases with $\theta > 0$ from the mean to the minimum inter-arrival time and $\rho_S(\theta)$ increases with $\theta > 0$ from the mean to the maximum service time.

Theorem 1 (Statistical performance bounds). *Consider a server as in Definition 1, with arrival and service parameters $(\sigma_A(-\theta),\rho_A(-\theta))$ and $(\sigma_S(\theta),\rho_S(\theta))$ as specified by Definition 2. For $n \geq 1$, the sojourn time $T(n) = D(n) - A(n)$ satisfies*

$$\mathsf{P}[T(n) > \tau] \leq \alpha e^{\theta\rho_S(\theta)} e^{-\theta\tau},$$

and the waiting time $W(n) = [D(n-1) - A(n)]^+$ satisfies

$$\mathsf{P}[W(n) > \tau] \leq \alpha e^{-\theta\tau}.$$

In the case of G|G arrival and service processes, the free parameter $\theta > 0$ has to satisfy $\rho_S(\theta) < \rho_A(-\theta)$ and

$$\alpha = \frac{e^{\theta(\sigma_A(-\theta)+\sigma_S(\theta))}}{1 - e^{-\theta(\rho_A(-\theta)-\rho_S(\theta))}}.$$

In the special case of GI|GI arrival and service processes, $\theta > 0$ has to satisfy $\rho_S(\theta) \leq \rho_A(-\theta)$ and $\alpha = 1$.

For the special case of GI|GI arrival and service processes, Theorem 1 recovers the classical bound for the waiting time of GI|GI|1 queues [34] in the max-plus system theory. Like [34] and subsequent works in the stochastic network calculus [14], [21], [24], [30], [35], [36], the proof uses Doob's martingale inequality [37]. The proof for the G|G arrival and service processes adapts the approach from [18, Ch. 7], [29] to max-plus systems. The important property of the G|G result is that it differs only by a constant factor α from the GI|GI result and otherwise recovers the characteristic exponential tail decay $e^{-\theta\tau}$ with the same maximal decay θ.

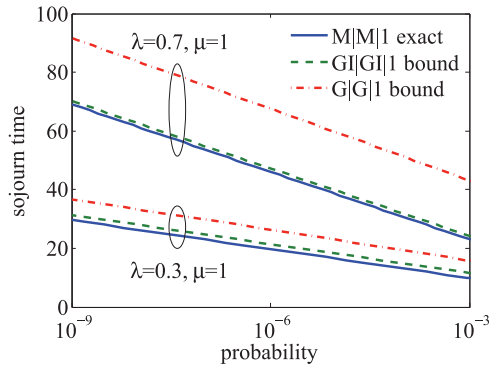


Fig. 2. M | M | 1 queue. The CCDF bounds show the true exponential tail decay.

Proof. We only show the proof of the sojourn time, as the proof of the waiting time follows similarly. G | G | 1 servers: We obtain from (3) for $\theta > 0$ that

$$\mathsf{E}\left[e^{\theta T(n)}\right] \leq \sum_{v=1}^{n} \mathsf{E}\left[e^{\theta S(v,n)}\right]\mathsf{E}\left[e^{-\theta A(v,n)}\right],$$

where we estimated the maximum by the sum of its arguments and used the statistical independence of arrivals and service.

By insertion of the (σ,ρ)-constraints from Definition 2 we have

$$\mathsf{E}\left[e^{\theta T(n)}\right] \leq e^{\theta(\sigma_A(-\theta)+\sigma_S(\theta)+\rho_S(\theta))} \sum_{v=1}^{n} e^{-\theta(\rho_A(-\theta)-\rho_S(\theta))(n-v)}.$$

Next, for $\rho_S(\theta) < \rho_A(-\theta)$ we estimate

$$\sum_{v=1}^{n} e^{-\theta(\rho_A(-\theta)-\rho_S(\theta))(n-v)} \leq \sum_{v=0}^{\infty} \left(e^{-\theta(\rho_A(-\theta)-\rho_S(\theta))}\right)^{v} = \frac{1}{1-e^{-\theta(\rho_A(-\theta)-\rho_S(\theta))}}.$$

With Chernoff's bound $\mathsf{P}[X \geq x] \leq e^{-\theta x}\mathsf{E}\left[e^{\theta X}\right]$ we obtain

$$\mathsf{P}[T(n) \geq \tau] \leq \frac{e^{\theta(\sigma_A(-\theta)+\sigma_S(\theta))}}{1-e^{-\theta(\rho_A(-\theta)-\rho_S(\theta))}} e^{\theta\rho_S(\theta)} e^{-\theta\tau}.$$

GI | GI | 1 servers: From (3) we have

$$T(n) = \max_{m\in[1,n]}\{S(n-m+1,n) - A(n-m+1,n)\}.$$

It follows that $\mathsf{P}[T(n) > \tau] = \mathsf{P}\left[\max_{m\in[1,n]}\{U(m)\} > e^{\theta\tau}\right]$ for $\theta > 0$ where

$$U(m) = e^{\theta(S(n-m+1,n)-A(n-m+1,n))}.$$

Using the representation of $A(m,n) = \sum_{v=m}^{n-1} A(v,v+1)$ and $S(m,n) = \sum_{v=m}^{n} S(v,v)$ by increment processes, we have

$$U(m+1) = U(m)e^{\theta(S(n-m,n-m)-A(n-m,n-m+1))}.$$

The conditional expectation can be computed as

$$\mathsf{E}[U(m+1)|U(m),U(m-1),\ldots,U(1)] = U(m)\mathsf{E}[e^{\theta S(n-m,n-m)}]\mathsf{E}[e^{-\theta A(n-m,n-m+1)}],$$

where we used the independence of the inter-arrival times and the service times. If $\rho_S(\theta) \leq \rho_A(-\theta)$, it holds that $\mathsf{E}\left[e^{\theta S(n-m,n-m)}\right]\mathsf{E}\left[e^{-\theta A(n-m,n-m+1)}\right] \leq 1$ and

$$\mathsf{E}[U(m+1)|U(m),U(m-1),\ldots,U(1)] \leq U(m),$$

i.e., $U(m)$ is a supermartingale. By application of Doob's inequality for submartingales [37, Theorem 3.2, p. 314] and the formulation for supermartingales [24], [36] we have for non-negative $U(m)$ for $m \geq 1$ that

$$x\mathsf{P}\left[\max_{m\in[1,n]}\{U(m)\} \geq x\right] \leq \mathsf{E}[U(1)]. \tag{7}$$

We derive

$$\mathsf{E}[U(1)] = \mathsf{E}\left[e^{\theta(S(n,n)-A(n,n))}\right] = \mathsf{E}\left[e^{\theta S(1,1)}\right].$$

Letting $x = e^{\theta\tau}$ we have from (7) that

$$\mathsf{P}[T(n) \geq \tau] \leq e^{\theta\rho_S(\theta)}e^{-\theta\tau},$$

which completes the proof. □

Example (The M | M | 1 Queue). The purpose of this example is to illustrate the relative tightness of the GI | GI and G | G bounds. We apply Theorem 1 to the M | M | 1 queue since exact reference results are available for this case. We note that the relation of stochastic network calculus and queueing theory has been investigated also in [24], [35]. Given iid exponential inter-arrival and service times with parameters λ and μ, respectively, (5) and (6) evaluate to

$$\rho_A(-\theta) = -\frac{1}{\theta}\ln\left(\frac{\lambda}{\lambda+\theta}\right), \tag{8}$$

for $\theta > 0$, and

$$\rho_S(\theta) = \frac{1}{\theta}\ln\left(\frac{\mu}{\mu-\theta}\right), \tag{9}$$

for $\theta \in (0,\mu)$. From the condition $\rho_S(\theta) \leq \rho_A(-\theta)$ it follows that $\theta \leq \mu - \lambda$ under the stability condition $\lambda < \mu$. By the choice of the maximal $\theta = \mu - \lambda$ we have from Theorem 1 that

$$\mathsf{P}[T(n) > \tau] \leq \frac{\mu}{\lambda}e^{-(\mu-\lambda)\tau}. \tag{10}$$

Compared to the exact CCDF of the sojourn time of the M | M | 1 queue, that is $\mathsf{P}[T(n) > \tau] = e^{-(\mu-\lambda)\tau}$ see, e.g., [38], the bound (10) has the same tail decay and differs only by the pre-factor μ/λ. Therefore the bound becomes tighter if the utilization is high, in which case μ/λ approaches one.

In Fig. 2, we illustrate the bounds from Theorem 1 compared to the exact M | M | 1 CCDF. Clearly, the curves show the same tail decay, where the GI | GI | 1 bound provides better numerical accuracy compared to the G | G | 1 bound that has parameter $\alpha > 1$. In the case of the G | G | 1 bound, the parameter θ is optimized numerically to obtain the smallest delay bound.

We note that the deviation of the bounds is due to relaxed assumptions. Particularly, Theorem 1 provides results for general, i.e., non-exponential and non-independent, inter-arrival and service times by substitution of the MGFs of the respective processes into Definition 2. We provide an example of how to use MGFs of empirical trace data in Section 3.

3 Multi-Queue Fork-Join Systems

The multi-queue fork-join system is the standard fork-join model. We will generally refer to them simply as "fork-join" systems. Later in Section 6 we will study a single-queue variation on the standard fork-join model.

In a fork-join system, each job $n \geq 1$ is composed of k tasks with service times $Q_i(n)$ for $i \in [1,k]$; i.e., the service requirements of the tasks may differ from each other and may or may not be independent. The tasks are distributed (fork) to k parallel servers and once all tasks of a job are served, the job leaves the system (join), see Fig. 1. The parallel servers are not synchronized; i.e., server i starts serving task i of job $n+1$ (assuming it is already in the system), once it finishes serving task i of job n, which departs from server i at $D_i(n)$. Job n has finished service once all of its tasks $i \in [1,k]$ have finished service. The following lemma shows that fork-join systems are servers under the max-plus algebra.

Lemma 2 (Multi-queue fork-join system). *Consider a fork-join system with k parallel servers as in Lemma 1. Let $Q_i(n)$ denote the service time of task i of job n where $i \in [1,k]$ and $n \geq 1$. Define for $n \geq m \geq 1$*

$$S(m,n) = \max_{i\in[1,k]} \left\{ \sum_{\nu=m}^{n} Q_i(\nu) \right\}.$$

The system is an exact $S(m,n)$ server.

Proof. Since a job departs from the system once all of its tasks $i \in [1,k]$ are completed, we have for $n \geq 1$ that

$$D(n) = \max_{i\in[1,k]} \{D_i(n)\}. \tag{11}$$

By insertion of Definition 1 for each of the servers $i \in [1,k]$, it follows that

$$D(n) = \max_{i\in[1,k]} \left\{ \max_{m\in[1,n]} \{A(m) + S_i(m,n)\} \right\}.$$

After reordering the maxima

$$D(n) = \max_{m\in[1,n]} \left\{ A(m) + \max_{i\in[1,k]} \{S_i(m,n)\} \right\},$$

we conclude that the fork-join system is an exact $S(m,n) = \max_{i\in[1,k]}\{S_i(m,n)\}$ server. In the last step, we invoke Lemma 1 with $Q_i(n)$ for each of the servers $i \in [1,k]$. □

Performance bounds are obtained using the MGF of the service process $S(m,n)$. For Lemma 2, we estimate the MGF for $n \geq m \geq 1$ by

$$\mathsf{E}\left[e^{\theta S(m,n)}\right] \leq \sum_{i=1}^{k} \mathsf{E}\left[e^{\theta \sum_{\nu=m}^{n} Q_i(\nu)}\right].$$

Assuming homogeneous tasks with parameters $(\sigma_Q(\theta), \rho_Q(\theta))$ for $i \in [1,k]$, it follows by insertion of Definition 2 that

$$\mathsf{E}\left[e^{\theta S(m,n)}\right] \leq k e^{\theta(\sigma_Q(\theta)+\rho_Q(\theta)(n-m+1))}.$$

This shows that the service process of the fork-join system has parameters

$$\sigma_S(\theta) = \sigma_Q(\theta) + \frac{\ln k}{\theta}, \tag{12}$$

and

$$\rho_S(\theta) = \rho_Q(\theta). \tag{13}$$

Corollary 1 (Multi-queue fork-join system). *Consider a fork-join system as in Lemma 2, with arrival and service parameters $(\sigma_A(-\theta), \rho_A(-\theta))$ and $(\sigma_Q(\theta), \rho_Q(\theta))$ as specified by Definition 2. For $n \geq 1$, the sojourn time satisfies*

$$\mathsf{P}[T(n) > \tau] \leq k\alpha e^{\theta\rho_Q(\theta)} e^{-\theta\tau},$$

and the waiting time of the task that starts service last

$$\mathsf{P}[W(n) > \tau] \leq k\alpha e^{-\theta\tau}.$$

In the case of G|G arrival and service processes, the free parameter $\theta > 0$ has to satisfy $\rho_Q(\theta) < \rho_A(-\theta)$ and

$$\alpha = \frac{e^{\theta(\sigma_A(-\theta)+\sigma_Q(\theta))}}{1 - e^{-\theta(\rho_A(-\theta)-\rho_Q(\theta))}}.$$

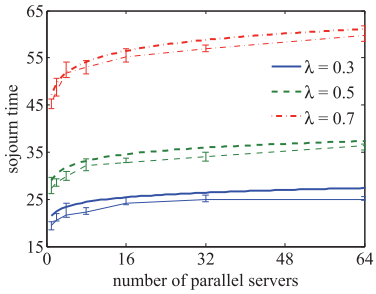
In the special case of GI|GI arrival and service processes, $\theta > 0$ has to satisfy $\rho_Q(\theta) \leq \rho_A(-\theta)$ and $\alpha = 1$.

We note that Corollary 1 does not assume independence of the parallel servers. Indeed, independence cannot be assumed as the waiting and sojourn times of the individual servers depend on the same arrival process [11], [13].

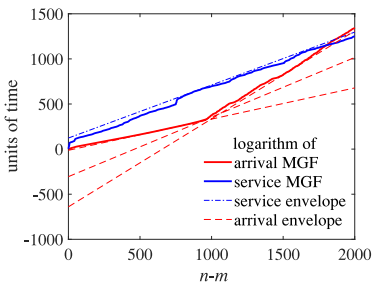
Proof. The proof depends on Lemma 2 that verifies that the fork-join system is a max-plus server. Hence, Theorem 1 applies, and by insertion of (12) and (13) the result of Corollary 1 for G|G arrival and service processes is obtained. We note that the waiting time of a job is defined to be that of its task that starts service last. This follows by insertion of (11) into the definition of waiting time $W(n) = [D(n-1) - A(n)]^+$.

As the increment process of $S(m,n)$ in Lemma 2 is non-trivial, we pursue a different approach[1] to show the result for the special case of GI|GI arrival and service processes. With (11), we derive the sojourn time $T(n) = D(n) - A(n)$ of job n as $T(n) = \max_{i\in[1,k]}\{D_i(n) - A(n)\}$ for $n \geq 1$. Hence, the sojourn time of job n is expressed as a maximum $T(n) = \max_{i\in[1,k]}\{T_i(n)\}$ of the sojourn times $T_i(n) = D_i(n) - A(n)$ of the individual tasks $i \in [1,k]$ of job n. Since the individual servers satisfy Lemma 1, we can invoke Theorem 1 for each of the servers $i \in [1,k]$ and use the union bound to obtain the result of Corollary 1. The waiting time $\max_{i\in[1,k]}\{W_i(n)\}$ where $W_i(n)$ is given by (4) can be derived in the same way. □

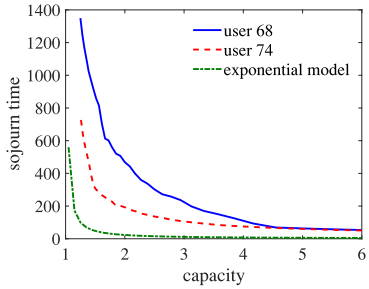
1. The approach applies also in the case of G|G arrival and service processes. We showed the alternative approach via (12) and (13) though, as it extends to multi-stage fork-join networks, see Section 3.2.



(a) scaling in $\ln k$



(b) (σ, ρ) envelope construction



(c) capacity provisioning

Fig. 3. Fork-join system with k servers. (a) Analytical CCDF bounds for $\varepsilon = 10^{-6}$ (thick lines) and simulation results (thin lines) for exponential inter-arrival and service times. (b) Emperical log MGFs of inter-arrival and task service time trace data (user 68) and several feasible (σ, ρ) lines. (c) Analytical sojourn time bounds for $\varepsilon = 10^{-6}$ and $k = 16$ for different server capacities.

Scaling of Fork-Join Systems. To investigate the scaling of fork-join systems with k parallel servers, we first note that the stability condition $\rho_Q(\theta) < \rho_A(-\theta)$ is independent of k. Hence, our first observation is that the speed of the tail decay of the performance bounds θ does not depend on the number of parallel servers. Next, we equate the sojourn time bound from Corollary 1 with ε and solve for

$$\tau \leq \rho_Q(\theta) + \frac{\ln k + \ln \alpha - \ln \varepsilon}{\theta}, \tag{14}$$

for $\theta > 0$ subject to the stability condition $\rho_Q(\theta) < \rho_A(-\theta)$. Eq. (14) expresses a sojourn time bound that is exceeded at most with probability ε. It exhibits a growth in $\mathcal{O}(\ln k)$. The growth is larger for smaller θ corresponding to a higher utilization. The result applies for general arrival and service processes subject to Definition 2 and generalizes the finding of $\mathcal{O}(\ln k)$ that is obtained in [13], [14] for iid service times.

In Fig. 3a, we consider a fork-join system with $k \geq 1$ parallel servers that serve jobs composed of k tasks each. For reasons of space we avoid defining further workload models and consider jobs with iid exponential inter-arrival times with parameter $\rho_A(-\theta)$, see (8), and tasks with iid exponential service times with parameter $\rho_Q(\theta)$, (9), where we let $\mu = 1$. The parameters of other distributions can be straightforwardly obtained from their MGFs, see e.g., [31], [39]. We show sojourn time bounds τ, where $\mathsf{P}[T(n) > \tau] \leq \varepsilon$ and $\varepsilon = 10^{-6}$. The curves show the characteristic logarithmic growth with k. This is also confirmed in simulation results (exact results are not known) that agree well with the sojourn time bounds.

3.1 Cluster Trace Data

Google's publicly-available Cluster2011 trace [40] records one month of activity in a cluster of approximately 12,000 machines, and includes events in the life-cycles of both jobs and tasks. For each user we obtain an arrival process $A(n)$ by extracting the user's job SUBMIT events. The number of tasks per job in the trace is usually much larger than the number of servers, and there are often failed and repeated tasks. To obtain a task service time process, $Q(n)$, we extract the service time of the first successful task from each of the user's jobs. In order to produce comparable processes we normalize the processes to have the same means, $1/\lambda$ and $1/\mu$. We multiply $A(n)$ by $1/(\lambda \overline{A})$ and multiply $Q(n)$ by $1/(\mu \overline{Q})$, where $\overline{A}$ and $\overline{Q}$ are the respective empirical averages. This scales the processes, but does not affect their correlations.

In order to apply Corollary 1 we need to estimate (σ_A, ρ_A), (σ_Q, ρ_Q), and $\theta > 0$. Taking the log of the MGFs in Definition 2 and dividing by $-\theta$ and θ, respectively, we have

$$-\frac{1}{\theta} \ln \mathsf{E}\left[e^{-\theta A(m,n)}\right] \geq \rho_A\ (n-m) - \sigma_A, \tag{15}$$

$$\frac{1}{\theta} \ln \mathsf{E}\left[e^{\theta Q(m,n)}\right] \leq \rho_Q\ (n-m+1) + \sigma_Q. \tag{16}$$

Let us focus on the arrivals. On the left side of (15) we compute the empirical log-MGF [33] from the arrival data. The right side of the inequality is a linear envelope function of the lag, $(n-m)$. If the inter-arrival times were independent, then the empirical log-MGF would be linear with slope ρ_A and intercept $\sigma_A = 0$. Fig. 3b, however, shows an example where the log-MGF is convex (user 68, $\theta = 0.27$). Therefore, there are a range of feasible (σ_A, ρ_A) lines satisfying (15). The empirical log-MGF of service times in this example appears close to linear. Stability requires that $\rho_Q < \rho_A$, i.e., the lines must intersect. For a given pair of processes, A and Q, we search of the space of feasible (σ_A, ρ_A), (σ_Q, ρ_Q), and $\theta > 0$ to find the minimal G|G bound in Corollary 1.

Fig. 3c demonstrates the practical task of capacity provisioning for a desired sojourn time bound τ with violation probability $\varepsilon = 10^{-6}$. The capacity of a server is μ/λ, the inverse of the load. For two users with particularly correlated arrivals, we normalize the arrival process to have $\lambda = 1$ and scale the service process to have the desired capacity μ/λ to compute corresponding delay bound quantiles. We observe a dramatic increase in the sojourn times of users 68 and 74 compared to the exponential model due to the correlations of the job arrivals.

3.2 Multi-Stage Fork-Join Networks

We contribute a new bound on the growth of end-to-end sojourn times for multi-stage fork-join networks, where we consider h fork-join stages in tandem, each with k parallel servers. We use subscript $i \in [1, k]$ to distinguish the servers of a stage and superscript $j \in [1, h]$ to denote the stages. Since jobs depart from each fork-join stage in the order of their arrival, the following lemma can be obtained by repeated application of [18, Theorem 6.3.6].

Lemma 3 (Multi-stage fork-join network). *Consider a multi-stage network of h fork-join systems as in Lemma 2 in tandem. Define for $n \geq m \geq 1$*

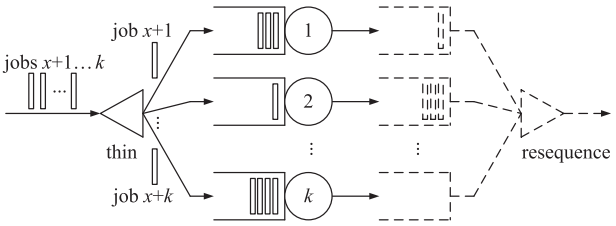


Fig. 4. Multi-queue load balancing system. Compared to a fork-join system, jobs are not divided into tasks. Instead, entire jobs are assigned to the servers, resulting in thinned arrival processes. The load balancing system does not maintain the order of jobs unless a resequencing step is added (dashed).

$$S^{\text{net}}(m,n) = \max_{\nu^j: m\le\nu^1\le\nu^2\le\ldots\le\nu^{h-1}\le n} \{S^1(m,\nu^1) + S^2(\nu^1,\nu^2) + \cdots + S^h(\nu^{h-1},n)\}.$$

The fork-join network is an exact $S^{\text{net}}(m,n)$ server.

Theorem 2 (Multi-stage fork-join network). *Consider a multi-stage fork-join network as in Lemma 3 with arrival and service parameters $(\sigma_A(-\theta),\rho_A(-\theta))$ and $(\sigma_Q(\theta),\rho_Q(\theta))$ as specified by Definition 2. Let the service times at each of the stages be independent. For $n\ge 1$, the end-to-end sojourn time satisfies*

$$\mathsf{P}[T(n) > \tau] \le k^h \alpha e^{\theta h \rho_Q(\theta)} e^{-\theta\tau},$$

where $\theta > 0$ has to satisfy $\rho_Q(\theta) < \rho_A(-\theta)$ and

$$\alpha = \frac{e^{\theta(\sigma_A(-\theta)+h\sigma_Q(\theta))}}{\left(1-e^{-\theta(\rho_A(-\theta)-\rho_Q(\theta))}\right)^h}.$$

Proof. First, we derive the MGF of $S^{\text{net}}(m,n)$ as in Lemma 3. It follows for $\theta > 0$ that

$$\mathsf{E}\left[e^{\theta S^{\text{net}}(m,n)}\right] \le \sum_{\nu^j\ge 0:\sum_{j=1}^h \nu^j = n-m} \mathsf{E}\left[e^{\theta S^1(m,m+\nu^1)}\right] \mathsf{E}\left[e^{\theta S^2(m+\nu^1,m+\nu^1+\nu^2)}\right]\cdots \mathsf{E}\left[e^{\theta S^h(m+\sum_{j=1}^{h-1}\nu^j, m+\sum_{j=1}^{h}\nu^j)}\right],$$

where we used a variable substitution, estimated the maximum by the sum of its arguments, and used the statistical independence of the stages. Given homogeneous stages that are (σ_S,ρ_S) constrained as specified by Definition 2 and using [39, Prop. 6.2] to replace the sum by a binomial coefficient, we have for $\theta > 0$ that

$$\mathsf{E}\left[e^{\theta S^{\text{net}}(m,n)}\right] \le \binom{n-m+h-1}{h-1} e^{\theta(h\sigma_S(\theta)+\rho_S(\theta)(n-m+h))}.$$

Next, we derive for the MGF of the sojourn time from (3) for $\theta > 0$ that

$$\mathsf{E}\left[e^{\theta T(n)}\right] \le \sum_{\nu=1}^{n} \mathsf{E}\left[e^{\theta S^{\text{net}}(\nu,n)}\right]\mathsf{E}\left[e^{-\theta A(\nu,n)}\right],$$

where we estimated the maximum by the sum of its arguments and used the statistical independence of arrivals and service. Considering (σ_A,ρ_A) constrained traffic as in Definition 2, we have

$$\mathsf{E}\left[e^{\theta T(n)}\right] \le e^{\theta(\sigma_A(-\theta)+h\sigma_S(\theta)+h\rho_S(\theta))} \sum_{\nu=1}^{n}\binom{n-\nu+h-1}{h-1} e^{-\theta(\rho_A(-\theta)-\rho_S(\theta))(n-\nu)}.$$

Next, we estimate for $\rho_S(\theta) < \rho_A(-\theta)$ that

$$\sum_{\nu=1}^{n}\binom{n-\nu+h-1}{h-1} e^{-\theta(\rho_A(-\theta)-\rho_S(\theta))(n-\nu)} \le \sum_{\nu=0}^{\infty}\binom{\nu+h-1}{h-1}\left(e^{-\theta(\rho_A(-\theta)-\rho_S(\theta))}\right)^{\nu} = \left(1-e^{-\theta(\rho_A(-\theta)-\rho_S(\theta))}\right)^{-h},$$

where we used that the argument of the sum is expressed as a negative binomial probability. With Chernoff's bound we have

$$\mathsf{P}[T(n)\ge\tau] \le \frac{e^{\theta(\sigma_A(-\theta)+h\sigma_S(\theta))}}{\left(1-e^{-\theta(\rho_A(-\theta)-\rho_S(\theta))}\right)^h} e^{\theta h\rho_S(\theta)} e^{-\theta\tau}.$$

Finally, we insert the service parameters of the tasks $\sigma_S(\theta)=\sigma_Q(\theta)+\ln(k)/\theta$ and $\rho_S(\theta)=\rho_Q(\theta)$ from (12) and (13) for each of the fork-join stages to complete the proof. □

Scaling of Multi-Stage Fork-Join Networks. To evaluate the growth of τ with h and k, we equate the sojourn time bound in Theorem 2 with ε and solve for

$$\tau = \sigma_A(-\theta) + h(\sigma_Q(\theta)+\rho_Q(\theta)) + \frac{1}{\theta}\left(h\ln k - h\ln\left(1-e^{-\theta(\rho_A(-\theta)-\rho_Q(\theta))}\right) - \ln\varepsilon\right)$$

that grows in $\mathcal{O}(h\ln k)$. The result compares to a growth in $\mathcal{O}(h\ln(hk))$ obtained previously in [1]. The improvement is achieved by taking advantage of the statistical independence of the stages, whereas [1] does not make this assumption.

4 Multi-Queue Load Balancing Systems

We compare the performance of fork-join systems to that of traditional load balancing systems, both with multiple queues. A schematic of such a system with k servers is depicted in Fig. 4. In this system jobs are not divided into tasks; instead each job is assigned in its entirety to one of the servers. Therefore, in the language of point processes, the external arrival process $A(n)$ is divided into k *thinned* processes $A_i(m)$. Thinning is an operation whereby some of the points in the point process are removed. In the case of a multi-queue load balancing system we have an external arrival process $A(n)$, specifying the arrival time of job n, and a collection of k thinned arrival processes, $A_i(m)$, $i\in[1,k]$, denoting the arrival time of the mth job of the ith thinned process at server i. The corresponding service time is $L_i(m)$. The opposite operation, where multiple point processes are combined, is called superposition. The thinned processes have the constraint that the superposition of the $A_i(m)$ must give the external arrival process, $A(n)$. That is, each job is mapped to exactly one server. Superposition of the departure processes $D_i(m)$, $i\in[1,k]$ gives the external

departure process $D(n)$ that may optionally be resequenced in the original order of $A(n)$.

In the case of random thinning, each job is assigned to one of the k servers according to iid discrete (not necessarily uniform) random variables with support $[1,k]$. From the iid property, the mapping of each job to a certain server $i \in [1,k]$ is an independent Bernoulli trial with parameter p_i where $\sum_{i=1}^{k} p_i = 1$. Let $X_i(m)$ denote the number of the job that becomes the mth job that is assigned to server i. It follows that $X_i(m)$ is a sum of m iid geometric random variables with parameter p_i; i.e., $X_i(m)$ is negative binomial. The arrival process at server i is

$$A_i(m) = A(X_i(m)), \tag{17}$$

for $m \geq 1$. Conversely, given jobs $1, 2, \ldots, n$ of the external arrival process, let $Y_i(n)$ denote the number of jobs assigned to server i. It follows that $Y_i(n)$ is binomial with parameter p_i. Further, it holds that $X_i(Y_i(n)) \leq n$ for $n \geq 1$.

In the case of deterministic thinning, a round-robin assignment of the jobs of an arrival process $A(n)$ to k servers results in the processes $A_i(m)$ as in (17) where

$$X_i(m) = k(m-1) + i, \tag{18}$$

for $m \geq 1$ and $i \in [1,k]$. Given jobs $1, 2, \ldots, n$, the number of jobs that are assigned to server i is

$$Y_i(n) = \left\lceil \frac{n-i+1}{k} \right\rceil, \tag{19}$$

for $n \geq 1$ and $i \in [1,k]$. To see this, note that job n of the external arrival process becomes the $m = \lceil n/k \rceil$th job of server $j = (n-1) \bmod k + 1$. Hence, $Y_i(n) = m$ for $i \leq j$ and $Y_i(n) = m-1$ for $i > j$. The same is verified for (19).

Corollary 2 (Multi-queue load balancing). *Assume arrivals with iid inter-arrival times and parameter $\rho_A(-\theta)$ for $\theta > 0$ as in (5). In the case of random thinning with probabilities p_i for $i \in [1,k]$, the thinned arrival processes have parameter*

$$\rho_{A_i}(-\theta) = -\frac{1}{\theta} \ln\left(\frac{p_i e^{-\theta \rho_A(-\theta)}}{1-(1-p_i)e^{-\theta \rho_A(-\theta)}} \right),$$

where $\theta > 0$ so that $e^{-\theta \rho_A(-\theta)} < 1/(1-p_i)$.

In the case of deterministic thinning, for $\theta > 0$ the thinned arrival processes have parameter

$$\rho_{A_i}(-\theta) = k \rho_A(-\theta).$$

Proof. The thinned arrivals are expressed by (17) as a doubly random process that has increments $A_i(\nu, \nu+1) = A(X_i(\nu), X_i(\nu+1))$ for $\nu \geq 1$. Considering iid inter-arrival times, with [39, Ex. 7j] the MGF of the thinned process is

$$\mathsf{M}_{A_i(\nu,\nu+1)}(-\theta) = \mathsf{E}\left[\left(\mathsf{M}_{A(1,2)}(-\theta) \right)^{X_i(1)} \right], \tag{20}$$

for $\nu \geq 1$. After some reordering, it follows that

$$\mathsf{M}_{A_i(\nu,\nu+1)}(-\theta) = \mathsf{M}_{X_i(1)}(\ln \mathsf{M}_{A(1,2)}(-\theta)). \tag{21}$$

Since $X_i(1)$ is a geometric random variable with MGF $\mathsf{M}_{X_i(1)}(\theta) = p_i e^{\theta}/(1-(1-p_i)e^{\theta})$ for $\theta < -\ln(1-p_i)$, we obtain by insertion of (21) into (5) that

$$\rho_{A_i}(-\theta) = -\frac{1}{\theta} \ln\left(\frac{p_i \mathsf{M}_{A(1,2)}(-\theta)}{1-(1-p_i)\mathsf{M}_{A(1,2)}(-\theta)} \right), \tag{22}$$

where $\theta > 0$ so that $\mathsf{M}_{A(1,2)}(-\theta) < 1/(1-p_i)$.

In the case of deterministic thinning, (20) simplifies to $\mathsf{M}_{A_i(\nu,\nu+1)}(-\theta) = (\mathsf{M}_{A(1,2)}(-\theta))^k$ so that (5) evaluates to $\rho_{A_i}(-\theta) = k\rho_A(-\theta)$. □

Since each of the servers in the load balancing system serves its arriving jobs independently, statistical performance bounds are obtained by insertion of the arrival parameters from Corollary 2 into Theorem 1. Further, the modularity of this approach allows for the case where the servers themselves are fork-join systems, as could be the case if a scheduler were performaning load balancing between racks in a data center. This case is handled by insertion of Corollary 2 into Corollary 1. We provide an example of such a hybrid system in [41].

The main effect of load balancing is captured in the stability condition in Theorem 1, which becomes $\rho_L(\theta) < \rho_{A_i}(-\theta)$, where $\rho_{A_i}(-\theta)$ increases with k. Given fixed $\rho_L(\theta)$, the increase of $\rho_{A_i}(-\theta)$ permits larger θ that yield a faster tail decay.

4.1 Tail Decay of Random versus Deterministic Load Balancing

For this comparison, we consider arrivals with iid exponential inter-arrival times and jobs with iid Erlang-k service times with parameters λ and μ, respectively. This choice of service time distribution will be motivated in Section 4.2.

Performance bounds for the multi-queue load balancing system are obtained from Theorem 1 using the parameters of the thinned arrival processes given in Corollary 2. Deterministic thinning gives processes $A_i(m)$ where the inter-arrival times are a sum of k exponential random variables; that is, Erlang-k distributed. It follows by insertion of $\rho_A(-\theta)$ from (8) into Corollary 2 that

$$\rho_{A_i}(-\theta) = -\frac{k}{\theta} \ln\left(\frac{\lambda}{\lambda+\theta} \right), \tag{23}$$

for $\theta > 0$. In the case of random thinning we have by insertion of $\rho_A(-\theta)$ from (8) into Corollary 2 with $p_i = 1/k$ that

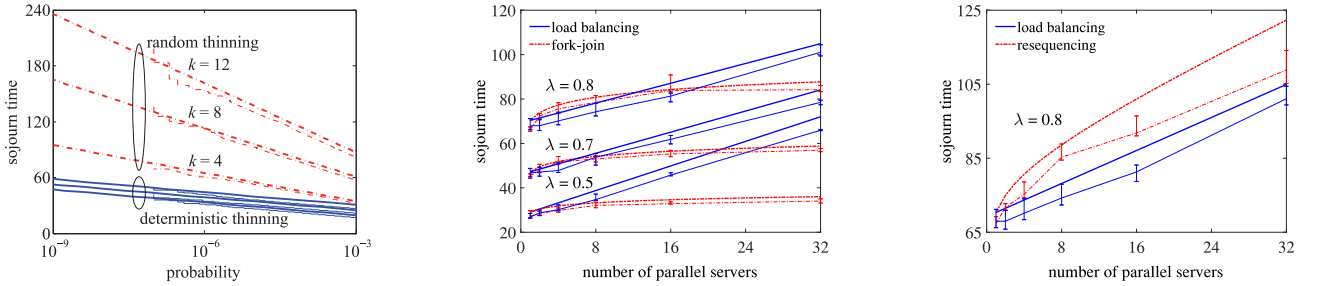
$$\rho_{A_i}(-\theta) = -\frac{1}{\theta} \ln\left(\frac{\lambda}{\lambda+k\theta} \right), \tag{24}$$

for $\theta > 0$. In this case, the inter-arrival times of the thinned processes are exponentially distributed with parameter λ/k. Lastly, the Erlang-k service times of the jobs have parameter

$$\rho_L(\theta) = \frac{k}{\theta} \ln\left(\frac{\mu}{\mu-\theta} \right), \tag{25}$$

for $\theta \in (0, \mu)$. For deterministic thinning, the maximal θ that satisfies the stability condition $\rho_L(\theta) \leq \rho_{A_i}(-\theta)$ is $\theta = \mu - \lambda$.

Fig. 5a contrasts the tail decay of deterministic and random thinning for $\mu = 1$, $\lambda = 0.5$, and $k \in \{4, 8, 12\}$. The speed of the tail decay does not depend on k for deterministic thinning. In contrast, for random thinning the tail decay becomes slower with increasing k. Deterministic thinning



(a) deterministic vs. random (b) load balancing vs. fork-join, $\varepsilon = 10^{-6}$ (c) load balancing with resequencing, $\varepsilon = 10^{-6}$

Fig. 5. Comparison of different multi-queue load balancing configurations. Analytical bounds (thick lines) and simulation results (thin lines). (a) Deterministic thinning outperforms random thinning. (b) Sojourn time bounds grow linearly with k in the case of load balancing systems with deterministic thinning and with $\ln k$ for fork-join systems. (c) Resequencing the departures of the load balancing system adds another delay that grows with $\ln k$.

generally outperforms random thinning. Hence, we only include deterministic thinning in the next section.

4.2 Comparison to Multi-Queue Fork-Join Systems

In order to facilitate a comparison between fork-join systems, where jobs are divided into tasks, and load balancing systems where they are not, the distribution of job sizes in the load balancing case must be the same as the distribution of the sum of the task sizes in the fork-join case. That is, the total job sizes must have the same distribution. One example of this is a fork-join system with exponential task service times, and a load balancing system with Erlang-k job service times, both with the same rate parameter, μ. In general, given iid task service times with parameter $\rho_Q(\theta)$ for $i \in [1,k]$ as defined by (6), the job service times have parameter $\rho_L(\theta) = k\rho_Q(\theta)$.

Considering the load balancing system, we obtain the stability condition $k\rho_Q(\theta) < k\rho_A(-\theta)$ by insertion of $\rho_L(\theta)$ and $\rho_{A_i}(-\theta)$ from Corollary 2 into Theorem 1. Hence, the maximal θ that achieves the stability condition is independent of k. As a consequence, the waiting time bound from Theorem 1 and the speed of the tail decay of the sojourn time bound do not depend on k. Regarding the sojourn time, we equate the bound from Theorem 1 with ε and solve for

$$\tau \leq k\rho_Q(\theta) + \frac{\ln\alpha - \ln\varepsilon}{\theta}, \tag{26}$$

for $\theta > 0$ under the stability condition $\rho_Q(\theta) < \rho_A(-\theta)$. Eq. (26) shows a linear growth with k. The result compares to the logarithmic growth established by (14) for the fork-join system. If k is large, the sojourn time of a job at the load balancing system is dominated by its service time, which depends linearly on k. The fork-join system avoids this effect, as the tasks of the jobs are served by k servers in parallel.

Fig. 5b compares the performance of the load balancing system with the fork-join system, as already evaluated in Fig. 3a, for $\mu = 1$, $\lambda \in \{0.5, 0.7, 0.8\}$, and $\varepsilon = 10^{-6}$. Clearly, the sojourn time bounds of the load balancing system with deterministic thinning grow at most linearly with k, as established by (26). For large k the sojourn time of a job is dominated by its service time, so that the curves that are depicted for different arrival rates λ converge slowly. The fork-join system mitigates the impact of large jobs by serving the tasks in parallel. It achieves a scaling in $\ln k$, see (14), that is due to the synchronization constraint of the join operation. The fork-join system mostly outperforms the load balancing system, except in the case of large λ and small k. The reason is that in a fork-join system the occurrence of a large task blocks all subsequent tasks of that server so the respective jobs cannot complete the join operation. In contrast, in a load balancing system there is no synchronization constraint, so subsequent jobs that are served by other servers can finish service earlier.

4.3 Resequencing

Unlike fork-join systems, load balancing systems do not guarantee that jobs depart in their order of arrival. An optional resequencing step is depicted in Fig. 4. It applies for example in case of a multi-path transmission of packet data, e.g., using multi-path TCP. As the resequencing step does not affect the waiting time of a job, we state the following corollary only for the sojourn time.

Corollary 3 (Load balancing with resequencing). *Consider a multi-queue load balancing system of k parallel servers as in Lemma 1 with resequencing. The thinned arrival processes have parameter $\rho_{A_i}(-\theta)$ as given in Corollary 2 and the jobs have service parameters $(\sigma_L(\theta), \rho_L(\theta))$ as given in Definition 2. For $n \geq 1$, the sojourn time satisfies*

$$\mathsf{P}[T(n) > \tau] \leq k\alpha e^{\theta\rho_L(\theta)} e^{-\theta\tau}.$$

In the case of GI|G arrival and service processes, the free parameter $\theta > 0$ has to satisfy $\rho_L(\theta) < \rho_{A_i}(-\theta)$ and

$$\alpha = \frac{e^{\theta\sigma_L(\theta)}}{1 - e^{-\theta(\rho_{A_i}(-\theta) - \rho_L(\theta))}}.$$

In the case of GI|GI arrival and service processes, $\theta > 0$ has to satisfy $\rho_L(\theta) \leq \rho_{A_i}(-\theta)$ and $\alpha = 1$.

Proof. Given the departure processes of each of the servers $D_i(n)$ for $i \in [1,k]$, the combined in-sequence departure process for $n \geq 0$ is

$$D(n) = \max_{i\in[1,k]}\{D_i(Y_i(n))\}, \tag{27}$$

where $D_i(0) = 0$ by convention. Note that in evaluating (27) one only has to verify the departure of job $Y_i(n)$ from each server $i \in [1,k]$, since the departure of job $Y_i(n)$ from server i implies the departure of all jobs $\nu \in [1, Y_i(n)]$ of the same server. I.e., $D_i(\nu) \leq D_i(Y_i(n))$ for

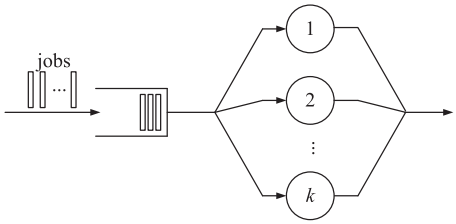


Fig. 6. Single-queue load balancing system. The system is non-idling; once a server finishes a job, the next job in the queue is assigned to that server.

$\nu \in [1, Y_i(n)]$, since each server implements first-in first-out order.

Next, we use Lemma 1 for each server $i \in [1,k]$ to obtain $D_i(Y_i(n)) = \max_{m\in[1,Y_i(n)]}\{A_i(m) + S_i(m, Y_i(n))\}$. By insertion into (27) we have

$$D(n) = \max_{i\in[1,k]}\left\{\sup_{m\in[1,Y_i(n)]}\{A_i(m) + S_i(m, Y_i(n))\}\right\}, \quad (28)$$

for $n \geq 1$, where $\sup\{\emptyset\} = 0$. We estimate the sojourn time $T(n) = D(n) - A(n) \leq D(n) - A_i(Y_i(n))$ for $n \geq 1$, where we used that $A(n) \geq A_i(Y_i(n))$ for $i \in [1,k]$. By insertion of (28), it follows for $n \geq 1$ that

$$T(n) \leq \max_{i\in[1,k]}\left\{\sup_{m\in[1,Y_i(n)]}\{S_i(m, Y_i(n)) - A_i(m, Y_i(n))\}\right\}.$$

By the union bound $\mathsf{P}[T(n) > \tau] \leq \sum_{i=1}^{k} \mathsf{P}[T_i(n) > \tau]$, where $T_i(n) \leq \sup_{m\in[1,Y_i(n)]}\{S_i(m, Y_i(n)) - A_i(m, Y_i(n))\}$. Finally, we estimate $\mathsf{P}[T_i(n) > \tau]$ using Theorem 1. □

To evaluate the scaling with k, we consider the case where the service time grows with k, expressed as $\rho_L(\theta) = k\rho_Q(\theta)$, as before. We investigate deterministic thinning and equate the sojourn time bound from Corollary 3 with ε to solve for

$$\tau \leq k\rho_Q(\theta) + \frac{\ln k + \ln \alpha - \ln \varepsilon}{\theta} \quad (29)$$

for $\theta > 0$ under the stability condition $k\rho_Q(\theta) < k\rho_A(-\theta)$. Eq. (29) shows two effects: a linear growth with k that is due to the increase of the job service time, as also observed for the load balancing system in (26), and a logarithmic term, $\ln k$, that is due to resequencing.

Scaling of Load Balancing with Resequencing. We show a numerical comparison of multi-queue load balancing systems with deterministic thinning, and with and without resequencing in Fig. 5c. We use the same parameters as above. The results clearly show the additional logarithmic delay due to resequencing. Compared to the fork-join system, resequencing consumes the advantage that load balancing systems showed for large λ in Fig. 5b.

5 Single-Queue Load Balancing Systems

A major drawback of both types of multi-queue systems is that servers may idle while tasks or jobs are queued at other servers. This is due to the early and static assignment of tasks to servers at their time of arrival. Single-queue systems assume we have additional feedback from the servers and can perform a dynamic assignment of tasks or jobs to the servers as they become available. This eliminates idling of servers when there are unserviced jobs or tasks in the system.

We assume that feedback and job/task assignment are instantaneous. In a real single-queue system there would be some overhead incurred from moving a task's code and data to its assigned server. One way to include this in our model is to incorporate the overhead into the service time distribution. Another complication of single-queue systems is that jobs may depart out of order; i.e., $D(n) \ngeq D(n-1)$. This implies also that the waiting time of job n cannot simply be determined from $D(n-1)$ as in (4) and Theorem 1.

In this section we derive performance bounds for single-queue load balancing systems without a resequencing constraint, i.e., $G \,|\, M \,|\, k$ queues, as shown in Fig. 6. First, we prove that the system satisfies the definition of max-plus server.

Lemma 4 (Single-queue load balancing system). *Consider a single-queue load balancing system with k parallel servers as in Lemma 1. Let $L(n)$ denote the service time of job n for $n \geq 1$. Given job n starts service at $V(n)$, define $Z(n)$ to be the time until the next server becomes idle. Trivially, $Z(n) = 0$ if there is an idle server at $V(n)$. Define for $n \geq m \geq 1$*

$$S(m,n) = L(n) + \sum_{\nu=m}^{n-1} Z(\nu).$$

i) The system is an exact $S(m,n)$ server.
ii) Given that the jobs have iid exponential service times with parameter μ. The non-zero elements of $Z(n)$ are iid exponential random variables with parameter $k\mu$.
iii) Replace the zero elements of $Z(n)$ by iid exponential random variables with parameter $k\mu$ and compute $S(m,n)$ as above. The system is an $S(m,n)$ server.

Proof. Using the definition of $Z(n)$, it holds for $n \geq 2$ that

$$V(n) = \max\{A(n), V(n-1) + Z(n-1)\}. \quad (30)$$

Further, $V(1) = A(1)$ and since $Z(n) = 0$ for $n \in [1, k-1]$ we have $V(n) = A(n)$ also for $n \in [2,k]$. By recursive insertion of (30) we obtain for $n \geq 1$ that

$$V(n) = \max_{m\in[1,n]}\left\{A(m) + \sum_{\nu=m}^{n-1} Z(\nu)\right\}. \quad (31)$$

Since $D(n) = V(n) + L(n)$ we have with (31) that $D(n) = \max_{m\in[1,n]}\{A(m) + S(m,n)\}$, which proves the first part.

Next, we consider iid exponential service times with parameter μ and investigate the distribution of $Z(n)$ for $n \geq 1$. Given all servers are busy after the start of service of job n at $V(n)$. This implies that there are another $k-1$ jobs with indices smaller n that are already in service at $V(n)$. Due to the memorylessness of the exponential distribution, the residual service time of each of these jobs as well as the service time of job n are iid exponential random variables with parameter μ. Since the minimum of k iid exponential random variables with parameter μ is an exponential random variable with parameter $k\mu$, it

follows that the time until the next server becomes idle is exponentially distributed with parameter $k\mu$.

For the last part, we use that exponential random variables are non-negative. If we replace all $Z(n)$ that are zero by iid exponential random variables with parameter $k\mu$, (31) becomes $V(n) \le \max_{m\in[1,n]}\{A(m) + \sum_{\nu=m}^{n-1} Z(\nu)\}$ for $n \ge 1$, and consequently $D(n) \le \max_{m\in[1,n]}\{A(m) + S(m,n)\}$. □

Considering iid exponential service times $L(n)$ with parameter μ, we have $\rho_L(\theta)$ for $\theta \in (0,\mu)$ as in (9). With Lemma 4 (iii), $Z(n)$ is composed of iid exponential random variables with parameter $k\mu$. Invoking (9) with parameter $k\mu$ gives

$$\rho_Z(\theta) = \frac{1}{\theta}\ln\left(\frac{k\mu}{k\mu-\theta}\right), \tag{32}$$

for $\theta \in (0, k\mu)$. With (9) and (32) the MGF of $S(m,n)$ in Lemma 4 (iii) is $\mathsf{E}\left[e^{\theta S(m,n)}\right] = e^{\theta(\rho_L(\theta)+\rho_Z(\theta)(n-m))}$ for $\theta \in (0,\mu)$. Hence, $S(m,n)$ satisfies Definition 2 with parameters $\sigma_S(\theta) = \rho_L(\theta) - \rho_Z(\theta)$ and $\rho_S(\theta) = \rho_Z(\theta)$.

Lemma 4 (iii) verifies that the single-queue load balancing system is an $S(m,n)$ server, where $S(m,n)$ is composed of independent increments. Hence, a sojourn time bound can be derived from Theorem 1 using the parameters $(\sigma_S(\theta), \rho_S(\theta))$ defined above. However, the waiting time bound of Theorem 1 uses a definition of waiting time $W(n) = [D(n-1) - A(n)]^+$ that does not apply to the single-queue load balancing system, where the departure of job $n-1$ does not generally mark the start of the service of job n. The following theorem first formulates the waiting time for single-queue load balancing systems and in a second step derives a sojourn time bound from the waiting time. The derivation avoids the technical limitation $\theta \in (0,\mu)$ that applies if (9) is inserted into Theorem 1 and thus enables tighter bounds.

Theorem 3 (Single-queue load balancing system). *Consider a single-queue load balancing system as in Lemma 4, with arrival parameters $(\sigma_A(-\theta), \rho_A(-\theta))$ as specified by Definition 2, iid exponential job service times with parameter μ, and parameter $\rho_Z(\theta)$ given by (32). For $n \ge 1$, the sojourn time satisfies*

$$\mathsf{P}[T(n) > \tau] \le \alpha\frac{\mu}{\mu-\theta}\left(e^{-\theta\tau} - e^{-\mu\tau}\right) + e^{-\mu\tau},$$

and the waiting time

$$\mathsf{P}[W(n) > \tau] \le \alpha e^{-\theta\tau}.$$

In the case of $G|M$ arrival and service processes, the free parameter $\theta \in (0, k\mu), \theta \ne \mu$ has to satisfy $\rho_Z(\theta) < \rho_A(-\theta)$ and

$$\alpha = \frac{e^{\theta\sigma_A(-\theta)}}{1 - e^{-\theta(\rho_A(-\theta)-\rho_Z(\theta))}}.$$

In the special case of $GI|M$ arrival and service processes, $\theta \in (0, k\mu), \theta \ne \mu$ has to satisfy $\rho_Z(\theta) \le \rho_A(-\theta)$ and $\alpha = 1$.

Proof. With Lemma 4 (iii) and (31), the waiting time $W(n) = V(n) - A(n)$ for $n \ge 1$ is estimated as

$$W(n) \le \max_{m\in[1,n]}\left\{\sum_{\nu=m}^{n-1} Z(\nu) - A(m,n)\right\},$$

where $Z(n)$ for $n \ge 1$ are iid exponential random variables with parameter $k\mu$. Thereafter, the derivation of the statistical waiting time bound closely follows the proof of Theorem 1 and is omitted.

To derive a sojourn time bound we use that $D(n) = V(n) + L(n)$ so that $T(n) = D(n) - A(n)$ can be expressed as $T(n) = W(n) + L(n)$, where we substituted $W(n) = V(n) - A(n)$. We use the waiting time bound from Theorem 3 to estimate the waiting time CDF[2] as $F_W(\tau) = \mathsf{P}[W(n) \le \tau] \ge 1 - \alpha e^{-\theta\tau}$. By convolution with the exponential job service time PDF $f_L(\tau) = \mu e^{-\mu\tau}$ for $\tau \ge 0$ we obtain the CDF of the sojourn time as

$$F_T(\tau) = \int_0^\tau F_W(\tau - x) f_L(x) dx$$

that evaluates for $\theta \ne \mu$ to

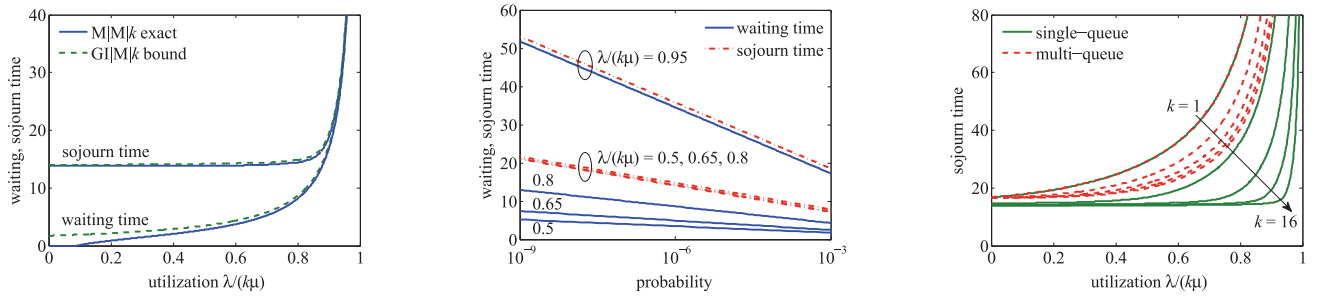
$$F_T(\tau) \ge 1 - e^{-\mu\tau} - \alpha\frac{\mu}{\mu-\theta}(e^{-\theta\tau} - e^{-\mu\tau}),$$

which completes the proof. □

Accuracy Compared with the $M|M|k$ Queue. To obtain exact reference results, we consider the special case of the $M|M|k$ queue. While the following conclusions can be readily obtained from queueing theory, we note that Theorem 3 is not limited to exponential arrivals. Fig. 7a compares the waiting time and the sojourn time bounds with the exact results for $k = 8$, $\varepsilon = 10^{-6}$, $\mu = 1$ and different utilizations defined as $\lambda/(k\mu)$. The exact waiting time distribution of the $M|M|k$ queue is $\mathsf{P}[W(n) > \tau] = P_k e^{-(k\mu-\lambda)\tau}$ where P_k is the probability of waiting, i.e., the probability that k or more jobs are in the system [42]. The bounds from Theorem 3 are obtained by insertion of $\rho_A(-\theta)$ from (8). From the stability condition $\rho_Z(\theta) \le \rho_A(-\theta)$ we find $\theta \le k\mu - \lambda$. By the choice of maximal θ, the waiting time bound $\mathsf{P}[W(n) > \tau] \le e^{-(k\mu-\lambda)\tau}$ exhibits the same exponential speed of decay and differs by the prefactor P_k that approaches one for high utilization. Fig. 7a confirms the good agreement of the bounds. Visible differences appear only in the case of the waiting time and low utilization.

Service versus Waiting Time. Regarding the sojourn time bound in Fig. 7a, two effects can be distinguished. These are expressed by the two parts of the sojourn time bound in Theorem 3 that decay as $e^{-\theta\tau}$ and $e^{-\mu\tau}$, respectively. In the case of high utilization, the sojourn time is dominated by the waiting time that decays with $e^{-\theta\tau}$ where $\theta = k\mu - \lambda$. Otherwise, if the utilization satisfies $\lambda/(k\mu) < 1 - 1/k$ (that is 0.875 for $k = 8$ here), it follows that $\theta > \mu$ so that the waiting time decays quickly and the sojourn time is mostly due to the service time of the job itself that decays slower with $e^{-\mu\tau}$. Fig. 7b details the effect, again for $k = 8$. For utilizations below 0.875, the waiting time decays faster than the service time so that the sojourn time changes only

2. We note that for $\alpha > 1$, a tighter bound can be derived, using that $F_W(\tau) = 0$ for $\tau < -\ln(1/\alpha)/\theta$. We omit the details for notational brevity.



(a) bound vs. exact result, $\varepsilon = 10^{-6}$ (b) tail decay (c) single- vs. multi-queue system, $\varepsilon = 10^{-6}$

Fig. 7. Single-queue load balancing system. (a) The bounds agree closely with the exact result. (b) For high utilization the sojourn time is dominated by the waiting time, otherwise by the service time. (c) Comparison of single-queue and multi-queue load balancing systems for $k = \{1, 2, 4, 8, 16\}$. The sojourn time decreases with k. The improvement is significantly larger in the case of the single-queue system that can sustain a utilization close to one if k is large.

marginally if the utilization is increased from 0.5 to 0.8. In contrast, once the utilization exceeds 0.875, the waiting time dominates.

Comparison to Multi-Queue Load Balancing. Fig 7c evaluates the sojourn time of the single-queue load balancing system for $k = \{1, 2, 4, 8, 16\}$ with respect to the multi-queue system. For comparability, we use the same technique to derive the sojourn time from the waiting time as in Theorem 3 for both systems. While for $k = 1$ the systems are identical, the single-queue load balancing system outperforms the multi-queue system for larger k. Given a target sojourn time bound, the single-queue system sustains a significantly higher utilization.

6 Single-Queue Fork-Join Systems

In a single-queue fork-join system, jobs are composed of k tasks that are stored in a single-queue. Once any of the k parallel servers becomes idle, it fetches the next task from the head of the queue. A diagram is shown in Fig. 8. Our analysis of single-queue fork-join systems follows the same essential steps as in the case of the single-queue load balancing systems with the additional synchronization constraint of the join operation.

Lemma 5 (Single-queue fork-join system). *Consider a single-queue fork-join system with k parallel servers as in Lemma 1. Let $Q_i(n)$ denote the service time of task i of job n for $n \geq 1$ and $i \in [1, k]$. Given task i of job n starts service at $V_i(n)$, define $Z_i(n)$ to be the time until the next server becomes idle. Trivially $Z_i(n) = 0$ if there is an idle server at $V_i(n)$. Define for $n \geq m \geq 1$*

$$S(m,n) = \max_{i \in [1,k]} \left\{ Q_i(n) + \sum_{j=1}^{i-1} Z_j(n) + \sum_{\nu=m}^{n-1} \sum_{j=1}^{k} Z_j(\nu) \right\}.$$

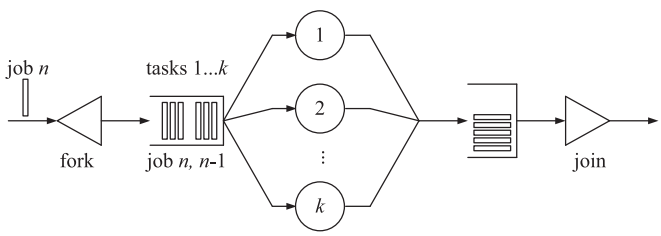


Fig. 8. Single-queue fork-join system. The system is non-idling; once a server finishes a job, the next task in the queue is assigned to that server. The join operation uses a random access buffer to complete jobs (possibly out of sequence) once all tasks are finished.

i) The system is an exact $S(m,n)$ server.
ii) Given that the tasks have iid exponential service times with parameter μ. The non-zero elements of $Z_i(n)$ are iid exponential random variables with parameter $k\mu$.
iii) Replace the zero elements of $Z_i(n)$ by iid exponential random variables with parameter $k\mu$ and compute $S(m,n)$ as above. The system is an $S(m,n)$ server.

Proof. Using the definition of $Z_i(n)$, it holds for $n \geq 1$ and $i \in [2, k]$ that

$$V_i(n) = \max\{A(n), V_{i-1}(n) + Z_{i-1}(n)\}, \tag{33}$$

and for $n \geq 2$ and $i = 1$

$$V_1(n) = \max\{A(n), V_k(n-1) + Z_k(n-1)\}. \tag{34}$$

Further, $V_1(1) = A(1)$ and since $Z_i(1) = 0$ for $i \in [1, k-1]$ we have $V_i(1) = A(1)$ also for $i \in [2, k]$. By recursive insertion of (33) and (34) we obtain for $n \geq 1$ and $i \in [1, k]$ that

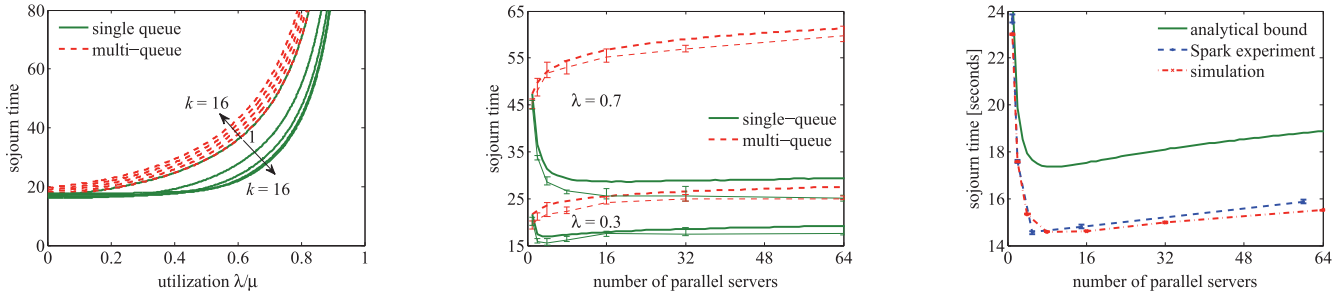
$$V_i(n) = \max_{m \in [1,n]} \left\{ A(m) + \sum_{j=1}^{i-1} Z_j(n) + \sum_{\nu=m}^{n-1} \sum_{j=1}^{k} Z_j(\nu) \right\}. \tag{35}$$

Above we used that $Z_i(n)$ is non-negative to reduce the number of terms that are evaluated by the maximum operator. Given $V_i(n)$, task i of job n finishes service after another $Q_i(n)$ units of time at $D_i(n) = V_i(n) + Q_i(n)$. Finally, job n is completed once all of its tasks have finished service at $D(n) = \max_{i \in [1,k]}\{V_i(n) + Q_i(n)\}$. Inserting (35) and reordering the maxima proves the first part.

The proof of the remaining parts is a notational extension of the proof of Lemma 4 that considers tasks instead of jobs. □

Considering exponential service times $Q_i(n)$ with parameter μ, we have $\rho_Q(\theta)$ as in (9) for $\theta \in (0, \mu)$. With Lemma 5 (iii), $Z_i(n)$ is composed of iid exponential random variables with parameter $k\mu$ that are characterized by $\rho_Z(\theta)$ given by (32) for $\theta \in (0, k\mu)$. The MGF of $S(m,n)$ in Lemma 5 (iii) is

$$\begin{aligned} \mathsf{E}\left[e^{\theta S(m,n)}\right] &\leq \sum_{i=1}^{k} e^{\theta(\rho_Q(\theta) + \rho_Z(\theta)((n-m)k+i-1))} \\ &= \beta e^{\theta(\rho_Q(\theta) + k\rho_Z(\theta)(n-m))}, \end{aligned}$$



(a) $k = \{1, 2, 4, 8, 16\}$, $\varepsilon = 10^{-6}$ (b) $\lambda = \{0.3, 0.7\}$, $\varepsilon = 10^{-6}$ (c) Spark, $\lambda = 0.7$, $\varepsilon = 10^{-3}$

Fig. 9. (a) and (b) Comparison of single-queue and multi-queue fork-join systems. Analytical bounds (thick lines) and simulation results (thin lines). While the sojourn time of the multi-queue system grows with $\ln k$, the single-queue system achieves a significant improvement due to the fact that it is non-idling if k is increased, particularly in the case of high utilization. The gain diminishes for large k and is eventually consumed by the synchronization constraint of the join operation (parameter β in Theorem 4). (c) Spark experiment. The bound predicts the trend of the sojourn time with k of the Spark system.

for $\theta \in (0, \mu)$ where

$$\beta = \frac{e^{\theta k \rho_Z(\theta)} - 1}{e^{\theta \rho_Z(\theta)} - 1}. \tag{36}$$

Hence, $S(m, n)$ satisfies Definition 2 with parameters $\sigma_S(\theta) = \rho_Q(\theta) - k\rho_Z(\theta) + \ln \beta/\theta$ and $\rho_S(\theta) = k\rho_Z(\theta)$.

Theorem 4 (Single-queue fork-join system). *Consider a single-queue fork-join system as in Lemma 5, with arrival parameters $(\sigma_A(-\theta), \rho_A(-\theta))$ as specified by Definition 2, iid exponential task service times with parameter μ, parameter $\rho_Z(\theta)$ as specified by (32), and β as given in (36). For $n \geq 1$, the sojourn time satisfies*

$$\mathsf{P}[T(n) > \tau] \leq \alpha\beta \frac{\mu}{\mu - \theta} \left(e^{-\theta\tau} - e^{-\mu\tau}\right) + e^{-\mu\tau},$$

and the waiting time of task i

$$\mathsf{P}[W_i(n) > \tau] \leq \alpha e^{\theta(i-1)\rho_Z(\theta)} e^{-\theta\tau}.$$

In the case of $G|M$ arrival and service processes, the free parameter $0 < \theta < k\mu, \theta \neq \mu$ has to satisfy $k\rho_Z(\theta) < \rho_A(-\theta)$ and

$$\alpha = \frac{e^{\theta\sigma_A(-\theta)}}{1 - e^{-\theta(\rho_A(-\theta) - k\rho_Z(\theta))}}.$$

In the special case of $GI|M$ arrival and service processes, $0 < \theta < k\mu, \theta \neq \mu$ has to satisfy $k\rho_Z(\theta) \leq \rho_A(-\theta)$ and $\alpha = 1$.

We note that the waiting time of the task of job n that starts service last is simply the waiting time of task k of job n as all other tasks start service before, i.e., no maximum as in the multi-queue fork-join system is needed.

Proof. Similar to the proof of Theorem 3, we start with the waiting time that is expressed as $W_i(n) = V_i(n) - A(n)$ for task $i \in [1, k]$ of job $n \geq 1$. With Lemma 5 (iii) and (35) we have for $n \geq 1$ that

$$W_i(n) \leq \max_{m \in [1,n]} \left\{ \sum_{j=1}^{i-1} Z_j(n) + \sum_{v=m}^{n-1} \sum_{j=1}^{k} Z_j(v) - A(m, n) \right\},$$

where $Z_i(n)$ for $n \geq 1$ and $i \in [1, k]$ are iid exponential random variables with parameter $k\mu$. The derivation of the statistical waiting time bound closely follows the proof of Theorem 1 and is therefore omitted.

A sojourn time bound for each task $i \in [1, k]$ of job n follows from its waiting time bound by convolution with the exponential task service time PDF as in the proof of Theorem 3. Finally, estimating the maximum sojourn time of all tasks $i \in [1, k]$ of job n by use of the union bound leads to parameter β defined by (36). □

Performance Gain of Single-Queue Fork-Join Systems. We compare the single-queue fork-join system with the multi-queue system from Section 3. Jobs have iid exponential inter-arrival times and are composed of k tasks with iid exponential service times. The parameters $\rho_A(-\theta)$ and $\rho_Q(\theta)$ are as specified by (8) and (9), respectively, where we let $\mu = 1$. For comparability, we use the same technique to derive the sojourn time from the waiting time as in Theorem 4 for both systems. In Fig. 9a, we fix $\varepsilon = 10^{-6}$ and show the impact of the utilization λ/μ for different $k \in \{1, 2, 4, 8, 16\}$. For $k = 1$ the single-queue and the multi-queue system are identical and the sojourn time bounds from Corollary 1 and Theorem 4 agree. For increasing k, the sojourn time bound of the multi-queue fork-join system shows logarithmic growth with k; i.e., the lines are equally spaced. This effect is due to the synchronization constraint of the join operation. In contrast, the sojourn time bounds of the single-queue fork-join system improve with k with decreasing gain. Here, two opposing effects are superimposed: 1.) the fact that the single-queue system is non-idling and assigns the task from the head of the queue immediately to the next idle server achieves a gain if k is increased, most visible for medium to high utilization; 2.) the synchronization constraint of the join operation, similar to the case of the multi-queue fork-join system. Fig. 9b depicts the effects for fixed $\lambda = 0.3$ and $\lambda = 0.7$, respectively, $\varepsilon = 10^{-6}$, and varying k. For small λ the gain of the single-queue system is small. For intuition, if all servers of the two systems are idle at the time of a job arrival, the single-queue and the multi-queue system perform identically and the sojourn time is determined by the task with the maximal service time. For large λ the advantage of being non-idling becomes more significant as k is increased, but for large k the synchronization constraint of the join operation eventually consumes the gain.

Spark Experiments. The default scheduler of Apache Spark is a prominent implementation of a single-queue system.

Fig. 9c shows results for $\lambda = 0.7$, $\mu = 1$, and $\varepsilon = 10^{-3}$ from experiments on a live Spark cluster [43]. The units are in seconds. Our simulation results match the Spark measurements almost perfectly. Similarly, the sojourn time bound from Theorem 4 captures the trend that is observed for Spark as k is increased. First the gain due to the non-idling implementation dominates, and that is later consumed by the synchronization constraint.

Spark experiments were carried out using our MRSperf benchmarking software.[3] We run a Spark master in standalone mode, and k single-core executors in Docker containers. The program submits map-only jobs according to a configurable random process. The jobs have k tasks whose runtimes are also random from a configurable distribution. Our simulator, forkulator, is event-driven and written in Java.[4] It supports numerous queue types and arrival and service processes. A more detailed description of both can be found in [41], [43].

7 Conclusions

We formulated a general model of parallel systems in max-plus system theory which allows us to derive bounds on the waiting and sojourn time distributions of a variety of systems. This included the full matrix of single- versus multi-queue and fork-join versus load-balancing. In the multi-queue fork-join case we extended known results to obtain a bound that applies in the $\mathrm{G}\,|\,\mathrm{G}$ case, and evaluated the bound based on empirical cluster trace data that exhibited correlations in the arrival process. We also obtained bounds for multi-queue fork-join systems with h independent stages that scale in $\mathcal{O}(h \ln k)$ compared to $\mathcal{O}(h \ln(hk))$ without independent stages. We found that the single-queue systems achieve a fundamental performance gain over multi-queue systems that is due to load balancing and possible overtaking of jobs, and observed that the Spark task manager is more accurately modeled as a single-queue system. We included an experimental validation using simulation, as well as MapReduce trace data and measurements obtained from Spark experiments which show that the analytical bounds closely predict the actual performance of systems.

Acknowledgments

This manuscript is a revised and extended version of the paper [1] that appeared in the IEEE Infocom 2016 proceedings. This work was supported in part by the European Research Council (ERC) under StG 306644.

References

[1] M. Fidler and Y. Jiang, "Non-asymptotic delay bounds for (k,l) fork-join systems and multi-stage fork-join networks," in *Proc. IEEE INFOCOM*, Apr. 2016, pp. 1–9.
[2] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters." *Commun. ACM*, vol. 51, no. 1, pp. 107–113, 2008.
[3] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," in *Proc. USENIX Conf. Hot Topics Cloud Comput.*, 2010, pp. 10–10.

3. Software available at https://github.com/brentondwalker/spark-arrivals

4. Software available at https://github.com/brentondwalker/forkulator

[4] L. Flatto and S. Hahn, "Two parallel queues created by arrivals with two demands," *SIAM J. Appl. Math.*, vol. 44, no. 5, pp. 1041–1053, 1984.
[5] R. Nelson and A. N. Tantawi, "Approximate analysis of fork/join synchronization in parallel queues," *IEEE Trans. Comput.*, vol. 37, no. 6, pp. 739–743, Jun. 1988.
[6] A. S. Lebrecht and W. J. Knottenbelt, "Response time approximations in fork-join queues," in *Proc. 23rd Annu. UK Performance Eng. Workshop*, Jul. 2007, pp. 1–8.
[7] S.-S. Ko and R. F. Serfozo, "Sojourn times in G/M/1 fork-join networks," *Naval Res. Logistics*, vol. 55, no. 5, pp. 432–443, May 2008.
[8] X. Tan and C. Knessl, "A fork-join queueing model: Diffusion approximation, integral representations and asymptotics," *Queueing Syst.*, vol. 22, no. 3–5, pp. 287–322, Sep. 1996.
[9] S. Varma and A. M. Makowski, "Interpolation approximations for symmetric fork-join queues," *Perform. Evaluation*, vol. 20, no. 1–3, pp. 245–265, May 1994.
[10] E. Varki, "Mean value technique for closed fork-join networks," *ACM Sigmetrics Perf. Eval. Rev.*, vol. 27, no. 1, pp. 103–112, May 1999.
[11] B. Kemper and M. Mandjes, "Mean sojourn times in two-queue fork-join systems: bounds and approximations," *OR Spektrum*, vol. 34, no. 3, pp. 723–742, Jul. 2012.
[12] F. Alomari and D. A. Menascé, "Efficient response time approximations for multiclass fork and join queues in open and closed queuing networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 6, pp. 1437–1446, Jun. 2014.
[13] F. Baccelli, A. M. Makowski, and A. Shwartz, "The fork-join queue and related systems with synchronization constraints: Stochastic ordering and computable bounds," *Adv. Appl. Probab.*, vol. 21, no. 3, pp. 629–660, Sep. 1989.
[14] A. Rizk, F. Poloczek, and F. Ciucu, "Stochastic bounds in fork-join queueing systems under full and partial mapping," *Queueing Syst.: Theory Appl.*, vol. 83, no. 3, pp. 261–291, Aug. 2016.
[15] G. Kesidis, B. Urgaonkar, Y. Shan, S. Kamarava, and J. Liebeherr, "Network calculus for parallel processing," in *Proc. MAMA Workshop ACM SIGMETRICS*, Jun. 2015, pp. 48–50.
[16] R. Nelson, D. Towsley, and A. N. Tantawi, "Performance analysis of parallel processing systems," *IEEE Trans. Softw. Eng.*, vol. 14, no. 4, pp. 532–540, Apr. 1988.
[17] Q. Yin, Y. Jiang, S. Jiang, and P. Y. Kong, "Analysis of generalized stochastically bounded bursty traffic for communication networks," in *Proc. 27th Annu. IEEE Conf. Local Comput. Netw.*, Nov. 2002, pp. 141–149.
[18] C.-S. Chang, *Performance Guarantees in Communication Networks*. Berlin, Germany: Springer-Verlag, 2000.
[19] J.-Y. Le Boudec and P. Thiran, *Network Calculus A Theory of Deterministic Queuing Systems for the Internet*. Berlin, Germany: Springer-Verlag, 2001.
[20] Y. Jiang and Y. Liu, *Stochastic Network Calculus*. Berlin, Germany: Springer-Verlag, Sep. 2008.
[21] F. Poloczek and F. Ciucu, "Contrasting effects of replication in parallel systems: From overload to underload and back," arXiv:1602.07978v1, Feb. 2016, https://dl.acm.org/citation.cfm?id=2901499
[22] F. Baccelli, G. Cohen, G. J. Olsder, and J.-P. Quadrat, *Synchronization and Linearity: An Algebra for Discrete Event Systems*. Hoboken, NJ, USA: Wiley, 1992.
[23] J. Xie and Y. Jiang, "Stochastic network calculus models under max-plus algebra," in *Proc. IEEE Global Telecommun. Conf.*, 2009, pp. 1–6.
[24] Y. Jiang, "Network calculus and queueing theory: Two sides of one coin," in *Proc. 4th Int. ICST Conf. Performance Evaluation Methodologies Tools*, 2009, pp. 1–12.
[25] R. Lübben, M. Fidler, and J. Liebeherr, "Stochastic bandwidth estimation in networks with random service," *IEEE/ACM Trans. Netw.*, vol. 22, no. 2, pp. 484–497, Apr. 2014.
[26] J. Liebeherr, *Duality of the Max-Plus and Min-Plus Network Calculus*. Breda, the Netherlands: Now Publishers, 2017.
[27] R. L. Cruz, "A calculus for network delay, part I and II: Network elements in isolation and network analysis," *IEEE Trans. Inf. Theory*, vol. 37, no. 1, pp. 114–141, Jan. 1991.
[28] F. Ciucu, A. Burchard, and J. Liebeherr, "Scaling properties of statistical end-to-end bounds in the network calculus," *IEEE/ACM Trans. Netw.*, vol. 14, no. 6, pp. 2300–2312, Jun. 2006.
[29] M. Fidler, "An end-to-end probabilistic network calculus with moment generating functions," in *Proc. 14th IEEE Int. Workshop Quality Service*, Jun. 2006, pp. 261–270.

[30] F. Ciucu and J. Schmitt, "Perspectives on network calculus - no free lunch but still good value," in *Proc. ACM SIGCOMM*, Aug. 2012, pp. 311–322.
[31] M. Fidler and A. Rizk, "A guide to the stochastic network calculus," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 1, pp. 92–105, Jan.-Mar. 2015.
[32] F. P. Kelly, "Notes on effective bandwidths," in *Stochastic Networks: Theory and Applications*. Oxford, U.K.: Oxford Univ., 1996, pp. 141–168.
[33] R. J. Gibbens, "Traffic characterisation and effective bandwidths for broadband network traces," in *Stochastic Networks: Theory and Applications*. Oxford, U.K.: Oxford Univ. Press, 1996, no. 4, pp. 169–179.
[34] J. F. C. Kingman, "A martingale inequality in the theory of queues," *Math. Proc. Cambridge*, vol. 60, no. 2, pp. 359–361, Apr. 1964.
[35] F. Ciucu, "Network calculus delay bounds in queueing networks with exact solutions," in *Proc. Managing Traffic Performance Converged Netw.*, Jun. 2007, pp. 495–506.
[36] Y. Jiang, "A note on applying stochastic network calculus," Technical Report, Norwegian University of Science and Technology (NTNU), 2010.
[37] J. L. Doob, *Stochastic Processes*. Hoboken, NJ, USA: Wiley, 1953.
[38] I. Adan and J. Resing, *Queueing Systems*. Eindhoven, the Netherlands: TU Eindhoven, 2015.
[39] S. Ross, *A First Course in Probability*, 7th ed. London, U.K.: Pearson, 2006.
[40] C. Reiss, A. Tumanov, G. R. Ganger, H. Katz, and M. A. Kozuch, "Heterogeneity and dynamicity of clouds at scale: Google trace analysis," in *Proc. ACM Symp. Cloud Comput.*, 2012, Art. no. 7.
[41] M. Fidler, B. Walker, and Y. Jiang, "Non-asymptotic delay bounds for multi-server systems with synchronization constraints," Tech. Rep., arXiv:1610.06309v1, Oct. 2016, https://arxiv.org/abs/1610.06309
[42] D. Gross, J. F. Shortle, J. M. Thompson, and C. M. Harris, *Fundamentals of Queueing Theory*, 4th ed. Hoboken, NJ, USA: Wiley, 2008.
[43] B. Walker, "Benchmarking and simulating the fundamental scaling behaviors of a MapReduce engine," *Workshop Information-Centric Fog Comput. IFIP Netw.*, pp. 1–6, Jun. 2017.

Markus Fidler (M'04–SM'08) received the doctoral degree in computer engineering from RWTH Aachen University, Germany, in 2004. He was a post-doctoral fellow of NTNU Trondheim, Norway, in 2005 and the University of Toronto, ON, Canada, in 2006. During 2007 and 2008, he was an Emmy Noether Research group leader with Technische Universität Darmstadt, Germany. Since 2009, he has been a professor of communications networks with Leibniz Universität Hannover, Germany. He is a senior member of the IEEE.

Brenton Walker received the bachelor of science degree from the University of Wisconsin-Madison and the PhD degree in mathematics from the University of Maryland-College Park, in 2014. He is a post-doctoral researcher with Leibniz Universität Hannover, Germany. He was previously an ERCIM fellow in SICS in Stockholm Sweden.

Yuming Jiang received the PhD degree from the National University of Singapore, in 2001. He has been a professor with NTNU Trondheim, Norway, since 2005. From 1996 to 1997, he worked with Motorola, Beijing, China, and from 2001 to 2003, with the Institute for Infocomm Research (I2R), Singapore. His research interests include the provision, analysis, and management of quality of service guarantees in communication networks, with a particular focus on stochastic network calculus.