Delay Asymptotics and Bounds for Multi-Task Parallel Jobs

Weina Wang
Coordinated Science Lab
University of Illinois at Urbana-Champaign
Urbana, IL
weinaw@cs.cmu.edu

Mor Harchol-Balter
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA
harchol@cs.cmu.edu

Haotian Jiang
Department of Physics
Tsinghua University
Beijing, China
jhtdavid@cs.washington.edu

Alan Scheller-Wolf
Tepper School of Business
Carnegie Mellon University
Pittsburgh, PA
awolf@andrew.cmu.edu

R. Srikant
Coordinated Science Lab
University of Illinois at Urbana-Champaign
Urbana, IL
rsrikant@illinois.edu

ABSTRACT

We study delay of jobs that consist of multiple parallel tasks, which is a critical performance metric in a wide range of applications such as data file retrieval in coded storage systems and parallel computing. In this problem, each *job* is completed only when *all* of its tasks are completed, so the delay of a job is the maximum of the delays of its tasks. Despite the wide attention this problem has received, tight analysis is still largely unknown since analyzing job delay requires characterizing the complicated correlation among task delays, which is hard to do.

We first consider an asymptotic regime where the number of servers, n, goes to infinity, and the number of tasks in a job, $k^{(n)}$, is allowed to increase with n. We establish the asymptotic independence of any $k^{(n)}$ queues under the condition $k^{(n)} = o(n^{1/4})$. This greatly generalizes the asymptotic-independence type of results in the literature where asymptotic independence is shown only for a fixed constant number of queues. As a consequence of our independence result, the job delay converges to the maximum of independent task delays.

We next consider the non-asymptotic regime. Here we prove that independence yields a stochastic upper bound on job delay for any n and any $k^{(n)}$ with $k^{(n)} \leq n$. The key component of our proof is a new technique we develop, called "Poisson oversampling". Our approach converts the job delay problem into a corresponding balls-and-bins problem. However, in contrast with typical balls-and-bins problems where there is a negative correlation among bins, we prove that our variant exhibits positive correlation.

A full version of this paper will all proofs appears in [28].

Categories and Subject Descriptors

G.3 [**Mathematics of Computing**]: Probability and Statistics—*Queueing Theory*

IFIP WG 7.3 Performance 2018. Dec. 5–7, 2018, Toulouse, France

General Terms

Theory, Performance

Keywords

Large systems, asymptotic independence, association of random variables, parallel jobs

1. INTRODUCTION

The problem

We consider a system with n servers, each with its own queue. Jobs arrive over time according to a Poisson process, and each job consists of some number of tasks, k, where $k \leq n$. Upon arrival, each job chooses k distinct servers uniformly at random and sends one task to each server. Each server serves the tasks in its queue in a First-In, First-Out (FIFO) manner. A job is considered to be completed only when *all* of its tasks are completed. Our goal is to compute the *distribution* of *job delay*, namely the time from when a job arrives until the whole job completes. If a job's tasks experienced independent delays, then computing the distribution of job delay would be easy: take the maximum of the independent task delays. Unfortunately, the task delays are not independent in general.

Our model is a generalization on the classic *fork-join* model, which is identical to our model except that it assumes that $k = n$: every job is forked to all n servers. In contrast, in our model, the fork is *limited* to k servers with $k \leq n$. So we will refer to our model as the *limited fork-join* model. Obtaining tight analytical job delay characterizations for fork-join systems is known to be notoriously difficult: exact analysis of fork-join remains an open problem except for the two-server case [7, 1].

Motivation

Delay of jobs, rather than delay of individual tasks, is a more critical performance metric in systems with parallelism, yet a fundamental understanding of job delay is still lacking. One example application is data file retrieval in coded storage systems [12, 21, 15, 14, 22]. Here a job is the retrieval

of a data file, which is stored as multiple data chunks. The data chunks are in a coded form such that any k-sized subset of them is enough to reconstruct the file. Coded file retrieval can be modeled via the so-called (n, r, k) model [21] where a job can request r data chunks with $r \geq k$ and the job is completed as long as k of them are completed. Existing analysis of the (n, r, k) model is usually not tight except for the light load regime [12, 15]. The special case where $r = d$ and $k = 1$, called the Redundancy-d model, is also highly non-trivial and was solved just last year [10]. Job delay in general (n, r, k) models remains wide open. Within the coded file retrieval setting, our limited fork-join model can be viewed as the (n, k, k) problem.

Another application is parallel computing systems such as the "map" phase of the popular MapReduce framework [5], where a job is divided into tasks that can run in parallel. A few papers have been written to analytically approximate the delay of MapReduce jobs. Please see Section 2 for more details of related work.

In the above applications, load-balancing policies (see, e.g., [32, 29, 15, 14, 22] are usually used for assigning tasks to servers. For scenarios where either low-overhead is desired or information accessibility is constrained (such as in a distributed setting), workload agnostic assignment policies [29, 14, 22] can be preferred. Our limited fork-join model assumes a random task assignment policy, which is suitable for such application scenarios.

Our approach and what makes this problem hard

The root of the hardness of analyzing job delay in our model is the complicated correlation among queues, which leads to the correlation among the delays of a job's tasks. If the task delays were independent, then the probability distribution of job delay would have a simple form. In this paper, we are interested in developing conditions and quantifying in what sense the job delay can be approximated by the job delay under the independence assumption.

Asymptotic Regime. We first study a regime where we prove that a job's tasks can be *viewed* as being independent: We focus on the asymptotic regime where the number of servers, n, goes to infinity. Here we are specifically interested in developing conditions under which the delays of a job's tasks are *asymptotically independent*, i.e., their joint distribution converges to the product distribution of their marginals.

Asymptotic independence of a number of queues in large systems is often called "chaoticity" and studied under the name "propagation of chaos." In many papers [26, 31, 8, 9], asymptotic independence is simply assumed to simplify analysis. In some load-balancing settings, asymptotic independence has been proven (e.g., [3, 32]). One strong restriction of the existing proofs is that only a *constant* number of queues are proven to be asymptotically independent. In contrast, our goal is to establish asymptotic independence for any k queues where k may grow with n; we write k as $k^{(n)}$ to explicitly indicate its dependence on n. The asymptotic independence of any $k^{(n)}$ queues implies the asymptotic independence of the delays of a job's tasks since they are sent to $k^{(n)}$ queues. Allowing $k^{(n)}$ to grow with n captures the trends that data files get larger and that jobs are processing larger and larger data sets [4].

When proving asymptotic independence of a constant number of queues in steady state, it is typical to start by showing asymptotic independence over a *constant time interval* $[0, t]$, where t is long enough for these queues to be close to steady state. Unfortunately, since $k^{(n)}$ grows with n in our model, to reach steady state, the system needs a time interval $[0, \tau^{(n)}]$, growing with n. This further complicates the analysis since asymptotic independence then needs to be established over this longer, *non-constant*, time interval.

Non-asymptotic regime. Next, we study the non-asymptotic regime. We show that for *any* n and *any* $k^{(n)} = k$ with $k^{(n)} \leq n$, the distribution of job delay is *stochastically upper bounded* by the distribution given by independent task delays, which we call the *independence upper bound*. Therefore, independence not only characterizes the *limiting* behavior of job delay in the asymptotic regime where $n \to \infty$, but also yields an upper bound for *any* n. I.e., the independence upper bound is asymptotically *tight*. The independence upper bound is also tighter than all the existing upper bounds in prior work [20, 14].

We prove the independence upper bound using the theory of associated random variables [6]. Association (also called positive association) is a form of positive correlation, and it has the property that if a set of random variables are associated, then the maximum of them is stochastically upper bounded by the maximum of independent versions of them. To show the independence upper bound, it thus suffices to show that the delays of a job's tasks are associated. Such an association result is known for the classical fork-join model with $k^{(n)} = n$, but not for the limited fork-join model when $k^{(n)} < n$. When proving association, a commonly used idea is to observe the system at each job arrival time, and show that the numbers of tasks sent to different queues are associated [18, 13, 22]. This corresponds to a balls-and-bins problem where $k^{(n)}$ balls are thrown into n bins in the same way that the tasks are sent to the queues. What is needed is that the numbers of balls thrown in different bins are associated, which is obviously true for $k^{(n)} = n$ since they are all equal to one, but not true when $k^{(n)} < n$. In fact, they are actually *negatively associated* by a classical result [11]. However, this does not mean that the *steady-state* queues are negatively associated, leaving the association problem for $k^{(n)} < n$ unsolved in the literature. As pointed out in [14], it was not known if independence yielded a bound, either lower or upper.

We develop a novel technique that we call "Poisson oversampling," where we *observe* the system not only when jobs arrive but also at the jump times of a Poisson process that is independent of everything else. This oversampling does not change the dynamics in the system since it is only a way of observing. But now at each observation time, there could be one or zero job arrivals. So in the corresponding balls-and-bins problem, there is certain probability that there are no balls at all. By properly choosing the observation rate, this extra randomness surprisingly makes the numbers of balls thrown in any $k^{(n)}$ bins (positively) associated, and further implies that the steady-state queues are associated. With this technique, we are able to prove the independence upper bound for any $k^{(n)} \leq n$ for the first time.

Our goal is to characterize the tail probability of the job delay distribution in steady state, since it is commonly used to quantify the quality of service. We study a system with n servers in which each job consists of $k^{(n)}$ tasks.

Our first result is that under the condition $k^{(n)} = o(n^{1/4})$, the queues at any $k^{(n)}$ servers are *asymptotically indepen-*

dent in steady state as $n \to \infty$, and thus the delays of a job's tasks are also asymptotically independent. It then follows that the job delay converges to the job delay given by the independence assumption. This result is established in Theorem 1 for generally distributed service times, and some explicit forms are given in Corollary 4.1 for exponentially distributed service times. This is the first asymptotically tight characterization of job delay in the limited fork-join model.

Our next result is that for *any* n and *any* $k^{(n)}$ with $k^{(n)} \leq n$, the job delay is stochastically upper bounded by the job delay given by the independence assumption. We refer to this upper bound as the *independence upper bound*. It is a new upper bound on job delay that is tighter than existing upper bounds. The technique we develop for the proof, named "Poisson oversampling", may be of independent interest for other related problems.

In this short version of our paper, we will state our results with no proofs. We refer the readers to a full version of this paper to [28] for proofs as well as more discussions and simulations.

2. RELATED WORK

In this section we discuss prior work on the limited fork-join model and some other related models. Prior work on the limited fork-join model [20, 14] has focused on the non-asymptotic regime and derived bounds on job delay. However, the bounds in [20, 14] do not have tightness guarantees. In particular, the upper bounds there are generally looser than the independence upper bound. Furthermore, none of the prior work has studied the asymptotic regime of the limited fork-join model. Below we give detailed discussions.

Limited fork-join model. Rizk et al. [20] give upper bounds on the tail probabilities of job delay in various settings. For Poisson arrivals and exponentially distributed service times, their upper bound is looser than the independence upper bound. For general service time distributions, their upper bound needs to be computed by numerically solving a non-linear equation. In contrast, we show that the independence upper bound holds and we also further establish asymptotic tightness of the independence upper bound.

Lee et al. [14] give upper and lower bounds on the *mean* job delay, not on the tail probabilities, assuming that service times follow an *exponential* distribution. Their upper bound is in general looser than the expectation of the independence upper bound, although the difference disappears as $n \to \infty$ when $k^{(n)} = o(n)$. Compared to this, we prove that the independence upper bound is indeed an upper bound for *any* n and $k^{(n)}$ with $k^{(n)} \leq n$. Besides, we prove it for very general service time distributions and in a stochastic dominance sense, which is stronger than the expectation sense. Also, there is a gap between their upper and lower bounds and there is no tightness analysis. Again, we establish asymptotic tightness of the independence upper bound.

There has also been work on variants of the limited fork-join model where each job consists of a random number of tasks. For example, Shah et al. [22] simply *assume* that the number of tasks in each job has a distribution such that the numbers of tasks sent to different queues are associated, thus obtaining the independence upper bound for their model. They further investigate different policies for assigning the tasks of a newly arrived job to servers, and show that the job delay under the two studied policies is shorter (in a proper sense) than the job delay under the random assignment in the limited fork-join model. Nelson et al. [19] consider a model where tasks wait in a central queue until some server becomes available. They show that the mean job delay is given by a set of recurrence equations, but no analytical form is derived. Kumar and Shorey [13] obtain upper and lower bounds on the mean delay when tasks are assigned to servers independently. But still, there are gaps between the upper and lower bounds.

Classic fork-join model. The classic fork-join model, where the number of tasks in a job is *equal* to the number of servers, n, has been widely studied in the literature. Similar to the limited fork-join model, tight characterizations of job delay are generally unknown except when $n = 2$. See [25] for a detailed survey. Here we just sample several most relevant papers. For general n, it has been proven that the mean delay of a job scales as $\Theta(\ln(n))$ as $n \to \infty$ under proper assumptions [18, 2]. Besides studying the limited fork-join model, Rizk et al. [20] also derive an upper bound on the tail distribution of the job delay for the classic fork-join model. Again, the tightness of the bound is not addressed.

MapReduce. Modeling MapReduce systems is challenging since the systems have many complex characteristics such as parallel servers, data locality, communication networks, etc. Most theoretical work on MapReduce does not provide analytical form bounds on the job delay. Papers such as [17, 33] and [23] design scheduling algorithms such that the job delay is guaranteed to be within a constant factor of the optimal, but do not provide analytical bounds. Tan et al. [24] quantifies the distribution tail of job delay when the map phase is abstracted as a single-server queue, resulting in a system with much higher efficiency, especially when the number of tasks in a job is large.

Asymptotic task delay. One component of the job delay in MapReduce is the task delay. Wang et al. [27] and Xie and Lu [30] bound the mean *task delay*, taking into consideration data locality; however they do not deal with the *job delay*. Bounding *job delay* would require characterizations of the correlation among queues. Ying et al. [32] study the task delay in a model where a load-balancing policy called batch-filling is used. They establish asymptotic independence for a *constant* number of queues, which is insufficient for models with jobs with a growing number of tasks.

3. MODEL AND NOTATION

Basic Notation. The symbols $\mathbb{R}_+$ and $\mathbb{Z}_+$ denote the set of nonnegative real numbers and nonnegative integers, respectively. We denote random variables by capital letters and vectors by bold letters. When a Markov chain $(\boldsymbol{X}(t), t \geq 0)$ has a unique stationary distribution, we denote by $\boldsymbol{X}(\infty)$ a random element whose distribution is the stationary distribution.

We denote by $\Rightarrow$ convergence in distribution (weak convergence) for random elements. We denote by $d_{TV}(\pi_1, \pi_2)$ the total variation distance between two probability measures π_1 and π_2 on a sigma-algebra σ of some sample space, i.e.,

$$d_{TV}(\pi_1, \pi_2) = \sup_{\mathcal{S} \in \sigma} |\pi_1(\mathcal{S}) - \pi_2(\mathcal{S})|. \tag{1}$$

Limited fork-join model. Our notation is summarized in Table 1. Recall that we consider a system with n servers, each with its own FIFO queue. We append the superscript

n	number of servers
superscript $^{(n)}$	quantities in the n-server system
$k^{(n)}$	number of tasks in a job
$\Lambda^{(n)}$	job arrival rate
λ	task arrival rate to each queue
$1/\mu$	mean of service time
ρ	load at each queue
$W_i^{(n)}(t)$	workload of server i's queue at time t
$T^{(n)}$	job delay
$\hat{T}^{(n)}$	job delay given by independent task delays
H_m	m-th harmonic number: $H_m = \sum_{j=1}^{m} \frac{1}{j}$

Table 1: Notation Table

$^{(n)}$ to related quantities to indicate that they are for the n-server system. We say that a quantity is a *constant* if it does not scale with n.

Jobs and tasks. Jobs arrive over time according to a Poisson process with rate $\Lambda^{(n)}$, and each job consists of $k^{(n)}$ tasks with $k^{(n)} \leq n$. Upon arrival, each job picks $k^{(n)}$ *distinct* servers uniformly at random from the n servers and sends one task to each server. We assume that $\Lambda^{(n)} = n\lambda/k^{(n)}$ for a *constant* λ, where the constant λ is the task arrival rate to each individual queue. Since different jobs choose servers independently, the task arrival process to each queue is also a Poisson process, and the rate is λ. The service times of tasks are i.i.d. following a cdf G with expectation $1/\mu$ and a finite second moment. We think of the service time of each task as being generated upon arrival: each task brings a required service time with it, but the length of the required service time is revealed to the system only when the task is completed. The load of each queue, $\rho = \lambda/\mu$, is then a constant and we assume that $\rho < 1$.

Queueing dynamics. It is not hard to see that each queue is an M/G/1 queue. But the queues are not independent in general since $k^{(n)}$ tasks arrive to the system at the same time. Let $W_i^{(n)}(t)$ denote the *workload* of server i's queue at time t, i.e., the total remaining service time of all the tasks in the queue, including the partially served task in service. So the workload of a queue is the waiting time of an incoming task to the queue before the server starts serving it. Let $\boldsymbol{W}^{(n)}(t) = \big(W_1^{(n)}(t), W_2^{(n)}(t), \ldots, W_n^{(n)}(t)\big)$. Then the workload process, $(\boldsymbol{W}^{(n)}(t), t \geq 0)$, is Markovian and ergodic. The ergodicity can be proven using the rather standard Foster-Lyapunov criteria [16], so we omit it here. Therefore, the workload process has a unique stationary distribution and $\boldsymbol{W}^{(n)}(t) \Rightarrow \boldsymbol{W}^{(n)}(\infty)$ as $t \to \infty$.

Job delay. We are interested in the distribution of job delay in steady state, i.e., the delay a job would experience if it arrives to the system and finds the system in steady state. Let a random variable $T^{(n)}$ represent this steady-state job delay. Specifically, the distribution of $T^{(n)}$ is determined by the workload $\boldsymbol{W}^{(n)}(\infty)$ in the following way. When a job comes into the system, its tasks are sent to $k^{(n)}$ queues and experience the delays in these queues. Since the queueing processes are symmetric over the indices of queues, without loss of generality, we can assume that the tasks are sent to the first $k^{(n)}$ queues for the purpose of computing the distribution of $T^{(n)}$. The delay of a task is the sum of its waiting time and service time. So the task delay in queue i, denoted by $T_i^{(n)}$, can be written as $T_i^{(n)} = W_i^{(n)}(\infty) + X_i$ with X_i being the service time. Recall that the X_i's are i.i.d.$\sim G$ and independent of everything else. Since the job is completed only when all its tasks are completed,

$$T^{(n)} = \max\left\{T_1^{(n)}, T_2^{(n)}, \ldots, T_{k^{(n)}}^{(n)}\right\}. \tag{2}$$

We will study the relation between $T^{(n)}$ and $\hat{T}^{(n)}$ with $\hat{T}^{(n)}$ defined as the job delay given by *independent* task delays. Specifically, $\hat{T}^{(n)}$ can be expressed as:

$$\hat{T}^{(n)} = \max\left\{\hat{T}_1^{(n)}, \hat{T}_2^{(n)}, \ldots, \hat{T}_{k^{(n)}}^{(n)}\right\}, \tag{3}$$

where $\hat{T}_1^{(n)}, \hat{T}_2^{(n)}, \ldots, \hat{T}_{k^{(n)}}^{(n)}$ are i.i.d. and each $\hat{T}_i^{(n)}$ has the same distribution as $T_i^{(n)}$. Again, due to symmetry, all the $T_i^{(n)}$'s have the same distribution. Let F denote the cdf of $T_i^{(n)}$, whose form is known from the queueing theory literature. Then, we have the following explicit form for $\hat{T}^{(n)}$:

$$\mathbb{P}\left(\hat{T}^{(n)} \leq \tau\right) = \left(F(\tau)\right)^{k^{(n)}}, \quad \tau \geq 0. \tag{4}$$

4. MAIN RESULTS

In Theorem 1, we establish asymptotic independence of any $k^{(n)}$ queues under the condition $k^{(n)} = o(n^{1/4})$ as the number of servers $n \to \infty$. The asymptotic independence is in the sense that the total variation distance between the distribution of the workloads of these queues and the distribution of $k^{(n)}$ independent queues goes to 0 as $n \to \infty$. Consequently, the distance between the distribution of job delay, $T^{(n)}$, and the distribution of the job delay given by independent task delays, $\hat{T}^{(n)}$, goes to 0. This result indicates that assuming independence among the delays of a job's tasks gives a good approximation of job delay when the system is large. Again, due to symmetry, we can focus on the first $k^{(n)}$ queues without loss of generality.

Theorem 1 *Consider an n-server system in the limited fork-join model with $k^{(n)} = o(n^{1/4})$. Let $\pi^{(n,k^{(n)})}$ denote the joint distribution of the steady-state workloads $W_1^{(n)}(\infty)$, $W_2^{(n)}(\infty), \ldots, W_{k^{(n)}}^{(n)}(\infty)$, and $\hat{\pi}^{(k^{(n)})}$ denote the product distribution of $k^{(n)}$ i.i.d. random variables, each of which follows a distribution that is the same as the distribution of $W_1^{(n)}(\infty)$. Then*

$$\lim_{n\to\infty} d_{TV}\left(\pi^{(n,k^{(n)})}, \hat{\pi}^{(k^{(n)})}\right) = 0. \tag{5}$$

Consequently, the steady-state job delay, $T^{(n)}$, and the job delay given by independent task delays as defined in (3), $\hat{T}^{(n)}$, satisfy

$$\lim_{n\to\infty} \sup_{\tau\geq 0} \left|\mathbb{P}\big(T^{(n)} \leq \tau\big) - \mathbb{P}\big(\hat{T}^{(n)} \leq \tau\big)\right| = 0. \tag{6}$$

For the special case where the service times are exponentially distributed, the job delay asymptotics have explicit forms presented in Corollary 4.1 below.

Corollary 4.1 *Consider an n-server system in the limited fork-join model with $k^{(n)} = o(n^{1/4})$, job arrival rate $\Lambda^{(n)} = n\lambda/k^{(n)}$, and exponentially distributed service times with mean $1/\mu$. Then the steady-state job delay, $T^{(n)}$, converges as:*

$$\lim_{n\to\infty} \sup_{\tau\geq 0} \left| \mathbb{P}\big(T^{(n)} \leq \tau\big) - \Big(1 - e^{-(\mu-\lambda)\tau}\Big)^{k^{(n)}} \right| = 0, \qquad (7)$$

Specifically, if $k^{(n)} \to \infty$ as $n \to \infty$, then

$$\frac{T^{(n)}}{H_{k^{(n)}}/(\mu-\lambda)} \Rightarrow 1, \quad \text{as } n \to \infty, \qquad (8)$$

where $H_{k^{(n)}}$ is the $k^{(n)}$-th harmonic number, and further,

$$\lim_{n\to\infty} \frac{\mathbb{E}\big[T^{(n)}\big]}{H_{k^{(n)}}/(\mu-\lambda)} = 1. \qquad (9)$$

The results above characterize job delay in the asymptotic regime where n goes to infinity. In Theorem 2 below, we study the non-asymptotic regime for any n and any $k^{(n)}$ with $k^{(n)} = k \leq n$, and we establish the independence upper bound on job delay.

Theorem 2 *Consider an n-server system in the limited fork-join model with $k^{(n)} = k \leq n$. Then the steady-state job delay, $T^{(n)}$, is stochastically upper bounded by the job delay given by independent task delays as defined in (3), $\hat{T}^{(n)}$, i.e.,*

$$T^{(n)} \leq_{st} \hat{T}^{(n)}, \qquad (10)$$

where "$\leq_{st}$" denotes stochastic dominance. Specifically, for any $\tau \geq 0$,

$$\mathbb{P}\big(T^{(n)} > \tau\big) \leq \mathbb{P}\big(\hat{T}^{(n)} > \tau\big) = 1 - \big(F(\tau)\big)^{k^{(n)}}. \qquad (11)$$

Proofs of these theorems appear in [28].

5. CONCLUSIONS

We study the limited fork-join model where there are n servers in the system and each job consists of $k^{(n)} \leq n$ tasks that are sent to $k^{(n)}$ distinct servers chosen uniformly at random. A job is considered complete only when all its tasks complete processing. We characterize the delay of jobs both in an asymptotic regime where $n \to \infty$ and in the non-asymptotic regime for any n and any $k^{(n)} = k$.

For the asymptotic regime, we show that under the condition $k^{(n)} = o(n^{1/4})$, the workloads of any $k^{(n)}$ queues in the n-server system are asymptotically independent, and the delay of a job therefore converges to the maximum of independent task delays. For the non-asymptotic regime, we show that the steady-state workloads of any $k^{(n)}$ queues are associated, and therefore assuming independent task delays yields an upper bound on the job delay. Our results provide the first tight characterization of job delay in the limited fork-join model, and the upper bound is tighter than other existing upper bounds.

From a technical perspective, we make the following two contributions: (1) Our asymptotic results open up new regimes for asymptotic independence: $k^{(n)}$ queues are shown to be asymptotically independent, where $k^{(n)}$ is allowed to grow with n instead of being a constant, as was previously studied. (2) We develop new proof techniques to establish association in steady state. We believe that the results and techniques in this paper will shed light on related problems such as order statistics in coded data storage systems, job redundancy, load-balancing algorithms.

6. ACKNOWLEDGMENT

This work was supported in part by NSF Grants CPS ECCS-1739189, ECCS 1609370, XPS-1629444, and CMMI-1538204, the U.S. Army Research Office (ARO Grant No. W911NF-16-1-0259), the U.S. Office of Naval Research (ONR Grant No. N00014-15-1-2169), DTRA under the grant number HDTRA1-16-0017, and a 2018 Faculty Award from Microsoft. Additionally, Haotian Jiang was supported in part by the Department of Physics at Tsinghua University.

References

F. Baccelli. Two parallel queues created by arrivals with two demands:the M/G/2 symmetrical case. Technical Report RR-0426, INRIA, July 1985.

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.*, 21:629–660, 1989.

M. Bramson, Y. Lu, and B. Prabhakar. Asymptotic independence of queues under randomized load balancing. *Queueing Syst.*, 71(3):247–292, July 2012.

Y. Chen, S. Alspaugh, and R. Katz. Interactive analytical processing in big data systems: A cross-industry study of MapReduce workloads. *Proc. VLDB Endow.*, 5(12):1802–1813, Aug. 2012. ISSN 2150-8097.

J. Dean and S. Ghemawat. MapReduce: Simplified data processing on large clusters. In *Proc. USENIX Conf. Operating Systems Design and Implementation (OSDI)*, pages 10–10, San Francisco, CA, 2004.

J. D. Esary, F. Proschan, and D. W. Walkup. Association of random variables, with applications. *Ann. Math. Statist.*, 38(5):1466–1474, 10 1967.

L. Flatto and S. Hahn. Two parallel queues created by arrivals with two demands I. *SIAM J. Appl. Math.*, 44(5): 1041–1053, 1984.

K. Gardner, M. Harchol-Balter, and A. Scheller-Wolf. A better model for job redundancy: Decoupling server slowdown and job size. In *IEEE Int. Symp. Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 1–10, London, United Kingdom, Sept. 2016.

K. Gardner, M. Harchol-Balter, A. Scheller-Wolf, and B. Van Houdt. A better model for job redundancy: Decoupling server slowdown and job size. *IEEE/ACM Trans. Netw.*, 25(6):3353–3367, Dec. 2017.

K. Gardner, M. Harchol-Balter, A. Scheller-Wolf, M. Velednitsky, and S. Zbarsky. Redundancy-d: The power of d choices for redundancy. *Oper. Res.*, 65(4):1078–1094, 2017.

K. Joag-Dev and F. Proschan. Negative association of random variables with applications. *Ann. Statist.*, 11(1):286–295, Mar. 1983.

G. Joshi, Y. Liu, and E. Soljanin. Coding for fast content download. In *Proc. Ann. Allerton Conf. Communication, Control and Computing*, pages 326–333, Monticello, IL, Oct. 2012.

A. Kumar and R. Shorey. Performance analysis and scheduling of stochastic fork-join jobs in a multicomputer system. *IEEE Trans. Parallel Distrib. Syst.*, 4(10):1147–1164, Oct. 1993.

K. Lee, N. B. Shah, L. Huang, and K. Ramchandran. The MDS queue: Analysing the latency performance of erasure codes. *IEEE Trans. Inf. Theory*, 63(5):2822–2842, May 2017.

B. Li, A. Ramamoorthy, and R. Srikant. Mean-field-analysis of coding versus replication in cloud storage systems. In *Proc. IEEE Int. Conf. Computer Communications (INFOCOM)*, pages 1–9, San Francisco, CA, Apr. 2016.

S. P. Meyn and R. L. Tweedie. Stability of Markovian processes III: Foster-Lyapunov criteria for continuous-time processes. *Adv. Appl. Probab.*, 25(3):518–548, 1993.

B. Moseley, A. Dasgupta, R. Kumar, and T. Sarlós. On scheduling in Map-Reduce and flow-shops. In *Proc. Ann. ACM Symp. Parallelism in Algorithms and Architectures (SPAA)*, pages 289–298, San Jose, CA, 2011.

R. Nelson and A. N. Tantawi. Approximate analysis of fork/join synchronization in parallel queues. *IEEE Trans. Comput.*, 37(6):739–743, June 1988.

R. Nelson, D. Towsley, and A. N. Tantawi. Performance analysis of parallel processing systems. *IEEE Trans. Softw. Eng.*, 14(4):532–540, Apr. 1988.

A. Rizk, F. Poloczek, and F. Ciucu. Stochastic bounds in fork–join queueing systems under full and partial mapping. *Queueing Syst.*, 83(3):261–291, Aug. 2016.

N. B. Shah, K. Lee, and K. Ramchandran. When do redundant requests reduce latency ? In *Proc. Ann. Allerton Conf. Communication, Control and Computing*, pages 731–738, Monticello, IL, Oct. 2013.

V. Shah, A. Bouillard, and F. Baccelli. Delay comparison of delivery and coding policies in data clusters. In *Proc. Ann. Allerton Conf. Communication, Control and Computing*, pages 397–404, Monticello, IL, Oct. 2017.

Y. Sun, C. E. Koksal, and N. B. Shroff. Near delay-optimal scheduling of batch jobs in multi-server systems. Technical report, The Ohio State University, 2017.

J. Tan, X. Meng, and L. Zhang. Delay tails in MapReduce scheduling. In *Proc. ACM SIGMETRICS/PERFORMANCE Jt. Int. Conf. Measurement and Modeling of Computer Systems*, pages 5–16, London, United Kingdom, 2012.

A. Thomasian. Analysis of fork/join and related queueing systems. *ACM Comput. Surv.*, 47(2):17:1–17:71, Aug. 2014.

A. Vulimiri, O. Michel, P. B. Godfrey, and S. Shenker. More is less: Reducing latency via redundancy. In *Proc. ACM Workshop Hot Topics in Networks (HotNets)*, pages 13–18, Redmond, WA, Oct. 2012.

W. Wang, K. Zhu, L. Ying, J. Tan, and L. Zhang. MapTask scheduling in MapReduce with data locality: Throughput and heavy-traffic optimality. *IEEE/ACM Trans. Netw.*, 24: 190–203, Feb. 2016.

W. Wang, M. Harchol-Balter, H. Jiang, A. Scheller-Wolf, and R. Srikant. Delay asymptotics and bounds for multi-task parallel jobs. *arXiv:1710.00296 [cs.PF]*, 2018.

Y. Xiang, T. Lan, V. Aggarwal, and Y.-F. R. Chen. Joint latency and cost optimization for erasure-coded data center storage. *IEEE/ACM Trans. Netw.*, 24(4):2443–2457, Aug. 2016.

Q. Xie and Y. Lu. Priority algorithm for near-data scheduling: Throughput and heavy-traffic optimality. In *Proc. IEEE Int. Conf. Computer Communications (INFOCOM)*, pages 963–972, Hong Kong, China, Apr. 2015.

Q. Xie, X. Dong, Y. Lu, and R. Srikant. Power of d choices for large-scale bin packing: A loss model. In *Proc. ACM SIGMETRICS Int. Conf. Measurement and Modeling of Computer Systems*, pages 321–334, Portland, OR, 2015.

L. Ying, R. Srikant, and X. Kang. The power of slightly more than one sample in randomized load balancing. In *Proc. IEEE Int. Conf. Computer Communications (INFOCOM)*, pages 1131–1139, Kowloon, Hong Kong, Apr. 2015.

Y. Zheng, N. B. Shroff, and P. Sinha. A new analytical technique for designing provably efficient MapReduce schedulers. In *Proc. IEEE Int. Conf. Computer Communications (INFOCOM)*, pages 1600–1608, Turin, Italy, Apr. 2013.