Analysis of Fork/Join and Related Queueing Systems

ALEXANDER THOMASIAN, Thomasian and Associates

Fork/join (F/J) requests arise in contexts such as parallel computing, query processing in parallel databases, and parallel disk access in RAID. F/J requests spawn K tasks that are sent to K parallel servers, and the completion of all K tasks marks the completion of an F/J request. The exact formula for the mean response time of $K = 2$-way F/J requests derived under Markovian assumptions ($R_2^{F/J}$) served as the starting point for an approximate expression for $R_K^{F/J}$ for $2 < K \leq 32$. When servers process independent requests in addition to F/J requests, the mean response time of F/J requests is better approximated by R_K^{max}, which is the maximum of the response times of tasks constituting F/J requests. R_K^{max} is easier to compute and serves as an upper bound to $R_K^{F/J}$. We discuss techniques to compute R_K^{max} and generally the maximum of K random variables denoting the processing times of the tasks of a parallel computation $\overline{X}_K^{max}$. Graph models of computations such as Petri nets—a more general form of parallelism than F/J requests—are also discussed in this work. Jobs with precedence constraints may require multiple resources, which are represented by a queueing network model. We also discuss various queueing systems related to F/J queueing systems and outline their analysis.

Categories and Subject Descriptors: C.4 [**Performance of Systems**]: Modeling Techniques; D.2.2 [**Design Tools and Techniques**]: Petri Nets; D.2.8 [**Metrics**]: Performance Measures; D.4.8 [**Performance**]: Queueing Theory, Stochastic Systems, Simulation

General Terms: Operating Systems, Synchronization

Additional Key Words and Phrases: Fork-join queueing system, split-merge queueing system, Markov chain model, matrix geometric method, queueing network model, vacationing server model, linear programming, order statistics, characteristic maximum, mirrored disk, RAID5 disk array, task systems, stream processing, MapReduce, serialization delay, lock contention, replicated databases, bulk arrivals, parallel processing, team service model, independent server model

ACM Reference Format:
Alexander Thomasian. 2014. Analysis of fork-join and related queueing systems. ACM Comput. Surv. 47, 2, Article 17 (July 2014), 71 pages.
DOI: http://dx.doi.org/10.1145/2628913

1. INTRODUCTION

Fork/join (F/J) requests spawn K independent tasks on arrival at the fork point, which are sent immediately to K parallel servers as show in Figure 1. The k^{th} task is send to the respective server, where it is served in first-come, first served (FCFS) order. Completed tasks are held in synchronization queues awaiting the completion of their siblings. After the last completed task reaches the synchronization queue, the tasks of F/J requests resynchronize at the join point, at which point the F/J request is considered completed.

Author's address: A. Thomasian, Thomasian and Associates, 17 Meadowbrook Road, Pleasantville, NY 10570; email: alexthomasian@gmail.com.
Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.
© 2014 ACM 0360-0300/2014/07-ART17 $15.00
DOI: http://dx.doi.org/10.1145/2628913

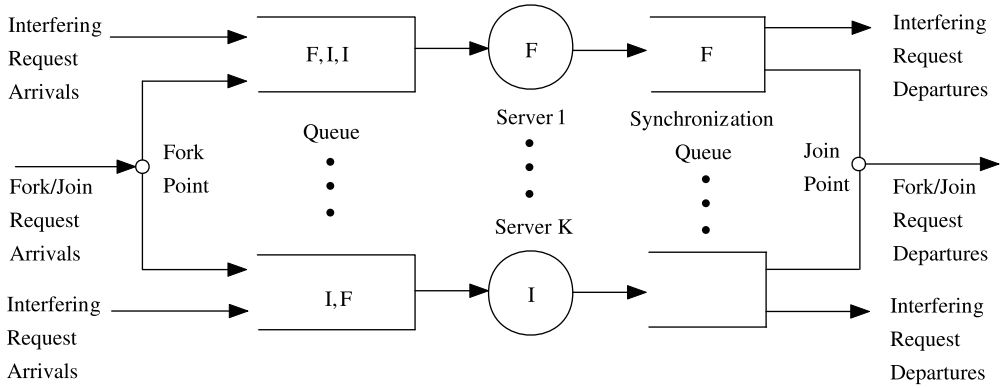

Fig. 1. Processing of F/J requests (denoted by F) and interfering requests (denoted by I). F/J requests are enqueued in synchronization queues until all of their siblings are completed.

The analysis of the splitting and matching (SPM) queueing system in Mandelbaum and Itzhak [1968] is perhaps the earliest work on this topic. It considers airline passengers and their luggage arriving together by airplane, but separated until they come together in the luggage claim area. The two-way system analyzed in the paper considers D/M/1 queues with deterministic (D) arrivals and mutually independent exponential service times (M) at single servers, according to the Kendall notation [Kleinrock 1975; Allen 1990].

Under favorable modeling assumptions, the task response times at individual servers can be estimated by using well-known results in queueing theory [Kleinrock 1975, 1976; Kobayashi and Mark 2008], but the synchronization delay is much more difficult to compute. There is no synchronization delay when the service times of an F/J request are equal at the K servers, but service times may differ from one F/J request to another. Synchronous disk interleaving described in Kim [1986] places consecutive blocks of a file at the same position at K disks and synchronizes disk rotations so that the same positioning time is incurred at all disks to access them as a means of increasing the disk transfer rate K-fold (see Appendix I). There are K identical M/G/1 queueing systems, whose mean response time can be obtained using Eq. (75) in Appendix II.

Split-merge (SM) is different from the F/J queueing system in that all of the tasks of an SM request need to complete service before the next SM request can be issued [Duda and Czachrski 1987]. There is a synchronization delay after the completion of the first task until all tasks are completed and before the next SM request can be issued. The SM paradigm is followed by asynchronous disk interleaving, where disk service times vary from disk to disk [Kim and Tantawi 1991]. The system can be analyzed by computing the expected value of the maximum of K random variables, which is denoted by $\overline{X}_K^{max}$ leads to a maximum throughput $\lambda_K^{SM} = (\overline{X}_K^{max})^{-1}$. In the case of an exponential service times with rate μ, $\overline{X}_K^{max} = H_K/\mu$, where $H_K = \sum_{k=1}^{K} 1/k$ is the Harmonic sum. Section 4.6.2 in Trivedi [2002] provides a concise proof that Eq. (20) simplifies to H_K/μ in the case of the exponential distribution. For large K, $H_K \approx \ln(K) + \gamma$, where $\ln$ is the natural logarithm and $\gamma \approx 0.57721$ is the Euler-Mascheroni constant [Havil 2003]. The maximum throughput of a K-way F/J system is independent of K and is given as $\lambda_K^{F/J} = \mu$, which exceeds λ_K^{SM} by a factor H_K.

The F/J queueing system is inefficient in that a task may be enqueued at a server, whereas other servers are idle. This is because each task needs to be processed at a specific server, as in the case of F/J requests issued to surviving disks in a redundant array of independent disks level 5 (RAID5) disk array to reconstruct blocks on a failed

disk [Chen et al. 1994; Thomasian and Blaum 2009] (see Appendix I). The server idleness problem is alleviated by centralized scheduling [Nelson et al. 1988], where tasks are held in a single queue as discussed in Section 6.1. In the case of the team service model (TSM), jobs require multiple servers for their execution, and the servers are acquired and released together [Omahen and Schrage 1972; Thomasian and Avizienis 1975]. The independent server model (ISM) is similar to TSM in that jobs require a variable number of servers simultaneously for their tasks, but the tasks can complete asynchronously [Green 1980]. The TSM and ISM request a variable number of servers, which do not exceed the number of available servers and do not make a distinction among servers. In transaction processing systems with predeclared locks, all requested locks should be acquired before a transaction starts execution so that there are delays in activating transactions when lock conflicts are encountered [Thomasian 1985].

Synchronization delays arise in the context of updating mirrored disks [Towsley et al. 1990b] as well as replicated databases [Nelson and Iyer 1985]. In the case of readers and writers paradigm, readers can be processed concurrently, whereas writers are processed singly. The threshold fastest emptying (TFE) scheduling policy outperforms FCFS scheduling in this case [Thomasian and Nicola 1993].

F/J queueing has been analyzed in the context of queueing network models in Heidelberger and Trivedi [1982, 1983], Baynat and Dallery [1983], and Varki and Dowdy [1996]. We make the point that $R_K^{F/J}$ response time in QNMs is affected by the probability distributions of task transitions as part of their execution in the QNM rather than just the mean number of visits to each QNM node (see Appendix III). General synchronization structures are considered in Thomasian and Bay [1986] and Menasce et al. [1991].

Serialization delays occur when jobs executing concurrently in a computer system are delayed in attempting to enter certain phases of their execution—that is, invoking a serially re-entrant procedure [Agrawal and Buzen 1983; Jacobson and Lazowska 1983; Thomasian 1983; Balbo et al. 1986]. These may be considered inverses of F/J parallelism, where the execution of independent jobs is throttled at certain points.

In RAID5 disk arrays, F/J requests arise in reconstructing missing data on a failed disk [Chen et al. 1994; Thomasian and Menon 1994; Tomasian and Blaum 2009] as well as parallel data access [Chen and Towsley 1993] (refer to Appendix I). In addition to F/J requests, surviving disks process independent disk accesses that interfere with the processing of F/J tasks as shown in Figure 1. F/J requests also arise when a logical disk request encompasses blocks on several disks. Updating small blocks in RAID5 requires reading the old data and parity blocks and updating them both, hence two consecutive two-way F/J requests and four disk accesses, known as the small write penalty (SWP) [Chen et al. 1994]. Updating data in mirrored disks requires a two-way F/J request. Estimating the mean F/J response time in RAID ($R_K^{F/J}$) is posed as a challenging problem in Chen et al. [1994], with a brief comment in Thomasian [1995]. In Appendix I, we discuss data layouts to reduce the cost of processing F/J requests in RAID5 disk arrays.

Web search results in F/J requests with a high degree of parallelism to attain a lower response time. Google's search engine returns thousands of answers in less than a second, based on the completion of all pertinent searches, and the results are returned in the order of diminished ranking. The analysis of a parallel search that concludes with the first result in Ghodsi and Kant [1991] is described in Section 6.1.

Parallel algorithms consist of multiple iterations, where the $(i + 1)^{st}$ iteration is started when the i^{th} iteration is completed [Sun and Peterson 2012]. Issuing tasks in parallel or gang scheduling is tantamount to a fork, and barrier synchronization to await the completion of all tasks is a join. Bounds on speedup and efficiency are obtained in Chang and Nelson [1995], when partial synchronization is required. The analysis is complicated, as some iterations start while others are still in progress.

Long tails in parallel processing are encountered in conjunction with the MapReduce programming framework developed at Google and its Hadoop open source implementation [Dean and Ghemawat 2010].

The stream processing paradigm fits a large class of applications [Cherniack et al. 2003] for which conventional database management systems (DBMSs) [Ramakrishnan and Gehrke 2003] fall short, although a streaming SQL language is under consideration. Distributed streams processing systems (DSPSs) are based on wireless sensor networks, which provide scalable load management and higher availability than centralized systems. In Section 7.6, we discuss F/J structures arising in the context of DSPSs based on Feng et al. [2007], Xia et al. [2007], and Zhao et al. [2010a, 2010b]. F/J queueing systems arise in other contexts, such as workflow systems [Rahman et al. 2010]; peer-to-peer systems [Schuler et al. 2004]; and complex event processing (CEP), a technology to predict high-level events resulting from specific sets of low-level events [Etzion and Niblett 2010].

Research on F/J queueing systems has been conducted in the context of computer science and engineering, operations research, and applied mathematics. Since only an outline of the analysis is provided for most papers, the readers are encouraged to read the original papers. This article should be of interest to students and researchers in this area. It can be used for supplementary reading in courses on performance analysis of computer systems based on queueing theory. Issues worthy of further research are presented throughout the article. Theoretical studies of F/J queueing systems are deemphasized in this work; three representative theoretical studies are listed in the References section [Baccelli and Makowski 1989; Baccelli et al. 1989; Baccelli and Liu 1990] but are not discussed further in the text.

1.1. Organization of the Article

The remainder of the article is organized as follows. In Section 2, classification of F/J and related queueing systems with references to previous work are presented. In Section 3, the analysis of F/J queueing systems is discussed. In Section 4, methods to estimate the expected value of the maximum for different distributions are explored. Sections 5 and 6 discuss replicated data and related queueing systems, respectively. In Section 7, the processing of F/J requests and jobs with precedence constraints in QNMs, serialization delays, and other forms of synchronization are presented. Section 8 concludes this article.

2. OVERVIEW: FORK/JOIN AND RELATED QUEUEING SYSTEMS

In addition to classifying F/J and related queueing systems, this section serves the role of reviewing previous work in more detail than in Section 1.

(1) *Single resource systems*: Most F/J studies consider the processing of tasks of F/J requests on single resource systems in the form of single-server queues. Multiserver queues are considered in Ko and Serfozo [2004].

(2) *Multiresource systems represented by QNMs*: The tasks of F/J requests are processed in the context of a QNM, which can be open, closed, and mixed or hybrid [Lavenberg and Sauer 1983a; Lazowska et al. 1984; Bolch et al. 2006; Kobayashi and Mark 2008] (see Appendix III).

(2.1) *Open QNMs*: Jobs that spawn F/J requests arrive according to a Poisson process, and their tasks leave the system after their processing is completed [Flatto and Hahn 1984; Nelson and Tantawi 1988; Thomasian and Tantawi 1994; Varki 2001].

(2.2) *Closed QNMs*: Closed QNMs process a fixed number of jobs exhibiting internal concurrency in the form of F/J requests [Heidelberger and Trivedi 1983; Liu and Perros 1991; Varki and Dowdy 1996; Varki 1999; Varki 2001; Casale et al. 2008; Varki et al.

2013]. A phase-type delay is considered as part of a closed QNM in Krishnamurthy et al. [2004].

(2.3) *Mixed/hybrid QNMs*: Tasks spawned by F/J requests in a closed QNM leave the system without requiring synchronization on completing their execution [Heidelberger and Trivedi 1982].

(2.4) *Single versus multiple job classes*: Internal parallelism in the form of asymmetric F/J requests with different processing requirements is considered in Baynat and Dallery [1993]. Multiple job classes are considered in the context of serialization delays [Thomasian and Nadji 1993] and in the context of task systems [Thomasian and Bay 1986].

(2.5) *Task systems*: A single instance of a complex task structure is considered in Thomasian and Bay [1986]. This work is extended in Menasce et al. [1995]. The execution of task systems in multiprocessors is studied in Kung [1984]. A tool for analyzing such QNMs is developed in Kapelnikov et al. [1989, 1992].

(3) *The arrival process*.

(3.1) Poisson arrivals or exponential interarrival times are the most commonly considered cases.

(3.2) Erlang and hyperexponential interarrival times [Varma and Makowski 1994].

(3.3) Phase-type interarrival times [Takahashi et al. 2000; Takahashi and Takahashi 2000].

(3.4) *Bulk arrivals* [Chaudhry and Templeton 1983; Nelson et al. 1988; Lebrecht et al. 2009]: The main difference with an F/J queueing system is that the tasks associated with an arrival can be served at any of the servers of a multiserver queueing system, and synchronization is not required.

(4) *Service time distributions*.

(4.1) *Exponential*: Combining with Poisson arrivals results in M/M/1 queues [Kleinrock 1975]. The exact formula for the 2-way F/J queueing system in Flatto and Hahn [1984] is extended to an approximate expression for $2 < K \geq 32$-way F/J queueing system in Nelson and Tantawi [1988]. An M/M/1 queueing system has been used in analyzing the performance of RAID5 arrays in normal, degraded, and rebuild modes in Menon [1994] (see Appendix I).

(4.2) *Erlang/hyperexponential*: This distribution is used as an approximation to disk service time in Kuratti and Sanders [1995] since the results in Kim and Agrawal [1985, 1989] can be used to estimate $R_k^{F/J}(\rho)$. Erlang and hyperexponential service times are considered in Varma and Makowski [1994].

(4.3) *General*: The sum of the components of disk service time specified in Appendix II has a general distribution. The M/G/1 queueing system with Poisson arrivals and general service times has been used in several studies of RAID5 performance [Chen and Towsley 1993; Thomasian and Menon 1994, 1997; Merchant and Yu 1997; Thomasian et al. 2007] (see Appendix II).

(4.4) Miscellaneous distributions, which have been used as approximations to general response time distributions for estimating the maximum response time ($R_K^{max}(\rho)$) in RAID5 disk arrays, are Coxian [Chen and Towsley 1993], Erlang [Thomasian and Menon 1994, 1997], extreme value [Merchant and Yu 1996; Thomasian et al. 2007], and normal [Kim and Tantawi 1991] to approximate disk service time in an SM queueing system.

(5) *Varieties of F/J systems*.

(5.1) *F/J and interfering requests*: A RAID5 disk array in degraded mode processes F/J requests on surviving disks to reconstruct blocks on the failed disk and interfering requests to access data on those disks directly [Thomasian and Menon 1994, 1997; Thomasian and Tantawi 1994; Thomasian et al. 2007].

(5.2) *FCFS scheduling with and without priorities*: F/J requests are processed in FCFS order at the same priority as interfering requests in Thomasian and Menon [1994, 1997], Thomasian and Tantawi [1994], and Thomasian [1997], whereas the processing of F/J requests may be prioritized providing lower response time for F/J requests used for materializing data blocks on failed disks. This effect is quantified by a numerical example in Appendix II.

(5.3) *Processor sharing (PS) scheduling policy*: This policy is considered at job and task level in Towsley et al. [1990a]. The latter does not allow parallel execution at job level, as task-level scheduling is more meaningful. The mean and the LST of the distribution conditioned on service time x for M/M/1 queues is given in Coffman et al. [1970], Kleinrock [1976], Harrison and Patel [1993], and Kobayashi and Mark [2008]. Note that PS outperforms FCFS scheduling when the coefficient of variation (CV) of service time exceeds unity (see Appendix II).

(5.4) *Symmetric versus F/J queueing systems*: Symmetric F/J queueing systems are considered in most studies of F/J queueing systems. Asymmetry may be due to (a) heterogeneous servers with different speeds or tasks with different service times [Balsamo et al. 1998], (b) unbalanced server utilizations from interfering requests with different intensities at the servers, (c) different service times required by the two branches of a two-way F/J read-modify-write (RMW) request [Thomasian and Menon 1994, 1997], (d) different processing requirements for the tasks spawned by an F/J request in a QNM [Baynat and Dallery 1993].

(5.5) Variable number of tasks per F/J request is considered in Varki et al. [2012], whereas most studies consider a fixed number of requests. A subset of surviving disks in clustered RAID5 disk is accessed according to the data layout specified in Appendix I.

(5.6) SM queueing systems process one F/J request at a time [Duda and Czachrski 1987; Kim and Tantawi 1991] so that the queueing is at the level of F/J requests rather than tasks enqueued at servers.

(6) *Varieties of F/J requests in RAID5*: These are described in Chen et al. [1994] and Thomasian and Blaum [2009], as well as Appendix I.

(6.1) *SWP*: Updating of a data block requires an F/J request to read the old data and parity blocks to compute the new parity, and another F/J request to overwrite them both with new values [Menon 1994; Chen et al. 1994; Thomasian et al. 2007; Thomasian and Blaum 2009].

(6.2) *Reconstruct writes*: Such updates are preferable when the majority of strips to be updated are cached. Check strips can be computed more efficiently by accessing the remaining data strips in the parity group via an F/J request [Thomasian and Blaum 2009].

(6.3) *Materializing logical block spanning multiple disks*: The strips constituting a large data block can be accessed in parallel, but this may be detrimental to performance at higher access rates [Chen and Towsley 1993].

(6.4) *Reconstructing blocks on failed disks*: This requires an F/J request to access the corresponding blocks in the parity group [Thomasian and Menon 1994, 1997; Merchant and Yu 1996; Thomasian et al. 2007].

(6.5) Synchronization delays due to *Concurrency Control (CC)* methods in databases in Section 6.5 [Thomasian 1998b, 2009].

(7) *Methods to estimate mean F/J response time and other metrics*.

(7.1) Simulation has been used in several studies to estimate the accuracy of F/J approximations [Thomasian and Menon 1994, 1997; Thomasian et al. 2007]. A simulation capability is implemented in packages for studying generalized stochastic Petri nets (GSPNs) [Sahner et al. 1996; Bolch et al. 2006; Dingle et al. 2009].

(7.2) *Continuous time Markov chain (CTMC) models*: The mean response time for two-way F/J queueing systems ($R_2^{F/J}$) was obtained in Nelson and Tantawi [1988] by

manipulating the z-transform of the corresponding state equilibrium equations, which are infinite in both dimensions [Flatto and Hahn 1984]. The matrix geometric method (MGM) has been used for analyzing F/J and related queueing systems in Gillent [1980], Rao and Posner [1985], Balsamo and Mura [1995, 1997], Balsamo et al. [1998], and Squillante et al. [1998].

(7.3) Empirical expressions for $R_K^{F/J}$ are obtained by matching the coefficients of an expression in ρ to simulation results [Nelson and Tantawi 1988; Thomasian and Tantawi 1994; Varma and Makowski 1994].

(7.4) Characteristic maximum defined in Blom [1958], Gumbel [1958], Gravey [1985], and Kruskal and Weiss [1985] is used in Chen and Towsley [1993], Kuratti and Sanders [1995], and Thomasian and Menon [1997] to estimate an approximation to R_K^{max}.

(8) *Performance metrics for F/J queueing systems.*

(8.1) *Mean F/J response time*: $R_K^{F/J}$, although R_K^{max} is reported in many studies instead without providing a justification.

(8.2) Moments of F/J queue lengths [Balsamo and Mura 1997].

(8.3) *Distribution of F/J response time* [Balsamo and Mura 1995]: Sojourn time distribution with GI/M/1 queues is obtained approximately in Ko and Serfozo [2008] using bounds, properties of D/M/1 and M/M/1 networks and exploratory simulations. An earlier study dealt with sojourn time distribution with multiserver queues [Ko and Serfozo 2004].

(8.4) *Various performance metrics* [Bolch et al 2006]. These following metrics can be normalized by dividing them by K. Given $\overline{R} = \sum_{k=1}^{K} R_k/K$, $G_K/K = \overline{R}/R_K^{F/J}$ is a measure of inefficiency due to the synchronization delay given by Equation (7).

(8.4.1) *Speedup*: $G_K = (K \times R)/R_K^{F/J}$ quantifies the effect of F/J parallelism, where R denotes the mean response time of individual tasks. $G_K = K$ if response times have fixed values, which would be the case with fixed service times when there is no queueing—for example, interarrival times are greater than or equal to service times. For unequal mean response times, $R_k, 1 \le k \le K$, $G_K = \sum_{k=1}^{K} R_k/R_K^{F/J}$. G_K is much smaller than K when tasks have highly skewed processing times, so the sum of response times is dominated by few of the longest tasks.

(8.4.2) *Synchronization overhead*: $S_K = \sum_{k=1}^{K} w_k/R_K^{F/J}$, the sum of waiting times (w_k) of tasks at synchronization queues divided by $R_K^{F/J}$.

(8.4.3) *Blocking factor*: $B_k = \sum_{k=1}^{K} Q_k$ is the sum of the mean number of tasks in synchronization queues:

(8.4.4) *Subtask dispersion*: This is the difference in time between the completion of the last and first subtask of an F/J request, where we have adopted the terminology in Tsimashenka and Knottenbelt [2013b]; thus, each task consists of K subtasks in a K-way F/J queueing system.

(8.4.5) *Transient solution*: A transient solution of response time in a two-server F/J queueing system with exponential, Erlang, and hyperexponential interarrival and service time distributions is provided in Kim and Agrawal [1989].

3. FORK/JOIN QUEUEING SYSTEM ANALYSES

Section 3.1 considers a K-way F/J queueing system with Poisson arrivals with rate λ and exponential service times with mean $\overline{x} = 1/\mu$ so that server utilizations are $\rho = \lambda\overline{x}$. An exact expression for $R_2^{F/J}(\rho)$ and an approximation for $R_K^{F/J}(\rho)$ for $2 < K \le 32$ is provided in Section 3.2. In Section 3.3, empirical formulas for K-way F/J response time with Poisson arrivals and general service times are presented. Methods to obtain the F/J response time with phase-type interarrival and service times are discussed in Section 3.4. The application of MGM to the analysis of F/J queueing systems is

discussed in Section 3.5. The processing of F/J requests with independent requests, which interfere with the processing of F/J requests, is discussed in Section 3.6.

3.1. Analyses of Markovian Queues

We consider a two-way F/J queueing system with arrival rate λ and mean service time $\overline{x} = 1/\mu$ at its two servers so that the utilization factor at each server is $\rho = \lambda\overline{x}$. A utilization factor $\rho < 1$ ensures that the queueing system is not saturated. Given the random variables of the response times at the two servers, $\tilde{R}_i(\rho), i = 1, 2$, with means $R_1(\rho)$, $R_2(\rho)$, the expected value of the maximum of the two response times is $R_2^{max}(\rho) = E[max\tilde{R}_1(\rho), \tilde{R}_2(\rho))]$; note that $R_2^{max}(\rho) \neq max[R_1(\rho), R_2(\rho)]$.

For M/M/1 queues, the response time distribution is exponential according to Eq. (80) in Appendix II with mean $R(\rho) = (\mu - \lambda)^{-1}$ is

$$F(t) = 1 - e^{-t/R(\rho)} = 1 - e^{-(\mu-\lambda)t},$$

so the maximum of K response times is [Allen 1990; Trivedi 2002]

$$R_K^{max}(\rho) = H_K R(\rho) = \left[\sum_{k=1}^{K} \frac{1}{k}\right] R(\rho).$$

For $K' > K$,

$$R_{K'}^{max}(\rho) - R_K^{max}(\rho) = \left[\sum_{k=K+1}^{K'} \frac{1}{k}\right] R(\rho).$$

Note the insignificant increase in R_{K+1}^{max} with respect to R_K^{max} for larger values of K.

For $K = 2$, the mean F/J response time $R_2^{F/J}(\rho) = (H_2 - \rho/8)R(\rho)$, which is also given later by Eq. (6), is less than $R_2^{max}(\rho)$, and the difference increases with ρ since $R(\rho)$ is also an increasing function of ρ:

$$R_2^{max}(\rho) - R_2^{F/J}(\rho) = \frac{\rho}{8} R(\rho).$$

An argument based on *associated* random variables is used in Nelson and Tantawi [1988] to show that $R_K^{max}(\rho)$ is an upper bound to $R_K^{F/J}(\rho)$. Two real-valued r.v.'s S and T are associated if $E[f(S)g(T)] \geq E[f(S)]E[g(T)]$, where f and g are nondecreasing functions [Barlow and Proschan 1975].

An approximation for $R_K^{F/J}(\rho), 2 \leq K \leq 32$ is developed in Nelson and Tantawi [1988] by utilizing Eq. (6) and a scaling approximation that the upper and lower bounds for F/J response time increase at the same rate: $H_K/\mu \leq R_K^{F/J}(\rho) \leq H_K R(\rho)$. The expansion in response time with respect to $R_2^{F/J}(\rho)$ is expressed as $R_K^{F/J}(\rho) = S_K(\rho)R_2^{F/J}(\rho)$, where $S_K(\rho) = \alpha(\rho) + \beta(\rho)H_K$. It follows from $S_2(\rho) = 1$, $\beta(\rho) = (1 - \alpha(\rho)/H_2$ so that $S_K(\rho) = \alpha(\rho) + (1 - \alpha(\rho))H_K/H_2$, where $\alpha(\rho)$ is to be determined. It follows from simulation results that $\alpha(\rho) \approx 4\rho/11$ and $S_K(\rho) = H_K/H_2 + (1 - H_K/H_2)\alpha(\rho)$, which leads to Eq. (3) in Table I. This formula has been used in RAID5 performance studies with Markovian assumptions in Menon [1994]—for example, to estimate $R_K^{F/J}(\rho)$ for reconstructing the blocks of a failed disk in a RAID5 disk array with $K+1$ disks. More accurate response times would have been obtained by using $R_K^{max}(\rho)$ rather than $R_K^{F/J}$, as elaborated in Section 3.6.

The coexistence of read requests, which are uniformly distributed at K servers, and writers, which are processed as K-way F/J requests, are considered in Nelson and Tantawi [1988]. The overall arrival rate is λ; a fraction r of requests are reads and

a fraction $w = 1 - r$ are writes. Both request types have a service rate μ. The mean response time of read requests is that of an M/M/1 queueing system, with an arrival rate $\lambda(w+r/K)$, so that $R_r(\rho) = [\mu - \lambda(w+r/K)]^{-1}$. Given Eq. (3), $R_k^{F/J}(\rho)$ is considered as $R_K^{F/J}(\lambda, \mu)$: $R_w(\rho) = R_K^{F/J}(w\lambda, \mu - \lambda r/K)$. The accuracy of this approach is investigated via simulation by setting $\lambda = 1$ and varying $0 \leq \mu < 1$.

Two other approximations to F/J response time with exponential distributions, which are given in Lebrecht and Knottenbelt [2007], are shown in Table I. The approximation derived based on Varma and Makowski [1994] is given by Eq. (4). The approximation obtained in Varki et al. [2013] is given by Eq. (5) and is the mean of pessimistic and optimistic bounds. According to Nelson and Tantawi [1988] the pessimistic bound is:

$$R_K^{F/J}(\rho) \leq R_K^{max}(\rho) = \frac{1/\mu}{1-\rho} H_K. \tag{1}$$

The optimistic bound is obtained by analyzing a CTMC in Varki et al. [2013]:

$$R_K^{F/J(opt)}(\rho) = \frac{1}{\mu}[H_K + S_{K(K-\rho)}], \quad S_{K(K-\rho)} = \sum_{j=1}^{K} \frac{1}{j} \frac{\rho}{j-\rho}. \tag{2}$$

In the case of a variable number of tasks in Varki et al. [2013], the upper limit of summation should be substituted with $k \leq K$. Validation results for $K = 1, \ldots, 100$ and $\rho = 0.1, 0.5, 0.9$ are given in this article. The error is mainly affected by ρ but remains flat for higher values of K.

A limited simulation study to assess the accuracy of the approximations given in Table I against simulation is given in Lebrecht and Knottenbelt [2007]. Also given is the maximum of the response time distributions given by Eq. (21) in Section 4.1. For $\lambda = 1, \mu = 1.1$, the utilization of each M/M/1 queue is $\rho = 0.90909$. The accuracy of these expressions decreases in the order that they are listed, with Eq. (21) overestimating $R_K^{F/J}(\rho)$.

3.2. Analysis of a Two-Way Fork/Join Queueing System

The analysis of F/J queueing systems is difficult even under Markovian assumptions due to the following reasons: (1) the model cannot be decomposed, as the arrival processes to the servers are correlated, and (b) the CTMC has K dimensions, each of which has an infinite number of states. Accurate solutions to unbounded queue sizes for a given arrival rate can be obtained by setting the queue capacities large enough so that the probability of lost requests due to the queue capacity being exceeded is negligibly small. Iterative numerical solution methods for large sets of linear equations are applicable in this case (see Section 5.3 in Bolch et al. [2006]).

The CTMC for the two-way F/J queueing systems with two unbounded state variables is solved in Flatto and Hahn [1984]. This analysis was extended to an M/G/2 system with a general symmetrical joint distribution in Baccelli [1985]. Complex variable function techniques are used in this study to obtain the transient and stationary behavior of the basic boundary value problem (a differential equation together with a set of additional constraints) [Boyce and DiPrima 2012].

In the following discussion, we elaborate on the analysis in Flatto and Hahn [1984], because it leads to the exact analysis of the two-way F/J queueing system. The arrival rate of F/J requests is set to one, and the service rates at the two servers are set to $\alpha > 1$ and $\beta > 1$, ensuring that the system is not saturated. Let i and j denote the number of customers at the two queues, then the state equilibrium probabilities $p_{i,j}$ can be obtained by inverting $P(z, w) = \sum_i \sum_j p_{i,j} z^i w^j$, which is a discrete transform on

Table I. Equations for Approximating Mean F/J Response Time with Markovian Assumptions

$$R_K^{F/J}(\rho) \approx \left[\frac{H_K}{H_2} + \left(1 - \frac{H_K}{H_2}\right)\frac{4\rho}{11}\right]\left(1.5 - \frac{\rho}{8}\right)(\mu - \lambda)^{-1}, 2 \le K \le 32. \quad (3)$$

$$R_K^{F/J}(\rho) \approx \left[H_K + \left(\left(\sum_{i=1}^{K}\binom{K}{i}(-1)^{i-1}\sum_{m=1}^{i}\binom{i}{m}\frac{(m-1)!}{i^{m+1}}\right) - H_K\right)\frac{\lambda}{\mu}\right](\mu - \lambda)^{-1}, \quad (4)$$

$$R_K^{F/J}(\rho) \approx \frac{1}{\mu}\left[H_K + \frac{\rho}{2(1-\rho)}\left(\sum_{i=1}^{K}\frac{1}{i-\rho} + (1-2\rho)\sum_{i=1}^{K}\frac{1}{i(i-\rho)}\right)\right]. \quad (5)$$

two variables. The following equations are obtained in Flatto and Hahn [1984]:

$$P(z, w) = \frac{N(z, w)}{Q(z, w)}, \quad |z|, |w| \le 1,$$

$$N(z, w) = \beta z(w - 1)P(z, 0) + \alpha w(z - 1)P(0, w),$$

$$Q(z, w) = 1 + \alpha + \beta zw - \alpha w - \beta z - z^2 w^2.$$

The state probabilities can be obtained by converting it to a boundary-value problem for $P(z, 0)$ and $P(0, w)$. The discussion illustrates the point that a simply stated queueing problem may require advanced mathematical methods for its analysis.

Using the results of the analysis in Flatto and Hahn [1984], the mean response time for a two-way symmetric F/J queueing system is derived in Appendix B in Nelson and Tantawi [1988]. The arrival rate of Poisson process is set to λ, and the service rates at the two servers are set to μ—that is, $\alpha = \beta = \mu$ with $\lambda < \mu$—so that the server utilizations, which are both equal to $\rho = \lambda/\mu$, are less than one:

$$R_2^{F/J}(\rho) = \left[H_2 - \frac{\rho}{8}\right]R(\rho) = \frac{12 - \rho}{8}R(\rho), \quad (6)$$

where $H_2 = 1.5$, and $R(\rho) = \overline{x}/(1 - \rho) = (\mu - \lambda)^{-1}$ is the mean response time of an M/M/1 queue.

The derivation of $R_2^{F/J}(\rho)$ is based on the observation that it is the sum of the delay at the server ($R(\rho)$) and the synchronization delay ($S_2(\rho)$):

$$R_2^{F/J}(\rho) = R(\rho) + S(\rho). \quad (7)$$

Let q_k denote the probability that the number of tasks in the second queue exceeds the number of tasks in the first queue by k. Due to symmetry $q_k = q_{-k}$, also $q_k = q_{k+1} + p_{k,0}$, where $p_{k,0}$ is the probability that there are k tasks in the first queue and zero tasks in the second queue. It follows that

$$q_k = \sum_{i=k}^{\infty} p_{i,0}.$$

The mean number of tasks being synchronized in the system is

$$\overline{N}_{Synch} = 2\sum_{k=1}^{\infty} kq_k = \sum_{i=1}^{\infty} i(i+1)p_{i,0} = \overline{I(I+1)}.$$

We have substituted q_k with the summation for $p_{i,0}$ and reordered the summations. The z-transform for $p_{i,0}$, given as $P(z,0) = (1-\rho)^{3/2}/\sqrt{1-\rho z}$, is used to compute $\overline{I} = dP(z)/dz|_{z=1} = \rho/2$ and $\overline{I(I-1)} = d^2P(z)/dz^2|_{z=1} = (\rho^2/4)/(1-\rho)$, yielding

$$\overline{N}_{Synch} = \overline{I(I-1)} + 2\overline{I} = \frac{\rho(1-\rho/4)}{1-\rho}.$$

It follows from Little's result that

$$S_2(\rho) = \frac{\overline{N}_{Synch}}{2\lambda} = \frac{1}{2}\left(1-\frac{\rho}{4}\right)R(\rho). \tag{8}$$

Inserting this expression into Eq. (7) yields Eq. (6).

3.3. General Service Times

Motivated by Eq. (22) in Section 4.1, an approximate formula for M/G/1 F/J synchronization is given in Thomasian and Tantawi [1994]:

$$R_K^{F/J}(\rho) = R(\rho) + F_K\sigma_R(\rho)\alpha_K(\rho), \tag{9}$$

where $R(\rho)$ and $\sigma_R(\rho)$ denote the mean and standard deviation of response time in an M/G/1 queue queue with server utilization ρ [Kleinrock 1975] (see Appendix II). Denoting the i^{th} moment of service time by $\overline{x^i}$, its standard deviation by σ_X, and the expected value of the maximum of K random variables by $\overline{X}_K^{max}$, we define

$$F_K = \frac{\bar{X}_K^{max} - \overline{x}}{\sigma_X}, \tag{10}$$

so that it follows from Eq. (9) that for $\rho = 0$, $\alpha_K(\rho) = 1$. It can be easily shown that for the exponential distribution, $F_K = H_K - 1$, and that $F_K = \sqrt{6}ln(K)/\pi$ for the extreme value distribution (EVD) [Johnson et al. 1995] (see Section 4.6).

Curve fitting with respect to simulation results for a given response time distribution can be used to obtain the expression for $\alpha_K(\rho)$ [Thomasian and Tantawi 1994]:

$$\alpha_K(\rho) = \frac{R_K^{F/J}(\rho) - R(\rho)}{F_K\sigma_R(\rho)}. \tag{11}$$

For a two-way F/J with exponential interarrival and service times, it follows from Eq. (6) that $\alpha_2(\rho) = 1 - \rho/4$—for instance, plotting $\alpha_2(\rho)$ versus ρ, we have a straight line with negative slope. For K-way F/J requests with Erlang distributions, surface fitting yields

$$\alpha_K(\rho) = 1 - (a - b/K)\rho.$$

$\alpha_K(\rho)$ versus ρ for $K = 2^k, 1 \le k \le 6$ is plotted in Thomasian and Tantawi [1994] for four service time distributions in Varma and Makowski [1994], which are specified in Section 3.4. Surface-fitting F/J response times with respect to K and ρ yields

$$\alpha_K(\rho) = 1 + a\rho + b\rho^2 - \rho(c + d\rho)\frac{log_2 K}{K+e}. \tag{12}$$

Note that $\alpha_K(\rho)$ is not dependent on K, as $K \to \infty$—that is, the increase in $R_K^{F/J}(\rho)$ is negligible for higher values of K.

Linear expressions for $\alpha_K(\rho)$ are observed for K-way F/J with the following distributions: exponential, Erlang with two stages ($\mathcal{E}_2$), and hyperexponential with two branches ($\mathcal{H}_2$) with a small CV. The slopes are negative, and it follows from Eq. (12)

that the slope decreases as K increases, which is tantamount to an increase in F/J response time. For $\mathcal{H}_2$ with a high CV, $\alpha_K(\rho)$ is initially quadratic with a positive slope for smaller values of ρ and when K is small achieves a maximum and then becomes linear with a negative slope as ρ increases. Further experimentation with curve fitting may yield simpler expressions for $\alpha_K(\rho)$.

The steps to obtain $R_K^{F/J}(\rho)$ for a given distribution are as follows:

(1) Compute $F_K = (\bar{X}_K^{max} - \overline{x})/\sigma_X$ for $1 \leq log_2(K) \leq 6$.
(2) Compute $R(\rho)$ and $\sigma_R(\rho)$ using the M/G/1 queueing model as ρ is varied (see Eq. (75) in Appendix II).
(3) Use simulation to estimate $R_K^{F/J}(\rho)$ for the set of values of K and ρ of interest.
(4) Obtain $\alpha_K(\rho)$ in Eq. (12) by solving the associated nonlinear equations. Surface fitting with the AGSS package [Lane and Welch 1987] was used for this purpose.

3.4. Analyses of Markovian Models

The analysis in Varma and Makowski [1994] combines light and heavy traffic limits to interpolate the F/J response time, yielding an approximation shown to be quite accurate by comparison against simulation results. Erlang and hyperexponential interarrival and service times are considered in this study.

A k-stage Erlang ($\mathcal{E}_k$) probability density function (pdf) is given as

$$f_X(x) = \frac{\mu(\mu x)^{k-1}}{(k-1)!}e^{-\mu x}, \quad k \geq 1, \quad t > 0. \tag{13}$$

The cumulative distribution function (CDF) for $\mathcal{E}_k$ is given by Eq. (31) in Section 4.3.

The Gamma distribution is a generalization of the Erlang distribution with a positive parameter α, instead of k [Trivedi 2002]. The pdf for the Gamma distribution is

$$f_X(x) = \frac{\mu(\mu x)^{\alpha-1}}{\Gamma(\alpha)}e^{-\mu x}, \quad \alpha > 0 \quad x > 0,$$

where the Gamma function is $\Gamma(\alpha) = \int_0^\infty t^{\alpha-1}e^{-t}dt$. Integration by parts yields $\Gamma(\alpha) = (\alpha-1)\Gamma(\alpha-1)$. When $\alpha = k$ is an integer $\Gamma(k) = (k-1)!$, which is the factorial function. The Laplace-Stieltjes transform (LST) of its pdf, mean, variance, and third moment of the Gamma distribution are

$$\mathcal{X}^*(s) = \left(\frac{\mu}{\mu+s}\right)^\alpha, \quad E[X] = \frac{\alpha}{\mu}, \quad \sigma_X^2 = \frac{\alpha}{\mu^2}, \quad E[X^2] = \frac{\prod_{i=0}^{2}(\alpha+i)}{\mu^3}.$$

The Beta function, which is related to the Gamma function, has two parameters [Johnson et al. 1995; David and Nagaraja 2003]:

$$B(\alpha, \beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}.$$

The pdf of the Beta distribution is

$$f_X(x;\alpha,\beta) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha,\beta)}, \quad 0 < x < 1. \tag{14}$$

The Beta distribution has an important application in the theory of order statistics in that the distribution of the k^{th} order statistic among n uniform distributions (0, 1) has a Beta distribution: $U_{(k)} \sim B(k, n+1-k)$ [David and Nagaraja 2003].

For a two-stage Erlang distribution $\mathcal{E}_2$, $\overline{x} = 1$, and $\sigma_X^2 = 1/2$, we have

$$\overline{X}_K^{max} = \frac{1}{\mu} \sum_{n=1}^{K} \binom{K}{n} (-1)^{n-1} \sum_{m=1}^{n} \binom{n}{m} \frac{m!}{2n^{m+1}},$$

which results in $F_K = \sqrt{2}(\bar{X}_K^{max} - 1)$. The analyses of RAID5 disk arrays in Thomasian and Menon [1994, 1997] and Kuratti and Sanders [1995] approximate disk service time by a k-stage Erlang distribution.

The pdf for the hyperexponential distribution with two branches ($\mathcal{H}_2$) is

$$b(t) = p_1 \mu_1 e^{-\mu_1 t} + p_2 \mu_2 e^{-\mu_2 t}, \ t > 0, \tag{15}$$

where $0 < p_1 < 1$, $p_2 = 1 - p_1$, $\mu_1 \neq \mu_2$. When $p_1/\mu_1 = p_2/\mu_2 = 0.5$, then $p_1 + p_2 = 0.5(\mu_1 + \mu_2) = 1$ yields $\mu_1 + \mu_2 = 2$. The first equality implies that both branches yield the same utilization, and the mean is given as $b = p_1/\mu_1 + p_2\mu_2 = 1$. Algebraic manipulation yields $\sigma_B^2 = 2/(\mu_1\mu_2) - 1$. The CDF for this distribution is given in Eq. (37).

$$\overline{X}_K^{max} = \sum_{n=1}^{K} (-1)^{m+1} \sum_{m=0}^{n} \binom{n}{m} \frac{p_1^m p_2^{n-m}}{m\mu_1 + (n-m)\mu_2}.$$

For a small CV, we have the parameters $\mu_1 = 0.5$, $\mu_2 = 1.5$, $p_1 = 0.25$, $p_2 = 0.75$ with $\sigma_B \approx 1.29$ in Varma and Makowski [1994]. For a large CV, $\mu_1 = 0.1$, $\mu_2 = 1.9$, $p_1 = 0.05$, $p_2 = 0.95$ with $c_B \approx 3$.

A two-way F/J queueing system is analyzed in Kim and Agrawala [1985] to yield the virtual waiting time CDF for the M/E$_k$/1 queue:

$$F_W(t) = 1 - e^{-k\mu t} \sum_{i=0}^{N} \frac{b_i [k\mu t]^i}{i!}. \tag{16}$$

N is chosen to be sufficiently large with respect to k to obtain an accurate solution. In addition,

$$b_i = \frac{\lambda}{k\mu} \sum_{j=1}^{k} b_{i-j}, \ i \geq k \qquad b_i = 1, \ 0 \leq i \leq k-1.$$

This solution method is extended in Kim and Agrawala [1989] to yield a transient solution for the virtual waiting or response time with FCFS scheduling. The analysis is applicable to exponential and hyperexponential interarrival times with Erlang service times.

An upper bound to the expected value of the maximum is obtained in Kuratti and Sanders [1995] by using the characteristic maximum [Blom 1958; Gumbel 1958; Gravey 1985] (see Section 4.9).

Upper and lower bounds to the response time of F/J queueing systems are derived in Lui et al. [1998]. The arrival process is phase type, and the service time is Erlangian. The capacity (C) and the difference in queue lengths (d) are utilized in both upper and lower bound models, whose spread is defined as $(R_u - R_\ell)/(R_u + R_\ell)$. Numerical results are provided for two cases: exponential service times, and arrival times that are exponential and $\mathcal{E}_2$. The trade-off between the computational cost and the tightness of the bounds is provided in Table 4 of Lui et al. [1998]. It is observed that the error percentage decreases with increasing C and d. The determination of these parameters to attain a given spread of bounds is listed as an area of further investigation.

3.5. Fork/Join and Related Models Analyzed Using the Matrix Geometric Method

The MGM was applied to the ISM in Gillent [1980]. A two-way SM queueing system using MGM is analyzed in Rao and Posner [1985] by treating one of the queues as bounded. Expressions are developed for the mean, the variance, and the distribution function of the sojourn time, as well as the interdeparture time of tasks. An efficient approximation scheme is also developed for systems with more than two branches.

A series of papers applying the MGM to analyzing F/J queueing systems is as follows. An approximate response time distribution for a modified F/J model is obtained in Balsamo and Mura [1995]. Moments of F/J response time are obtained in Balsamo and Mura [1997]. An F/J queueing system with unequal service rates is analyzed in Balsamo et al. [1998], which uses lower and upper lower bound models to obtain the two bounds.

MGM is used in Takahashi et al. [2000] and Takahashi and Takahashi [2000] to analyze F/J systems, where the distribution of interarrival times is phase type from an infinite population, but arrivals that find the input buffer full are lost.

Analysis of F/J synchronization in a closed QNM with two-phase Coxian inputs feeding a two-way F/J queue is discussed in Krishnamurthy et al. [2004]. The system is analyzed from the viewpoint of queue lengths of the F/J node and also its interdeparture times. It is observed that variability of interarrival times has a significant impact on performance.

A variation of the classical F/J queueing systems is analyzed exactly using the MGM for small problems and approximately based on fixed point iteration for large problems in Squillante et al. [2008]. Each task has multiple iterations with different phases of execution, which include vacations and communication among sibling tasks. Dynamic polices are considered for scheduling to maintain effective server utilization. Simulation results are presented to ascertain the accuracy of the approximate solution.

In Section 5.1, we summarize the application of the MGM to the analysis of a mirrored disk system based on Towsley et al. [1990b].

3.6. Fork/Join and Interfering Requests

The processing of F/J requests and read requests uniformly distributed over the K servers was considered in Section 3.1, based on Section IV.B in Nelson and Tantawi [1988]. Limited simulations lead to the conclusion that Eq. (3) provides a good approximation when the fraction of read requests is $r = 0.75$.

Let $R_K^{F/J/I}(\rho)$ denote the mean response time of F/J requests with interfering requests. It is observed from the simulation study in Thomasian and Tantawi [1994] with Poisson arrivals and the four service time distributions in Varma and Makowski [1994] that $R_K^{F/J}(\rho) < R_K^{F/J/I}(\rho)$.

$R_K^{F/J/I}(\rho)$ increases as interfering requests constitute a higher fraction of server load at the same total utilization, $\rho_{F/J/I} = \rho_{F/J} + \rho_I$. When $\rho_I/\rho \to 1$, then $R_K^{F/J/I}(\rho) \to R_K^{max}(\rho)$ and otherwise $R_K^{F/J/I}(\rho) \approx R_K^{F/J}(\rho)$ [Thomasian and Tantawi 1994; Thomasian 1997]. This is due to the fact that when interfering requests are present, the response times of F/J tasks become less synchronized, so the F/J response time can be approximated by the maximum of the response times of K tasks. Stated differently, interfering requests result in a higher variability in the response times of F/J tasks for a fixed overall server utilization (ρ). The following bounds hold according to the simulation results in Thomasian and Tantawi [1994]:

$$R_K^{F/J}(\rho) \le R_K^{F/J/I}(\rho) \le R_K^{max}(\rho). \tag{17}$$

This discussion is motivated by the performance analysis of RAID5 disk arrays in degraded mode and that disk accesses to surviving disks due to F/J requests are

processed together with interfering requests, as explicated in Appendix I and based on the analyses in Thomasian and Menon [1994, 1997] and Thomasian et al. [2007].

4. THE EXPECTED VALUE OF THE MAXIMUM

Section 4.1 discusses order statistics [David 1970; David and Nagaraja 2003] in the context of computing the expected value of the maximum of response time $R_K^{max}(\rho)$ or execution times $\overline{X}_K^{max}$. Section 4.2 provides $\overline{X}_K^{max}$ for the exponential distribution [Kleinrock 1975; Allen 1990; Trivedi 2002]. Also provided is a recurrence relation for this maximum given in Harrison and Zertal [2007] and a new summation for the LST of the maximum. Section 4.3 utilizes the Erlang distribution $\mathcal{E}_k$ with k exponential stages with rate μ to approximate disk response times to estimate $R_K^{max}(\rho)$ in RAID5 disk arrays. Section 4.4 discusses methods to obtain the three parameters of $\mathcal{H}_2$-that is, a hyperexponential distribution with two branches [Allen 1990]. Section 4.5 discusses the use of the Coxian distribution with two parameters q and μ [Chen and Towsley 1993]. Section 4.6 discusses the application of EVD, which was used in Merchant and Yu [1996] and Thomasian et al. [2007]. Section 4.7 discusses the use of the normal distribution to approximate disk service time as in Kim and Tantawi [1991]. Section 4.8 provides an approximation for the maximum of generally distributed random variables based on Harrison and Zertal [2007]. Section 4.9 discusses the characteristic maximum [Gravey 1985; Kruskal and Weiss 1985], which was used in Kim and Tantawi [1991], Thomasian and Menon [1994, 1997], and Kuratti and Sanders [1995] to reduce the cost of computing the maximum as in the case of $\mathcal{E}_k$ and Coxian distributions. We conclude with a discussion of long-tailed distributions in Section 4.10.

4.1. Order Statistics

The K^{th} order statistic, denoted by Y_K, or the maximum of K independent, identically distributed (i.i.d.) random variables with distribution $F_X(y)$, is given as follows [Allen 1990; Trivedi 2002]:

$$F_{Y_K}(y) = [F_X(y)]^K. \tag{18}$$

More generally, the k^{th} order statistic of K random variables is given as

$$F_{Y_k}(y) = \sum_{j=k}^{K} \binom{K}{j} F_X(y)^j [1 - F_X(y)]^{K-j}, \quad 1 \le k \le K. \tag{19}$$

For positive random variables, the mean and the n^{th} moment of Y_K are given as

$$\overline{Y}_K = \int_0^\infty y dF_X^K(y) = \int_0^\infty \left[1 - F_X^K(y)\right] dy, \tag{20}$$

$$\overline{Y_K^n} = \int_0^\infty y^n dF_X^K(y) = K \int_0^\infty y^n f_X(y) [F_X(y)]^{K-1} dy. \tag{21}$$

The mean response time of the K-way SM queueing system defined in Section 1 can be obtained by representing it by an M/G/1 queueing system. This requires the first two moments of service time: $\overline{Y_K^n}, n = 1, 2$ as inputs to Eq. (75) for M/G/1 queues described in Appendix II.

Numerical methods can be applied to evaluate these integrals, but it is easier to carry out the integration symbolically for distributions, which are exponential in form, such as $\mathcal{E}_k$, $\mathcal{H}_2$, or Coxian [Kleinrock 1975; Allen 1990; Trivedi 2002]. The computational cost increases with the number of ways (K), and the number of the stages for Erlang

and Coxian distributions (k). The characteristic maximum [Gravey 1985] can be used to deal with the high computational cost problem (see Section 4.9).

The following approximation for the expected value of the maximum of K random variables is given in David [1970] and David and Nagaraja [2003]:

$$\overline{X}_K^{max} \approx \mu_X + \sigma_X G(K). \tag{22}$$

For the exponential distribution, $G(K) = H_K - 1$, and for the uniform distribution, $G(K) = \sqrt{3}(K-1)/(K+1)$. Alternatively, for the uniform distribution $(0, 2a)$ with mean $\mu_X = a$ and variance $\sigma_X^2 = a^2/12$, $\overline{X}_K^{max} = 2aK/(K+1)$, which approaches $2a$, as $K \to \infty$ as would be expected. A tight upper bound to $G(K)$ is [Davis 1970]

$$G(K) \le \frac{K-1}{\sqrt{2K-1}}. \tag{23}$$

When the mean is zero and the standard deviation is equal to one, then $\overline{X}_K^{max} = G(K)$. Equality in Eq. (23) holds for the distribution

$$P(x) = \frac{b}{K}\left(\frac{1+bx}{K}\right)^{1/(K-1)}, \quad -\frac{\sqrt{2K-1}}{K-1} \le x \le \sqrt{2K-1}.$$

For a symmetric distribution $P(x) = 1 - P(-x)$,

$$\overline{X}_K^{max} \le \frac{K}{2}\left[\frac{2\left(1 - 1/\binom{2K-2}{K-2}\right)}{2n-1}\right]^{1/2}.$$

In Lebrecht et al. [2009], the logistic function $f(t) = [1 + e^{a-bt}]^{-1}$ is fitted to $[F_X(y)]^K$, whose LST is the Hypergeometric2F1(a, b, c, s) function [Weisstein 2012],

$$\mathcal{X}^*(s) = \mathrm{H}ypergeometric2F1[1, s/b, (b+s)/b, -e^a],$$

which solves the hypergeometric differential equation,

$$z(1-z)y'' + [c - (a+b+1)z]y' - aby = 0.$$

If independent, not identically distributed (INID) random variables have distributions $F_i(x)$, $1 \le i \le K$, then the distribution of the first and K^{th} order statistic are given as follows:

$$F_{(1)}(t) = 1 - \prod_{i=1}^{K}[1 - F_i(t)], \quad F_{(K)}(t) = \prod_{i=1}^{K} F_i(t). \tag{24}$$

The CDF for the dispersion, which is the time from the completion of the first to the K^{th} F/J task $(X_{(K)} - X_{(1)})$ is given in Tsimashenka and Knottenbelt [2013b] as

$$F_{\mathrm{d}isp.}(x) = \sum_{i=1}^{K}\int_{-\infty}^{\infty}\prod_{j=1, j\ne i}[F_j(x+t) - F_j(x)]dx. \tag{25}$$

4.2. Exponential Distribution

Given that the pdf of the exponential distribution is $f(t) = \lambda e^{-\lambda t}$, the LST of its pdf, its i^{th} moment, variance, and CV are

$$\mathcal{F}^*(s) = \frac{\lambda}{s+\lambda}, \quad \overline{t^i} = \frac{i!}{\lambda^i}, \quad \sigma_t^2 = \frac{1}{\lambda^2}, \quad c = \frac{\sigma}{\overline{t}} = 1. \tag{26}$$

Let X_i denote the i^{th} exponentially distributed random variable (r.v.) with probability distribution $F_i(t) = 1-e^{-\lambda_i t}$ and pdf $f_i(t) = \lambda_i e^{-\lambda_i t}$, $1 \le i \le K$. The CDF for the maximum follows from Eq. (18):

$$F_{Y_K}(t) = \prod_{i=1}^{K} F_i(t) = \prod_{i=1}^{K}(1 - e^{-\lambda_i t}).$$

The CDF, pdf, and the LST for $K = 2$ are [Allen 1990; Trivedi 2002]:

$$\begin{aligned} F_{Y_2}(t) &= F_1(t)F_2(t) = 1 - e^{-\lambda_1 t} - e^{-\lambda_2 t} + e^{-(\lambda_1+\lambda_2)t}. \\ &= f_{Y_2}(t) = \frac{dF_{Y_2}(t)}{dt} = f_1(t)F_2(t) + f_2(t)F_1(t) \\ &= \lambda_1 e^{-\lambda_1 t} + \lambda_2 e^{-\lambda_2 t} - (\lambda_1 + \lambda_2)e^{-(\lambda_1+\lambda_2)t}. \\ \mathcal{L}^*_{Y_2}(s) &= \int_0^\infty f_{Y_2}(t)e^{-st}dt = \frac{\lambda_1}{s+\lambda_1} + \frac{\lambda_2}{s+\lambda_2} - \frac{\lambda_1+\lambda_2}{s+\lambda_1+\lambda_2}. \end{aligned}$$

The expected value of the maximum can be obtained by taking the derivative of the LST with respect to s and setting $s = 0$ [Allen 1990; Trivedi 2002] (see Appendix II):

$$\overline{Y}_2^{max} = -\frac{d\mathcal{L}^*_{Y_2}(s)}{ds}|_{s=0} = \frac{1}{\lambda_1} + \frac{1}{\lambda_2} - \frac{1}{\lambda_1+\lambda_2}. \tag{27}$$

A general expression for $\overline{Y}_K^{max}$ for $K > 2$ is given by Eq. (67) in Section 7.3.

The LST of the maximum of K exponentially distributed random variables can be expressed as follows, using Eq. (27) as a starting point:

$$\mathcal{L}^*_{Y_K}(s) = \sum_{i=1}^{K}(-1)^{i-1}\sum_{n=1}^{\binom{K}{i}}\sum_{k=1}^{n}\frac{\sum_{\ell=1}^{i}\lambda_{(k+\ell-1)_K}}{s+\sum_{\ell=1}^{i}\lambda_{(k+\ell-1)_K}}, \tag{28}$$

where $(k+\ell-1)_K$ denotes a summation modulo K.

The maximum of K random variables with negative exponential distribution with parameters $\underline{\alpha} = (\alpha_1, \ldots, \alpha_K)$ and pdf $f_n(\underline{\alpha}, t)$ with LST $\mathcal{L}^*(\underline{\alpha}, s)$ can be expressed as a recurrence, where $\backslash j$ indicates the exclusion of α_j [Harrison and Zertal 2007]:

$$\left(s + \sum_{j=1}^{m}\alpha_j\right)\mathcal{L}^*_m(\underline{\alpha}, s) = \sum_{j=1}^{m}\alpha_j\mathcal{L}^*_{m-1}(\underline{\alpha}_{\backslash j}, s), 1 \le m \le K. \tag{29}$$

The n^{th} moment of the maximum of K exponentially distributed i.i.d. random variables is then

$$M_K(\underline{\alpha}, n) = \frac{nM_K(\underline{\alpha}, n-1)}{\sum_{j=1}^{K}\alpha_j} + \frac{\sum_{j=1}^{K}M_{K-1}(\underline{\alpha}_{\backslash j}, k)}{\sum_{j=1}^{K}\alpha_j}. \tag{30}$$

4.3. Erlang Distribution

The Erlang distribution ($\mathcal{E}_k$) with $k > 1$ exponential stages with parameter μ [Kleinrock 1975; Allen 1990; Trivedi 2002] has been used in Thomasian and Menon [1994, 1997] and Kuratti and Sanders [1995] to approximate the response time distribution for disk access time. Its two parameters can be obtained using the first two moments of the

M/G/1 queue for obtaining R_K^{max} in RAID5 disk arrays (see Appendix II):

$$F_X(t) = 1 - e^{-\mu t} \sum_{j=0}^{k-1} \frac{(\mu t)^j}{j!},\ t > 0,\ k \geq 1. \tag{31}$$

The Erlang distribution has an LST, mean, variance, and squared CV given as follows:

$$\mathcal{E}^*(s) = \left(\frac{\mu}{s+\mu}\right)^k, \quad \overline{x} = \frac{k}{\mu}, \quad \sigma_X^2 = \frac{k}{\mu^2}, \quad c_X^2 = \frac{\sigma_X^2}{(\overline{x})^2} = \frac{1}{k} < 1.$$

Let X denote the r.v. for single read (SR) service time in RAID5, which is given in Appendix II as the sum of components of disk service time. The squared CV for the response time of SR requests is given (incorrectly) in Thomasian and Menon [1997]:

$$c_{SR}^2(\rho) = \frac{\sigma_{SR}^2(\rho)}{R_{SR}^2(\rho)} = \frac{c_X^2 + \frac{\rho s_X}{3(1-\rho)} + \frac{\rho^2(1+c_X^2)^2}{4(1-\rho)^2}}{1 + \frac{\rho(1+c_X^2)}{1-\rho} + \frac{\rho^2(1+c_X^2)^2}{4(1-\rho)^2}}. \tag{32}$$

The coefficient of skewness $s_X = \overline{x^3}/(\overline{x})^3$ is the third moment divided by the mean cubed. The disk utilization ρ is the sum of disk utilizations due to F/J and independent requests. For $\rho \to 0$ $c_{SR}^2(\rho) \to c_X^2$ and for $\rho \to 1$ $c_{SR}^2(\rho) \to 1$—that is, an exponential distribution regardless of the service time distribution.

The response time of SR requests to small randomly placed disk blocks in Thomasian and Menon [1997] has $c_{SR}^2(\rho) < 1$ for the disk characteristics under consideration, so the response time can be approximated by an Erlang or Coxian distribution [Chen and Towsley 1993]. When $y = 1/c_R^2$ is not an integer given $k = \lceil y \rceil$ and $j = \lfloor y \rfloor$, $R_y^{max} = (k-y)R_j^{max} + (y-j)R_k^{max}$, where we temporarily simplify the notation by eliding ρ. Another approach to deal with the problem is given in the context of the Coxian distribution by Eq. (39).

$R_K^{max}(\rho)$ for service times approximated by $\mathcal{E}_k$ distribution is given in Thomasian and Menon [1997] as:

$$R_K^{max}(\rho) = \int_0^\infty \left[1 - \prod_{i=1}^{K} \left(1 - e^{-\mu_i t} \sum_{j=0}^{k_i-1} \frac{(\mu_i t)^j}{j!}\right)\right] dt. \tag{33}$$

In the special case $K = 2$, we have

$$R_2^{max}(\rho) = \sum_{i=1}^{2} R_i(\rho) - \sum_{m=0}^{k_1-1} \sum_{n=0}^{k_2-1} \binom{m+n}{m} \frac{\mu_1^m \mu_2^n}{(\mu_1+\mu_2)^{m+n+1}}, \tag{34}$$

where $\mu_1 = k_1/R_1(\rho)$ and $\mu_2 = k_2/R_2(\rho)$.

An Erlang distribution with different service rates per stage, $\lambda_i, 1 \leq i \leq m$, is a hypoexponential distribution, whose mean and variance are as follows [Allen 1990]:

$$\mu = \sum_{i=1}^{m} \lambda_i^{-1}, \quad \sigma^2 = \sum_{i=1}^{m} \lambda_i^{-2}. \tag{35}$$

Given the mean μ and $c^2 = \sigma^2/\mu^2$, the two parameters of a two-stage Erlang distribution can be obtained using Eq. (35) with $m = 2$ [Crow et al. 2007]:

$$\lambda_i^{-1} = \frac{\mu}{2}\left(1 \pm \sqrt{2c^2 - 1}\right), \quad i = 1, 2.$$

The shifted exponential distribution

$$f_X(x) = \lambda e^{-\lambda(x-d)}, \quad x > d, \tag{36}$$

with mean $\mu = \lambda^{-1} + d$ and variance $\sigma^2 = \lambda^{-2}$, can be fitted to a CV $0 < c < 1$ by setting $\lambda = \sigma^{-1}$ and $d = \mu - \lambda^{-1}$.

When the distribution of response time in an M/G/1 queue is the sum of several exponential distributions, it can then be used in conjunction with Eq. (20) to compute R_K^{max}.

It is observed that although this method outperforms the approximation in Harrison and Zertal [2007], the error in both cases increases with K for low variance Erlang distribution. The error decreases with increasing K for Eq. (21), but the reverse is true for Eq. (30).

4.4. Hyperexponential Distribution

The CDF for $\mathcal{H}_2$ is a weighted sum of two exponentials, whose pdf is given by Eq. (15), is

$$F_X(x) = 1 - pe^{-\mu_1 x} - (1-p)e^{-\mu_2 x}, \quad x > 0. \tag{37}$$

The settings $0 < p < 1$ and $\mu_1 \neq \mu_2$ ensure that the hyperexponential distribution does not degenerate to an exponential distribution.

The moments of the hyperexponential distribution are weighted sums of the exponential distribution:

$$\overline{x} = \frac{1}{\mu} = \frac{p}{\mu_1} + \frac{1-p}{\mu_2}, \quad \overline{x^2} = \frac{2p}{\mu_1^2} + \frac{2(1-p)}{\mu_2^2}.$$

Its squared CV can be easily shown to be greater than one: $c^2 = \overline{x^2}/(\overline{x})^2 - 1 > 1$. The LST is also a weighted sum of the LSTs of the exponential distribution given by Eq. (26).

Several algorithms are presented in Allen [1990] to match the moments of response time distribution with a hyperexponential distribution to obtain its three parameters.

Given an input distribution, three algorithms (3.2.2, 3.2.3, and 3.2.4) are given in Allen [1990] to obtain the parameters of the matching $\mathcal{H}_2$.

Algorithm 3.2.2 makes the assumption that the utilizations of the branches of the hyperexponential distribution are balanced: $p/\mu_1 = (1-p)/\mu_2$. When $\mathcal{H}_2$ is a service time distribution, this implies that they contribute equally to server utilization. It follows from $p/\mu_1 + (1-p)/\mu_2 = 1/\mu$ that $\mu_1 = 2p\mu$ and $\mu_2 = 2(1-p)\mu$. Substituting μ_1 and μ_2 into $c^2 = \overline{x^2}/\overline{x}^2 - 1$ leads to

$$p \;/\; 1-p = \frac{1}{2}\left(1 \pm \sqrt{\frac{c^2-1}{c^2+1}}\right).$$

Algorithm 3.2.3 in obtains the parameters of an $\mathcal{H}_2$ distribution, which has the same third moment as a Gamma distribution:

$$\mu_1 \;/\; \mu_2 = 2\mu\left(1 \pm \sqrt{\frac{c^2-0.5}{c^2+1}}\right),$$

$$p = \frac{\mu_1(\mu_2 - \mu)}{\mu(\mu_2 - \mu_1)}, \quad 1 - p = \frac{\mu_2(\mu - \mu_1)}{\mu(\mu_2 - \mu_1)}.$$

Algorithm 3.2.4 in obtains the three parameters of $\mathcal{H}_2$ by using the first three moments of the response time ($k_i, 1 \leq i \leq 3$). The following relationships hold:

$k_1 = 1/\mu$, $c^2 = k_2/k_1^2 - 1$, and $k_1 k_3 > 1.5k_2^2$ ($k_1 k_3 \geq k_2^2$ in general [Whitt 1982]). Given $x = k_1 k_3 - 1.5k_2^2$, $y = k_2 - 2k_1^2$, and $v = (x + 1.5y^2 - 3k_1^2 t)^2 + 18k_1^2 y^3$, we have

$$\mu_1\ \mu_2 = 6k_1 y\big[x + 1.5y^2 + 3k_1^2 y \pm \sqrt{v}\big]^{-1},$$

and $p = (1/\mu - 1/\mu_2)/(1/\mu_1 - 1/\mu_2)$.

Note that given the first two moments of the target distribution, an additional assumption is used by Algorithms 3.3.2 and 3.3.3 to obtain the three parameters for $\mathcal{H}_2$, whereas Algorithm 3.3.4 uses the three moments of the distribution. It remains to be determined if the form of the distribution has a significant effect on R_K^{max}.

4.5. Coxian Distribution

We follow the analysis of RAID5 disk arrays in Chen and Towsley [1993], which considers small strip sizes, so that a request to larger block sizes requires accesses to multiple disks via an F/J request. Disk response time in this study is approximated by a Coxian distribution with k stages with parameters q and μ, where q is the probability of a transit to the next stage. Setting the rate of all stages to μ simplifies the discussion:

$$f(x) = (1-q)u_0(x) + (1-q)\sum_{i=1}^{k-1} \frac{q^i \mu(\mu x)^{i-1} e^{-\mu x}}{(i-1)!} + \frac{q^k \mu(\mu x)^{k-1} e^{-\mu x}}{(k-1)!}, \tag{38}$$

where $u_0(x)$ is the unit impulse function at the origin $\int_{-\infty}^{0+} u_0(x)dx = 1$ [Kleinrock 1975]. The value of k can be determined using the inequalities

$$\frac{1}{c_R^2} \leq k \leq \frac{1}{c_R^2} + 1. \tag{39}$$

Note that $k \to \infty$ as $c \to 0$. Given the first two moments of the input distribution: R and $\overline{R^2}$, the values of q and μ are obtained by solving the two equations:

$$R = \sum_{i=1}^{k} \frac{q^i}{\mu}, \quad \overline{R^2} = 2\sum_{i=1}^{k} i\frac{q^i}{\mu^2}.$$

Algorithm 3.2.5 in Allen [1990] specifies a two-stage Coxian distribution, whose stages are specified as $\tilde{T}_1$ and $\tilde{T}_2$ with means $T_1 = 1/\mu_1$ and $T_2 = \mu_2$. The probability that the second stage is visited is q. The r.v. representing the Coxian distribution is $\tilde{X} = \tilde{T}_1 + q\tilde{T}_2$, and its mean, second moment, and variance are

$$\overline{x} = T_1 + qT_2 = \frac{1}{\mu_1} + \frac{q}{\mu_2}, \quad \overline{x^2} = 2\big(T_1^2 + qT_2^2 + qT_1T_2\big), \quad \sigma_X^2 = \frac{1}{\mu_1^2} + \frac{q(2-q)}{\mu_2^2}. \tag{40}$$

Given a target distribution with a mean $1/\mu$ and squared CV c^2, we have two equations in three unknowns. One method to deal with this problem is to vary $0 < q < 1$ such that it results in positive values for μ_1 and μ_2. The approach in Allen [1990] is based on Marie [1978] and adds a third condition that $1/\mu_1 = q/\mu_2$, which implies that the two stages contribute equally to the mean. This yields $q = \mu_2/\mu_1$, which is acceptable if $\mu_2 < \mu_1$. This leads to $\mu_1 = 2\mu$ and $\mu_2 = 2\mu q$ The value of q can be obtained by plugging in μ_1 and μ_2 into the equality $\sigma_X^2 = c^2\mu^2$, which yields $q = 1/(2c^2)$. This leads to $\mu_2 = \mu/c^2$. More details can be found in the discussion of branching Erlang distribution in Sauer and MacNair [1983] and Section 3.2.1.2 in Lipsky [2008].

Given the first three moments of a target distribution, whose CV is greater than one, it can be matched to an $\mathcal{H}_2$ or a Coxian distribution with two stages. According to

Example 3.2.11 in Allen [1990], the same service rates are obtained for the two stages, although the transition probabilities are different, since the first stage is visited in all cases for the Coxian distribution. Interestingly, both pdfs have the same LST. It is to be determined if this approach yields a Coxian distribution, which better represents the original distribution.

The use of percentiles in modeling the CPU service time distribution is proposed in Lazowska [1977]. A technique to determine the parameters of Coxian distribution, which matches the mean and percentiles of a given distribution, is given in Lazowska and Addison [1979].

4.6. Extreme Value Distribution

A simple method to obtain the maximum of K i.i.d random variables is to approximate them with an EVD [Johnson et al. 1995]:

$$F_Y(y) = P(Y < y) = exp\left(-e^{-\frac{y-a}{b}}\right). \tag{41}$$

Noting that $\gamma = 0.577721$ is the Euler constant, the mean and variance of this distribution are

$$\overline{Y} = a + \gamma b, \quad \sigma_Y^2 = \frac{\pi^2 b^2}{6}.$$

The maximum of K random variables with this distribution is [Johnson et al. 1995]

$$Y_{max}^K = (a + \gamma b) + bln(K) = \overline{Y} + \frac{\sqrt{6}ln(K)}{\pi}\sigma_Y. \tag{42}$$

Given the mean and variance of SR requests $R_{SR}(\rho))$ and $\sigma_{SR}^2(\rho)$, respectively, the values for a and b can be obtained by matching them to $\overline{Y}$ and σ_Y^2: $b = \sqrt{6}\sigma_{SR}(\rho)/\pi$ and $a = R_{SR}(\rho) - \gamma b$ [Merchant and Yu 1996; Thomasian et al. 2007]. It follows that

$$R_K^{max}(\rho) = R_{SR}(\rho) + (\sqrt{6}ln(K)/\pi)\sigma_{SR}(\rho). \tag{43}$$

Calibration against simulation results in Thomasian et al. [2007] shows that $R_K^{max}(\rho)$ can be estimated more accurately by dividing the second term in the summation by 1.27.

Approximations for the probability distribution of the maximum of n i.i.d. nonnegative random variables and the first few moments of the distribution are introduced in Crow et al. [2007]. Distributions with heavy tails are not considered. A minimum threshold n^* is obtained depending on the underlying distribution so that $n > n^*$ is required for the asymptotic extreme-value approximations to be accurate. The threshold n^* tends to increase as c^2 increases above one or decreases below one. In the case of the Gumbel distribution [Gumbel 1958] with r.v. W,

$$\begin{aligned} G(x) &= p(W \le x) = exp(-e^{-x}), \\ E[W] &= \gamma = 0.57772, \\ Var[W] &= \sigma_W^2 = \pi^2/6 = 1.6449. \end{aligned}$$

It is shown in Gumbel [1954] that the maximum value in a sample of random variables following an exponential distribution approaches this distribution with increasing sample size. Let M_K denote the maximum of K exponentially distributed random variables with mean $m_1 = 1$. It follows that

$$M_K \approx ln(K) + W,$$

$$E[M_K] \approx ln(K) + 0.57772,$$

$$Var[M_K] \approx Var[W] = \pi^2/6.$$

Table II. Approximate Versus the Exact Value for $E[Y_K]$ for an Exponential with Mean One

K	H_k	$ln(K)+\gamma$	$1+\frac{\sqrt{6}}{\pi}ln(K)$	$-ln(1-\phi^{(1/K)})$
2	1.500000	1.270868	1.540445	1.407446
4	2.083333	1.964015	2.080889	2.032872
8	2.717857	2.657163	2.621335	2.691543
16	3.380729	3.350310	3.161780	3.367299
32	4.058495	4.043457	3.702225	4.051712
64	4.743891	4.736604	4.242671	4.740482
128	5.433147	5.429751	4.783116	5.431438
256	6.124345	6.122898	5.323561	6.123490

The q^{th} quantile satisfying $G(x_{(q)}) = q$ is given as

$$x_{(K,q)} \approx ln(K) - lnln(1/q),$$

where $P(M_K \leq x_{(K,q)}) = q$ is the q^{th} quantile. The approximation for the quantiles is inaccurate for small values of K.

The EMMA method described in Sun and Peterson [2012] obtains the expected value of Y_K $Y_K = max_{i=1}^{n}(X_i)$, where X_i, $1 \leq i \leq n$ are i.i.d. random variables with CDF $F_X(x)$. It is shown next that $E[Y_K]$, the expected value of the maximum, satisfies the following equality:

$$(P(X_i < E[Y_K]))^K = [F_X(E[Y_K])]^K \approx \phi = 0.570376.$$

Three types of distributions for order statistics types I, II, III, known as Gumbel, Frechet, and Weibull, cover the asymptotic extreme distributions for individual distributions.

In the case of type I distribution $F_X(x) = exp(-exp(x-a)/b)$, a is the mode and hence the location of the peak, and b is the standard deviation and hence a measure of dispersion. Setting $x = Y_K = a + \gamma b$ yields

$$F_X(E[Y_K]) = exp\left[-exp\left(-\frac{(a+\gamma b)-a}{b}\right)\right] = exp(-exp(-\gamma)] = \phi \approx 0.570376.$$

$E[Y_K]$ can be obtained by solving the following equation:

$$E[Y_K] \approx F^{-1}(\sqrt[K]{\phi}). \tag{44}$$

In the case of the exponential distribution with rate μ,

$$(1-e^{-\mu E[Y_K]})^K = \phi \quad \Rightarrow \quad E[Y_K] = -\frac{1}{\mu} ln(1-\sqrt[K]{\phi}).$$

This is an approximation for types II and III EVDs as $b \to \infty$, for which the two CDFs converge to a type I CDF as shown in Sun and Peterson [2012].

The accuracy of the method is investigated with respect to the distributions and parameters in Table 1 in Sun and Peterson [2012], where according to Figure 4 for $K =$ 500, 1,000, 1,500, the exponential distribution introduces the lowest error and the Beta distribution the highest. Table II compares the accuracy of different approximations for the exponential distribution.

With type II and III EVDs where b is small, a different value of ϕ is appropriate. For the distribution given by Eqs. (18) and (19), more accurate results are obtained by setting $\phi = 0.46$. The case of n_i, $1 \leq i \leq m$ random variables in m heterogeneous classes

with $\sum_{i=1}^{m} n_i = n\ E(Y_n)$ can be determined as follows:

$$\prod_{i=1}^{m} F_i(E(Y_n)))^{n_i} \approx \phi.$$

4.7. Normal Distribution

The normal distribution is used as an approximation to disk service time (whose components are discussed in Appendix II) in Kim and Tantawi [1991]. The analysis requires the expected value of the maximum of K i.i.d. random variables with the normal distribution. The approximation in Arnold [1980] which uses $G(K) = \sqrt{2ln(K)}$ in Eq. (22), is

$$E[Y_K] \approx \mu_X + \sigma_X\sqrt{2ln(K)}.$$

A more accurate approximation is given in Johnson et al. [1995] as

$$E[Y_K] \approx \mu + \sigma\left[\sqrt{2ln(K)} - \frac{ln(ln(K)) - ln(4\pi) + 2\gamma}{2\sqrt{2\ln(K)}}\right]. \tag{45}$$

There is a bias associated with Eq. (45), which is corrected by subtracting $\delta(K) = 0.1727K^{-0.2750}$ from the expression in the brackets [Petzold 2000].

We also have

$$Var[Y_K] \approx \frac{1.64492\sigma^2}{2ln(K)}.$$

4.8. Maximum of Random Variables with a General Distribution

We summarize the derivation in Harrison and Zertal [2007] of an approximation for the moments of the maximum of random variables with a general distribution. Given $T = max(T_1, T_2)$, where $F_i(t)$ is the distribution of T_i and $\mathcal{F}_i^*(s)$ denotes its LST, its expected value is expressed as

$$\begin{aligned} E[T] &= E[T_1] + P(T_2 > T_1) \times E[T_2 - T_1 | T_2 > T_1] \\ &= E[T_1] + \int_0^\infty F_1(t) dF_2(t) \times E[T_2 - T_1 | T_2 > T_1]. \end{aligned}$$

Let m_i and M_i denote the first two moments with $\alpha_i = m_i^{-1}$. When T_2 is exponentially distributed,

$$E[T] = m_1 + \mathcal{F}_1^*(\alpha_2) \times E[T_2 - T_1 | T_2 > T_1].$$

For an exponentially distributed T_1, $E[T_2 - T_1 | T_2 > T_1] = M_2/(2m_2)$, which leads to

$$E[T] = m_1 + \frac{M_2\mathcal{F}_1^*(m_2^{-1})}{2m_2}.$$

Assuming that Eq. (29) is applicable to the nonexponential random variables, the k^{th} moment of the maximum of K random variables with $\underline{\alpha} = (\alpha_1, \ldots, \alpha_K)$ and $I(1, \alpha_1, M_1) = 1/\alpha_1$ is

$$I(k, \underline{\alpha}, \underline{M}) = \frac{1}{k}\sum_{i=1}^{k} I(k-1, \underline{\alpha}_{\setminus i}, \underline{M}_{\setminus i}) + \frac{1}{2}\alpha_i M_i L_{k-1}\left(\underline{\alpha_i}_{\setminus i}, \alpha_i\right). \tag{46}$$

When all random variables have the same distribution with mean $\alpha_i = \alpha$ and $M_i = M$ for $1 \le i \le N$,

$$I(k, \alpha, M) = \frac{1}{\alpha} + \frac{M\alpha}{2}(H_k - 1).$$

Simulation results to assess the accuracy of the approximation are given in Harrison and Zertal [2007], as well as in Lebrecht at Knottenbelt [2007] using Eq. (21). The following distributions are considered: $\mathcal{E}_2$, $\mathcal{E}_3$, $\mathcal{E}_4$, and Pareto-β with $\beta = 4$ and $\beta = 5$, which is given as

$$F_p(x) = 1 - \alpha(x + \gamma)^{-\beta}, \quad \beta > 2.$$

This condition is required for the first two moments to be finite, $\alpha = \gamma^\beta$ and $\gamma = \beta - 1$, so that $m = 1$ and $M = 2 + 2/(\beta - 2)$.

The error due to Eq. (46) and Eq. (21) remains small for $\mathcal{E}_2$ but increases rapidly with $K = 16$ for $\mathcal{E}_3$ and $\mathcal{E}_4$. Interestingly, both equations yield accurate results for the Pareto distribution, with the latter equation yielding more accurate results in both cases.

4.9. The Characteristic Maximum

The characteristic maximum has been used in several studies to reduce the computational cost of computing the maximum for Erlang and Coxian distributions. The discussion that follows is based on Gravey [1985].

Let $X_1, X_2, \ldots, X_K$ denote K nonnegative random variables with a distribution function $F_X(x)$. Since $Y_K = max(X_1, X_2, \ldots, X_K)$ is nonnegative [Gravey 1985],

$$E[Y_K] = \int_0^\infty P(Y_K > x)dx.$$

When X_i are i.i.d.,

$$P(Y_K > x) = 1 - [1 - P(X > x)]^K = \sum_{k=1}^{K} (-1)^{k+1} \binom{K}{k} [P(X > x)]^k.$$

In the general case when the random variables are not independent, using the fact that $P(Y_k > x) = max(1, KP(x))$ leads to the following expression for all m:

$$g(m) = m + K \int_m^\infty = KE[X] + \int_0^\infty \left[\frac{1}{K} - P(X > x) \right] dx.$$

Let m_K be the greatest lower bound of the set $\{x : P(X > x) \leq 1/K\}$. Since $F_X(x)$ is right continuous,

$$x \geq m_K : \ P(X < x) \leq 1/K \ \ x < m_K : \ P(X > x) > 1/K.$$

If X has a density, then simply

$$P(X > m_K) = \frac{1}{K}.$$

Denoting the smallest upper bound $g(m_K)$ by M_K yields

$$M_K = m_K + K \int_{m_K}^\infty P(X > x)dx. \tag{47}$$

For the exponential distribution, $1 - F_X(x) = e^{-\mu x} = 1/K$ yields $m_K = ln(K)/\mu$ and $E[Y_K] = H_K/\mu \approx (ln(K) + \gamma)/\mu$ [Trivedi 2002] so that $E[X_{max}(K)] - m_K = \gamma/\mu$.

For the $\mathcal{E}_k$ distribution in Eq. (13), m_K can be obtained by solving

$$e^{-\mu m_K} \sum_{i=0}^{k-1} \frac{(\mu m_K)^i}{i!} = \frac{1}{K}.$$

It follows that

$$M_K = \frac{k}{\mu}\left[1 + Ke^{-\mu m_K}\frac{(\mu m_K)^k}{k!}\right].$$

If X is a discrete r.v., then m_K and M_K are given as

$$m_k = \mathrm{m}in[x, P(X > x) \le 1/K],$$

$$M_K = m_K + K\sum_{k=m_K}^{\infty} P(X > k). \tag{48}$$

For the geometric distribution $P(X > k) = p^{k+1}$,

$$P(Y_K > x) = \sum_{k=1}^{K}\binom{K}{k}(-1)^{k-1}p^{k(x+1)}.$$

The expected value of the maximum is difficult to evaluate for large K:

$$E[Y_K] = \sum_{k=1}^{K}\binom{K}{k}(-1)^{k+1}\frac{p^k}{1-p^k},$$

so alternatively M_K can be used for this purpose,

$$M_K = m_K + Kp^{m_K+1}/(1-p),$$

where $m_K = min(k : p_k \le 1/K) = -ln(K)/ln(p)$.

For the Poisson distribution, $(X = k) = e^{-\theta}\theta^k/k!$ and $\overline{X} = \sigma_X = \theta$,

$$M_k = m_K + Ke^{-\theta}\sum_{k=m_K+1}^{\infty}\sum_{i=k+1}^{\infty}\theta^i/i!,$$

which can be simplified to

$$M_K = m_K(1 - KP(X > m_K)) + K\theta P(X > m_K - 1).$$

For the standard normal distribution, with mean equal to zero and standard deviation equal to one,

$$\Phi(x) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{x} e^{-t^2/2}dt,$$

and the characteristic maximum is given as follows [Lai and Robbins 1976; Kruskal and Weiss 1985]:

$$\frac{e^{-x^2}}{\sqrt{2\pi}(x + x^{-1})} < 1 - \Phi(x) < \frac{e^{-x^2}}{\sqrt{2\pi}x},$$

$$\sqrt{2ln(K) - ln(ln(K)) - 3} < m_K < \sqrt{2ln(K) - ln(ln(K))}, \quad K \ge 5.$$

For a normal distribution with mean μ_X and standard deviation σ_X [Kruskal and Weiss 1985], $m_K \approx \mu_X + \sigma_X\sqrt{2ln(K)}$.

A correction to the approximation to $m_K = F_X(K/(K+1))$ was proposed in Blom [1958]:

$$m_k = F_X^{-1}\left(\frac{K-\alpha}{K-\alpha-\beta+1}\right).$$

For example, for $K = 20$, $\alpha = 0.4886$ and $\beta = 0.3140$.

4.10. Long-Tailed Distributions and Skew

A distribution is long tailed if its complementary CDF, $F^c(t) = 1 - F(t)$, decays slower than the exponential distribution [Feldmann and Whitt 1998]. The Pareto (a, b) $F^c(t) = (1+bt)^{-a}$ and Weibull(c, a) $F^c(t) = e^{-(t/a)^c}$ are two examples of such a distribution [Trivedi 2002]. To analyze single-server queues with the Pareto and Weibull distributions, these distributions can be approximated with a mixture of I hyperexponentials:

$$\mathcal{F}^*(s) = \int_{t=0}^{\infty} e^{-st}\left[\sum_{i=1}^{I} p_i e^{-\mu_i t}\right] dt = \sum_{i=1}^{I} \frac{p_i \mu_i}{\mu_i + s},$$

where $\mu_i \neq \mu_j$ for $i \neq j$ and $\sum_i^I p_i = 1$. The least squares or expectation-maximization (EM) method [Gupta and Chen 2010] was applied to estimate p_i and μ_i for $i \geq 1$.

Distributions that are exponential in form can be used in estimating $R_K^{max}(\rho)$ via symbolic integration with Eq. (20), when an expression for the maximum of the long-tailed distributions is not available. Phase approximations to general service times, which have been considered in Malhotra and Reibman [1993] are discussed in Sahner et al. [1996]. The Weibull and lognormal distribution have been determined to be a better match to the time to disk failure than the exponential [Schroeder and Gibson 2007].

Long tails for file length distributions have been observed in Crovella and Bestavros [1996]. Such files when transmitted affect Internet performance, resulting in heavy-tailed response times [Scharf 2005]. The following Pareto distribution has been observed through measurement:

$$F_R(t) = 1 - (k/t)^{\alpha}, \quad t > k, \quad F_R(t) = 0, t \leq k.$$

The j^{th} moment of the i^{th} order statistic of K such random variables according to Johnson et al. [1995] is

$$m_{i.j} = \frac{\Gamma(K+1)\Gamma(K-j+1-i/\alpha)}{\Gamma(K+1-i/\alpha)\Gamma(K-j+1)} k^i,$$

where $\Gamma(x+1) = x! \approx (\frac{x}{e})^x \sqrt{2\pi x}$ for large x. This is known as Stirlings first approximation, and substituting e^{-x} with $e^{-x+1/12}$ provides a more accurate expression, which as an upper bound. Long delays can be dealt with by (1) allowing incomplete results, (2) restricting to fast servers, and (3) timeouts.

The Zipf law deals with the frequency of words, which can be expressed as $p_i = c/i, 1 \leq i \leq N$ with $c = 1/H_N$. The "80–20" rule states that 80% of transactions access 20% of the database. This rule applies in a fractal fashion—for example, 80% of 80% or 64% of transactions access 20% of 20% or 4% of the database. An example of Zipf law approximately satisfying this rule is given as [Knuth 1997]

$$p_n = c/n^{(1-\theta)}, 1 \leq n \leq N.$$

where $\theta = ln(0.80)/ln(0.20) = 0.1386$ and $c = 1/H_N^{(1-\theta)}$. Zipf distribution is related to the Pareto distribution and power laws [Adamic 2010].

Early query optimizers for relational databases were based on the unrealistic assumption that the values of attributes of columns in tables are uniformly distributed.

A highly skewed Zipf distribution for column values of relational tables was observed in Lynch [1988]. The inefficiency of plans for processing relational queries based was fixed by maintaining histograms of column values in relational DBMSs, as this leads to improved query optimization plans [Ramakrishnan and Gehrke 2003]. The first implementation of histograms in the context of Berkeley's Ingres relational database was carried out under the author's supervision [Kooi 1980].

The notion that skew results in degraded performance in parallel processing of relational operations was illustrated in the context of the Gamma parallel database machine in Schneider and DeWitt [1989]. Four parallel join algorithms were evaluated: (1) Grace join, (2) Hybrid hash-join, (3) Simple hash-join, and (4) Sort-Merge. Hashing algorithms distribute data of the two relations to be joined over the nodes of the machine. The Hybrid hash-join algorithm is found to be superior except when the join attributes of the inner relation are skewed, in which case the Sort-Merge join exhibits robust performance. Sampling can be used to build instant histograms for dealing with skew encountered in join processing.

The execution time of parallel hash join and semijoin are derived in Seetha-Lakshmi and Yu [1990], and their effectiveness explored with respect to the skew in the distribution of join attribute values. This results in highly variable processing times in parallel database machine, with some processors becoming a bottleneck whereas others are underutilized. An analytic performance model of concurrent processing parallel Hybrid hash and Grace joins to investigate the effect of skew is developed in Mourad et al. [1994].

Heavy-tailed characteristics have been observed in MapReduce clusters [Tan et al. 2012]. Facebook's datacenter has 20,000 machines that processed 25,000 MapReduce jobs per day in 2010. A MapReduce has a Map and Reduce phase [Dean and Ghemawat 2010]. Mapper tasks search through large volumes of data to produce key-value pairs, which are then used by Reducer tasks to combine information according to key values [Li et al. 2014]. Each phase is processed by multiple affiliated tasks. The tasks of the Map phase process fixed-sized chunks of input data to generate intermediate results in the form of key-value pairs. Reduce tasks start only after these key-value pairs are available, sorting and merging according to the key value. Tasks are initiated by a job tracker, which keeps track of task progress at slave nodes. The Fair Scheduler launches Reduce tasks as soon as Map tasks produce sufficient intermediate results. This is intended to maximize the overlap between the two sets of tasks. These reduce tasks may have data that needs to be processed by all Reducers of the same job. The Fair Scheduler has been known to be susceptible to starvation, since Mappers are run independently and complete quickly, whereas Reducers requiring the output from all Mappers are long running.

Analytical models are proposed for the default first-in, first-out (FIFO) and the Fair and Coupling schedulers proposed in Tan et al. [2012]. The problem would be much simpler if there were no Reduce tasks and a large job could cause long delays for short jobs. In fact, 91% of MapReduce jobs are Mapper-only, whereas only 7% of jobs are Reduce-mostly. As far as Fair scheduling is concerned, the first job is assigned all 28 Mappers, until the second job arrives, after which they share the Mappers equally until the third job arrives. In effect, this a PS discipline. Given seven slots for Reduce tasks, second and third jobs cannot share until the first job completes.

The Coupling scheduler proposed in Tan et al. [2012] launches Reducers according to the progress of Mappers. The Map phase in this study is modeled as an M/G/1 queue [Kleinrock 1975] (see Appendix II). Jobs arrive according to a Poisson distribution with rate λ. File sizes follow a Pareto distribution, and job service times are heavy tailed, with mean service time $E[B]$ and server utilization $\rho = \lambda E[B] < 1$. The r.v. B is slowly

varying:

$$P(B > x) = \frac{\ell(x)}{x^{\alpha}}, x \geq 0, \alpha \geq 0, \lim_{x\to\infty} \frac{\ell(kx)}{\ell(x)} = 1, k > 0.$$

FIFO scheduling is modeled by two tandem queues, whereas for Fair and Coupling scheduling, the Mappers join the Map queue, which is PS, and Reducers the Reduce queue. The analysis of the scheduling methods is quite complex and is not repeated here. An order of magnitude gain by the Coupling scheduler is possible under certain conditions, although a constant gain is possible in general. The results of a trace-driven simulation study with 12 jobs submitted at fixed intervals are given in Table 1. Jobs are characterized with the number of Mappers and Reducers. The starvation time of a job is defined as the average of durations from the completion time of a Mapper and starting time of Reducers. The simulation yields the starvation times $W_{Coupling}$ and W_{Fair}. The Coupling scheduler expedites the processing time of 10 jobs out of 12 according to Figure 3.

5. REPLICATED DATA

Data replication at multiple geographically distributed sites is a common form of distributed databases. Data is initially stored in triplicate for increased availability but is later reverted to a more space-efficient erasure coding format such as RAID5 afterward [Thomasian and Blaum 2009]. In Section 5.1, application of the MGM is discussed in the context of reading and writing of replicated data in mirrored disks. The MGM analysis of replicated databases, where the number of replicas is greater than two, is discussed in Section 5.2. In the case of the readers and writers problem, M readers are processed concurrently, whereas writers are processed singly. The TFE policy, which improves performance with respect to the FCFS policy, is described and analyzed in this context in Section 5.3.

5.1. Updating Mirrored Disk Systems

We provide a rather detailed analysis of mirrored disk scheduling to demonstrate the complexity of scheduling of read and update requests in a relatively simple context. Requests are processed in FCFS order, and update requests need to wait until both disks are available.

The processing of read and update requests in mirrored disks lends itself to a variety of single queue (SQ)and multiple queue (MQ) scheduling policies [Towsley et al. 1990b]. SQ policies are classified as primary/secondary (PSSQ) and equitable SQ (ESQ) policies.

With PSSQ, one disk is termed *primary* and the second disk *secondary*. PSSQ policies are classified as serial PSSQ (S-PSSQ) and concurrent PSSQ (C-PSSQ) policies. S-PSSQ requires both disks to be available for a read or update request to be processed. C-PSSQ allows concurrency by allowing reads to be processed when the primary disk is idle. ESQ does not make a distinction between primary and secondary disks.

Concurrent read ESQ (CR-ESQ) allows concurrent reads, but no concurrency between reads and updates. Concurrent read update ESQ (CRU-ESQ) allows read concurrency with updates. Minimum read ESQ (MR-ESQ) is similar to S-PSSQ in that each request is supposed to wait until both disks are idle. Read requests are initiated at both disks, and when the processing at one disk is completed, the read request at the other disk is aborted.

There are two variations to MQ: distributed MQ (DMQ) and common MQ (CMQ). DMQ maintains separate disk queues. There are three variations for processing reads: (1) a read request is routed uniformly, (2) (resp. 3) read requests are sent to both disks, and as soon as one of the reads completes, the other read request is aborted (resp. not aborted). CMQ maintains all requests in a common queue and issues them as soon as

one disk becomes available. An update may be held at a disk's temporary queue until the disk becomes available.

An analysis of the C-PSSQ policy using the MGM [Neuts 1981; Nelson 1995; Latouche and Ramaswami 1999; Bolch et al. 2006] is described later. Reads are processed at the primary disk, whereas updates are started when both disks are available.

The MGM, similarly to birth–death processes [Kleinrock 1975], accommodates an infinite number of requests but has a state vector instead of a single state for varying number of requests in the system. The state of the system is specified as $[N(t), S(t)]$, where $N(t)$ is the number of requests in the waiting queue, and $S(t)$ is the execution state at the disks as follows:

—0: both disks are idle
—R: primary disk is reading
—WR: primary disk reading and secondary disk writing
—W_p: if only the primary disk is writing
—W_s: if only the secondary disk is writing
—$2W$: if both disks are writing

Note that WR is the only state with two requests in service. We are interested in the stationary probabilities of the CTMC $(N(t), S(t))$, which are specified as $p(i, j) = Prob(N = i, S = j)$, where $i \geq 0$ and $j \in \{0, R, WR, W_p, W_s, 2W\}$. The stationary probability vector $Y = (y(0), y(1), \ldots, y(i), \ldots)$ satisfies $YQ = 0$, where Q is the infinitesimal generator [Kleinrock 1975]:

$$Q = \begin{pmatrix} B_1 & B_0 & 0 & 0 & 0 & . \\ B_2 & A_1 & A_0 & 0 & 0 & . \\ 0 & A_2 & A_1 & A_0 & 0 & . \\ 0 & 0 & A_2 & A_1 & A_0 & . \\ . & . & . & . & . & . \end{pmatrix}.$$

A key step in applying the MGM is identifying Q's constituent matrices: $A_0 = \lambda \times I_{5\times 5}$, where $I_{n\times n}$ denotes the $n \times n$ identity matrix. Matrices for B_0, B_1, B_2, A_1, and A_2 are given in Towsley et al. [1990b].

The stability condition is $\pi A_2 e > \pi A_0 e$, where $\pi A = 0$, $\pi e = 1$ and $A = A_0 + A_1 + A_2$. e is a column vector with all ones, and e_j is a column vector of all zeroes, except a one in the j^{th} position.

It is shown in Neuts [1981] that Y is given as

$$y(i) = y(1)R^{i-1}, \quad i > 1. \tag{49}$$

R is the minimal solution (as explained in Section 9.1 in Nelson [1995]) of the following quadratic equation

$$A_0 + RA_1 + R^2A_2 = 0, \tag{50}$$

which can be solved using the initialization $R(0) = 0$ and the following iteration:

$$R(n+1) = -A_0A_1^{-1} - R^2(n)A_2A_1^{-1}, \ n \geq 0.$$

Using $B = \left(\begin{smallmatrix} B_1 & B_0 \\ B_2 & A_1 + RA_2 \end{smallmatrix}\right)$, Y can be obtained by solving

$$(y(0), y(1))B = 0, \quad y(0)e + y(1)[I - R]^{-1}e = 1.$$

The stability condition is

$$\lambda < \frac{(2 - p_r)\mu}{3 - 3p_r + p_r^2}.$$

Table III. Summary of Configurations for Replicated Databases (Based on Table I in Nelson and Iyer [1985])

Paper#	Arrival	Priority	Scheduling	Writing
Coffman et al. [1981a]	Saturated	Preemptive	Synch.	Sequential/Nonsynch.
Coffman et al. [1981b]	Saturated	Preemptive	Nonsynch.	Parallel
Baccelli and Coffman [1982]	Poisson	Preemptive	Nonsynch.	Sequential
Nelson and Iyer [1985]	Poisson	Nonpreempt.	Synch.	Sequential Parallel
Nelson and Iyer [1985]	Poisson	Nonpreempt.	Nonsynch.	Parallel

For $p_r = 0$, we have $\lambda_{max} = \mu/H_2$. The mean queue length is

$$E[N] = y(1)[I - R]^{-2}e - y(1)[I - R]^{-1}e_2.$$

This discussion illustrates some of the complexities associated with analyzing queueing systems using the MGM. The article concludes with graphs depicting the maximum throughputs versus p_r and response times versus λ to compare the performance of various scheduling policies.

5.2. Replicated Databases

Replication allows higher access efficiency by distributing the read load over K database copies. The read-one, write-all paradigm simply reads one copy and updates all K copies to maintain consistency. More sophisticated methods are required to deal with node failures and network partitions, which are beyond the scope of this discussion [Ceri et al. 1991].

Reads and updates are handled in FCFS order, but multiple reads can be issued simultaneously. Databases are updated *in parallel* or *sequentially*. In the *synchronous* case, the database can only be read when no database copy is being updated, whereas in the *nonsynchronous* case, any available database copy can be read. In the former case, all F/J requests to update the replicas should be completed before a read can be issued. Updates are issued only after all reads are completed.

Saturated read queues, as well as Poisson arrivals for reads, have been considered in replicated databases. The read throughput in the former case is the performance metric of interest. The assumption of saturated read queues is relieved in Baccelli and Coffman [1982] and Nelson and Iyer [1985]. Modeling assumptions in earlier studies of replicated databases are summarized in Table 1 in Nelson and Iyer [1985], which is reproduced in Table III. The first three studies postulate a preemptive priority for writes versus reads, whereas a nonpreemptive policy is considered in Nelson and Iyer [1985].

The analysis is simplified by assuming that both reads and writes, which are served in FCFS order, have the same service rate. Reads constitute a fraction r of all requests. The analysis of the CTMC, which is carried out using the MGM method, yields the maximum system throughput:

$$\lambda^{max}_{non\_synch} = \frac{\mu}{1-r}\left[\sum_{i=1}^{k-1}\frac{1}{i} + \frac{1}{k(1-r)}\right]^{-1}. \tag{51}$$

The maximum throughput for the synchronous policy with parallel updates is

$$\lambda^{max}_{synch} = \frac{\mu}{1-r}\left[\sum_{i=1}^{k}\frac{1}{i} + \sum_{i=1}^{k-1}\frac{r^i}{i} + \frac{r^k}{k(1-r)}\right]^{-1}. \tag{52}$$

Nonsynchronous scheduling outperforms synchronous scheduling because $\lambda_{non\_synch} > \lambda_{synch}$. It is easy to verify that in both cases, the maximum throughput for $r = 1$ equals $k\mu$, and for $r = 0$ it is equal to μ/H_k.

The number of replicas is optimized in Nelson and Iyer [1985] to maximize the throughput. In the synchronous case, if $r < 0.5$, then the optimal number of replicas is $k_{opt} = 1$; otherwise, it is the closest integer to $1/(1-r)$. In the nonsynchronous case, $k_{opt} = 1$ if $r < 1/g$, where $g = (1+\sqrt{5})/2 = 1.618$ is the *golden ratio*; otherwise, $k_{opt} = 1 + r^{k_{opt}}/(1-r)$. In Baccelli and Coffman [1982], the degree of replication is optimized to minimize the read delay or to maximize the attainable read throughput.

5.3. Readers and Writers Synchronization

The readers/writers paradigm allows concurrent processing of readers, but only one writer can be processed at a time. FCFS processing of readers and writers may result in a lower degree of concurrency—for example, in the worst case, arrivals alternate between readers and writers so that readers cannot be processed concurrently.

An infinite backlog of readers processed with a degree of concurrency M and writer arrivals with rate λ is considered in Courcoubetis and Reimann [1987]. Markovian decision processes (MDPs) [Howard 1960] are used to show that TFE policy optimizes rewards for (1) reader and writer completions, and (2) reader completions and writer waiting times.

According to TFE scheduling policy, when the number of enqueued writers exceeds a threshold K, then the processing of readers in progress is completed but no new readers are started. Once the processing of writers is started, the processing of writers continues until the writer queue is emptied, at which point the processing of M readers are started. Completed readers are immediately replaced by a new reader as long as the number of enqueued writers is less than K.

The two performance metrics of interest are the mean response time for writers and reader throughput. These are obtained in Thomasian and Nicola [1993] using the vacationing server model (VSM) [Takagi 1991]. Vacations correspond to reader processing time, whereas busy periods correspond to writer processing time. For the analysis to be tractable, the readers that are processed concurrently should follow an exponential distribution with rate ν, whereas the writers may have a general service time with moments denoted by $\overline{x^k}$. The fraction of time spent in serving writers is then $\rho_w = \lambda\overline{x}$. The mean number of writers in the system is the sum of writers in a regular M/G/1 queue,

$$\overline{N}_w^{M/G/1} = \rho_w + \frac{\lambda^2\overline{x^2}}{2(1-\rho_w)}, \tag{53}$$

and the extra writers accumulated during the vacation period awaiting the completion of readers. A corrected version of Eq. (3.8) in Thomasian and Nicola [1993] for the mean number of writers is

$$\overline{N}_w^{VSM} = \overline{N}_w^{M/G/1} + \frac{\alpha^{(2)}(1)}{2\alpha^{(1)}(1)}, \tag{54}$$

where $\alpha(z)$ is given as

$$\alpha(z) = M\nu \sum_{m=0}^{M-1} (-1)^m \binom{M-1}{m} \frac{z^K}{(m+1)\nu + \lambda(1-z)}.$$

The throughput for readers is

$$\gamma(K) = (1 - \rho_w) M\nu \frac{K + \frac{\lambda}{\nu}}{K + \frac{\lambda}{\nu} H_M}. \tag{55}$$

$K \to \infty$ results in the least interruption in reader processing and maximizing reader throughout: $\gamma_{max} = (1 - \rho_w)M\nu$. This results in a saturated queue for writers.

To determine the maximum throughput of the readers and writers queueing system with the FCFS policy, let Λ_{FCFS} denote the total arrival rate. Given the fraction of writers $f_w = 1 - f_r$, then $\lambda = \Lambda_{FCFS} f_w$ and $\gamma = \Lambda_{FCFS} f_r$. It is shown in Thomasian and Nicola [1993] that the maximum throughput with the FCFS policy is

$$\Lambda_{FCFS}^{max} = \left[\frac{f_w}{\mu} + \frac{f_w}{\nu} \sum_{i=1}^{M} \frac{f_r^i}{i} + \frac{f_r^{M+1}}{M\nu} \right]^{-1}. \tag{56}$$

The maximum throughput with a TFE policy with a threshold K: T_K is the positive root of the following quadratic equation,

$$a(b + cH_M)T_K^2 + (Kb + Kc - a)T_K - K = 0,$$

where $a = f_w/\nu$, $b = f_w/\mu$, and $c = f_r/(M\nu)$. It can be shown that T_K is an increasing function of K and that even $T_1 > \Lambda_{FCFS}^{max}$. This is because TFE is more efficient than FCFS in that it completes the processing of writers before switching back to readers.

Assuming that both reader and writer service times are exponentially distributed, we analyzed the corresponding CTMC, which is infinite in one dimension. The two analyses lead to different algebraic expressions for γ, which yield numerically equivalent results.

Thresholds for both readers and writers are proposed for a system with both reader and writer arrivals in Thomasian [1998a]. The balancing of the response times for readers versus writers is attained by utilizing an additional threshold for enqueued readers.

6. RELATED QUEUEING SYSTEMS

Parallel processing and bulk arrival systems are discussed in Section 6.1. We next discuss the issue of subtask dispersion in SM and F/J queueing systems following the discussion in Tsimashenka and Knottenbelt [2013b]. Jobs of the TSM require a variable number of servers to be acquired concurrently according to the job class, and all servers are released together as discussed in Section 6.3. The ISM requires a variable number of servers with one server per task to be acquired concurrently. Tasks complete their processing and release servers asynchronously as discussed in Section 6.4. Transaction processing systems where transactions are blocked due to lock conflicts is discussed in Section 6.5.

All three systems have a similarity to F/J queueing systems in that multiple servers are required for processing jobs. The tasks of F/J requests should be served at specific servers, but this is not so for bulk arrival systems, TSM, and ISM. With the same arrival rate, number of requested servers, and service times, bulk arrival systems are expected to provide the lowest mean response time, followed by F/J systems, ISM, and TSM.

6.1. Parallel Processing and Bulk Arrival Systems

Four parallel processing systems with K servers are considered in Nelson et al. [1988]: (1) distributed splitting (DS), (2) centralized splitting (CS), (3) distributed no splitting (D/NS), and (4) centralized no splitting (C/NS).

DS corresponds to the F/J queueing system, whereas CS is a multiserver queueing system with bulk arrivals. CS is less restrictive than DS in that the same server may serve multiple tasks from the same batch arrival. This requires all requests to be held in a central queue. Given Poisson arrivals and exponential service times, the $M^X/M/K$ bulk arrival system is analyzed in Nelson et al. [1988] utilizing the z-transform method [Kleinrock 1975] (see Appendix II). An analysis of batch arrivals in a multiserver queueing system under Markovian assumptions was also analyzed in Coffman [1967].

D/NS routes jobs consisting of a sequence of K tasks uniformly to K servers. Assuming Poisson arrivals and exponential service times, this is an $M/E_K/1$ queue whose mean response time,

$$R_{M/E_K/1}(\rho) = \left[K - (K-1)\frac{\rho}{2}\right] R(\rho),$$

where $R(\rho) = (\mu - \lambda)^{-1}$ and $\rho = \lambda/\mu$, follows from Eq. (75) arrival rate λ/K and $\overline{x} = K/\mu$ and $c_X^2 = 1/K$. Numerical examples show that the splitting case outperforms the no splitting case, which is due the fact that there is higher probability of having idle processors in the no splitting case.

C/NS corresponds to an M/G/m queue, which has been analyzed using the diffusion approximation method in Yao [1985]. Assuming that the number of exponentially distributed tasks is geometrically distributed, their sum is exponentially distributed and the resulting M/M/m queueing system is easy to analyze [Kleinrock 1975; Allen 1990; Trivedi 2002].

The mean waiting time for a GI/G/m queueing system can be approximated as follows [Allen 1990]:

$$W \approx \frac{P_q/\mu}{m(1-\rho)}\left[\frac{c_a^2 + c_b^2}{2}\right].$$

P_q is the probability that an arriving request has to queue—that is, all m servers in an M/M/m queue are busy: $P_q = \sum_{k=m}^{\infty} p_k$, where p_k are the steady-state probabilities. c_a^2 and c_b^2 are the squared CVs for the interarrival time and service times. Setting $c_a^2 = 1$ provides a good approximation for M/G/m.

The splitting model yields a better response time than the no splitting model, as it allows a more balanced utilization of servers. Centralized systems are preferable to decentralized systems, because the latter may result in queueing at some servers, whereas other servers are idle. The relative performance of D/S versus C/NS depends on the value of ρ, because for high values of ρ, D/S encounters high synchronization delays.

Bulk arrivals with general service times are considered in Lebrecht et al. [2009], hence an M^X/G/1 queue [Chaudhry and Templeton 1983]. Of interest in this study is the time to complete all requests rather than a random request in a batch.

Considered in Ghodsi and Kant [1991] is a sequential task that spawns K tasks on $M(\geq K)$ processors. Only one spawned task needs to complete, after which remaining tasks are aborted. F/J tasks are preceded by a sequential task and are spawned after this initial task is completed. The task arrival process is Poisson with rate λ, and the analysis is simplified by assuming that all tasks have exponentially distributed service rate μ. Given that at least two tasks need to be completed, the minimum mean service time per job is $2/\mu$ and the utilization factor is $\rho = 2\lambda/(M\mu)$. The analysis is based on applying the z-transform to the two-dimensional Markov chain model to obtain the mean number in system, from which the mean response time R follows. Priority is given to completing the tasks of a job rather than activating the header task of the next job, which may not be the best policy. For various values of ρ, it is shown that R decreases as K increases ($1 \leq K \leq M$). Note that the mean of the minimum of k exponentials is

$\overline{X}_K^{min}] = 1/(K\mu)$, which decreases slowly as K increases. This effect leads to the gradual reduction in R, which is plotted versus K for different values of ρ.

6.2. Scheduling for Minimizing Task Dispersion

Task dispersion is the time difference between the completion of the last and first subtask of an F/J request. This is especially an issue when subtasks have variable service times. The strategy proposed in Tsimashenka and Knottenbelt [2013b] is to reduce the mean response time of tasks while minimizing the dispersion of its subtasks by delaying the activation of certain subtasks that are at the head of the parallel service queues. More specifically,

> In particular, at every time instant at which a hitherto-unserviced subtask **S** reaches the front of a parallel queue, we take the following control actions:
>
> 1. If any of the siblings of **S** have already completed service then the best mean subtask dispersion and task response time with respect to **S**s task are simultaneously achieved by immediately beginning service of **S** and also of any of its siblings that are at the front of their parallel queue.
> 2. Otherwise all siblings are still present in the parallel queues and we apply delays to **S** and its siblings that are at the front of their parallel queues and which have not yet entered service. We choose appropriate delays (which may include zero delays) by observing that, from the point of view of subtask S and its siblings, the system at that instant is equivalent to an N-server split-merge system in which parallel server i has service time distribution Erlang$(q_i(t) + 1, \mu_i)$, where $q_i(t)$ is a number of subtasks in front of **S** or its sibling subtask in parallel queue i at time t. The $q_i(t)$ form vector $\underline{q(t)} = (q_1(t), q_2(t), \ldots, q_N(t))$. We can then exploit the optimization method we developed in our previous work [Tsimashenka and Knottenbelt 2011, 2013a; Tsimashenka et al. 2012] to determine a vector of (near)-optimal deterministic subtask delays $\underline{d} = (d_1, d_2, \ldots, d_N)$. Here element d_i denotes the deterministic delay, which should notionally be applied to parallel server i. In fact, we only adopt the delays corresponding to **S** and its siblings that are at the front of their parallel queues and which have not yet entered service (note this may involve overwriting a currently pending delay).

The objective function of the optimization is mean subtask dispersion, computed as the difference between the maximum and minimum heterogeneous order statistics of the SM-equivalent system with delays. Referring to Eqs. (24) and (25) and using the linearity property of the expectation operator over dependent variables,

$$\begin{aligned} E[D_d] &= E\left(E\left[X^d_{(N)} - X^d_{(1)}\right]\right) = \left(E\left[X^d_{(N)}\right] - E\left[X^d_{(1)}\right]\right) \\ &= \int_0^\infty \left(\prod_{i=1}^{N}(1 - F_i(x - d_i))\right) - \left(\prod_{i=1}^{N} F_i(x - d_i)\right) dx, \end{aligned}$$

where $F_i(x - d_i)$ is a shifted Erlang$(q_i(t) + 1, \mu_i)$ CDF (see Eq. (36) in Section 4.3). When optimizing, we solve for $d_{min} = argmin E[D_d]$. Numerical integration is used to evaluate the objective function. Results for five different SM and F/J queueing systems are provided in this study.

An option to reduce subtask dispersion in the presence of interfering requests is to process the subtasks of an F/J request at a higher priority than independent requests. This results in reduced mean task dispersion at the cost of higher mean response time

for interfering requests. In the context of a RAID5 disk array with $K+1$ disks, one of which is failed, the mean overall response time of $R_o = (K \times R + R_K^{F/J}/(K+1)$ will be higher than the case when all requests are processed at the same priority. A compromise is to process all subtasks at the same priority but to increase the priority of F/J subtasks when a certain fraction of subtasks complete or reach the head of the queue.

6.3. Team Service Model

According to the TSM, jobs require a variable number of processors to commence their execution. All servers requested by a job are acquired and released at the same time [Thomasian and Avizienis 1975; Thomasian 1978a, 1978b]. When the number of idle processors available for scheduling is less than the number of processors required by enqueued jobs, there is forced server idleness, especially if a strict FCFS policy is followed. It is shown later that full server utilization may not be achievable even if a preemptive policy is followed, so setting $\rho = 1$ may yield an unattainable arrival rate. There is the possibility of varying the degree of job concurrency—for example, a computation requiring four processors can be processed as two (resp. four) jobs requiring two (resp. one) processors. There is no forced idleness when single processors are requested, but there is extra job setup overhead that is not considered in this study. In fact, there are efficiencies in processing multiple tasks of a job, such as obviating the need to store temporary results in main memory [Thomasian 1978a], which are beyond the scope of this discussion.

Consider a TSM with Poisson arrivals with rate Λ, s servers, K job classes with frequencies f_k with $\sum_{k=1}^{K} f_k = 1$. Jobs in class k denoted by $\mathcal{C}_k$ have frequencies f_k, so their arrival rate is $\lambda_k = \Lambda f_k$, mean service time $\overline{x}_k = 1/\mu_k$, and require r_k, $1 \le k \le K$ servers. Setting $r_k = k$ and $K = s$ simplifies the discussion. A TSM can be specified succinctly as

$$\mathcal{T} = \{s, K, \Lambda, \underline{f}, \underline{r}, \underline{\overline{x}}\}.$$

The server utilizations are

$$\rho = \frac{\lambda}{s} \sum_{K=1}^{K} f_k r_k \overline{x}_k.$$

The apparent maximum arrival rate for $\rho = 1$ is

$$\Lambda^{max} = s \left[\sum_{k=1}^{K} f_k r_k \overline{x}_k \right]^{-1}, \tag{57}$$

which is not attainable for some TSMs even by the best preemptive scheduling policies, as explained next.

A TSM with $s = 2$ servers (processors) and $K = 2$ job classes is analyzed in Omahen and Schrage [1972]. $\mathcal{C}_1$ and $\mathcal{C}_2$ jobs require one and two servers for their execution, respectively. In fact, a multiresource queueing system with two processors and two memory units is considered in this study, where jobs require a single processor and one or two memory units. We simplify the discussion by postulating a single resource system with two servers and two job classes that request one or two servers, since this model entails in the same degree of concurrency. The execution time of $\mathcal{C}_1$ jobs, which are processed concurrently, should be exponential, whereas singly processed $\mathcal{C}_2$ jobs are allowed a general service time distribution. Denoting the service times as

$\overline{x}_k = 1/\mu_k, k = 1, 2$, the utilization factor in this case is

$$\rho = \lambda \left[\frac{f_1}{2\mu_1} + \frac{f_2}{\mu_2} \right] = \lambda \frac{f_1\mu_2 + 2f_2\mu_1}{2\mu_1\mu_2}.$$

Λ^{max} corresponding to $\rho = 1$ is then

$$\Lambda^{max} = \frac{2\mu_1\mu_2}{f_1\mu_2 + 2f_2\mu_1}. \tag{58}$$

Λ^{max} can be attained by the last two out of five queueing disciplines considered and analyzed in Omahen and Schrage [1972]: (1) FCFS; (2) nonpreemptive $\mathcal{C}_1$ static priority; (3) preemptive $\mathcal{C}_1$ static priority; (4) preemptive $\mathcal{C}_1$ conditional priority, requiring two jobs to start execution; and (5) preemptive $\mathcal{C}_2$ static priority. Disciplines (2) and (3) both use the following priority ordering for activating jobs: $(2\mathcal{C}_1, \mathcal{C}_1, \mathcal{C}_2)$. Disciplines (4) and (5), which are preemptive, prioritize processing according job combinations $(2\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_1)$ and $(\mathcal{C}_2, 2\mathcal{C}_1, \mathcal{C}_1)$, respectively. Note that these are full-capacity disciplines in that they attain full server utilization.

FCFS scheduling is analyzed using Takacs' integrodifferential equation (see Section 5.12 in Kleinrock [1975]), whereas preemptive methods are analyzed using the delay-cycle analysis method (see Section 3.3 in Kleinrock [1976] and Conway et al. [1967]).

The maximum throughput attainable by FCFS policy is

$$\lambda_1^{max} = \frac{2\mu_1\mu_2}{f_1^2\mu_2 + 2f_2^2\mu_1 + 2f_1f_2(\mu_1 + \mu_2)} < \Lambda^{max}.$$

It is easy to see that $\lambda_1^{max} < \Lambda^{max}$—that is, $\rho = 1$ cannot be achieved by this policy. Disciplines (2) and (3) have a lower maximum throughput than Λ^{max} as well, whereas the maximum throughput for disciplines (4) and (5) equals Λ_{max}, thus they are full capacity as stated earlier. To summarize:

$$\Lambda^{max} = \lambda_4^{max} = \lambda_5^{max} \geq \lambda_2^{max} \geq \lambda_3^{max} \geq \lambda_1^{max}.$$

The capacity bound for multiresource systems, such as a system consisting of a CPU and memory units, is defined in Omahen [1977] as the smallest input rate guaranteed to saturate the system. The system should spend specific fractions of time in processing various job combinations to achieve the capacity hound.

A linear programming (LP) formulation [Dantzig 1998] to determine the capacity bound in a single resource system is similarly used in Thomasian [1978a, 1978b]. It is shown that full server utilization is not achievable for certain compositions of requests, and this is only possible when all states of the solution of the LP formulation utilize all s servers.

To gain insight into the maximum throughput attainable by the TSM, we consider an example:

$$s = K = 4,\ f_k = 1/K,\ r_k = k,\ \overline{x}_k = 1,\ 1 \leq k \leq K.$$

There are 11 execution states {1, 2, (1,1), 3, (1,2), (1,1,1), 4, (1,3), (2,2), 2,1,1 (1,1,1,1)}.

The LP formulation sets the mean number of jobs in execution in different classes $1 \leq k \leq K$ and execution states $1 \leq i \leq I$ equal to the production of their arrival rate and mean service time:

$$-f_i\overline{x}_i\lambda + \sum_{k=1}^{K} n_{i,k}p_k = 0, \quad 1 \leq i \leq I, \qquad \sum_{k=1}^{K} p_k = 1.$$

It is shown that full server utilization and the maximum throughput are attained when the states appearing in the LP solution are full capacity (i.e., utilize all s servers).

This point is illustrated by the following examples. We vary f_k such that the mean number of required servers remains equal to what is attained by uniform frequencies: $\overline{r} = \sum_{k=1}^{K} f_k r_k = 2.5$. The maximum system throughput with $\overline{x}_k = 1,\ 1 \leq k \leq K$ is $\Lambda^{max} = s/[\sum_{k=1}^{K} r_k \overline{x}_k] = 1.6$ using Eq. (57).

The frequency vector $\underline{f} = (\epsilon, 0.5 - \epsilon, 0.5 - \epsilon, \epsilon)$, with $\epsilon \to 0$, also requires $\overline{r} = 2.5$ servers. The LP formulation yields $\lambda^{max} = 1.33333 < \Lambda_{max}$ and execution state probabilities: $P(2\mathcal{C}_2) = 1/3$ and $P(\mathcal{C}_3) = 2/3$. Λ^{max} is attained when all job classes have the same frequency in the arrival stream (i.e., $f_k = 0.25, 1 \leq k \leq 4$), and the associated state probabilities are $P(2\mathcal{C}_2) = 0.2$, $P(\mathcal{C}_1, \mathcal{C}_3) = 0.4$, $P(\mathcal{C}_4) = 0.4$. Λ^{max} can be attained since all servers are utilized in the three states. All $\mathcal{C}_1$ and $\mathcal{C}_3$ jobs are processed together, so that $P(4\mathcal{C}_1) = 0$.

The arithmetic processors (APs) postulated in Thomasian [1978a] have two inputs for vector data, and the output can be transmitted directly to one or more APs via an interconnection network (IN). APs can process one operation per cycle. The fact that the IN provides limited interconnectivity restricts the allocation of otherwise available APs. There is also the issue of limited connectivity of address generators (AGs), which access data for APs. There is a trade-off between the complexity of the IN and utilization of APs and AGs. Allocating multiple APs to a vector computation such as $D \leftarrow A \times B + C$ (two in this case) obviates the need to save temporary results in main memory and preserves memory bandwidth.

Response times in TSM performance can be improved with respect to a strict FCFS policy by using lookahead to allocate combinations of jobs with full server utilization. Preemptive policies are required to allocate job combinations, which fully utilize all servers. Overall response time can be improved by prioritizing the allocation of combinations of smaller jobs based on the solution of the LP formulation [Thomasian and Avizienis 1975].

The assumption that the service times for both job classes are exponentially distributed allows an analysis based on solving the corresponding CTMC. The z-transform method is applied to the analysis of the CTMC, whose states are defined by (i, j), where i denotes the number of jobs in the systems, and $j \in \{0, 1, 2, 3\}$ is the execution state. In the state probability vector, $p_{0,0}$ is the empty state, $p_{i,1}$ and $p_{i,2}$ are states with one and two $\mathcal{C}_1$ jobs in execution, and $p_{i,3}$ is the state with one $\mathcal{C}_2$ job in execution. The three execution states are repeated for different job populations. Given Poisson arrivals with rate Λ and a fraction f_k for $\mathcal{C}_k$ requests and service rates μ_k, we have the following state equilibrium equations [Thomasian 1978a]:

$$\Lambda p_{0,0} = \mu_1 p_{1,1} + \mu_2 p_{1,3};$$

$$(\Lambda + \mu_1) p_{(1,1)} = \Lambda f_1 p_{0,0} + 2\mu_1 p_{2,2} + \mu_2 f_1 p_{2,3};$$

$$(\Lambda + \mu_2) p_{1,3} = \Lambda f_2 p_{0,0} + \mu_1 p_{2,1} + \mu_2 f_2 p_{2,3};$$

$$(\Lambda + \mu_1) p_{2,1} = \Lambda f_2 p_{1,1} + 2\mu_1 f_2 p_{3,2} + \mu_2 f_1 f_2 p_{3,3};$$

$$(\Lambda + 2\mu_1) p_{2,2} = \Lambda f_1 p_{1,1} + 2\mu_1 f_1 p_{3,2} + \mu_2 f_1^2 p_{3,3};$$

$$(\Lambda + \mu_2) p_{2,3} = \Lambda p_{1,3} + \mu_1 p_{3,1} + \mu_2 f_2 p_{3,3};$$

The transition pattern is repeated beyond $i > 3$:

$$(\Lambda + \mu_1)p_{i,1} = \Lambda p_{i-1,1} + 2\mu_1 f_2 p_{i+1,2} + \mu_2 f_1 f_2 p_{i+1,3};$$

$$(\Lambda + 2\mu_1)p_{i,2} = \Lambda p_{i-1,2} + 2\mu_1 f_1 p_{i+1,2} + \mu_2 f_1^2 p_{i+1,3},$$

$$(\Lambda + \mu_2)p_{i,3} = \Lambda p_{i-1,3} + \mu_1 p_{i+1,1} + \mu_2 f_2 p_{i+1,3}.$$

The analysis leads to the z-transforms for the three execution states. We obtain unknown probabilities noting that z-transforms evaluate to one for $z = 1$. L'Hôpital's rule is applied whenever necessary. The mean number of jobs in the two classes follows from the z-transform.

Larger TSMs with $K = s = 3$ were solved numerically in Thomasian [1978a].

6.4. Independent Server Model

According to the ISM, customers require a variable number of servers simultaneously to commence service [Green 1980]. This implies that some servers may be idle while there are customers awaiting service. The ISM is different from a batch arrival system, as customers in a batch may acquire servers singly. In addition, F/J tasks are intended for specific servers and may enqueue for initially busy servers. The ISM differs from the TSM, referred to as *joint service* in Green [1980], in that the servers are released asynchronously. Customer service is considered completed when all of its tasks complete their execution.

There are s servers and as many customer classes, which request $1 \le i \le s$ servers with probability c_i, $1 \le i \le s$. With a per-task service rate μ, the service time distribution is given as

$$B_s(t) = \sum_{i=1}^{s} c_i (1 - e^{-\mu t})^i, \quad \sum_{i=1}^{s} c_i = 1.$$

A *queueing period* Q begins when a customer arrives at an empty queue waiting for service and ends when the queue becomes empty again. Note that this is unlike an M/G/m queueing system, where servers cannot be idle while the queue is nonempty. A *nonqueue period* $\overline{Q}$ begins when the preceding queueing period ends and ends when a queue forms. Together, they constitute a cycle. The customers in a queueing period enter service according to an *interservice time* B. Customers that initiate a queueing period enter service after an *initial delay* D. Expected values of B and D are derived next.

A proposition in Green [1980] states that all s servers are busy when a customer enters service in a queueing period. Furthermore, (1) the sequence of interservice times $\{B_n, n \ge 1\}$ are i.i.d. random variables, and (2) the sequence of queueing cycles forms a renewal process and the queue-length process is regenerative.

Let p_q denote the probability that there is a queue and p_d the probability that a customer arriving at an empty queue is delayed. The LST for the waiting time is based on Eq. (59).

$$\mathcal{W}^*(s) = (1 - p_q)(1 - p_d) + (1 - p_q)p_d\mathcal{D}^*(s) + p_q \frac{(1 - \mathcal{D}^*(s))(1 - \lambda E[B])}{[s - \lambda + \lambda\mathcal{B}^*(s)]E[D]} B^*(s).$$

Given $B^*(s)$ and $D^*(s)$ for the LST of B and D and $E(B)$ and $E(D)$ for their mean, the LST for the waiting time is

$$\Omega^*(s) = \frac{(1 - \lambda E[B])[\lambda(\mathcal{D}^*(s) - \mathcal{B}^*(s)) - s]}{(1 - \lambda(E[B] - E[D])(\lambda - s - \lambda\mathcal{B}^*(s))}. \tag{59}$$

The probability that the system empties is given as

$$\pi_0 = \frac{1 - \lambda E[B]}{1 - \lambda(E[B] - E[D])}. \tag{60}$$

Thus, Eq. (59) can be rewritten as

$$\mathcal{W}^*(s) = \pi_0 + (1 - \pi_0)\frac{(1 - \mathcal{D}^*(s))(1 - \lambda E[B])}{(s - \lambda + \lambda \mathcal{B}^*(s))E[D]}. \tag{61}$$

$E[Q]$ (resp. $E[\overline{Q}]$) is the duration of a queueing period (resp. nonqueue) period. Given the expression for $E[Q]$ by Eq. (62) and expression for $E[\overline{Q}]$ by Eq. (63), the stationary probability that there is a queue is

$$p_q = \frac{E[Q]}{E[\overline{Q}] + E[Q]}.$$

The duration of the queueing period is given similarly to the mean duration of the delay cycle in M/G/1 [Conway et al. 1967; Kleinrock 1976], except $E[D]$ replaces $E[B]$ in the numerator:

$$E[Q] = \frac{E[D]}{1 - \lambda E[B]} = \frac{E(D)}{1 - \rho}. \tag{62}$$

$E[Q]$ remains finite and a steady-state solution exists provided that $\rho = \lambda E[B] < 1$.

To derive $E(\overline{Q})$, we consider the embedded Markov chain at arrival and server completions with an absorbing state $s + 1$ [Trivedi 2002]. Let $v_{i,j}$ denote the expected number of visits to state j starting with state i before absorption. These are obtained by solving

$$V = (I - T)^{-1},$$

where T is the matrix of transient states with $s + 1$ rows and columns given as

$$T = \begin{pmatrix} 0 & c_1 & c_2 & \dots & c_s \\ \frac{\mu}{\lambda+\mu} & 0 & \frac{c_1\lambda}{\lambda+\mu} & \dots & \frac{c_{s-1}\lambda}{\lambda+\mu} \\ 0 & \frac{2\mu}{(\lambda+2\mu)} & 0 & \dots & \frac{c_{s-2}\lambda}{\lambda+2\mu} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0. \end{pmatrix},$$

Since the expected time per visit spent in state j is $(\lambda + j\mu)^{-1}$, it follows that

$$E[\overline{Q}] = \sum_{j=0}^{s} \frac{v_{s,j}}{\lambda + j\mu}. \tag{63}$$

Let L and $\overline{L}$ denote the number of arriving customers during queueing and nonqueue periods in a cycle, then $E[\overline{L}] = \lambda E[\overline{Q}]$ and $E[L] = \lambda E[Q]$. The fraction of customers arriving at an empty queue that must wait is $p_d = 1/E[\overline{L}]$.

The condition for a steady-state probability distribution is

$$\rho = \frac{\lambda}{\mu} \sum_{k=1}^{s} \sum_{j=0}^{k-1} \frac{c_k}{s - j} < 1.$$

The mean interservice time ($E[B]$) and initial delay ($E[D]$) are derived in the Appendix of Green [1980]. The probability that k out of i servers will complete service by time t is

$$F_{k,i}(t) = \sum_{j=k}^{i} \binom{i}{j} (1 - e^{-\mu t})^j (e^{-\mu t})^{i-j}, \quad k \le i.$$

Since all servers are busy after an interservice time commences,

$$B(t) = \sum_{k=1}^{s} F_{k,s}(t)c_k.$$

When i servers are busy, the mean time for a server to become free is $1/(i\mu)$, thus

$$E[B] = \sum_{k=1}^{s}\sum_{j=0}^{k-1} \frac{c_k}{(s-j)\mu}.$$

Let $H(i, j)$ denote the probability that a customer finds i busy servers and needs j servers:

$$H(i, j) = \overline{q}_i c_j / p_d,\ i > s - j, \text{and } H(i, j) = 0,\ i \le s - j.$$

$\overline{q}_i$ is the distribution of the number of busy servers during a nonqueue period. It follows that

$$D(t) = \sum_{i=1}^{s}\sum_{k=1}^{i} F_{k,i}\overline{q}_i c_{s-i+k}/p_d.$$

When i servers are busy, the mean time for a server to become free is

$$E[B] = \sum_{k=1}^{s}\left[\frac{1}{s\mu} + \frac{1}{(s-1)\mu} - \cdots + \frac{1}{s-k+1)\mu}\right]c_k, \tag{64}$$

$$E[D] = \sum_{i=1}^{s}\sum_{k=1}^{i}\left[\frac{1}{s\mu} + \frac{1}{(s-1)\mu} + \cdots + \frac{1}{(i-k+1)\mu)}\right]\frac{\tilde{q}_i c_{s-i+k}}{p_d}. \tag{65}$$

Alternative service disciplines are considered in Green [1981]. For the constant total service rate model, the smallest number of servers (SNOS) discipline serves the customer in the queue requiring the fewest number of servers. In the ISM with FCFS (resp. SNOS), $p_q = E[Q]/(E[\overline{Q}] + E[Q])$ (resp. $p'_q = E[Q']/(E[\overline{Q'}] + E[Q'])$ denotes the probability that a queue exists for FCFS (resp. SNOS). It is shown in Green [1981] that $p_q < p'_q$—that is, increased queueing in SNOS. SNOS* selects a customer (if any) that will utilize all servers. A customer is enqueued if it cannot utilize all servers. It can be shown that SNOS* reduces the waiting time with respect to FCFS: $E[W'] < E[W]$.

The system where the customers request a variable number of servers has been analyzed using the MGM in Gillent [1980]. The analysis in Green [1980] is extended in Ittimakin and Cao [1991], where the waiting time distribution is obtained using *randomization* [Gross and Miller 1984; Nelson 1995]. The numerical example in this work considers $\lambda = 1$, $\mu = 1.4$, $s = 4$, $\underline{c} = (0.1, 0.2, 0.3, 0.4)$, thus $\rho = 0.9286$. Note that this utilization takes into account inefficiencies associated with synchronization delays, since otherwise $\rho' = (\lambda/(s\mu))\sum_{i=1}^{s} i f_i = 0.5357$.

6.5. Arbitrary Constraints for Job Activation

We consider transactions with predeclared lock requests, which is only possible at a course granularity of locking—for example, tables in a relational DBMS [Ramakrishnan and Gehrke 2003]. The activation of transactions is delayed when requested locks are held in a conflicting mode by currently active transactions until all required locks are released [Thomasian 1985]. Only exclusive lock requests are considered in this study. The processing time of activated transactions is determined by the composition of transactions executing in the system (see Appendix III).

$$T = \begin{pmatrix} T_{1,1} & 0 & \frac{f_3}{1-f_2}T_1 & \frac{f_4}{1-f_2}T_1 & \frac{f_5}{1-f_2}T_1 & \frac{f_1 f_2}{1-f_2}T_1 \\ 0 & T_{2,2} & \frac{f_3}{1-f_1}T_2 & \frac{f_4}{1-f_1}T_2 & \frac{f_5}{1-f_1}T_2 & \frac{f_1 f_2}{1-f_1}T_2 \\ f_1(1-f_2)T_3 & f_2(1-f_1)T_3 & T_{3.3} & f_4 T_3 & f_5 T_3 & 2f_1 f_4 T_3 \\ f_1(1-f_2)T_4 & f_2(1-f_1)T_4 & f_3 T_4 & T_{4,4} & f_5 T_4 & 2f_1 f_2 T_4 \\ f_1(1-f_2)T_5 & f_2(1-f_1)T_5 & f_3 T_5 & f_4 T_5 & T_{5,5} & 2f_1 f_2 T_5 \\ (1-f_2)T'_2 & (1-f_1)T'_1 & 0 & 0 & 0 & T_{6,6} \end{pmatrix}$$

Fig. 2. Representation of the T matrix for a closed task system with two tasks.

In Thomasian [1985], we consider multiple transaction classes $\mathcal{C}_k$ with frequencies $f_k, 1 \leq k \leq K$ in the arrival stream, thus $\sum_{k=1}^{K} f_k = 1$. Compatible transaction classes that can be processed concurrently can be deduced from lock request tables. To shorten the discussion, we do not provide lock request tables but only compatible transaction classes in two cases with "fine" and "coarse" granularity of locking. There are 14 (resp. 6) feasible transaction execution states when the locking granularity is fine (resp. course) in the two cases. Only the latter case with $K = 5$ classes, which allows the concurrent execution of transaction classes 1 and 2, is considered next for the sake of brevity.

The mean response times of different transaction classes is the sum of the delay in activating transactions in strict FCFS order and their execution time at the computer system, which is represented by a QNM—for example, the central server model (CSM) in Figure 6 (see Appendix III). The solution of the closed QNM for various transaction execution states that $\mathcal{S}_i, 1 \leq i \leq 5$ yields T_i $1 \leq i \leq 5$ and $\mathcal{S}_{1,2}$ with transaction throughputs $T'_i, i = 1, 2$. Using hierarchical decomposition, these throughputs are then incorporated into a higher-level two-dimensional CTMC. The first dimension is the number of transactions in the system, and the second dimension is the execution state.

The recursive queueing analyzer (RQA) [Wallace and Rosenberg 1966] succinctly specifies the sparse regularly structured state transition matrix (Q). RQA is applicable to the solution of two-dimensional CTMCs with one infinite dimension. The number of states in this dimension, which is also the queue size, is set to be sufficiently large so that the fraction of transactions lost due to the finite capacity is negligibly small for the given arrival rate. An iterative method of the form $\underline{\pi}(k+1) = \underline{\pi}(k)(c\boldsymbol{Q} + I)$, where c is a constant, $\boldsymbol{Q}$ is the transition rate matrix, and I is the unity matrix [Bolch et al. 2006]. The index of the idle execution state is set to one. States executing single transactions are numbered from 2 to 6, and states with $1 < i \leq N$ transactions are numbered $6(i-1)+j, 1 \leq j \leq 6$. Generally, a set of repeating elements are given by $(\alpha_i, \beta_i, \gamma_i, B_i)$, where α_i is the value of the rate and (β_i, γ_i) are the coordinates of the transition, so elements repeat at $(\beta_i + k\delta, \gamma_i + k\delta)$, where δ is a fixed increment: $B_i = \beta_i + k_{max}\delta$. Given the state probabilities, we can obtain the performance metrics of interest, such as the mean number of transactions in different classes: $\overline{N}_c, 1 \leq c \leq C$.

In Thomasian [1985] we also explore the aggregation of transaction classes into one, such that the system is specified by its throughput characteristic in processing k transactions in a single class: $T(k), 1 \leq k \leq K_{max}$ and $T(k) = T(K_{max}), k \geq K_{max}$, which is an approximation used for dealing with the maximum degree of concurrency [Lazowska et al. 1984]. Transaction throughput with a single executing transaction is a weighted sum of individual transaction classes. With two transactions in execution and an infinite backlog of transactions processed in FCFS order, we have the transition rate matrix T among the execution state, with $T_{j,j} = -\sum_{i \neq j} f_{j,i}, j = 1, 6$ in Figure 2. Solving the set of linear equations $\underline{P}T = 0$ yields the state probabilities $\underline{P}$.

Transaction throughputs for the five classes with $k = 2$ transactions are as follows:

$$\begin{aligned} T_1(2) &= P(\mathcal{S}_1)T_1(\mathcal{S}_1) + P(\mathcal{S}_{1,2})T'_1(\mathcal{S}_{1,2}), \\ T_2(2) &= P(\mathcal{S}_2)T_2(\mathcal{S}_2) + P(\mathcal{S}_{1,2})T'_2(\mathcal{S}_{1,2}), \\ T_j(2) &= P(S_j)T_j(\mathcal{S}_j), \quad j = 3, 5. \end{aligned}$$

Execution states with m transactions in the system are $T_i(m)/T_j(m) = f_i/f_j$ and $T_k(m) = f_k T(m)$, where $T(m) = \sum_{\forall k} T_k(m)$. For $m = 2$ transactions in the system, the mean number of transactions in different classes is

$$\overline{N}_1(2) = P(S_1)\left(1 + \frac{f_1}{1 - f_2}\right) + P(S_{1,2}) + f_1 \sum_{j=3}^{5} P(S_j),$$

$$\overline{N}_2(2) = P(S_2)\left(1 + \frac{f_2}{1 - f_1}\right) + P(S_{1,2}) + f_2 \sum_{j=3}^{5} P(S_j),$$

$$\overline{N}_j(2) = P(S_j)(1 + f_j) + \frac{P(S_1)f_j}{1 - f_2} + \frac{P(S_2)f_j}{1 - f_1}, 3 \le j \le 5.$$

The mean number of transactions in the system is determined by solving the higher-level model, which is a birth–death process to obtain P_k. The mean number of transaction classes in execution remains the same beyond K_{max}, whereas the number of transactions in different classes among enqueued transactions is determined by their frequencies because the service discipline is FCFS.

A probabilistic analysis of transaction processing systems with predeclared lock requests at a fine granularity is provided in Thomasian and Ryu [1983]. It is assumed that the lock requests are uniformly distributed over all the database locks—that is, the Maxwell-Boltzmann statistic, unlike the Bose-Einstein statistic postulated in Potier and LeBlanc [1980].

7. FORK/JOIN AND TASK SYSTEMS IN QUEUEING NETWORK MODELS

We discuss analytic methods to estimate F/J response times processed in QNMs, which are described in Appendix III. In Section 7.1, jobs executing concurrently are serialized in entering certain phases of their execution. This material is presented first because it is used to cover details of how hierarchical decomposition works in QNMs. In Section 7.2, we use numerical examples to show that the mean F/J response time is affected by job transition probabilities, not just job service demands, which in addition to degree of concurrency are required to analyze QNMs. In Section 7.3, we discuss two performance studies that deal with the analysis of multiprogrammed computer systems with jobs spawning F/J requests, which do or do not require further synchronization. In Section 7.4, we discuss the analysis of task systems, where F/J requests are embedded in a computation that is specified by a directed acyclic graph (DAG). Section 7.5 discusses graph models of computation. In Section 7.6, we discuss stream processing in sensor networks, and in Section 7.7, we present method and tools for analyzing concurrent systems.

7.1. Serialization Delays in Queueing Network Models

Integrity of file accesses is preserved by protecting them with shared and exclusive locks. Jobs are enqueued in a FCFS queue. When the lock being held in exclusive mode is released, then all jobs at the head of the queue requesting a shared lock, or one job requesting an exclusive lock, are activated. The job at the head of the queue may be

blocked until all jobs holding shared locks are completed. Nonstrict FCFS scheduling as in Section 5.3 can be used to improve throughput. The serialization delay is the waiting time to entering a serialization phase (SP). Only exclusive lock requests are considered in the following discussion.

QNMs with serialization delays are not product form, and hence approximate methods are required to analyze them. An iterative solution method considered in Thomasian [1983] represents the possible delay in entering SP by a pseudoserver in the QNM. The pseudoserver denoting the serialization delay is bypassed if no job occupies the SP. The probability of being blocked in entering the s^{th} SP (SP_s) is $P_s(M) \approx 1 - [1 - R_s(M)/R(M)]^{M-1}$, where $R_s(M)$ is the mean residence time in SP_s and $R(M)$ is the mean residence time in the QNM, which is the sum of residence times in $S+1$ phases $R(M) = \sum_{s=0}^{S} R_s(M)$, where $s = 0$ is the nonserialized phase. The delay in entering SP_s at the respective pseudoserver is $\alpha R_s(M)$, where α depends on the distribution of residence time and the placement of requests to enter the SP. Note that this analysis is applicable to low SP utilizations, as it does not take into account the possibility of queueing for SPs.

Two other studies dealing with the analysis of QNMs with serialization delays are as follows. The aggregate server method (ASM) in Agrawal and Buzen [1983] inflates the service times of nonserialized customers to take into account the effect of serialized customers. Iteration is used to determine the inflation factors of service times at aggregate servers. The extension of this method to multiclass networks with shared serialized phases is computationally difficult.

The reduction technique is developed in Jacobson and Lazowska [1983] to analyze QNMs with single-threaded serialization. The average subsystem population approximation (ASPA) uses state-dependent servers, with the number of servers equal to the constraint. According to Table VIII in Appendix III, as the population size increases, the error association with ASPA decreases, whereas the error for the ASM stays at the same level. This method can be extended to multiple classes as long as different classes have identical service times at the SPs.

Aggregation and decomposition for estimating serialization delays introduced in Thomasian [1983] is shown to be more accurate than the iterative method in the same study. At this point, we repeat the analysis for a single job class and two SPs ($S = 2$) in the context of the CSM shown in Figure 6 in Appendix III. The underlying computer system consists of a CPU and disk (denoted by "c" and "d"). Transitions from the nonserialized phase SP_0 to SP_s, $s = 1, S$ occur during CPU processing with probability P_{c0cs}, $1 \le s \le S$. The self-transition P_{c0c0} denotes the completion of a job, at which point a new job is introduced in its place (see Appendix III). After completing processing at the CPU, jobs make transitions to disks with probability $P_{c0d0} = 1 - \sum_{s=0}^{S} P_{c0cs}$. Jobs at SP_s $s \neq 0$ return to SP_0 with probability P_{csc0}, or require disk processing with probability P_{csds}, $1 \le s \le S$. Jobs at SP_s return to the CPU after disk access. The top three rows of the transition probability matrix correspond to transitions from the CPU, whereas the bottom three rows represent transitions from the disk back to the CPU. Three consecutive columns in the matrix correspond to the three phases at the CPU and the disks:

$$P = \left(\begin{array}{ccc|ccc} P_{c0c0} & P_{c0c1} & P_{c0c2} & P_{c0d0} & 0 & 0 \\ P_{c1c0} & 0 & 0 & 0 & P_{c1d1} & 0 \\ P_{c2c0} & 0 & 0 & 0 & 0 & P_{c2d2} \\ -- & -- & -- & -- & -- & -- \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{array} \right),$$

The cycling of jobs between the CPU and the disk may be considered as constituting three QNMs with three job classes for phases 0, 1, and 2. For a given composition of jobs, the state completion rates of job classes can be computed easily if the QNM is product form (see Appendix III). The rows and columns of the CTMC can be renumbered to emphasize the frequent transitions that the job classes make between the CPU and the disk and the rarer transitions among SPs. For example, execution in SP_0 can be represented as

$$\begin{pmatrix} P_{c0c0} & P_{c0d0} \\ 1 & 0 \end{pmatrix},$$

with $P_{c0c0} + P_{c0d0} = 1 - \epsilon$, where ϵ is to take into account the transitions to SP_1 and SP_2: $\epsilon = P_{c0c1} + P_{c0c2}$. Given that the transitions among phases are rare, we can obtain the throughput for each phase by solving the QNM with three job classes [Lavenberg and Sauer 1983a; Bolch et al. 2006; Kobayashi and Mark 2008].

The states of the CTMC are specified by the numbers of jobs in the three phases: (m_0, m_1, m_2) with $m_0 + m_1 + m_2 = M$. For $M = 4$ and one SP, the number of states of the CTMC can be reduced from 15 to 12 by coalescing states (1,2,1) and (1,1,2) into (1,1,1) and also states (0,3,1), (0,2,2), and (0, 1, 3) into (0,1,1). When the processing at SP_1 at (1,1,1) completes, then there is a transition to (2,0,1) if the blocked request was for SP_2, and otherwise the transition is to (2,1,1). With $M = 10$ and $S = 5$, with a detailed state space representation (SSR), there are 3,003 states, whereas only 242 states are required when states are lumped.

To study the effect of lumping of the states of the CTMC on state space reduction, we next consider two job classes, two SPs, and five SSRs [Thomasian and Nadji 1993]. Several simplifying assumptions are made here, such as assuming that jobs in different classes have the same service demands in SPs. Analytic expressions to enumerate the number of states are not repeated here for the sake of brevity.

SSR_1 specifies the number of nonserialized jobs and the contents of the two queues (the first request is active, and the other requests are queued): $(M_1 - m^1, M_2 - m^2; II, I; I, I, II)$, thus the number of jobs in the two SPs are $m^1 = 2$ and $m^2 = 3$.

SSR_2 specifies the identity of job classes currently active in the serialized phases, which is $(M_1 - m^1, M_2 - m^2; II; I)$ for the previous example.

SSR_3 specifies the number of jobs in each class in each phase. In the current example, $(M_1 - m^1, M_2 - m^2; 1, 1; 2, 1)$ with $m^1 = 3$ and $m^2 = 2$. The identity of the job class completing an SP is determined probabilistically by the number of jobs in that class divided by the total number of jobs at the SP.

SSR_4 specifies the total number of job classes in each SP. For the current example, $(M_1 - m^1, M_2 - m^2; 2; 3)$.

SSR_5 uses a Boolean variable to specify whether an SP is active or not. For the current example, $(M_1 - m^1, M_2 - m^2; True; True)$. Only currently active SPs may have a completion.

7.2. Effect of Transition Probabilities on Fork/Join Processing in Queueing Network Models

We first consider the execution of a single K-way F/J request, whose tasks have identical service demands processed by a multiprogrammed computer system with maximum MPL $M_{max} \geq K$ so that all tasks can be activated simultaneously. Given the service demands of the tasks, task throughputs $T(k), K \geq k \geq 1$ can be obtained by solving the underlying QNM. Using the hierarchical decomposition principle, a higher-level CTMC can be used to represent the behavior of the system. The completion rates in a pure death process [Kleinrock 1975] are $T(k), K \geq k \geq 1$, thus $\mathcal{S}_k \xrightarrow{T(k)} \mathcal{S}_{k-1}, K \geq k \geq 1$. The mean residence time in state $\mathcal{S}_k$ is $R(k) = k/T(k)$, so $R^{F/J}(K) = \sum_{k=1}^{K} R(k)$.

The mean completion time of F/J requests in a closed QNM is sensitive to the distribution of the number of cycles made by tasks. To simplify the discussion in computing task completion rates, we consider a *cyclic server* QNM with two nodes: a processor and a disk. Following a disk access, the job leaves the system or returns to the processor. The number of cycles required by a job may be fixed or follow a geometric distribution: $p_n = (1-p)p^{n-1}, n \geq 1$ with a mean $\overline{n} = 1/(1-p)$. A fixed number of jobs can be specified by a nonhomogeneous CTMC [Kleinrock 1975]. The number of jobs completed in a time interval approaches a Poisson process when $p \to 1$, signifying a large number of cycles, since transitions leading to a job completion are less frequent than other transitions. Poisson interdeparture times imply exponentially distributed residence times [Salza and Lavenberg 1981]. It is shown in Appendix II of Kleinrock [1975] that the geometrically distributed sum of exponential random variables is also exponentially distributed. The argument based on thinning point processes in Salza and Lavenberg [1981] does not require the per cycle residence times to be exponentially distributed or even i.i.d.

Consider a single job with two heterogeneous tasks, $\mathcal{T}_1$ and $\mathcal{T}_2$, which are processed concurrently in execution state $\mathcal{S}_{1,2}$. The completion of $\mathcal{T}_i$ in this state leads to $\mathcal{S}_{j\neq i}$, and the completion of task $\mathcal{T}_i$ at $\mathcal{S}_i$ leads to $S_\emptyset$, which signifies the completion of both tasks. Given that $T_i(\mathcal{S}_{1,2})$ is the completion rate of $\mathcal{T}_i$, $i = 1, 2$ at this state, and the mean residence time is $R(\mathcal{S}_{1,2}) = [T_1(1,2) + T_2(1,2)]^{-1}$. Assuming that the transition rates are exponentially distributed, the probability that $\mathcal{T}_i$ completes first is $p_i = T_i(1,2)/[T_1(1,2) + T_2(1,2)], i = 1, 2$. The completion rate at $\mathcal{S}_i$ is $T_i(i)$ for $i = 1, 2$. The mean residence time at $\mathcal{S}_i$, $i = 1, 2$ is $R(\mathcal{S}_i) = [T_i(i)]^{-1}$. The time to complete the two tasks is then

$$C_{geo}(\mathcal{T}_1, \mathcal{T}_2) = R(\mathcal{S}_{1,2}) + p_1 R(\mathcal{S}_2) + p_2 R(\mathcal{S}_1). \tag{66}$$

As a numerical example, consider $\mathcal{T}_1$ and $\mathcal{T}_2$, which require $\overline{n}_1 = 5$ and $\overline{n}_2 = 10$ cycles on the average. Setting the time per visit at the processor and the disk to $\overline{x} = 1$ results in service demands $C_1 = D_1 = 5$ for $\mathcal{T}_1$ and $C_2 = D_2 = 10$ for $\mathcal{T}_2$. The mean residence time of these tasks when processed singly is $R(\mathcal{S}_1) = C_1 + D_1 = 10$ and $R(\mathcal{S}_2) = C_2 + D_2 = 20$. The throughput for the two tasks when processed together can be obtained by solving the corresponding QNM using, which yields $T_1(1,2) = 1/15$ and $T_2(1,2) = 1/30$. The mean residence time in state $\mathcal{S}_{1,2}$ is then $R(\mathcal{S}_{1,2}) = 1/(T_1(1,2) + T_2(1,2)) = 10$. The probability that $\mathcal{T}_1$ and $\mathcal{T}_2$ finishes first is $q_1 = T_1(1,2)R(\mathcal{S}_{1,2}) = (1/15)10 = 2/3$ and $q_2 = T_2(\mathcal{S}_{1,2})R(\mathcal{S}_{1,2}) = (1/30)10 = 1/3$, respectively. It follows from Eq. (66) that $C_{geo}(\mathcal{T}_1, \mathcal{T}_2) = 10 + \frac{2}{3}(20) + \frac{1}{3}(10) \approx 26.77$.

When the number of cycles is fixed and $n_1 = 5$ and $n_2 = 10$, then the two tasks share $min(n_1, n_2) = 5$ cycles together. The mean residence time per cycle according to Eq. (82) in Appendix III is $r(2) = (M + N - 1)\overline{x} = 3$. After $\mathcal{T}_1$ completes, $\mathcal{T}_2$ has five remaining cycles, where each cycle takes $r(1) = 2$ time units. The completion time is $C_{fixed}(\mathcal{T}_1, \mathcal{T}_2) = R'(\mathcal{S}_{1,2}) + R'(\mathcal{S}_2) = 5 \times 3 + 5 \times 2 = 25$, thus $C_{fixed} < C_{geo}$. In counting the number of remaining cycles, in effect we have used the hybrid simulation method in Schwetman [1978].

As a final example, we consider two identical tasks in two cases when the number of cycles is geometrically distributed with a mean $\overline{n} = 5$ and a fixed number of cycles $n = 5$. The service demands are $X = C = D = 5$ in both cases. The two tasks with geometrically distributed processing times have an equal probability of finishing first. Due to symmetry, regardless of which task finishes first, $C_{geo}(\mathcal{T}_1, \mathcal{T}_2) = (N + M - 1)X + NX = 15 + 10 = 25$. For fixed n, the two tasks almost finish together, and it takes $r(2) = 3$ time units per cycle: $C_{fixed}(\mathcal{T}_1, \mathcal{T}_2) = 5 \times 3 = 15$.

It follows that the completion time of F/J requests, and tasks systems in general, is sensitive to the number of cycles. A first-order (homogeneous) Markov chain and

$$
\begin{aligned}
&Q_9 \left(\xrightarrow{0.5} Q_{10} \xrightarrow{0.5} F^2 \right) \\
&Q_{10} \left(\xrightarrow{0.5} F^1 \xrightarrow{0.5} Q_{11} \right) \\
&F^1\{Q_1 \rightarrow Q_2 \rightarrow; Q_3; Q4 \rightarrow Q_5\}\, J^1 \rightarrow Q^9 \\
&F^2\{Q_6; Q_7 \rightarrow Q_8\} J^2 \rightarrow Q_{12} \\
&Q_{11} \rightarrow Q_{12} \rightarrow Q_9
\end{aligned}
$$

Fig. 3. Figure 1 from Baynat and Dallery [1993] with two F/J queues.

geometrically distributed number of cycles are postulated in the studies considered in the following sections.

7.3. Fork/Join Requests with and without Synchronization

A general model for jobs exhibiting internal concurrency is presented in Baynat and Dallery [1993]. The single-server queues of the QNM in Figure 1 are given by Figure 3. Q_i denotes the i^{th} node in the QNM; F^i and J^i are fork and join points; semicolons inside curly brackets delineate the branches of an F/J request—two for F^1 and three for F^2; arrows denote transitions among the nodes of the QNM and also F/J points; and numbers above arrows denote transition probabilities when applicable. The QNM with multiple job classes is analyzed using the aggregation method [Lazowska et al. 1984]. This is an example of a heterogeneous F/J queueing system, with a variable number of nodes visited by tasks.

The model proposed and analyzed in Heidelberger and Trivedi [1982] considers a multiprogrammed computer system with a degree of concurrency M, where jobs spawn asynchronous tasks that contend for resources with their parent jobs. There is an overhead associated with spawning tasks, so tasks should be spawned if they represent significant processing with respect to task spawning overhead. An issue is choosing a threshold to determine whether to execute a computation sequentially or forking parallel tasks to speed up the execution. A high threshold might not create enough tasks to take advantage of available multiprocessing capability, whereas with a small threshold, the overhead of task creation and management may be unacceptable.

The effect of these asynchronous tasks on system load is taken into account by assuming that they are arrivals from a Poisson stream with rate $\lambda = T(M)$, where $T(M)$ is the throughput of the closed jobs. To take into account the effect of resource contention due to asynchronous tasks in the mixed QNM, the service demands of closed jobs are inflated by dividing them by the complement of server utilizations due to asynchronous tasks, which constitute an open (o) job class. At node n, which is assumed to be a single server, the service demands of closed (c) jobs are adjusted as $D'_{n,c} = D_{n,c}/(1-\rho_{n,o})$, where $\rho_{n,o} = \lambda D_{n,o}$, and where $D_{n,o}$ is the service demand of asynchronous tasks [Lavenberg 1983; Lazowska et al. 1984; Bolch et al. 2006; Kobayashi and Mark 2008]. The iteration is repeated until it converges. In the first iteration, with $T(M) = 0$, there is no interference by spawned tasks, and the jobs in the closed QNM exhibit the highest throughput. In the second iteration, the lowest throughput is attained by closed jobs, since their service demands are at the highest due to inflated service demands. This model is also discussed in Section 10.6.1 in Bolch et al. [2006].

A limited task structure comprising a parent task $\mathcal{T}_0$ that spawns two parallel tasks $\mathcal{T}_1$ and $\mathcal{T}_2$ is considered in Heidelberger and Trivedi [1983]. Two approximation methods are considered in this study. Method 1 utilizes a hierarchical decomposition method [Lavenberg 1983; Lazowska et al. 1984; Bolch et al. 2006; Kobayashi and Mark 2008],

where the states of the CTMC at the higher level are specified as the combination of tasks executing in the system: $\underline{a} = (a_0, a_1, a_2)$, with $0 \leq a_0 \leq M$, $0 \leq a_1 \leq M - a_0$, $M \leq a_0 + a_1 + a_2$, and $0 \leq a_1 + a_2 \leq M - 2a_0$. The completion rate of tasks is obtained by solving closed QNMs with three job classes, assuming that the QNM is product form and using one of the methods described in Appendix III.

The number of siblings $\mathcal{T}_i$ that have completed their processing is $w_i = M - a_0 - a_i, i = 1, 2$. When $\mathcal{T}_1$ completes, there is no guarantee that its sibling $\mathcal{T}_2$ is among the w_2 completed $\mathcal{T}_2$ tasks and vice versa. Assuming that the siblings are completed in FCFS order, when $\mathcal{T}_1$ completes and $a_2 > 0$, then it includes its matching sibling $\mathcal{T}_2$. We have the transition from (a_0, a_1, a_2) to (a_0+1, a_1-1, a_2-1). When $\mathcal{T}_1$ completes its processing, the probability that its sibling $\mathcal{T}_2$ has completed its processing is w_2/a_1. The system is analyzed by solving the higher-level Markov chain model.

Method 2 introduces a delay server into the QNM, whose service time is set to the maximum of residence times of the T spawned tasks: $E[D_0] = E[max(\tilde{R}_1, \tilde{R}_2, \ldots, \tilde{R}_T)]$. Assuming that residence times are exponentially distributed, the expected value of the maximum of T random variables is Trivedi [2002]:

$$E[D_0] = \sum_{i=1}^{T} \frac{1}{\lambda_i} - \sum_{i<j} \frac{1}{\lambda_i + \lambda_j} + \sum_{i<j<k} \frac{1}{\lambda_i + \lambda_j + \lambda_k} - \cdots + (-1)^{T-1} \frac{1}{\lambda_{i_1} + \lambda_{i_2} + \ldots + \lambda_{i_T}}.$$

The preceding equation is used in the iterative solution, which converges in practice but is less accurate than the decomposition method.

Spawning of subtransactions when a transaction is blocked due to lock conflicts after requesting a taken lock is proposed in Franaszek et al. [1992]. The spawned subtransaction executes without requesting locks, and this serves the role of prefetching objects for further execution. The subtransaction is aborted when the main transaction is unblocked, but prefetched database objects can be used to speed up the execution of the main transaction.

An iterative method based on nearly complete decomposability and the Gauss-Seidel method is applied in Liu and Perros [1991] to a closed QNMs consisting of a fork-node preceding the F/J queues. The analysis turns out to be exact with $K = 3$ siblings for the subnetwork of F/J queues.

The analysis of a K-way F/J queueing system with a subnetwork of K parallel queues is undertaken in Varki [2001]. Key results are summarized by two theorems. Theorems 2.1 states that for parallel subsystems with homogeneous exponential servers in open and closed networks, the mean response time at an F/J subnetwork with K parallel servers and mean service time $\overline{x}$ is

$$R_{P_K}(M) \leq \overline{x}[H_K + A_{P_K}], \tag{67}$$

where A_{P_K} is the mean number of jobs observed by an arrival. In the case of closed QNMs, the encountered queue length is that of a QNM with one less job $A_{P_K} \approx Q_K(M-1)$, which is due to the arrival theorem [Lavenberg and Sauer 1983a; Lazowska et al. 1994] (see Appendix III).

In a closed network consisting only of K parallel queues (P_K), Theorem 4.1 states that $R_{P_K}(M) \leq \overline{x}[H_k + M - 1]$. Equality holds for $K = 2$, thus this is an exact result.

Geometric bounds (GBs) are a noniterative analysis technique for closed QNMs [Casale et al. 2008]. The noniterative solution method is more accurate than the balanced job bounds (BJBs) [Lazowska et al. 1984] and has been used in analyzing F/J queueing systems [Casale et al. 2008]; in addition, it has been shown to yield more accurate results than BJB, which was used for this purpose in Varki et al. [2013] and the PB approximate method in Hsieh and Lam [1987].

The number of circulating jobs in the closed QNM considered in Casale et al. [2008] is M, and there are N F/J subnetworks with P_n-way F/J queueing systems with equal service demands (at each subnetwork) $D_n = v_n \overline{x}_n, 1 \leq n \leq N$, where v_n is the mean number of visits to the subnetwork and $\overline{x}_n$ is the mean service time. Using Eq. (67), the mean number of jobs at subnetwork n with service demands D_n is

$$R_n^{F/J}(M) \approx D_n\big[H_{P_n} + Q_n^{F/J}(M-1)\big], \tag{68}$$

$$Q_n^{F/J}(M) \approx D_n T(M)[H_{P_n} + Q_n(M-1)], \tag{69}$$

from which the throughput follows: $T(M) = M/\sum_{n=1}^{N} R_n(M)$.

An approximation for the queue length in the n^{th} subnetwork with $P_n = k$ parallel queues is given as Casale et al. [2008]

$$Q_n^{F/J}(M) \approx H_k\left(\frac{y_n(M)}{1-y_n(M)} - \frac{(y_n(M))^{M+1}}{1-y_n(M)}\right), \tag{70}$$

where $y_n(M) = D_n M/(\sum_{j=1}^{M} D_j H_j + D_{max} M) < 1$ with $D_{max} = max\{D_1, D_2, \ldots, D_k\}$. Numerical results are used to compare the accuracy of GB with an exact method to solve product-form QNMs, such as mean-value analysis (MVA) (see Appendix III). GB is more accurate than the BJB method in computing $R_K^{F/J}(M), 2 \leq M \leq 50$, where $P_1 = 1$, $P_2 = 2$, $P_3 = 3$, and $D_1 = 1$, $D_2 = 2$, $D_3 = 3$.

7.4. Analysis of Task Systems

In a task system, we have a set of tasks with precedence relationships specified by a DAG. Fixed task execution times were used in early studies [Coffman and Denning 1972], whereas task execution time variability has been considered in more recent studies [Pinedo 2012]. A large variety of scheduling policies have been proposed to assign tasks to processors to optimize certain criteria, such as minimizing the completion time of all tasks, known as the makespan [Coffman and Denning 1972; Pinedo 2012].

Jobs with precedence constraints to be processed on multiprogrammed computer systems represented by QNMs are considered in Thomasian and Bay [1986]. Jobs are activated as soon as precedence and other constraints such as the maximum MPL are satisfied. The makespan can be reduced by prioritizing the processing of tasks in the Critical Path (CP). When there are multiple computers, the makespan can be reduced by appropriate job allocation.

Jobs are specified by their service demands on multiprogrammed computer systems represented by a QNM. In the case of heterogeneous computers, the service demands should be adjusted according to processor speeds, disk access times, and so forth. Hierarchical decomposition is used to analyze the system. At the lower level, the analysis determines job throughputs for various compositions of tasks executing at the computer systems. The QNM is analyzed using the Buzen's convolution algorithm for multiple classes [Lavenberg and Sauer 1983a; Lazowska et al. 1984; Bolch et al. 2006; Kobayashi and Mark 2008]. Using the hierarchical decomposition method, at the higher level, we have a CTMC whose transition rates are set to job throughputs obtained by solving the closed QNMs [Lavenberg 1983; Lazowska et al. 1984; Bolch et al. 2006; Kobayashi and Mark 2008].

We consider the task system in Figure 4 (from Thomasian and Bay [1986]) to illustrate the discussion. Given that the precedence relationships constitute a DAG, the CTMC for the analysis of the task system is also a DAG, as shown in Figure 5. It is constructed and solved level by level, thus obviating the need to solve a large number

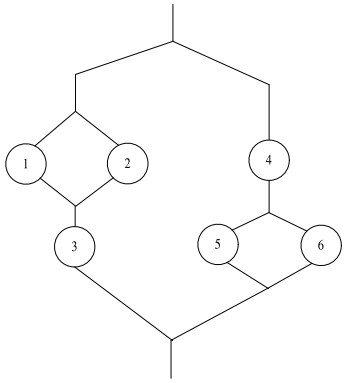

Fig. 4. Task system example in Thomasian and Bay [1986].

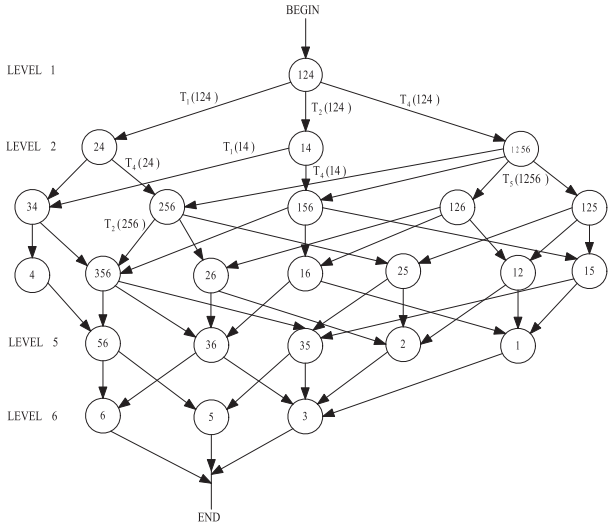

Fig. 5. Markov chain model for analyzing example task system from Thomasian and Bay [1986].

of linear equations corresponding to the execution states of the CTMC. The number of CTMC levels equals the number of tasks, and the number of states in the CTMC also increases accordingly. At one extreme, we have a 10-way F/J request where there are $\binom{10}{5} = 252$ states at level five of the CTMC, with 5 out of 10 tasks completed. The other extreme is when the 10 tasks are executed sequentially and thus the resulting CTMC is a pure death process.

The outline of the procedure to compute the makespan is as follows. Let R and S be two states where state S precedes state R. The completion rate at state S is $T(S) = \sum_{i \in \{S\}} T_i(S)$, where $\{S\}$ denotes the set of tasks executing at that state. The probability that the completion of a task at S leads to R is denoted by $b_R(S) = T_R(S)/T(S)$.

Let R^- denote all states preceding R, then the mean delay from the initial state to the completion of state S is given as the weighted sum:

$$D(R) = M(R)p(R) + \sum_{S \in R^-} b_R(S)D(S).$$

where $p(R)$ is the probability of reaching state R from the initial state, and $M(R)$ is the residence time in that state. $p(R)$ is given as

$$p(R) = \sum_{S \in R^-} p(S) b_R(S).$$

Denoting the set of states at level ℓ by S_ℓ following normalization we have $\sum_{S \in S_\ell} p(S) = 1$. The time until the initiation of task $\mathcal{T}_i$ is given by I_i. We assume that $\mathcal{T}_i$ is initiated at the completion of states S_i, which leads to a transition to state R at which $\mathcal{T}_i$ starts its execution:

$$I_i = \sum_{S \in S_i} D(S) b_R(S).$$

Given that $\mathcal{T}_i$ completes its execution in states $S \in S'_i$, which leads to state R, then the time to complete $\mathcal{T}_i$ is

$$C_i = \sum_{S \in S'_i} D(S) b_R(S).$$

The execution time of $\mathcal{T}_i$ is $E_i = C_i - I_i$.

Unnormalized state probabilities are computed level by level after setting the probability of the initial state to one:

$$P(R) = \sum_{S \in R^-} P(S) T_R(S) / T(R).$$

These probabilities are normalized with respect to the normalization constant: $N = \sum_{\forall R} P(R)$.

The completion time of the task system (C) is the initiation of the "End" dummy task, which succeeds all tasks. Given that A_i denotes all states in which $\mathcal{T}_i$ is active, its execution time is given by

$$E_i = C \sum_{S \in A_i} P(S) = \sum_{S \in A_i} p(S) / T(S).$$

The state space explosion problem has been addressed in studies dealing with generating CTMCs from Petri nets [Bolch et al. 2006]. A compact state representation can be adopted a priori, as in the case of serialization delays [Thomasian and Nadji 1993]. The truncation of continuous stochastic logics (CSLs) [Zhao and Ciardo 1993] is considered in the context of the Stochastic Model checking Analyzer for Reliability and Timing (SMART) tool [Ciardo et al. 2006].

As an example of the effect of state truncation, consider two tasks $\mathcal{T}_1$ and $\mathcal{T}_2$ executing at two independent servers with rates $\mu_1 = 1$ and $\mu_2 = 0.1$ so that their mean execution times are $\bar{t}_1 = 1/\mu_1 = 1$ and $\bar{t}_2 = 1/\mu_2 = 10$. The execution state $\{\mathcal{T}_1, \mathcal{T}_2\}$ with mean holding time $\bar{t}_{(1,2)} = 1/(\mu_1+\mu_2) \approx 0.91$ is followed by two states with mean $\bar{t}_1 = 1/\mu_1 = 1$ and $\bar{t}_2 = 1/\mu_2 = 10$, which are transitioned with probability $p_1 = \mu_1/(\mu_1 + \mu_2) \approx 0.91$ and $p_2 = 1 - p_1$. The completion time of the two tasks is $\bar{T} = \bar{t}_{(1,2)} + p_1\bar{t}_2 + p_2\bar{t}_1 \approx 10.1$. Ignoring that possibility of $\mathcal{T}_2$ completing first, $\bar{T}' = \bar{t}_{(1,2)} + \bar{t}_2 = 10.91$, thus the completion time is overestimated. We next set $\mu_1 = 1$ and $\mu_2 = 0.01$ so that $\bar{t}'_{1,2} = 1/1.01$, $p_1 = 1/1.01$ and the completion time of the two tasks is $\bar{T} = \bar{t}'_{(1,2)} + p_1\bar{t}'_2 + p_2\bar{t}'_1 \approx 100.98$. State truncation or ignoring the possibility that $\mathcal{T}_2$ completes first yields $\bar{T}' = 100.99$, which is still an overestimate but very accurate.

This study has motivated several studies, most notably Menasce et al. [1995], where the emphasis is on processor heterogeneity; thus, larger tasks benefit from processing

on a fast processor, whereas other tasks are processed on ordinary processors. A distinction is made between uniprogrammed and multiprogrammed parallel systems and between space- and time sharing in the latter case. From the viewpoint of F/J synchronization, it is best that all tasks be allocated approximately equal amounts of processor time. Gang scheduling or coscheduling schedules all executable tasks together. A distinction is made in selecting tasks first and processors next. Several dynamic processor assignments are proposed based on their speed. The Largest Task First/Minimum Finish Time First static scheduling policy is shown to outperform others.

The GRAVITY topology considered in Menasce et al. [1995] consists of four k_i-way F/J requests with $k_1 = k_2 = 8$, $k_3 = 12$, and $k_4 = 10$. There are five sequential synchronization tasks with service rate μ_s. The service rate of parallel tasks is μ_p with $\delta = \mu_s/\mu_p$. The fraction of sequential processing F_S is given as

$$F_S = \frac{5 + 4\delta}{5 + \delta(2H_8 + H_{12} + H_{10})}.$$

In Kung [1984], the interest is in assessing the speedup attainable by multiprocessors on jobs represented as DAGs. There are the following attributes: (1) job arrivals, either at time zero or from a Poisson source; (2) the DAG, either fixed or variable; (3) service time of each task, either constant or exponentially distributed; and (4) the number of processors, either fixed or infinite. A common concurrency measure is obtained by first converting the DAG into a CTMC, where each state represents a possible set of tasks that can be executed in parallel. Two algorithms are presented for assigning the tasks to processors: the first one minimizes the expected time to complete all jobs, whereas the other maximizes the utilization of the processors.

A tool based on an amalgamation of queueing network theory and graph models of program behavior is proposed in Kapelnikov et al. [1989]. Parallel computations with looping constructs are allowed in Kapelnikov et al. [1992].

7.5. Graph Models of Computation

Complex relationships among tasks can be specified by Petri nets [Peterson 1981], which are bipartite directed multigraphs specified as $G(V, A)$, where the vertices $V = P \cup T$ consist of *places* and *transitions*. The places are denoted by circles and transitions by bars (lines). An arc from place p_i to a transition t_j defines it to be an input to the transition. There are also arcs from transitions to places. The marking μ of a Petri net is the assignment of tokens to its places. A transition with two input places fires by removing tokens from both places and placing them in the respective output place.

The UCLA graph model of parallel computation (discussed in Section 8.6 of Peterson [1981]) can be easily converted to a Petri net and vice versa. It is stated that the attempt to show that they are equivalent in the Appendix of Gostelow [1971] shows a misunderstanding of Petri nets.

Stochastic Petri nets (SPNs), which associate a time delay with transitions, were proposed in Molloy [1982]. GSPNs differ from SPNs in that they offer immediate transitions, which fire before all others. GSPNs have been used in Balbo et al. [1986]. to represent synchronization and concurrency effects in software systems, such as serialization delays. A translation to a CTMC is still required to solve the system, which then leads to the analysis of the CTMC as in Thomasian [1983] and Thomasian and Nadji [1993]. Performance evaluation of multiprocessor system based on their GSPN model is discussed in Ajmone-Marsan et al. [1986], whereas Ajmone-Marsan et al. [1995] is a more comprehensive text on this matter.

The critical path method (CPM) determines the minimum time to execute a set of tasks with precedence constraints. Tasks have fixed execution times. When there are no resource constraints, the earliest completion times of tasks can be computed easily.

The completion time of tasks can be determined by filling a table so that tasks in column $n+1$ have tasks in column n as their predecessors. An LP formulation can be used to determine the tasks in the critical path (CP) [Dantzig 1998].

The Program Evaluation and Review Technique (PERT) consists of tasks whose processing time is approximated by the Beta distribution given by Eq. (14). In determining the execution time of PERT tasks, the most favorable (a_i), least favorable (b_i), and most likely (m_i) execution times for task T_i are assumed to be known. The mean and variance for T_i can be specified as

$$E(T_i) = (a_i + 4m_i + b_i)/6, \quad \mathrm{V}ar(T_i) = (b_i - a_i)^2/36.$$

With several simplifying assumptions, the mean and the variance of the critical path (CP) is given as

$$E[T_{CP}] = \sum_{i \in CP} T_i, \quad \mathrm{V}ar[T_{CP}] = \sum_{i \in CP} \mathrm{V}ar(T_i).$$

7.6. Stream Processing in Sensor Networks

Distributed Streams Processing System (DSPS) has been developed to analyze large amounts of data from dispersed sources, such as distributed sensor systems. An SPDS utilizes multiple servers that cooperate in processing streams. Each stream requires processing at multiple servers. Servers and communication links have finite computing power and transmission bandwidth; therefore, a fundamental problem in SPDS is to optimize the utilization of resources and to maximize the concurrent throughput for all output streams. This is modeled as a multicommodity flow model for the preceding problem, which is solved using a distributed resource allocation algorithm that guarantees the optimality [Xia et al. 2006].

A distributed algorithm for SPDS, which incorporates a push-and-pull–based admission control mechanism and a pressure-based μc rule [Kleinrock 1976] for resource allocation, is presented in Feng et al. [2007]. It is shown that the algorithm guarantees the stability of the network and converges to the optimal solution.

The throughput scalability of large SPDS with TCP-type reliable data transfer is considered in Xia et al. [2007]. The fork/join queueing network with blocking ($FJQN/B$) model captures three key aspects of stream processing applications. Joins are used to model the simultaneous consumption of input data streams, involving aggregation, correlation, and merging. Forks are used to model the production of multiple output data streams. Blocking occurs when output buffers are full.

FJQN/Bs are represented by $N = (V, E, N)$, where V is a set of N servers, and $E \subset V^2$ is a set of directed edges indicating the flow of jobs. B is the size of the buffer associated with each edge. Server i initiates a service period when there is at least one job in each one of its upstream buffers and there is space for at least one job in its downstream buffers. Server i is starved when at least one of the immediate upstream buffers is full or at least one immediate downstream buffer is full.

QNMs with blocking and various forms of synchronization arise in stochastic modeling of manufacturing systems [Curry and Feldman 2011]. Disassembly/assembly is the terminology used for F/J requests in this context.

Three types of blocking are considered in QNMs with finite buffers [Perros 1994]: (1) *blocking after service*, where a completed job occupies the server until the destination queue at the destination node has an empty slot; (2) *blocking after service*, where the service of a job is delayed if the queue at the destination node is full; and (3) *repetitive blocking*, where the service of a job is repeated if the output queue is full. There is an exact solution for a QNM with $N = 2$ nodes and blocking after service, but the

approximate solution for $N > 2$ is costly and only yields the throughput [Bolch et al. 2006].

Operation research methods developed for manufacturing systems are applicable to stream processing (e.g., see Curry and Feldman [2011]). Given the difficulty of analyzing QNMs with blocking performances, bounds on the asymptotic throughput are used in Xia et al. [2007] to derive necessary conditions under which "the network stays throughput scalable." Presented are explicit conditions under which the system throughput would scale and the exact speed of the throughput degradation as the system size grows.

Joins in stream processing consume data from multiple streams for correlating, merging, and aggregating data, whereas the forks generate multiple output dataflows for duplication. Stream processing can be represented by a DAG with three types of nodes: Sources (S), Processing (P), and Destination (D), as shown in Figure 1 in Zhao et al. [2010a]. A convex optimization algorithm for the sink nodes is formulated in the paper. Due to the distributed and time-varying nature of the system, distributed algorithms are required for its solution. Two sets of algorithms—dual and primal—are proposed in the paper, which have associated advantages and disadvantages.

This problem is pursued further in Zhao et al. [2010b] to study the resource allocation problem in a node-capacitated processing network with arbitrary combinations of F/J semantics. F/J contention is resolved by introducing lossiness. The resource management problem is formulated as a convex optimization problem under given task-to-server mapping.

Two types of fork semantics and two types of join semantics, where each task is associated with a unique join and a unique fork type. Each task is denoted by $v(a_1, a_2)$, where a_1 specifies its join type and a_2 its fork type. A similar notation was proposed in conjunction with UCLA's graph model of parallel computation (see Section 7.5).

(1) Synchronous or AND-join, denoted as $\otimes$-join, requires data from all input streams simultaneously (e.g., to merge an audio stream and a video stream into a multimedia stream)
(2) Asynchronous or OR-join, denoted by $\oplus$-join, processes data independently from each input stream, either from the left or right input link (e.g., multipath routing of packets)
(3) Synchronous or AND-fork, denoted by $\otimes$-fork, produces multiple output streams simultaneously (e.g., splitting a multimedia stream into an audio stream and a video stream)
(4) Asynchronous or OR-fork, denoted by $\oplus$-fork, chooses one of the output links to send the data it produces (e.g., an $\oplus$-fork with two outputs will send output data over either the left output link or the right output link, but not both simultaneously)

Message reassembly in packet-switched networks when packets follow different paths [Kleinrock 1976] is a variant of F/J queueing with nonhomogeneous service times. Cyclic scheduling of successive packets of a long message along disjoint routes from source to destination seems to be desirable for load-balancing purposes, but this may result in higher buffering requirements and packet reassembly delays at the destination [Kleinrock 1976]. In fact, packet reassembly delays occur even when packets follow the same path, which is due to interfering packets along the path. Two methods that alleviate this delay are message switching, which does not benefit from the pipelining effect of packet switching, and circuit switching, which requires setup time. The analysis in Dou et al. [2000] compares the delay for the three methods to determine the best method under varying parameters.

7.7. Methods and Tools for Analyzing Concurrent Systems

SPNs proposed in Molloy et al. [1982] associate a time delay with transitions in Petri nets [Peterson 1981]. Provided that these times are exponentially distributed, SPNs can be easily translated into an equivalent CTMC, which can then be solved using well-known techniques for solving CTMCs (e.g., see Chapter 3 of Bolch et al. [2006]). GSPNs are extensions of SPNs that allow immediate transitions and have been extended to stochastic reward networks (SRNs) by utilizing Guards to specify conditions for transitions and reward rates [Bolch et al. 2006].

The application of the Symbolic Hierarchical Automated Reliability and Performance Evaluator (SHARPE) package for analyzing GSPNs is presented in Sahner et al. [1996] and Bolch et al. [2006]. An extension to this package to deal with general distribution of holding times in semi-Markov models appears in Malhotra and Reibman [1993]. A similar transformation from task graphs in Figure 4 to a Markov chain in Figure 5 in Section 7.3 are applicable to analyzing GSPNs.

Platform-Independent Petri Net Editor 2 (PIPE2) is an open source tool that supports the design and analysis of GSPNs [Dingle et al. 2009]. Several analysis tools are provided, such as a tool for the steady-state analysis of CTMCs with up to $\mathcal{O}(10^8)$) states and a semi-Markov response time analyzer that performs iterative numerical analyses of passage times in very large semi-Markov models and GSPNs.

Research queueing (RESQ), which originated at the University of Texas at Austin and was further developed at IBM Research, has high-level primitives for simulating extended QNMs [Sauer and MacNair 1983] (see http://www.ca.com). RESQ allows active queues with FCFS, PS, priorities without and with preemption; Fission and Fusion nodes specify fork and a join points as in Figure 1. At the CPU, which has a small preemption overhead, it allows a preemptive resume policy. A file transfer from disk needs to be continued from the point where it was interrupted, and this requires a repeated seek and latency—that is, a preemptive repeat policy with resampling. The problem with RESQ and similar packages is that it is not easy to extend the package to handle variations from what is provided by the package.

8. CONCLUSIONS

F/J queueing systems arise in many contexts, some of which have been described in this article. Formulas and analytic solutions methods to analyze the performance of F/J queueing systems can be used to evaluate overall system performance and to optimize it.

F/J queueing systems are difficult to analyze, as exemplified by the analysis of two-way F/J queueing systems under Markovian assumptions, which yields the mean response time $R_2^{F/J}$ [Flatto and Hahn 1984]. Approximate methods are a more promising approach in this direction, such as the use of simulation results to derive $R_K^{F/J}, 2 < K \leq 32$ under Markovian assumptions [Nelson and Tantawi 1988]. It is shown in this study that $R_K^{F/J}$ is upper bounded by the maximum of K independent response times—that is, $R_K^{F/J} < R_K^{max}$.

Curve fitting was used in Thomasian and Tantawi [1994] to obtain approximations to $R_K^{F/J}$ with M/G/1 queues. This study also investigates the effect of interfering requests on $R_K^{F/J}$. It is observed that as the fraction of interfering requests increases, R_K^{max} becomes a better approximation to $R_K^{F/J}$.

F/J requests occur in the context of RAID5 disk arrays with a single disk failure. Since disks process interfering requests, R_K^{max} is a better approximation to F/J response time. A common method is to obtain the moments and the pdf of response time of individual tasks and approximate it with a distribution amenable to computing R_K^{max},

such as Erlang, Coxian, hyperexponential, or EVD. The two moment approximations for maxima is carefully explored in Crow et al. [2007]. The computational cost for the first two distributions increases rapidly with the number of stages representing the distribution when the CV is smaller than one. The characteristic maximum can be utilized in this case to reduce the computational cost.

The expected value of the maximum of K execution times ($\overline{X}_K^{max}$) is of interest in parallel processing systems. The method proposed in Sun and Peterson [2012] yields accurate results for some distributions.

Tasks of an F/J request may require service at multiple resources that comprise a QNM. The task system in Thomasian and Bay [1986] represents job-level parallelism with precedence constraints. There is a state-explosion problem in the respective CTMC generated from the task system, even when hierarchical decomposition is applied, which can be trimmed by ignoring states with a low occurrence probability.

APPENDIX I: RAID5 DISK ARRAYS

The initial RAID classification of levels 1 through 5 was extended to two additional levels (0 and 6) [Chen et al. 1994]. RAID0 does not provide redundancy but balances disk load via striping—that is, the partitioning of large files into fixed size strips that are allocated round-robin across N disks. This allows higher data transfer rates via parallel data access to disks with balanced utilizations.

Strips should be large enough to accommodate most logical disk requests in one disk access, but not too large to allow efficient parallel access to very large files. The extreme case of parity striping does not stripe data at all [Gray et al. 1990] but allocates strips sequentially on consecutive disk drives, providing enough disk space on each disk for parity blocks. This allows efficient sequential accesses to large files at disk transfer rate. The queueing analysis in Chen and Towsley [1993] compares the effect of access skew on mean response time in a RAID5 with parity striping and a RAID5 (described later) with strips smaller than the size of logical blocks being accessed. In the latter case, F/J requests are required to materialize larger logical requests. Given that the stripe width is W_s, the distribution of the number of accessed blocks is assumed to follow a quasigeometric distribution with $P(B = 1) = \sigma$ and

$$P(B = i) = (1 - \sigma)p(1 - p)^{i-1}/P, \quad 2 \le i \le W_s,$$

with $P = (1-p)[1-(1-p)^{W_s-1}]$ in Chen and Towsley [1993]. A slightly different access pattern is postulated for this purpose in Kuratti and Sanders [1995].

The analysis of a closed RAID0 system is considered in Lee and Katz [1991, 1993], where L processes generate $n < N$ requests uniformly over the N disks of RAID0, thus the probability of accessing each disk is $p = n/N$. Due to symmetry, disk utilizations are balanced and given by $U \approx [1 + (1/p - 1)/L]^{-1}$. Variations of this formula are compared against simulation results and are shown to be adequately accurate. The analysis is used to determine the optimal strip size in Chen et al. [1994].

Replication and erasure coding are the main methods for coping with disk failures. Striping is applicable to disk arrays in both categories. The simplest form of replication is mirroring or duplexing classified as RAID1. RAID1 is costly in that it doubles disk space and disk access requirements (for writing). The writing of data to RAID1 is considered completed when both disks are updated by a two-way F/J request, but a shortcut is to consider the update completed as soon as the data is written to at least one of the disks at the the first available empty disk space or to nonvolatile storage (NVS) [Thomasian 2006].

An analysis of disk access traces of online transaction processing (OLTP) applications [Ramakrishnan et al. 1992] shows that disk accesses are to small, randomly placed

Table IV. Disk Numbers Followed by Two Rows with Naive and Permuted Placement in Clustered RAID5 with $N = 10$ and $G = 4$

0	1	2	3	4	5	6	7	8	9
D_0	D_1	D_2	P_{0-2}	D_3	D_4	D_5	P_{3-6}	D_6	D_7
D_8	P_{6-8}	D_9	D_{10}	D_{11}	P_{9-11}	D_{12}	D_{13}	D_{14}	P_{12-14}
D_0	D_4	D_3	P_{3-6}	D_6	D_5	P_{0-2}	D_2	D_7	D_1
D_8	P_{9-11}	D_{11}	D_{13}	D_{14}	D_{12}	D_{10}	D_9	P_{12-14}	P_{6-8}

Note: The parity updates will not be evenly distributed because only one half of the disks holds parities.

disk blocks that incur high disk positioning times. Most disk accesses are reads, and therefore RAID1 benefits OLTP workloads. Disk accesses are results of misses at the main memory buffer and the cache of the disk array controller (DAC), which acts as its extension. This cache is partially NVS and duplexed, providing the same level of reliability as disks and allowing writes to be deferred so that disk reads can be processed at a higher priority. In turn, reads associated with F/J requests to materialize data on a failed disk can be processed at a higher priority than ordinary reads to equalize disk access times. A scheme that alternates in utilizing mirrored disks for reading and writing is described in Polyzois et al. [1993] and further extended in Thomasian and Liu [2005].

When a disk fails in basic mirroring, the rate of read requests to the surviving disk is doubled. A solution to this problem is to distribute the data of each disk over $n > 1$ disks so that if a disk fails, the load increase at surviving disks is $1/n$ [Thomasian and Xu 2008; Thomasian and Tang 2012]. Clustered RAID5 to attain a similar effect is described later.

RAID5 utilizes the simplest form of erasure coding by dedicating the capacity of one disk out of $N > 2$ disks to parity. Unlike RAID4, which places all parity strips on one disk, parity strips in RAID5 are placed in right to left repeating diagonal, the so-called left-symmetric parity layout [Chen et al. 1994].

The updating of small data blocks in RAID5 requires an initial F/J request to read the old data and parity blocks, d_{old} and p_{old}, and computing the new parity as $p_{new} = p_{old} \oplus d_{old} \oplus d_{new}$; the new data and parity blocks, d_{new} and p_{new}, are then written to disk by a second F/J request [Thomasian et al. 2007]. Uniform updating of disk blocks will result in uniform disk accesses.

When a single disk in a RAID5 array with N disks fails, its blocks are reconstructed by accessing the corresponding blocks at $N - 1$ surviving disks and using eXclusive-ORing (XORing) to reconstruct the missing block [Chen et al. 1994]. This results in the doubling of disk loads due to SRs in OLTP applications [Thomasian et al. 2007]. Disk loads in normal and degraded mode with a mixture of read and write requests are derived in Thomasian [2005]. We continue the discussion with SR requests to simplify the discussion and because they result in the maximum performance degradation with a single disk failure.

Clustered RAID5 or parity declustering with a parity group size $G < N$ results in a reduction in the increased load. In the case of SR disk accesses, the load on surviving disks is doubled, whereas with clustered RAID, the load increase is $\alpha = (G - 1)/(N - 1) < 1$. Data layouts for clustered RAID5 based in balanced incomplete block designs (BIBDs) or nearly random permutations (NRPs) [Merchant and Yu 1996] are reviewed in Thomasian [2005] and Thomasian and Blaum [2009]. BIBD designs are not available for all values of N and G, for example, no layouts for $N = 33$ with $G = 12$, but $G = 11$ and $G = 13$ can be used instead [Hall 1986]. NRP is more flexible in that it provides data layouts for values of α and hence G, which are not available with BIBD. The NRP data layout with $N = 10$, $\alpha = 1/3$, or $G = 4$ is shown in Table IV.

The NRP permutation algorithm for an array A with N elements can be specified as follows.

—Set $n = N$.
—L: Pick a random number k between 1 and n.
—If $k \neq n$ then $A_n \leftrightarrow A_k$.
—Set $n \leftarrow n - 1$ and go back to L if $n < 2$.
—If $N \text{mod} G = 0$ then the random permutation is used once and otherwise repeated $K = \text{LCM}(N, G)/N$ times, where $\text{LCM}(N, G)$ denotes the least common multiple of N and G.

Given the random permutation

$$\underline{P_1} = \{0, 9, 7, 6, 2, 1, 5, 3, 4, 8\}$$

for the first row in Table IV and since $K = \text{LCM}(10, 4)/10 = 2$ the same permutation is repeated on the second row as well.

From the viewpoint of F/J requests, the identity of the subset of disks to be accessed is determined by the row number in RAID5, which is used as a seed (x_0) to a linear congruential random number generator (RNG), such as $x_n = bx_{n-1} + c(\text{mod} m)$ [Shedler 1983]. A higher-level simulation model may randomly select $G - 1$ disks out of $N - 1$ as targets of F/J requests. Consider a 2-way F/J queue with arrival rate λ and unequal routing probabilities for their tasks, say, $p_1 > p_2$. The queuelength of the first synchronization queue will grow with rate $(p_1 - p_2)\lambda$ and the departure rate of the system will be $\lambda' = \lambda p_2$.

Consider a RAID5 disk array, processing SR requests with mean service time $\overline{x}_{SR}$, with an initial disk utilization $\rho = \lambda \overline{x}_{SR}$, where λ is the arrival rate of requests to each disks (see Appendix II). Approximating disk access times with an exponential distribution, as was done in Menon [1994], implies an M/M/1 queueing system with a mean response time (see Appendix II):

$$R(\rho) = \frac{\overline{x}_{SR}}{1 - \rho}. \tag{71}$$

When a disk fails, each access to this disk will entail an F/J request to surviving disks, and thus the arrival rate to surviving disks is doubled to 2λ. Since there are interfering requests (see Section 3.6) and disk response times are exponentially distributed $R^{max}_{N-1} = R'(2\rho)H_{N-1}$, where $R'(2\rho) = \overline{x}/(1 - 2\rho)$. In clustered RAID5 with parity group size $G R^{max}_{G-1} = R((1+\alpha)\rho)H_{G-1}$. Since $H_G < H_N$ for $G < N$ and reduced mean response time $R^{max}_{G-1} < R^{max}_{N-1}$ due to lower disk utilization.

For $N = 10$, $\rho = 0.25$, the response time is $1.33\overline{x}_{SR}$ in normal mode and $5.66\overline{x}_{SR}$ in degraded mode, whereas it is $2.75\overline{x}_{SR}$ for clustered RAID5 with $G = 4$ and $\alpha = 1/3$. In both cases, R^{max} can be reduced by processing F/J requests at a higher priority. Noting that after a disk fails half of surviving disk utilizations are due to F/J requests and using Eq. (76) in Appendix II, we have $R^{max}_{N-1} = R(2\rho)H_{N-1} = 3.77\overline{x}_{SR}$ and $R^{G-1}_{max} = R((1+\alpha)\rho)H_{G-1} \approx 2\overline{x}_{SR}$.

Rebuild is the systematic reconstruction of the contents of a failed disk on a spare disk by accessing the contents of all surviving disks and XORing the strips in a stripe to reconstruct the missing strip on the failed disk [Menon 1994; Thomasian and Menon 1994, 1997; Kuratti and Sanders 1995; Thomasian and Blaum 2009]. The rebuild process in RAID5 may fail due to a second disk failure or unreadable disk sectors, which is the more likely reason for failed rebuilds. RAID6 with two check strips per stripe protects against two disk failures as well as unreadable disk blocks. The significantly high probability of encountering unreadable disk blocks during rebuild is the main

reason for the introduction of RAID6 and two-disk failure-tolerant (2DFT) disk arrays in general [Thomasian and Blaum 2009].

APPENDIX II: M/G/1 QUEUEING SYSTEM

The queueing of RAID5 disk arrays ignores the processing required at the DAC to determine if a requested block is cached, and if not, to locate and access it from disk. This is due to the availability of extremely fast microprocessors. The determination of queueing delays at disks is simplified by assuming Poisson arrivals, which implies exponential interarrival times from an infinite number of sources. This is a good approximation when I/O requests originate from a sufficiently large number of sources [Buzen and Goldberg 1974], since we are considering a high-performance OLTP system where I/O requests are generated by transactions processed at a high degree of concurrency.

Approximating disk service times with an exponential distribution leads to an M/M/1 queueing model, which has been used in several early studies of RAID5 [Menon 1994] (see Eq. (71)). Disk service time is more accurately represented by general service times of the M/G/1 queueing model [Kleinrock 1975; Allen 1990; Takagi 1991; Bolch et al. 2006; Kobayashi and Mark 2008], which has been utilized in various works [Chen and Towsley 1993; Thomasian and Menon 1994, 1997; Kuratti and Sanders 1995; Merchant and Yu 1996; Thomasian et al. 2007].

In the case of SR requests, disk service time is the sum of seek time of the disk arm, rotational latency, and data transfer time: $\tilde{x}_{SR} = \tilde{x}_{seek} + \tilde{x}_{lat} + \tilde{x}_{xfer}$ [Thomasian et al. 2007]. The service time of single write (SW) accesses incur an additional head settling time, which is about 0.1msec. The i^{th} moment of SR accesses is then

$$\overline{x_{SR}^i} = E\big[(x_{seek} + x_{lat} + x_{xfer})^i\big].$$

The disk utilization factor, which is the product of the arrival rate of disk accesses, mean disk service time ($\overline{x}_{SR}$), divided by the number of disks, $\rho = \lambda \overline{x}_{disk}/m$, should be smaller than one to avoid an excessive queue length [Kleinrock 1975]. Assuming that the three components of x_{SR} are independent, $\sigma_{SR}^2 = \sigma_{seek}^2 + \sigma_{lat}^2 + \sigma_{xfer}^2$, and $c_{SR}^2 = \sigma_{SR}^2/(\overline{x}_{SR})^2$.

The LST of the pdf $f(t)$ for a positive r.v. is [Kleinrock 1975; Allen 1990; Trivedi 2002]

$$\mathcal{F}^*(s) = \int_{t=0^-}^{\infty} f(t)e^{-st}dt. \tag{72}$$

The pdf of disk service time which is denoted by $b(x)$ has an LST $\mathcal{B}^*(s) = \int_0^\infty e^{-sx}b(x)dx$. Assuming that the three components of disk service time are independent, the LST of disk requests is the product of the LSTs of its three components:

$$\mathcal{B}^*(s) = \mathcal{X}_{seek}^*(s)\mathcal{X}_{lat}^*(s)\mathcal{X}_{xfer}^*(s).$$

The i^{th} moments of waiting time in M/G/1 queues ($\overline{w^i}$) require the arrival rate of requests (λ) and the moments of service time: $\overline{x^j}$, $j = 1, i+1$. We have omitted the subscript SR in what follows to simplify the discussion. The mean waiting time and variance with FCFS scheduling is

$$W = \frac{\lambda \overline{x^2}}{2(1-\rho)} = \frac{\rho \overline{x}\big(1 + c_X^2\big)}{2\big(1-\rho\big)}, \tag{73}$$

$$\sigma_W^2 = \overline{w^2} - W^2 = W^2 + \frac{\lambda \overline{x^3}}{3(1-\rho)}. \tag{74}$$

The moments of waiting time can be obtained by taking the derivatives of its LST given later by Eq. (77) [Kleinrock 1975].

Poisson arrivals see time averages (PASTA) [Bolch et al. 2006; Kobayashi and Mark 2008] leads to a direct method to obtain W as the sum of the mean waiting time for the completion of requests in the queue and the request at the possibly busy server, $W = \overline{N}_q\overline{x} + \rho\overline{x'}$, where $\overline{N}_q = \lambda W$ is the mean queue length and $\overline{x'} = \overline{x^2}/(2\overline{x})$ is the mean residual service time, which leads to Eq. (73). The exponential distribution with mean residual lifetime $\overline{x'} = \overline{x} = 1/\mu$ manifests the memoryless property (see Section 4.2). This analysis can be easily extended to obtain the mean waiting time with VSM by considering arrivals to an empty server and denoting the residual lifetime of vacation times $y' = \overline{y^2}/(2\overline{y})$ $W_{VSM} = W_{M/G/1} + (1-\rho)\overline{y'}$.

The waiting time W for M/G/1 queues is the same for scheduling policies, which do not take into account service time, such as FCFS, last-come, first served (LCFS), and Random policies, but the higher moments of waiting time (σ_W^2) are different [Conway et al. 1967; Takagi 1991], The mean and variance of response time are

$$R = \frac{\lambda\overline{x^2}}{2(1-\rho)} + \overline{x}, \quad \sigma_R^2 = \sigma_W^2 + \sigma_X^2. \tag{75}$$

The latter equation holds because the waiting time is independent of service time for these three disciplines. The mean number of requests in the system is given by Eq. (53) in Section 5.3.

Two approximations for percentiles in M/G/1 queues proposed by IBM's James Martin are quoted in Allen [1990]: $W_{90\%} \approx W + 1.3\sigma_W$ and $W_{95\%} \approx W + 2.0\sigma_W$. Further bounds and approximations are given in Kleinrock [1976] and Bolch et al. [2006].

SR accesses in Thomasian et al. [2007] may be given higher nonpreemptive priority [Kleinrock 1976] with respect to SW accesses, as the former affect application response time, whereas the latter do not. To simplify the discussion, we postulate accesses to RAID0, which do not incur SWP. The arrival rates for read and write requests are λ_r and λ_w. Let $\overline{x_{SR}^i}$ and $\overline{x_{SW}^i}$ denote the i^{th} moment of disk access time for SR and SW accesses, respectively. Given $W_0 = (\lambda_r\overline{x_{SR}^2} + \lambda_w\overline{x_{SW}^2})/2$, the mean waiting time for higher-priority SR requests is [Kleinrock 1976; Lavenberg 1983; Allen 1990; Takagi 1991]

$$W_{SR}^P = \frac{W_0}{1-\rho_{SR}}, \tag{76}$$

where $\rho_{SR} = \lambda_{SR}\overline{x_{SR}}$ is the disk utilization due to read requests *only*. In a RAID5 disk array in degraded mode, the mean response times for F/J requests can be reduced by processing SR requests due to F/J requests at a higher priority than interfering SR requests [Thomasian et al. 2007].

In queueing theory, we are concerned with positive continuous distributions, which are specified by their pdf, $f(t), t \geq 0$. The CDF is $F(t) = \int_{0^-}^{t} f(y)dy$. The pdf waiting time in an M/M/1 queueing system is [Kleinrock 1975; Allen 1990]

$$w(t) = (1-\rho)u_0(t) + \lambda(1-\rho)e^{-\mu(1-\rho)t}, \quad t \geq 0.$$

The first term is a unit impulse at $t = 0$ with magnitude $1-\rho$ [Kleinrock 1975], which is the fraction of time $P(\tilde{w} = 0)$. The CDF of waiting time is given as

$$W(t) = \int_{0^-}^{t} w(t)dt = 1 - \rho e^{-\mu(1-\rho)t} \quad t \geq 0,$$

where integration starting at 0^- is intended to capture the impulse at $t = 0$.

The LST of $w(t)$ is

$$\mathcal{W}^*(s) = 1 - \rho + \frac{\lambda(1-\rho)}{s + \mu(1-\rho)}. \tag{77}$$

Noting the Taylor series expansion of e^{-st}, it is easy to compute the i^{th} moment of waiting time ($\overline{w^i}$) by taking the i^{th} derivative of $\mathcal{W}*(s)$ and setting $s = 0$:

$$\overline{w^i} = (-1)^i \frac{d^i \mathcal{W}^*(s)}{ds^i}|_{s=0} = \int_0^\infty t^i w(t) dt. \tag{78}$$

The LSTs for the response time pdf in M/G/1 queues ($\mathcal{S}^*(s)$) for FCFS scheduling is the product of the LSTs of waiting time and service time, as the waiting time is independent of service time [Kleinrock 1975; Allen 1990; Trivedi 2002]:

$$\mathcal{S}^*(s) = \mathcal{W}^*(s)\mathcal{B}^*(s) = \frac{s(1-\rho)\mathcal{B}^*(s)}{s - \lambda + \lambda\mathcal{B}^*(s)}. \tag{79}$$

For some service time distributions, the LST can be easily inverted to yield the pdf of the response time. In the case of the exponential distribution $\mathcal{B}^*(s) = \mu/(s+\mu)$, using Eq. (79) leads to

$$\mathcal{S}^*(s) = \frac{\mu(1-\rho)}{s + \mu(1-\rho)} \Leftrightarrow r(t) = \mu(1-\rho)e^{-\mu(1-\rho t)}. \tag{80}$$

The above equation for M/M/1 queueing system with $\mathcal{B}^*(s) = \mu/(s+\mu)$ can be obtained directly by noting that due to PASTA the distribution of requests encountered by an arrival is $p_k = (1-\rho)\rho^k$ and that due to the memoryless property of the exponential distribution the residual service time of the request in service is the same as the service time. Using the equation for the LST of residual lifetime in Kleinrock [1975], in the case of the exponential distribution we have $\hat{\mathcal{B}}^*)(s) = (1 - \mathcal{B}^*(s))/(s/\mu) = \mathcal{B}^*(s)$, so that:

$$\mathcal{S}^*(s) = \hat{\mathcal{B}}^*(s)\mathcal{W}^*(s) = \mathcal{B}^*(s) \sum_{k\geq 0} [\mathcal{B}^*(s)]^k p_k = \frac{(1-\rho)\mu}{s+\mu} \sum_{k\geq 0} \left(\frac{\lambda}{s+\mu}\right)^k.$$

The pdf in the case of $\mathcal{H}_2$ and $\mathcal{E}_2$, M/D/1 distributions is given in [Kleinrock 1975; Allen 1990; Nelson 1995].

The mean response time in Processor Sharing (PS) queues with arrival rate λ and mean service time $\overline{x}$ is independent of the higher moments of service time and is given as $R_{PS}(x) = \overline{x}/(1-\rho)$. It can be easily shown that for $c_X^2 > 1$, $R_{PS} < R_{FCFS}$, where the latter is given by Eq. (75).

In the case of M/M/1 queues with arrival rate λ, service rate $\mu = 1/\overline{x}$ with $\rho = \lambda\overline{x}$, the conditional LST of response time with PS is given as [Coffman et al. 1970; Kleinrock 1976; Harrison and Patel 1993; Kobayashi and Mark 2008]

$$\mathcal{S}^*(s|x) = \frac{(1-\rho)(1-\rho r^2)e^{[-\lambda(1-r)+s]\overline{x}}}{(1-\rho r)^2 - \rho(1-r)^2 e^{-(\mu/r - \lambda r)\overline{x}}}, \tag{81}$$

where r is the smaller root of the quadratic equation $\lambda r^2 - (\lambda + \mu + s)r + \mu = 0$. Note that $dr/ds|_{s=0} = (\lambda - \mu)^{-1}$, which is required in further steps. Unconditioning on x for the exponential distribution,

$$\mathcal{S}^*(s) = \int_0^\infty \mathcal{S}^*(s|x)\mu e^{-\mu x} dx.$$

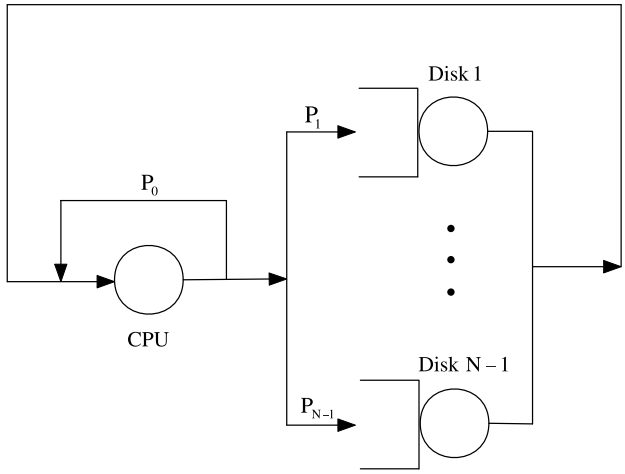

Fig. 6. The closed queueing network model of the central server model.

The moments of response time can be obtained by differentiating $\mathcal{S}^*(s|x)$ so that $R_{PS}(x) = x/(1-\rho)$ and

$$R_{PS} = \int_0^\infty R_{PS}(x)\mu e^{-\mu x} dx = \frac{\overline{x}}{1-\rho}.$$

The equations for $R_{PS}(x)$ and R_{PS} for a general distribution are derived in Section 4.4 in Kleinrock [1976]. The analysis of PS for M/G/1 queues results in rather complex expressions, which are beyond the scope of this discussion.

An M/G/1 queueing system alternates between idle and busy periods. The duration of idle periods is simply the interarrival time $1/\lambda$. Denoting the mean duration of the busy period by $\overline{b}$, the fraction of time that the server is busy is $\overline{b}/(\overline{b} + 1/\lambda) = \rho$, thus $\overline{b} = \overline{x}/(1-\rho)$ [Kleinrock 1975]. The busy period of an M/G/1 queueing system, whose first customer has an exceptional service time with mean $\overline{z}$, is called a delay cycle and its mean duration is $\overline{d} = \overline{z}/(1-\rho)$ [Kleinrock 1976].

APPENDIX III: INTRODUCTION TO QUEUEING NETWORK MODELS

QNMs have been used successfully in capturing the effect of resource contention among jobs utilizing multiple active resources, such as processors and disks in multiprogrammed computer systems [Kleinrock 1976; Lavenberg 1983; Lazowska et al. 1984; Bolch et al. 2006; Kobayashi and Mark 2008]. Active resources are referred to as service stations, which may be considered as the nodes of a graph. There are N nodes, and jobs follow the transition probability matrix $\mathcal{P} = [P_{i,j}]$, $1 \le i, j \le N$ in making transitions among the nodes (see Section 7.1 for an example).

Each service station is a single- or multiserver queueing system or a delay server, which has an infinite number of servers and thus incurs no queueing delay. QNMs have been classified into *open*, *closed*, and *mixed categories*. In open QNMs, jobs arrive and depart from the system, whereas in closed QNMs, the number of jobs remains fixed, as in a multiprogrammed computer system with an infinite backlog of jobs, which remains at its maximum multiprogramming level (MPL).

In open and closed systems, there is a special transition signifying the completion of a job. In the case of open QNMs, arriving jobs eventually leave the system. In the case of the CSM shown in Figure 6, the transition from the CPU back to itself (with probability P_0) signifies a job completion, and the probability of transition from the

CPU to the disks is $P_n, 1 \leq n \leq N-1$. The mean number of visits to the CPU (resp. disks) is $v_0 = 1/P_0$ (resp. $v_n = (P_n/(1-P_0)(1/P_0-1)) = P_n/P_0, 1 \leq n \leq N-1$). Note that the number of visits to the CPU exceeds the total number of visits to disks by one. The cyclic server model differs from CSM in that the special transition occurs following a disk access.

Solving the set of linear equations $\underline{v} = \underline{v}\mathcal{P}$ yields the mean number of visits to the nodes $v_n, 1 \leq n \leq N$ with respect to a reference node, since there are $N-1$ independent equations in N unknowns. The product of v_n and the mean service time per visit ($\overline{x}_n$) at node n is the *service demand* or loading at that node: $D_n = v_n\overline{x}_n, 1 \leq n \leq N$. The service demand, which is the mean total time spent per job at the devices of a computer system, can be estimated by measurement packages, such as those described in Chapter 16 of Lazowska et al. [1984]. There may be multiple job classes with different transition probabilities, mean service times, service demands, and different degrees of concurrency.

Instead of specifying the degree of concurrency for K job classes, there is a need to specify their frequencies, $f_k, 1 \leq k \leq K$, so that completed jobs are replaced by a job possibly in class k according to $\underline{f}$. Several instances of this model in the context of analyzing CC methods are given in Thomasian [2009], where jobs in class k have $k+1$ steps, usually with identical processing times.

A QNM is *product form* if its service stations belong to the four categories satisfying the conditions of the BCMP theorem: (1) FCFS scheduling at nodes with $m \geq 1$ servers with exponential service times, (2) PS queues, (3) infinite servers (IS) or delay servers with no queueing, and (4) last-come, first-served, preemptive resume (LCFS/PR). Service stations in categories 2-4 allow general service times. Most multiprogrammed computer systems restrict the number of concurrent jobs due to memory size constraints.

Closed QNMs satisfying the BCMP theorem can be analyzed using efficient solution methods such as Buzen's convolution algorithm and mean values analysis (MVA). Both solution methods require the degree of job concurrency (M), the number of service stations is N, the number of servers at each node, $\underline{k} = (k_1, k_2, \ldots, k_N)$; and the vector of service demands $\underline{D} = (D_1, D_2, \ldots, D_N)$. The performance metric of interest is the system throughput ($\overline{T}(M)$), which is an increasing function of degree of job concurrency (M) provided that it is not constrained by memory space, cache coherence protocols, latching, and OS bottlenecks in general [Lazowska et al. 1984].

The mean residence or the sojourn time in closed QNMs follows from Little's result, which states that the mean number of jobs at a system equals the product of their mean resident time and their throughput, hence $R(M) = M/T(M)$ [Kleinrock 1975; Lavenberg 1983; Lazowska et al. 1984; Trivedi 2002; Bolch et al. 2006; Kobayashi and Mark 2008]. The mean queue length at the n^{th} node is denoted by $Q_n(M)$, which includes jobs in service, thus $\sum_{n=1}^{N} Q_n(M) = M$. The mean residence at the node n is $R_n(M) = Q_n(M)/T(M)$. The utilization of the n^{th} service station with m_n servers is $U_n(M) = D_nT(M)/k_n$. As discussed in Section 7.2, the mean response time of F/J requests is sensitive to the distribution of the number of cycles.

Open QNMs whose nodes satisfy the criteria of the BCMP theorem can be solved one station at a time given the arrival rate of jobs (Λ) as if they are Poisson arrivals to obtain the mean residence time at the nodes $R_n(\Lambda)$. Postulating one entry and one exit point, which is the CPU, the mean residence time in the system is $R(\Lambda) = \sum_{n=1}^{N} R_n(\Lambda)$. The mean number of jobs at the systems is $\overline{M} = \Lambda R(\Lambda)$.

In the case of multiprogrammed computer systems with external arrivals and a constraint on the maximum MPL for different classes, several solution methods have been developed that are based on represented the computer system with a

flow-equivalent service center (FESC). In the case of a single job class, the computer system is represented by its throughput characteristic $T(m), 1 \leq m \leq M$ and $T(m) = T(M), m \geq M$. For an arrival rate λ, the system can be analyzed as a birth–death process $p(M) = p(M-1)\lambda/T(M), M \geq 1$ [Thomasian 1985]. With two job classes with M_1 and M_2 jobs per class, the per-class throughputs are specified as $T_1(m_1, m_2)$ and $T_2(m_1, m_2)$ for $0 \leq m_1 \leq M_1$ and $0 \leq m_2 leq M_2$. Efficient solution methods in this case are discussed in Lazowska et al. [1984] and Thomasian and Bay [1984].

A particularly simple solution is available for closed QNMs with balanced service demands at single servers: $\overline{x}_n = \overline{x}$ and $v_n = v$ with $D_n = D, 1 \leq n \leq N$ [Thomasian and Nadji 1981; Nadji and Thomasian 1984; Lazowska et al. 1984]. We first use MVA, which is based on the arrival theorem: "a job arriving at node n^{th} node in a closed QNM with M jobs encounters a queue-length as if there are $M-1$ jobs: $A_n(M) = Q_n(M-1)$" [Lavenberg and Sauer 1983a; Lazowska et al. 1984; Bolch et al. 2006; Kobayashi and Mark 2008]. In a balanced QNM, due to symmetry, the number of jobs at the nodes are $Q_n(M) = M/N$ and hence $A_n(M) = (M-1)/N, 1 \leq n \leq N$. The mean job residence time at the n^{th} node per visit is $r_n(M) = [1 + (M-1)/N]\overline{x}, 1 \leq n \leq N$, and multiplying by v and noting $D = v\overline{x}$ yields the mean residence time of a job at node n as $R_n(M) = ((N+M-1)/N)D$. The mean residence time of jobs is then $R(M) = NR_n(M) = (M+N-1)D$, and the throughput of the closed QNM is

$$T(M) = \frac{M}{R(M)} = \frac{M}{(N+M-1)D}, \quad T^{BJB}(M) = \frac{M}{R(M)} = \frac{M}{(N+M-1)D_a}. \tag{82}$$

In a QNM whose service demands are not balanced setting $D = D_a = \sum_{n=1}^{N} D_n/N$ yields the Balanced Job Bound (BJB), which is an upper bound to $T(M)$ according to Lazowska et al. [1984]. The *Asymptotic Job Bound (AJB)* yields the maximum throughput attainable by the QNM as $M \to \infty : T^{AJB} = min\{k_n/D_n, 1 \leq n \leq N\}$ [Kleinrock 1976; Lazowska et al. 1984]. Consider a QNM with a large number of single server queues with a single large service demand D_{max}, while the other $N-1$ service demands are negligibly small, so that $D_a \approx D_{max}/N$ and it is easy to see that $T^{BJB}(M)$ significantly overestimates T^{AJB}_{max}.

These studies utilize Buzen's convolution algorithm to derive Eq. (82). In a QNM with N single-server nodes and M jobs, the probability of being in state $\underline{m} = (m_1, m_2, \ldots, m_N)$ with $\sum_{n=1}^{N} m_n = M$ is

$$p(\underline{m}) = \frac{{D_1}^{m_1} {D_2}^{m_2} \ldots {D_N}^{m_N}}{G(M)},$$

where $G(M)$ is the normalization constant, since according to Buzen's convolution algorithm, $T(M) = G(M-1)/G(M)$ [Lavenberg and Sauer 1983a; Trivedi 2002; Bolch et al. 2006; Kobayashi and Mark 2008]. When $D_n = D, \forall n$ and noting that the number of QNM states is $S(M,N) = \binom{M+N-1}{N-1}$, then $G(M) = S(M,N)D^M$. For unequal service demands, a sequence of approximation formulas using successive terms in a Taylor series expansion was developed in Thomasian and Nadji [1981] and Nadji and Thomasian [1984] for single and two job classes. More recent work in this area is the GB (geometric bounds) noniterative analysis technique for closed QNMs [Casale et al. 2008], which was discussed in Section 7.3.

Limited exact results are available for response time distribution in QNMs. In the case of N tandem M/M/1 queues with service rates $\mu_n, 1 \leq n \leq N$ with an arrival rate λ to the first queue, provided $\rho = \lambda/(mu_n) < 1$, we have

$$\mathcal{S}^*(s) = \prod_{n=1}^{N} \frac{\mu_i - \lambda}{\mu_i - \lambda + s}.$$

This result holds since the departures from the first $N-1$ M/M/1 queues are Poisson, but the last queue can be an M/G/1 queue. This approach can be extended to tree networks; however, problems arise otherwise when there is reconvergent fanout when overtaking is a possibility, for example:

$$Q_1 \xrightarrow{p} Q_2, Q_1 \xrightarrow{1-p} Q_3, Q_2 \rightarrow Q_3.$$

Overtaking is possible in the path $\{Q_1, Q_2, Q_3\}$, since a customer at Q_2 may arrive at Q_3 after a customer from Q_1. Approximate methods to obtain the response time distributions are reviewed in Harrison and Patel [1993].

ACKNOWLEDGMENTS

Valuable information from Petzold [2000] was pointed out to the author by Dr. Asser Tantawi from IBM Research. Ms. Yujie Tang of Shenzhen Institutes of Science and Technology (SIAT), and currently a Ph.D. student at the University of Waterloo, assisted in the preparation of this article. Anonymous reviewers pointed out new areas for F/J processing.

REFERENCES

L. A. Adamic. 2010. Zipf, power-law, Pareto—a ranking tutorial. Retrieved June 29, 2014, from http://www.hpl.hp.com/research/idl/papers/ranking.

S. C. Agrawal and J. P. Buzen. 1983. The aggregate server method for analyzing serialization delays in computer system. *ACM Transactions on Computer Systems* 1, 2, 116–143.

M. Ajmone-Marsan, G. Balbo, and A. Conte. 1986. *Performance Evaluation of Multiprocessor Systems*. MIT Press.

M. M. Ajmone-Marsan, G. Balbo, S. Donatelli, and G. Franceschinis. 1995. *Modelling with Generalized Stochastic Petri Nets*. John Wiley & Sons.

A. O. Allen. 1990. *Probability and Statistics, and Queueing Theory with Computer Science Applications* (2nd ed.). Academic Press, New York, NY.

B. C. Arnold. 1980. Distribution-free bounds on the mean of the maximum of a dependent sample. *SIAM Journal on Applied Mathematics* 38, 1, 163–167.

F. Baccelli, W. A. Massey, and D. F. Towsley. 1989. Acyclic fork-join queuing networks. *Journal of the ACM* 36, 3, 615–642.

F. Baccelli and E. G. Coffman Jr. 1982. A data base replication analysis using an M/M/m queue with service interruptions. *ACM Performance Evaluation Review* 11, 4, 102–107.

F. Baccelli. 1985. Two parallel queues created by arrivals with two demands: The M/G/2 symmetrical case. INRIA Rapport de Recherche, No. 426. INRIA, Valbonne, France.

F. Baccelli and A. Makowski. 1989. Queueing models for systems with synchronization delays. *Proceedings of the IEEE* 77, 1, 138–161.

F. Baccelli and Z. Liu. 1990. On the execution of parallel programs on multiprocessor systems: A queuing theory approach. *Journal of the ACM* 37, 2, 373–414.

G. Balbo, S. C. Bruell, and S. Ghenta. 1986. Combining queueing network and generalized stochastic Petri net models for the analysis of some software blocking phenomena. *IEEE Transactions on Software Engineering* SE-12, 4, 561–575.

S. Balsamo and I. Mura. 1995. Approximate response time distribution in fork and join systems. In *Proceedings of the ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*. 305–306

S. Balsamo and I. Mura. 1997. Queue length moments in fork and join queuing networks with general service times. In *Computer Performance Evaluation.* Lecture Notes in Computer Science, Vol. 1245. Springer, 218–231.

S. Balsamo, L. Donatiello, and N. M. van Dijk. 1998. Bound performance models of heterogeneous parallel processing systems. *IEEE Transactions on Parallel and Distributed Systems* 9, 10, 1041–1056.

R. E. Barlow and F. Proschan. 1975. *Statistical Theory of Reliability and Life Testing*. Rinehart & Winston.

B. Baynat and Y. Dallery. 1993. A decomposition approximation method for closed queueing networks with fork/join subnetworks. In *Proceedings of the IFIP WG10.3 International Conference on Decentralized and Distributed Systems*. 199–210.

G. Blom. 1958. *Statistical Estimates and Transformed Beta Variables*. John Wiley & Sons.

G. Bolch, S. Greiner, H. de Meer, and K. S. Trivedi. 2006. *Queueing Networks and Markov Chains* (2nd ed.). Wiley-Interscience, New York, NY.

W. E. Boyce and R. C. DiPrima. 2012. *Elementary Differential Equations and Boundary Value Problems* (10th ed.). Wiley.

J. P. Buzen and P. S. Goldberg. 1974. Guidelines for the use of infinite source queueing models in the analysis of computer system performance. In *Proceedings of the 1974 AFIPS National Computer Conference (NCC'74)*, Vol. 43. AFIPS Press, Montvale, NJ, 371–374.

G. Casale, R. R. Muntz, and G. Serazzi. 2008. Geometric bounds: A non-iterative analysis technique for closed queueing networks. *Transactions on Parallel and Distributed Systems* 57, 6, 780–794.

S. Ceri, M. A. W. Houtsma, A. M. Keller, and P. Samarati. 1991. *A classification of update methods for replicated databases*. Technical Report STAN-CS-91-1392. Stanford University, Stanford, CA.

C.-S. Chang and D. N. Nelson. 1995. Bounds on the speedup and efficiency of partial synchronization in parallel processing systems. *Journal of the ACM* 42, 1, 204–231.

M. L. Chaudhry and J. G. C. Templeton. 1983. *A First Course in Bulk Queues*. John Wiley & Sons.

P. M. Chen, E. K. Lee, G. A. Gibson, R. H. Katz, and D. A. Patterson. 1994. RAID: High-performance, reliable secondary storage. *ACM Computing Surveys* 26, 2, 145–185.

S. Chen and D. F. Towsley. 1993. The design and evaluation of RAID 5 and parity striping disk array architectures. *Journal of Parallel and Distributed Computing* 10, 1–2, 41–57.

M. Cherniack, H. Balakrishnan, M. Balazinska, D. Carney, U. Cetintemel, Y. Xing, and S. B. Zdonik. 2003. Scalable distributed stream processing. In *Proceedings of the Conference on Innovative Data Systems Research (CIDR)*.

H. Ciardo, R. L. Jones III, A. S. Miner, and R. Siminiceanu. 2006. Logic and stochastic modeling with SMART. *Performance Evaluation* 63, 6, 578–608.

E. G. Coffman Jr. 1967. Bounds on parallel-processing of queues with multiple phase jobs. *Naval Research Logistics Quarterly* 14, 345–366.

E. G. Coffman, Jr., R. R. Muntz, and H. Trotter. 1970. Waiting time distributions for processor-sharing systems. *Journal of the ACM* 17, 1, 123–130.

E. G. Coffman Jr. and P. J. Denning. 1972. *Operating Systems Principles*. Prentice Hall.

E. G. Coffman Jr., H. O. Pollak, E. Gelenbe, and R. C. Wood. 1981a. An analysis of parallel-read sequential write systems. *Performance Evaluation* 1, 1, 62–69.

E. G. Coffman Jr., E. Gelenbe, and B. Plateau. 1981b. Optimization of the number of copies in a distributed data base. *IEEE Transactions on Software Engineering* SE-7, 1, 78–84.

R. W. Conway, W. L. Maxwell, and L. W. Miller. 1967. *Theory of Scheduling*. Addison-Wesley, Reading, MA.

C. A. Courcoubetis and M. I. Reimann. 1987. Optimal control of a queueing system with simultaneous service requirements. *IEEE Transactions on Automatic Control* AC-32, 8, 717–727.

M. Crovella and A. Bestavros. 1996. Self-similarity in World Wide Web traffic: Evidence and causes. In *Proceedings of the ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*. 160–169.

C. S. Crow IV, D. A. Goldberg, and W. Whitt. 2007. Two-moment approximations for maxima. *Operations Research* 55, 3, 532–548.

G. l. Curry and R. M. Feldman. 2011. *Manufacturing Systems: Modeling and Analysis* (2nd ed). Springer.

G. B. Dantzig. 1998. *Linear Programming and Extensions*. Princeton University Press.

H. A. David. 1970. *Order Statistics*. Wiley.

H. A. David and H. N. Nagaraja. 2003. *Order Statistics* (3rd ed.). Wiley.

J. Dean and S. Ghemawat. 2010. MapReduce: A flexible data processing tool. *Communications of the ACM* 53, 1, 72–77.

N. J. Dingle, W. J. Knottenbelt, and T. Suto. 2009. PIPE2: A tool for the performance evaluation of generalised stochastic Petri nets. *SIGMETRICS Performance Evaluation Review* 36, 4, 34–39.

C. Dou, C.-T. Lin, S.-W. Wang, and K.-C. Leu. 2000. Performance analysis of packet-level scheduling in an IP-over-ATM network with QoS control. *IEICE Transactions on Communications* E83-B, 7, 1534–1543.

A. Duda and T. Czachrski. 1987. Performance evaluation of fork and join primitives. *Acta Informatica* 24, 5, 525–533.

O. Etzion and P. Niblett. 2010. *Event Processing in Action*. Manning Publications, Stamford, CT.

A. Feldmann and W. Whitt. 1998. Fitting mixtures of exponentials to long-tail distributions to analyze network. *Performance Evaluation* 31, 3–4, 245–279.

H. Feng, Z. Liu, C. H. Xia, and L. Zhang. 2007. Load shedding and distributed resource control of stream processing networks. *Performance Evaluation* 64, 912, 1102–1120.

L. Flatto and S. Hahn. 1984. Two parallel queues created by arrivals with two demands: I. *SIAM Journal on Applied Mathematics* 44, 1041–1053.

P. A. Franaszek, J. T. Robinson, and A. Thomasian. 1992. Concurrency control for high contention environments. *ACM Transactions on Database Systems* 17, 2, 304–345.

M. Ghodsi and K. Kant. 1991. Performance analysis of parallel search algorithms on multiprocessor systems. *Performance Evaluation* 13, 1, 67–81.

F. Gillent. 1980. *An Application of the Matrix Geometric Analytic Method*. Master's Thesis. Free University of Brussels.

K. P. Gostelow. 1971. *Flow of Control, Resource Allocation, and the Proper Termination of Programs*. Ph.D. Thesis. Computer Science Departments, University of California at Los Angeles, UCLA-ENG-7179.

A. Gravey. 1985. A simple construction of an upper bound for the mean of the maximum of N identically distributed random variables. *Journal of Applied Probability* 22, 844–851.

J. Gray, B. Horst, and M. Walker. 1990. Parity striping of disk arrays: Low-cost reliable storage with acceptable throughput. In *Proceedings of the 16th International Conference on Very Large Data Bases (VLDB'90)*. 148–159.

L. Green. 1980. A queueing system in which customers require a random number of servers. *Operations Research* 26, 6, 1335–1346.

L. Green. 1981. Comparing operating characteristics of queues in which customers require a random number of servers. *Management Science* 27, 1, 65–74.

D. Gross and D. R. Miller. 1984. The randomization technique and a modeling tool and solution procedure for transient Markov processes. *Operations Research* 32, 2, 343–361.

E. J. Gumbel. 1954. Statistical theory of extreme values and some practical applications. *Applied Mathematics Series 33*, U.S. Department of Commerce, National Bureau of Standards.

E. J. Gumbel. 1958. *Statistics of Extremes*. Columbia University Press, New York, NY.

M. R. Gupta and Y. Chen. 2010. *Theory and Use of the EM Method*. New Publisher Inc.

M. Hall. 1986. *Combinatorial Theory* (2nd ed.). Wiley-Interscience, New York, NY.

P. G. Harrison and N. M. Patel. 1993. *Performance Modeling of Communication Networks and Computer Architectures*. Addison-Wesley, Reading, MA.

P. H. Harrison and S. Zertal. 2007. Queueing models of RAID systems with maxima of waiting times. *Performance Evaluation* 64, 7–8, 664–689.

J. Havil. 2003. *Gamma: Exploring Euler's Constant*. Princeton University Press.

P. Heidelberger and K. S. Trivedi. 1982. Queueing network models for parallel processing with asynchronous tasks. *IEEE Transactions on Computers* C-31, 11, 1099–1109.

P. Heidelberger and K. S. Trivedi. 1983. Analytic queueing models for programs with internal concurrency. *IEEE Transactions on Computers* C-32, 1, 73–82.

R. A. Howard. 1960. *Dynamic Programming and Markov Processes*. MIT Press.

C.-T. Hsieh and S. S. Lam. 1987. Two classes of performance bounds for closed queueing networks. *Performance Evaluation* 7, 1, 3–30.

P. Ittimakin and E. P.-C. Cao. 1991. Stationary waiting time distribution of a queue in which customers required a random number of servers. *Operations Research* 39, 4, 633–638.

P. A. Jacobson and E. D. Lazowska. 1983. A reduction technique for evaluating queueing networks with serialization delays. In *Proceedings of the 9th International Symposium on Computer Performance Modelling, Measurement and Evaluation*. 45–59.

N. L. Johnson, S. Kotz, and N. Balakrishnan. 1995. *Continuous Univariate Distributions* (Vol. 2, 2nd ed.). Wiley Series in Probability and Statistics.

A. Kapelnikov, R. R. Muntz, and M. D. Ercegovac. 1989. A modeling methodology for the analysis of concurrent systems and computations. *Journal of Parallel and Distributed Computing* 6, 3, 568–597.

A. Kapelnikov, R. R. Muntz, and M. D. Ercegovac. 1992. A methodology for performance analysis of parallel computations with looping constructs. *Journal of Parallel and Distributed Computing* 14, 2, 105–120.

C. Kim and A. K. Agrawala. 1985. *Virtual waiting time of an Erlangian single server queueing system*. Technical Report UMIACS-TR-1986-6. University of Maryland, College Park, MD.

C. Kim and A. K. Agrawala. 1989. Analysis of the fork-join queue. *IEEE Transactions on Computers* C-38, 2, 250–255.

M. Y. Kim. 1986. Synchronized disk interleaving. *IEEE Transactions on Computers* C-35, 11, 978–988.

M. Y. Kim and A. N. Tantawi. 1991. Asynchronous disk interleaving: Approximating access delays. *IEEE Transactions on Computers* C-40, 7, 801–810.

R. Kooi. 1980. *The Optimization of Queries in Relational Databases*. Ph.D. Dissertation. Case Western Reserve University, Cleveland, OH.

L. Kleinrock. 1975. *Queueing Systems, Vol. I: Theory*. Wiley-Interscience, New York, NY.

L. Kleinrock. 1976. *Queueing Systems, Vol. II: Applications*. Wiley-Interscience, New York, NY.

D. E. Knuth. 1997. *The Art of Computer Programming Vol. 2: Seminumerical Algorithms* (3rd ed.). Addison-Wesley, Reading, MA.

S.-S. Ko and R. F. Serfozo. 2004. Response times in M/M/s fork-join networks. *Advances in Applied Probability* 36, 3, 854–871.

S.-S. Ko and R. F. Serfozo. 2008. Sojourn times in G/M/1 forkjoin networks. *Naval Research Logistics* 55, 5, 432–443.

H. Kobayashi and B. L. Mark. 2008. *System Modeling and Analysis: Foundations of Systems Performance Evaluation*. Pearson Prentice-Hall.

A. Krishnamurthy, R. Suri, and M. K. Vernon. 2004. Analysis of a fork/join synchronization station with inputs from Coxian servers in a closed queuing network. *Annals of Operations Research* 125, 1–4, 69–94.

C. P. Kruskal and A. P. Weiss. 1985. Allocating independent subtasks on parallel processors. *IEEE Transactions on Software Engineering* SE-11, 10, 1001–1016.

K. C.-Y. Kung. 1984. *Concurrency in Parallel Processing Systems*. Ph.D. Dissertation. Computer Science Department, University of California at Los Angeles.

A. Kuratti and W. H. Sanders. 1995. Performance analysis of the RAID5 disk array. In *Proceedings of the International Computer Performance and Dependability Symposium*. 236–245.

T. L. Lai and H. Robbins. 1976. Maximally dependent it random variables. In *Proceedings of the National Academy of Sciences* 73, 2, 286–288.

T. P. Lane and P. D. Welch. 1987. The integration of a menu-oriented graphical statistical system with its underlying general purpose language. In *Proceedings of the 19th Symposium on the Interface of Computer Science and Statistics*. 267–373.

G. Latouche and V. Ramaswami. 1999. *Introduction to Matrix Analytic Methods in Stochastic Modeling*. SIAM Press.

S. S. Lavenberg (Ed.). 1983. *Computer Performance Modeling Handbook*. Academic Press, New York, NY.

S. S. Lavenberg and C. H. Sauer. 1983a. Analytical results for queueing models. In *Computer Performance Modeling Handbook*, S. S. Lavenberg (Ed.). Academic Press, New York, NY.

S. S. Lavenberg and C. H. Sauer. 1983b. Approximate analysis of queueing networks. In *Computer Performance Modeling Handbook*, S. S. Lavenberg (Ed.). Academic Press, New York, NY.

E. D. Lazowska. 1977. The use of percentiles in modeling CPU service time distribution. In *Computer Performance*, K. Mani Chandy and M. Reiser (Eds.). North-Holland, 53–64.

E. D. Lazowska and C. A. Addison. 1979. Selecting parameter values for servers of the phase type. In *Performance of Computer Systems*, M. Arato et al. (Eds.). North-Holland, 407–420.

E. D. Lazowska, J. Zahorjan, G. S. Graham, and K. C. Sevcik. 1984. *Quantitative Systems Performance: Computer System Analysis Using Queueing Network Models*. Prentice Hall.

A. S. Lebrecht and W. J. Knottenbelt. 2007. Response time approximations in fork-join queues. In *Proceedings of the 23rd Annual UK Performance Engineering Workshop (UKPEW'07)*.

A. S. Lebrecht, N. Dingle, P. Harrison, W. Knottenbelt, and S. Zertal. 2009. Using bulk arrivals to model I/O request response time distributions in zoned disks and RAID systems. In *Proceedings of the 3rd International ICST Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS'09)*.

E. K. Lee and R. H. Katz. 1991. *An analytic performance model of disk arrays and its applications*. Technical Report UCB/CSD 91/660. University of California at Berkeley.

E. K. Lee and R. H. Katz. 1993. An analytic performance model of disk arrays. In *Proceedings of the ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*. 98–109.

F. Li, B. C. Ooi, T. Özsu, and S. Wu. 2014. Distributed data management using MapReduce. *ACM Computing Surveys* 46, 3, Article 31.

L. Lipsky. 2008. *Queueing Theory: A Linear Algebraic Approach* (2nd ed.). Springer-Verlag, New York, NY.

Y. C. Liu and H. G. Perros. 1991. A decomposition procedure for the analysis of a closed fork/join queueing system. *IEEE Transactions on Computers* C-40, 3, 365–370.

J. C. S. Lui, R. R. Muntz, and D. F. Towsley. 1998. Computing performance bounds of fork-join parallel programs under a multiprocessing environment. *IEEE Transactions on Parallel and Distributed Systems* 9, 3, 295–311.

C. A. Lynch. 1988. Selectivity estimation and query optimization in large databases with highly skewed distribution of column values. In *Proceedings of the 14th International Conference on Very Large Data Bases (VLDB'88)*. 240–251.

M. Malhotra and A. L. Reibman. 1993. Selecting phase approximations for semi-Markov models. *Stochastic Models* 9, 4, 473–506.

M. Mandelbaum and A.-B. Itzhak. 1968. Introduction to queueing with splitting and matching. *Israel Journal of Technology* 6, 5, 376–382.

R. A. Marie. 1978. Methodes iterative de resolution de modeles mathematique do systemes informatique. *RAIRO Informatique Computer Sciences* 12, 107–122.

D. A. Menasce, D. Saha, S. C. S. Porto, V. Almeida, and S. K. Tripathi. 1995. Static and dynamic processor scheduling disciplines in heterogeneous parallel architectures. *Journal of Parallel and Distributed Computing* 28, 1, 1–18.

J. Menon. 1994. Performance of RAID5 disk arrays with read and write caching. *Journal of Distributed and Parallel Databases* 11, 3, 261–291.

A. Merchant and P. S. Yu. 1996. Analytic modeling of clustered RAID with mapping based on nearly random permutation. *IEEE Transactions on Computers* C-45, 3, 367–373.

M. K. Molloy. 1982. Performance analysis using stochastic Petri nets. *IEEE Transactions on Computers* 31, 9, 913–917.

A. N. Mourad, R. J. T. Morris, A. N. Swami, and H. C. Young. 1994. Limits of parallelism in hash join algorithms. *Performance Evaluation* 20, 1–3, 301–316.

B. Nadji and A. Thomasian. 1984. Throughput estimation in queueing networks. *Computer Performance* 5, 4, 197–206.

R. D. Nelson and B. R. Iyer. 1985. Analysis of a replicated database. *Performance Evaluation* 5, 3, 133–148.

R. D. Nelson and A. N. Tantawi. 1988. Approximate analysis of fork-join synchronization in parallel queues. *IEEE Transactions on Computers* C-37, 6, 736–743.

R. D. Nelson, D. F. Towsley, and A. N. Tantawi. 1988. Performance analysis of parallel processing systems. *IEEE Transactions on Software Engineering* SE-24, 4, 532–540.

R. D. Nelson. 1995. *Probability, Stochastic Processes, and Queueing Theory*. Springer-Verlag.

M. F. Neuts. 1981. *Matrix Geometric Solutions in Stochastic Models: An Algorithmic Approach*. John Hopkins University Press.

K. J. Omahen and L. Schrage. 1972. A queueing analysis of a multiprocessor system with shared memory. In *Proceedings of the Symposium on Computer Communication Networks and Teletraffic.* 77–88.

K. J. Omahen. 1977. Capacity bounds for multiresource queues. *Journal of the ACM* 24, 4, 646–663.

H. Perros. 1994. *Queueing Networks with Blocking*. Oxford University Press.

J. L. Peterson. 1981. *Petri Net Theory and the Modeling of Systems*. Prentice Hall.

M. Petzold. 2000. A note on the first moment of extreme order statistics from the normal distribution. Working paper. Goteborg University, Sweden.

M. Pinedo. 2012. *Scheduling: Theory, Algorithms, and Systems* (4th ed.). Springer.

C. A. Polyzois, A. Bhide, and D. M. Dias. 1993. Disk mirroring with alternating deferred updates. In *Proceedings of the International Conference on Very Large Data Bases (VLDB'93)*. 604–617.

D. Potier and Ph. LeBlanc. 1980. Analysis of locking policies in database management systems. *Communications of the ACM* 23, 10, 584–593.

M. Rahman, R. Ranjan, and R. Buyya. 2010. Cooperative and decentralized workflow scheduling in global grids. *Future Generation Computer Systems* 26, 5, 753–768.

K. K. Ramakrishnan, P. Biswas, and R. Karedla. 1992. Analysis of file I/O traces in commercial computing environments. In *Proceedings of the Joint ACM SIGMETRICS/Performance'92 Conference on Measurement and Modeling of Computer Systems*. 78–90.

R. Ramakrishnan and J. Gehrke. 2003. *Database Management Systems*. McGraw-Hill.

B. M. Rao and M. J. M. Posner. 1985. Algorithmic and approximation analyses of the split and match queue *Stochastic Models* 1, 3, 433–456.

R. A. Sahner, K. S. Trivedi, and A. Puliafito. 1996. *Performance and Reliability Analysis of Computer Systems*. Kluwer Academic Publishers, Norwell, MA.

S. Salza and S. S. Lavenberg. 1981. Approximating response time distributions in closed queueing network models of computer performance. In *Proceedings of the 8th International Symposium on Computer Performance Modelling, Measurement and Evaluation*. 133–144.

C. H. Sauer and E. A. MacNair. 1983. Extended queueing network models. In *Computer Performance Modeling Handbook*, S. S. Lavenberg (Ed.). Academic Press, New York, NY.

M. Scharf. 2005. On the response time of large-scale composite Web services. In *Proceedings of the 19th International Teletraffic Congress (ITC 19)*.

D. A. Schneider and D. J. DeWitt. 1989. A performance evaluation of four parallel join algorithms in a shared-nothing multiprocessor environment. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 110–121.

B. Schroeder and G. A. Gibson. 2007. Understanding disk failure rates: What does an MTTF of 1,000.000 hours mean to you? *ACM Transactions on Storage Systems* 3, 3, Article No. 7.

G. S. Shedler. 1983. Generation methods for discrete event simulation. In *Computer Performance Modeling Handbook*, S. S. Lavenberg (Ed.). Academic Press, New York, NY.

H. D. Schwetman. 1978. Hybrid simulation models of computer systems. *Communications of the ACM* 21, 9, 718–723.

M. Seetha-Lakshmi and P. S. Yu. 1990. Effectiveness of parallel joins. *IEEE Transactions on Knowledge and Data Engineering* 2, 4, 410–424.

M. S. Squillante, Y. Zhang, A. Sivasubramaniam, and N. Gautam. 2008. Generalized parallel-server fork-join queues with dynamic task scheduling. *Annals Operations Research* 160, 1, 227–255.

C. Schuler, R. Weber, H. Schuldt, and H.-J. Schek. 2004. Scalable peer-to-peer process management: The OSIRIS approach. In *Proceedings of the 2nd International Conference on Web Services (ICWS'04)*. 26–34.

J. Sun and G. D. Peterson. 2012. An effective execution time approximation method for parallel computing. *IEEE Transactions on Parallel and Distributed Systems* 24, 11, 2024–2032.

H. Takagi. 1991. *Queueing Analysis: Foundations of Performance Evaluation, Vol. 1: Vacation and Priority Systems, Part 1*. North-Holland.

M. Takahashi, H. Sawa, and T. Fujisawa. 2000. On a synchronization queue with two input buffers. *Queueing Systems* 15, 127–146.

M. Takahashi and Y. Takahashi. 2000. Synchronization queue with two MAP inputs and finite buffers. In *Proceedings of the 3rd International Conference on Matrix Analytic Methods in Stochastic Models*. 375–390.

J. Tan, X. Meng, and L. Zhang. 2012. Delay tails in MapReduce scheduling. In *Proceedings of the ACM SIGMETRICS/Performance Conference on Measurement and Modeling of Computer Systems*. 5–16.

A. Thomasian and A. Avizienis. 1975. Dynamic scheduling of tasks requiring multiple processors. In *Proceedings of the 11th IEEE Computer Society International Conference (COMPCON'75 Fall)*. 77–80.

A. Thomasian. 1978a. *A Design Study of a Shared-Resource Array Processing System*. Ph.D. Dissertation. Technical Report UCLA-ENG-7702. Computer Science Department, University of California at Los Angeles.

A. Thomasian. 1978b. A capacity bound for parallel processing. In *Proceedings of the 1978 IEEE International Conference on Parallel Processing (ICPP'78)*. 193–196.

A. Thomasian and B. Nadji. 1981. Aggregation of stations in queueing network models of multiprogrammed computers. In *Proceedings of the ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*. 86–96.

A. Thomasian. 1983. Queueing network models to estimate serialization delays in computer systems. In *Proceedings of the IFIP Working Group 7.3 Performance'83 Conference on Computer System Modeling and Analysis*. 61–79.

A. Thomasian and I. K. Ryu. 1983. A decomposition solution to the queueing network model of the centralized DBMS with static locking. In *Proceedings of the ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*. 82–92.

A. Thomasian and P. F. Bay. 1984. Analysis of queueing network models with population size constraints and delayed blocked customers. In *Proceedings of the ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*. 202–216.

A. Thomasian. 1985. Performance evaluation of centralized databases with static locking. *IEEE Transactions on Software Engineering* SE-11, 4, 346–355.

A. Thomasian and P. F. Bay. 1986. Analytic queueing network models for parallel processing of task systems. *IEEE Transactions on Computers* C-35, 12, 1045–1054.

A. Thomasian and V. Nicola. 1993. Performance evaluation of a threshold policy for scheduling readers and writers. *IEEE Transactions on Computers* C-42, 1, 83–98.

A. Thomasian and B. Nadji. 1993. State representation trade-offs in Markov chain models for serialization delays in computer systems. *Computer Systems: Science and Engineering* 8, 3, 154–165.

A. Thomasian and J. Menon. 1994. Performance analysis of RAID5 disk arrays with a vacationing server model for rebuild mode operation. In *Proceedings of the 10th IEEE International Conference on Data Engineering (ICDE'94)*. 111–119.

A. Thomasian and A. N. Tantawi. 1994. Approximate solutions for M/G/1 fork/join synchronization. In *Proceedings of the Winter Simulation Conference*. 361–368.

A. Thomasian. 1995. Surveyors' forum: High-performance secondary memory. *ACM Computing Surveys* 27, 2, 292–295.

A. Thomasian and J. Menon. 1997. RAID5 performance with distributed sparing. *IEEE Transactions on Parallel and Distributed Systems* 8, 6, 340–357.

A. Thomasian. 1997. Approximate analyses for fork/join synchronization in RAID5. *Computer Systems: Science and Engineering* 12, 5, 329–337.

A. Thomasian. 1998a. Threshold policies for scheduling readers and writers. *Information Sciences* 104, 3–4, 157–180.

A. Thomasian. 1998b. Concurrency control: Methods, performance, and analysis. *ACM Computing Surveys* 30, 1, 70–119.

A. Thomasian. 2005. Access costs in clustered RAID disk arrays. *Computer Journal* 48, 8, 702–713.

A. Thomasian and C. Liu. 2005. Performance comparison of mirrored disk scheduling methods with a shared non-volatile cache. *Distributed and Parallel Databases* 18, 3, 253–281.

A. Thomasian. 2006. Mirrored disk routing and scheduling. *Cluster Computing* 9, 4, 475–484.

A. Thomasian, C. Han, and G. Fu. 2007. Performance evaluation of two-disk failure tolerant arrays. *IEEE Transactions on Computers* C-56, 6, 799–814.

A. Thomasian and J. Xu. 2008. Reliability and performance of mirrored disk organizations. *Computer Journal* 51, 6, 615–629.

A. Thomasian and M. Blaum. 2009. Two disk failure tolerant disk arrays: Organization, operation, and coding methods. *ACM Transactions on Storage* 5, 4, Article No. 7.

A. Thomasian. 2009. Performance analysis of transaction processing systems. In *Encyclopedia of Database Systems*, Ling Liu and M. Tamer Özsu (Eds.), 2085–2093.

A. Thomasian and Y. Tang. 2012. Performance, reliability, and performability of a hybrid RAID array and a comparison with traditional RAID1 arrays. *Cluster Computing* 15, 3, 239–253.

D. F. Towsley, C. G. Rommel, and J. A. Stankovic. 1990a. Analysis of fork-join program response times on multiprocessors. *IEEE Transactions on Parallel and Distributed Systems* 1, 3, 286–303.

D. F. Towsley, S. Chen, and S.-P. Yu. 1990b. Performance analysis of a fault tolerant mirrored disk system. In *Proceedings of the 14th IFIP W.G. 7.3 International Symposium on Computer Performance Modelling, Measurement and Evaluation*. 239–253.

K. S. Trivedi. 2002. *Probability and Statistics with Reliability, Queueing, and Computer Science Applications* (2nd ed.). Wiley-Interscience, New York, NY.

I. Tsimashenka and W. J. Knottenbelt. 2011. Reduction of variability in split-merge systems. In *Proceedings of the Imperial College Computing Student Workshop (ICCSW'11)*. 101–107.

I. Tsimashenka, W. J. Knottenbelt, and P. G. Harrison. 2012. Controlling variability in split-merge systems. In *Analytical and Stochastic Modeling Techniques and Applications (ASMTA'12)*. Lecture Notes in Computer Science, Vol. 7314. Springer, 165–177.

I. Tsimashenka and W. G. Knottenbelt. 2013a. Trading off subtask dispersion and response time in split-merge systems. In *Proceedings of the 20th International Conference on Analytical and Stochastic Modelling Techniques and Applications (ASMTA'13)*. Lecture Notes in Computer Science, Vol. 5984. Springer, 431–442.

I. Tsimashenka and W. J. Knottenbelt. 2013b. Reduction of subtask dispersion in fork-join systems. In *Proceedings of the 10th European Performance Evaluation Workshop (EPEW'13)*. 325–336.

E. Varki and L. W. Dowdy. 1996. Exact response time analysis of two server systems. In *Proceedings of the 4th International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS'96)*. 287–295.

E. Varki. 1999. Mean value technique for closed fork-join networks. In *Proceedings of the ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*. 103–112.

E. Varki. 2001. Response time analysis of parallel computer and storage systems. *IEEE Transactions on Parallel and Distributed Systems* 12, 11, 1146–1161.

E. Varki, A. Merchant, and H. Chen. 2012. The M/M/1 fork-join queue with variable subtasks. Available at http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.100.3062

E. Varki, L. Dowdy, and C. Zhang. 2013. Quick performance bounding techniques for computer and storage systems with parallel resources. Available at http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.4.4105

S. Varma and A. M. Makowski. 1994. Interpolation approximations for symmetric fork-join queues. *Performance Evaluation* 20, 1–3, 245–265.

V. L. Wallace and R. S. Rosenberg. 1966. Markovian models and numerical analysis of computer system behavior. In *Proceedings of the AFIPS Spring Joint Computer Conference*, Vol. 27. 141–148.

W. Whitt. 1982. Approximating a point process by a renewal process: Two basic methods. *Operations Research* 30, 1, 125–147.

E. W. Weisstein. 2012. Hypergeometric Function, MathWorld: A Wolfram Web Resource. Retrieved June 29, 2014, from http://mathworld.wolfram.com/HypergeometricFunction.html.

C. H. Xia, J. Broberg, Z. Liu, and L. Zhang. 2006. Distributed resource allocation in stream processing systems. In *Proceedings of the International Symposium on Distributed Computing (DISC'06)*. Lecture Notes in Computer Science, Vol. 4167. Springer, 489–504.

C. H. Xia, Z. Liu, D. F. Towsley, and M. LeLarge. 2007. Scalability of fork/join queueing networks with blocking. In *Proc.eedings of the ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*. 133–144.

D. D. Yao. 1985. Refining the diffusion approximation for the M/G/m queue. *Operations Research* 33, 11, 1266–1277.

H. Zhao, C. H. Xia, Z. Liu, and D. F. Towsley. 2010a. Distributed resource allocation for synchronous fork and join processing networks. In *Proceedings of the 29th IEEE International Conference on Computer Communications*. 446–450.

H. Zhao, C. H. Xia, Z. Liu, and D. F. Towsley. 2010b. A unified modeling framework for distributed resource allocation of general fork and join processing networks. In *Proceedings of the ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*. 299–310.

Y. Zhao and G. Ciardo. 2013. Tackling truncation errors in CSL model checking through bounding semantics. In *Proceedings of the Computer Performance Engineering 10th European Workshop (EPEW'13)*. Lecture Notes in Computers Science, Vol. 8168. Springer, 58–73.

Received July 2012; revised May 2014; accepted May 2014