Response Time Analysis of Parallel Computer and Storage Systems

Elizabeth Varki, *Member*, *IEEE*

Abstract—Fork-join structures have gained increased importance in recent years as a means of modeling parallelism in computer and storage systems. The basic fork-join model is one in which a job arriving at a parallel system splits into K independent tasks that are assigned to K unique, homogeneous servers. In this paper, a simple response time approximation is derived for parallel systems with exponential service time distributions. The approximation holds for networks modeling several devices, both parallel and nonparallel. (In the case of closed networks containing a stand-alone parallel system, a mean response time bound is derived.) In addition, the response time approximation is extended to cover the more realistic case wherein a job splits into an arbitrary number of tasks upon arrival at a parallel system. Simulation results for closed networks with stand-alone parallel subsystems and exponential service time distributions indicate that the response time approximation is, on average, within 3 percent of the seeded response times. Similarly, simulation results with nonexponential distributions also indicate that the response time approximation is close to the seeded values. Potential applications of our results include the modeling of data placement in disk arrays and the execution of parallel programs in multiprocessor and distributed systems.

Index Terms—Performance evaluation, fork-join networks, parallel computer and storage systems, mean-value analysis.

1 Introduction

The power of parallelism is being exploited in computing and storage systems because of the increased efficiency it provides. In the case of parallel computing, a program divides into subtasks that execute in parallel on different nodes of a system. In the case of storage systems, it is an I/O request that is distributed across multiple disks. In order to predict the performance of such systems, it becomes important to be able to model the parallel behavior.

The performance of parallel systems is often analyzed using queueing models since they provide a favorable balance between efficiency and accuracy [14]. Queueing systems that model parallel devices are called *fork-join* systems. The name is derived from the manner in which jobs arriving at a parallel system are executed. An arriving job *forks* into tasks that execute independently on different service centers of a parallel device. On completing execution, each task waits at the *join* point for its sibling tasks to complete execution. A job finally leaves the parallel device once all its tasks complete execution.

The parameters of the fork-join model are introduced in Fig. 1, which show both an open and a closed parallel network. The upper half of Fig. 1 shows an open network containing a parallel subsystem, P_K, that consists of $K > 1$ identical service centers with exponential service time distributions with mean $s = \mu^{-1}$. Upon arrival at the parallel subsystem, a job forks into K independent and identical tasks, where tasks k, $k = 1, 2, \cdots, K$ are assigned to the kth service center. The service discipline is first-come-first-served. The interarrival time to the parallel subsystem is exponentially distributed with mean λ^{-1}. The lower half of Fig. 1 shows a closed network where jobs arrive at P_K from a server with mean service time λ^{-1}.

The fork-join structure is central to all parallel systems and, hence, has been extensively studied in performance evaluation. See Section 5 for a short description of prior work. Though the job structure is simple, the fork-join network is non-product-form [5]. Hence, none of the solution techniques developed for product-form systems can be used in the modeling and analysis of parallel systems. Consequently, much of the performance evaluation work in fork-join systems has been in the development of approximation or bounding techniques for the mean response time of such systems. Our work is along the lines of this earlier literature in terms of coming up with close approximations for the mean response time of fork join systems. Where this paper differs from the earlier papers is that 1) the response time approximation given here considers the effect of other devices (and not just that of a stand-alone parallel subsystem) on the mean response time of the parallel subsystem in a closed network and 2) the performance technique based on the approximation is computationally efficient for increasing values of K and workload intensity. Further, the paper provides exact mean response time values for stand-alone P_2 subsystems and response time bounds for P_K $(K > 2)$ stand-alone parallel subsystems in closed networks. The response time approximation is also shown to be valid for parallel subsystems with nonexponential servers and for parallel subsystems where arriving jobs split into an arbitrary number of tasks. The response time approximation holds for parallel subsystems in both closed and open networks.

• *E. Varki is with the Department of Computer Science, University of New Hampshire, Durham, NH 03824. Email: varki@cs.unh.edu.*

Manuscript received 11 Dec. 1998; revised 17 Mar. 2000; accepted 13 June 2001.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number 108433.

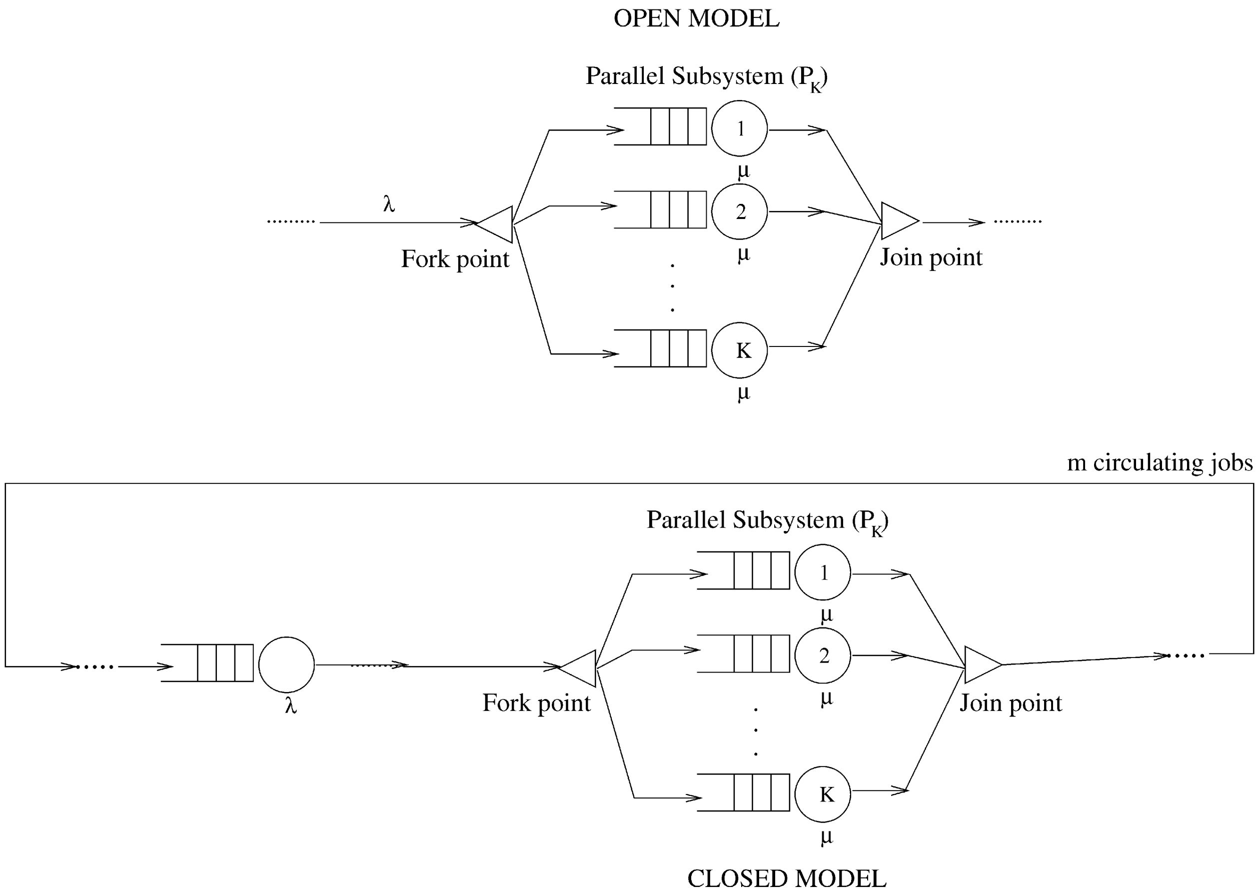


Fig. 1. K-sibling fork-join model.

However, one of the arguments in the approximation is the longest arrival queue length at a server from among the K servers and this argument is difficult to compute for parallel subsystems in open networks. Consequently, a limitation of this paper is that no performance evaluation techniques for parallel subsystems in open networks are provided.

The notation used in the paper is given in Appendix B; the superscript K being used only when required explicitly. The rest of the paper is organized as follows: The next section presents the derivation of the response time expression. In Section 3, the response time expression is extended to parallel systems where arriving jobs split into an arbitrary number of tasks. Section 4 presents performance techniques for different types of closed networks based on the response time approximation. Section 5 presents a brief description of prior work in the area of fork-join networks. Finally, the conclusion and future research directions are provided in Section 6.

2 Derivation of Response Time Approximation

The derivation of the response time approximation is approached at two levels. First, a simple intuitive argument for the approximation is provided by defining a simple way of looking at jobs executing on parallel systems. Second, a more formal proof of the approximation is provided by analyzing parallel systems from alternative frameworks and employing Markov diagrams. Note that the intuitive argument is independent of the second, more formal proof.

2.1 Intuitive Argument for Response Time Approximation

By definition, the response time of a job in a parallel system is the time taken from its arrival instant until all its tasks at K independent servers finish execution. Typically, this is expressed as the sum of a job's service time and wait time. However, since tasks of a job are assigned to independent servers, it is possible for the execution time of jobs to "overlap" in that some tasks of the arriving job are executed along with tasks of preceding and following jobs. For example, task 1 of job 1 could be executing on server A while task 1 of job 2 could be executing on server B as task 2 of job 1 has finished execution and is at the join point. Thus, in parallel systems, the distinction between when a job is waiting and when it is being serviced gets blurry because of the overlap in execution of tasks of different jobs. Accordingly, it becomes important to be able to come up with a set of definitions (for the wait time and service time of a job) which avoids the confusion generated by the task overlaps of different jobs in a parallel systems.

Accordingly, for simplicity in argument, we define the following terms as they apply to parallel systems:

Definition 1. *A job is at the **head** of a parallel system when there are no jobs ahead of it in the parallel system or, equivalently, when the last task of this job starts executing. (When a job* *reaches the head of the parallel system, there are no tasks of earlier arrived jobs executing at any of the service centers or, waiting at the join point.)*

Here, the **last** task refers to the last task of a job to reach the head of its queue and start executing from among the K tasks of the job. (Note that the last task to start executing does not necessarily finish last as its service time depends on the pick from the distribution describing service times.)

Definition 2. *The **wait time** of a job in a parallel system is the time taken from arrival instant until the job reaches the head of the parallel system or, equivalently, the time taken from arrival instant until its last task starts executing.*

Definition 3. *The **service time** of a job in a parallel system is the time taken from the instant the job gets to the head of the parallel system until this job leaves the join point or, equivalently, the time from when the last task begins executing until this job leaves the join point.*

A well-known result in probability theory is that when there are K identical tasks executing on parallel servers, the time taken to finish executing the K tasks is simply $s * O_K$, the mean of the Kth order statistic of task execution times, where s represents the mean execution time of a task and O_K is a scaling factor that is dependent on the service time distribution of a server. Since the service time of a job has been defined to start only when its last task begins executing, it is possible that when this job starts service, some or all of its remaining $K-1$ tasks have finished execution and are at the join point. Hence, the mean service time of a job executing on parallel servers is at most $s * O_K$.

Similarly, the wait time of a job is the time elapsed between its arrival instant until its last task begins executing. When the last task begins executing, all its sibling tasks must be either executing or waiting at the join point. In other words, the wait time of a job is the maximum of the K wait times of its individual tasks which is equal to the mean of the Kth order statistic of task wait times. Since this value is difficult to compute, we generate an optimistic bound for the mean *job wait time* by using the mean *task wait time* at the longest server queue. Let A_{P_K} represent the mean number of waiting tasks ahead of this job's task at the longest server queue. Then, on average, the task assigned to this longest queue would begin executing only after a time lapse of $s * A_{P_K}$, where s is the average time taken to execute a task. Thus, the mean wait time of a job must be at least $s * A_{P_K}$.

Now, the response time of a job in a parallel system is equal to the sum of its service time and wait time. The mean service time of a job has been to shown to be at most equal to $s * O_K$, whereas the mean wait time has been shown to be at least equal to $s * A_{P_K}$. Since one is an upper bound and the other is a lower bound, a response time bound cannot be defined. However, we can potentially derive an approximation by considering the following. We know that the service time is bounded between s and $s * O_K$. Since the distribution of service time of the job is unknown between these two values, we can employ the average of these two values as an approximation. This is equivalent to assuming a uniform (and noninformed) distribution over the range s and $s * O_K$. Similarly, since the wait time for a job is the maximum of the waiting times of the individual tasks and we have considered the mean wait time of the task at the longest queue, we would expect the actual job wait time to be not far from the bound $s * A_{P_K}$. Hence, it is reasonable to state that an approximate value of the mean response time R_{P_K} of a parallel system can be written as:

$$R_{P_K} \approx= s[O_K + A_{P_K}], \qquad (1)$$

where s is the mean task execution time, O_K is the Kth order-statistic scaling factor, and A_{P_K} is the maximum number of tasks at a server queue seen by a job just prior to arrival at P_K. Since every job within P_K must have one task at this longest queue, the term A_{P_K} also represents the mean number of jobs in P_K seen upon arrival.

As of this point, we have not made any assumption about the distribution of task service times and, hence, the result above applies to any parallel system with an arbitrary service time distribution. However, if we were to impose the restriction of an exponential distribution, then the scaling factor O_K is equal to the Kth harmonic number H_K $(= \sum_{i=1}^{k} \frac{1}{i})$. In the next section, we formally prove the approximation in the instance of parallel systems with exponential service time distributions. The approximation is shown to be a strict equality in the case of stand-alone P_2 in closed networks. In all other cases, which include parallel subsystems (not necessarily stand-alone P_K subsystems) in both open and closed networks, the equation is shown to be a bound.

2.2 Derivation of Response Time Approximation Using Alternate Representations of P_K

The response time expression is formally derived here for parallel subsystems with exponential task service times by employing alternate representations of P_K and then equating the parameters of the alternate, but equivalent representations. Note that the result obtained here with exponential service time distributions is a pessimistic bound (and not just an approximation) for P_K.

The three models or alternate representations of the basic P_K model are labeled by us as the serial-join model, the state-dependent model, and the hybrid model. Of these, the first has been discussed in the literature (though not explicitly referred to as a "serial-join" model). Detailed explanations of the alternate, but equivalent representations follow.

2.2.1 Equivalent Models of P_K

Serial-Join Model. In [22], the response time of a job in a parallel subsystem is divided into two phases, namely, the response time of a task at a server queue and the time spent by a job at the join point. For purposes of exposition, we refer to this representation as the serial-join model, where the time spent in the two phases is represented as the time spent at two nonparallel subsystems S_a^K and S_j^K. The mean service time at S_a is equal to that of a service center within P_K. The subsystem S_j in turn models the average delay encountered by a job at the join point of P_K (Fig. 2).

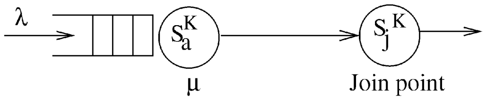


Fig. 2. Equivalent serial-join model of P_K.

State-Dependent Model. This representation of P_K is derived from a Markov analysis of P_K. The state of P_K is represented by the vector $(n_1, \cdots, n_K)$, each element of which represents the number of tasks at each of the K servers of P_K. Since all service centers are identical, the n_is are ordered such that $n_1 \leq n_2 \leq \cdots \leq n_K$. Thus, n_K is equal to the number of tasks at the longest server queue or, equivalently, represents the total number of jobs in P_K since each job is constrained to split into K tasks, one for each server.

Example 1. Consider a 4-sibling fork-join subsystem, P_4.

The state (3, 3, 3, 3) represents a state where there are three jobs in P_4. All four tasks of each job are at the queues (since $n_i = 3$ for all four servers) and there are no tasks at the join point.

The state (0, 1, 2, 3) also represents a state with three jobs in P_4 (since $n_4 = 3$). $n_1 = 0$ implies that the first task of all three jobs has finished execution. $n_2 = 1$ implies that job_3 (the last job) has three active tasks while both job_2 and job_1 have less than three active tasks. $n_3 = 2$ implies that job_2 has two active tasks while job_1 has one active task.

Fig. 3 shows the Markov diagram for P_2 in a network, whether closed or open. The diagram maps the states of P_K when there are zero, one, two, and three jobs in P_2. Each column of the diagram represents states with n number of jobs in the subsystem. Each row of the diagram represents states with n tasks at the join point. The horizontal transition arcs represent the arrival of jobs at P_2 at rate λ. The downward transition arcs, $\vec{t_1}$, represent the movement of a task to the join point. The diagonal transition arcs, $\vec{t_2}$, represent the movement of the last task of a job to the join point at which instant this job departs P_2. The time spent by a job in P_2 can be factored into two phases, namely, $phase_2$ and $phase_1$, in order. In $phase_2$, two tasks of the job are waiting for or receiving, service at the service centers of P_2. In $phase_1$, only one task of the job is at the service center while its sibling task waits at the join point.

Next, we analyze the Markov diagram of P_3 given in Fig. 4. The horizontal transition arcs again represent the movement of jobs into the parallel subsystem at rate λ. The arcs $\vec{t_k}$ $(k = 1, 2, 3)$ represent the movement of the kth task of a job to the join point. Just as in the case of P_2, the response time of a job in P_3 can be factored into three phases. In general, the response time of a job in P_K can be factored into K phases, namely, $phase_K, \cdots, phase_1$, in order, where $phase_k$ represents the situation when k tasks of the job are at the service centers. A $phase_k$ ends with the movement of one of the executing tasks to the join point, at which point the corresponding job moves to $phase_{k-1}$ of its response time.

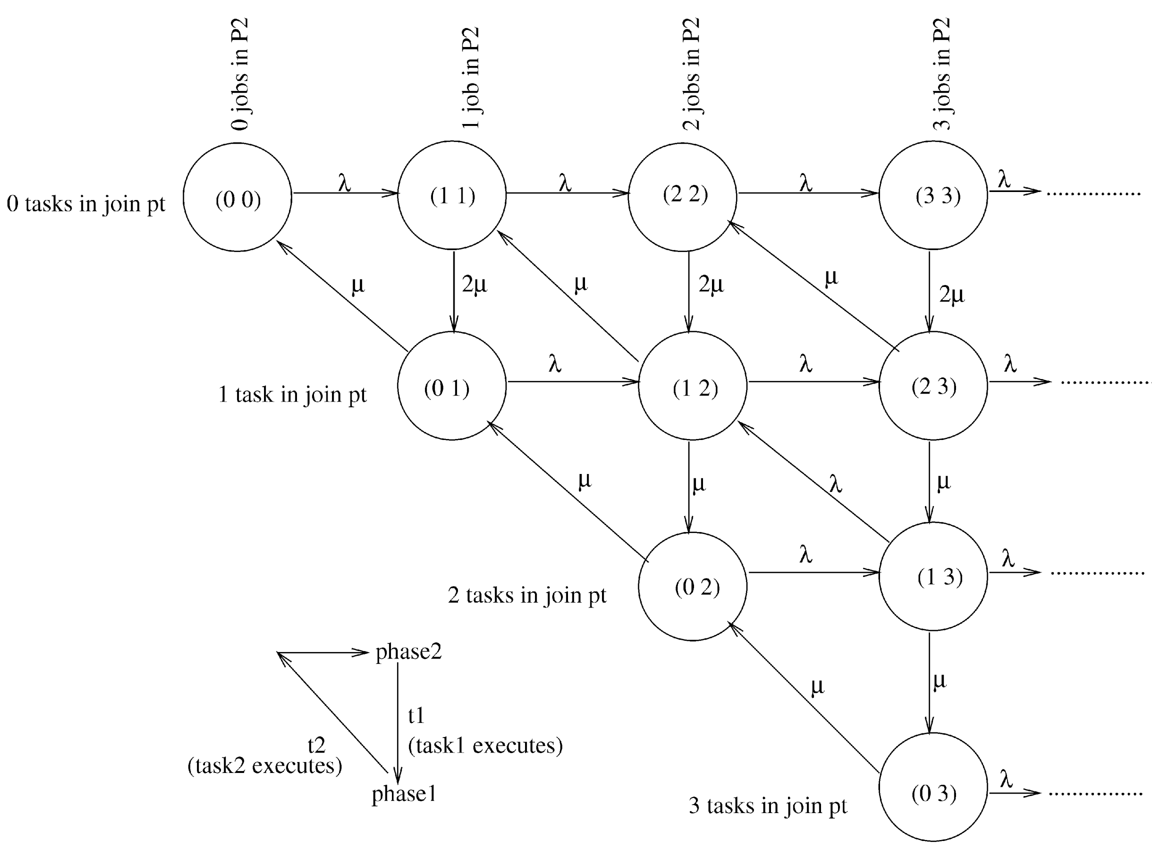


Fig. 3. Markov diagram of P_2.

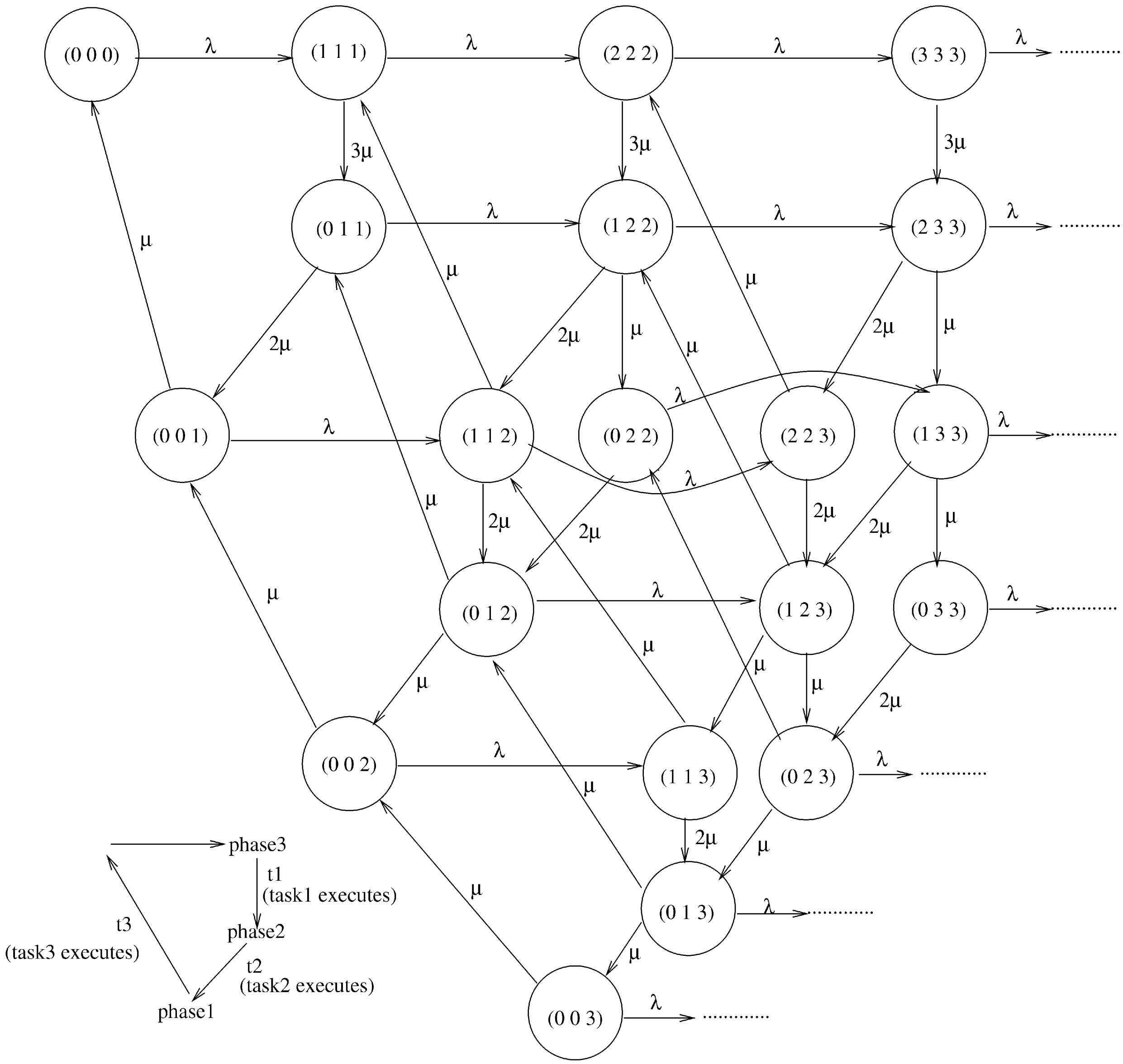


Fig. 4. Markov diagram of P_3.

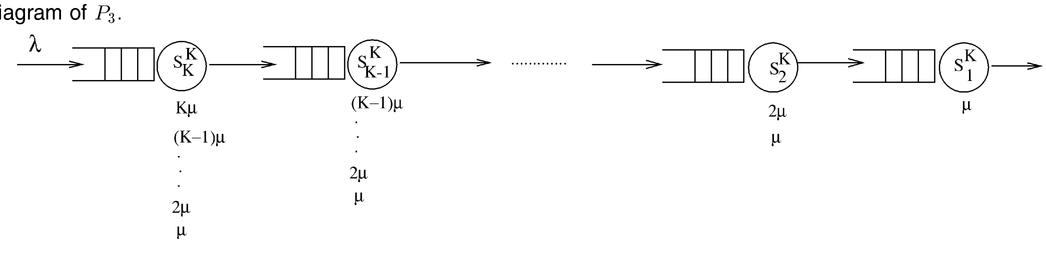


Fig. 5. Equivalent state-dependent model of P_K.

The time spent completing each phase of a job's response time in P_K can be viewed as the time spent getting service at K nonparallel subsystems $S_K^K, S_{K-1}^K, \cdots, S_1^K$ (or, $S_K, S_{K-1}, \cdots, S_1$ for notational simplicity), in order. A job at server S_k is in $phase_k$ of its response time. The parallel subsystem P_K can be mapped onto K nonparallel subsystems $S_K, S_{K-1}, \cdots, S_1$, as shown in Fig. 5. The mean service rate at service center S_k varies according to the number of customers at service centers $S_{k-1}, \cdots, S_2$, and S_1 as analyzed from the Markov process for P_K. (Refer to Appendix A for details.)

Let n_{S_k} represent the number of jobs in S_k. By construction, n_{S_k} represents the number of jobs in P_K with k active tasks. A state, $(n_{S_K}; n_{S_{K-1}}; n_{S_{K-2}}; \cdots; n_{S_2}; n_{S_1})$, of the state-dependent model is equivalent to the state

$$(n_{S_K}, n_{S_K} + n_{S_{K-1}}, n_{S_K} + n_{S_{K-1}} + n_{S_{K-2}}, \cdots,$$
$$n_{S_K} + .. + n_{S_3} + n_{S_2}, n_{S_K} + .. + n_{S_2} + n_{S_1})$$

of P_K. For purposes of distinction, the separator ";" is used between the elements of the vector representing a state of the state-dependent model, whereas the separator "," is

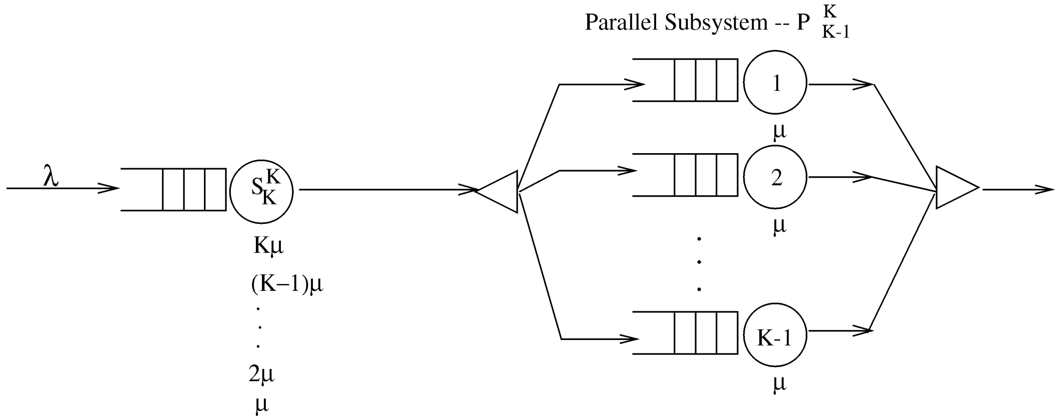


Fig. 6. Equivalent hybrid model of P_K.

used between the elements of a vector representing a state of P_K. The next example illustrates this mapping.

Example 2. State (3, 3, 3, 3) of P_4 represents the situation when all tasks of the three jobs within P_4 are at the servers. Thus, all three jobs are in the first phase ($phase_4$) of their response time within P_4 which implies that all three jobs are at S_4. This state is equal to the state (3; 0; 0; 0) of the state-dependent model.

State (0, 1, 2, 3) of P_4 represents the situation with job_1 in $phase_1$ of its response time with one active task; job_2 in $phase_2$ of its response time with two active tasks; job_3 in $phase_3$ of its response time with three active tasks. Thus, job_1 is at server S_1, job_2 is at S_2, and job_3 is at S_3. This is equivalent to state (0; 1; 1; 1) of the state-dependent model.

Hybrid Model. The hybrid model is a combination of the parallel subsystem and the state dependent model. This model is based on the fact that, once the first task of a job arriving at P_K is executed, the behavior of the remaining $K-1$ tasks of this job can be modeled by a P_{K-1} subsystem. The response time of the first task of the job is modeled by the state-dependent server S_K. Thus, the servers S_K and P_{K-1} of the hybrid system are equivalent to the subsystem P_K (Fig. 6).

In general, once the first k tasks of a job are executed, the behavior of the remaining tasks of the job can be modeled by a P_{K-k} subsystem. The response time of the first k tasks can be modeled by state-dependent servers $S_K, S_{K-1}, \cdots, S_{K-k+1}$. Thus, the system containing servers $S_K, S_{K-1}, \cdots, S_{K-k+1}$ and P_{K-k} are equivalent to subsystem P_K. In general, state $(n_{S_K}; n_{S_{K-1}}; \cdots; n_{S_2}; n_{S_1})$ of the state-dependent model is equivalent to state

$$(n_{S_K}; n_{S_{K-1}}; \cdots; n_{S_k}; (n_{S_{k-1}}, n_{S_{k-1}} + n_{S_{k-2}}, \cdots, n_{S_{k-1}} + .. + n_{S_2} + n_{S_1}))$$

of a hybrid model containing $S_K, \cdots, S_k$ and P_{k-1}. The next example illustrates this mapping.

Example 3. State (3, 3, 3, 3) of P_4 is equivalent to state (3; 0; 0; 0) of the state-dependent model as shown in Example 2. This state is equal to state (3; (0, 0, 0)) of the hybrid model containing S_4, P_3. This state is also equal to state (3; 0; (0, 0)) of the hybrid model containing S_4, S_3, P_2.

State (0, 1, 2, 3) of P_4 is equivalent to state (0; 1; 1; 1) in the state-dependent model as shown in Example 2. The first task of all three jobs are at the join point, so there are no jobs at S_4. Since all three jobs have less than four active tasks, their state can be represented by states in P_3. This state is equal to state (0; (1, 2, 3)) of the hybrid model containing S_4, P_3.

Job_3 has three executing tasks and is in $phase_3$ of its response time. Thus, job_3 is at server S_3. Both job_2 and job_1 have less than three active tasks and, therefore, their state can be represented by states in P_2. This state is equal to state (0;1;(1,2)) of the hybrid model containing S_4, S_3, P_2.

Equivalence of Models. The serial-join model has been established to be equivalent to P_K in [21]. In this section, two other alternate representations of P_K are shown, namely, the state-dependent model and the hybrid model. Now, every state in P_K can be mapped to a state in the state-dependent model and a state in the hybrid model and vice-versa. By construction, the rates along the transition arcs in the Markov diagrams for P_K, the state-dependent model, and the hybrid model are all equal. Hence, the parallel subsystem P_K, its state-dependent model, and its hybrid model have identical Markov processes and are equivalent models. The Markov diagrams of the state-dependent model and the hybrid model are shown in the Appendix in Fig. 12 and Fig. 13, respectively.

2.2.2 Formal Derivation

The response time expression is derived by equating R_{P_K}, the mean response time of P_K, to response times of the equivalent models as shown below.

1. $R_{P_K} = R_{S_a} + R_{S_j}$ of the serial-join model.
2. $$R_{P_K} = R_{S_K} + R_{S_{K-1}} + \cdots + R_{S_1}$$
   of the state-dependent model.
3. $R_{P_K} = R_{S_K} + R_{P_{K-1}}$ of the hybrid model.

Next, the response time of servers S_a and S_j of the serial-join model are equated to parameters in the state-dependent model and the hybrid model as explained in Lemmas 2.1 and 2.2.

Lemma 2.1. $R_{S_a} = s[1 + A_{S_a}]$, *where*

$$A_{S_a} = A_{S_K S_K} + \frac{K-1}{K} A_{S_K S_{K-1}} + \cdots + \frac{2}{K} A_{S_K S_2} + \frac{1}{K} A_{S_K S_1}.$$

Here, A_{S_a} refers to the mean number of jobs in S_a seen just prior to arrival at S_a and $A_{S_K S_i}$ represents the mean number of jobs in S_i seen by a job just prior to arrival at S_K.

Proof. Given in Appendix B. □

Lemma 2.2. $R_{S_j} = \frac{1}{2} R_{S_1^K}$, *when* $K = 2$

$$= \frac{1}{K}\left[R_{P_{K-1}^K} + R_{P_{K-2}^K} + \cdots + R_{P_2^K} + R_{S_1^K}\right], \forall K > 2.$$

Proof. Given in Appendix C. □

Next, R_{P_K} is written as the sum of R_{S_a} and R_{S_j}. This leads to R_{P_K} being expressed in terms of H_K and the arrival instant queue lengths at subsystems in the state-dependent model as shown in Lemma 2.3.

Lemma 2.3. $R_{P_K} = s\left[H_K + AQ_K^K\right]$, *where*

$$\begin{aligned}
AQ_k^K &= \\
&\left(A_{S_k^K S_k^K} + \frac{k-1}{k} A_{S_k^K S_{k-1}^K} + \cdots + \frac{1}{k} A_{S_k^K S_1^K}\right) \\
&+ \frac{1}{k}\left(A_{S_{k-1}^K S_{k-1}^K} + \frac{k-2}{k-1} A_{S_{k-1}^K S_{k-2}^K} + \cdots + \frac{1}{k-1} A_{S_{k-1}^K S_1^K}\right) \\
&+ \frac{1}{k-1}\left(A_{S_{k-2}^K S_{k-2}^K} + \frac{k-3}{k-2} A_{S_{k-2}^K S_{k-3}^K} + \cdots + \frac{1}{k-2} A_{S_{k-2}^K S_1^K}\right) \\
&+ \frac{1}{k-2}\left(A_{S_{k-3}^K S_{k-3}^K} + \frac{k-4}{k-3} A_{S_{k-3}^K S_{k-4}^K} + \cdots + \frac{1}{k-3} A_{S_{k-3}^K S_1^K}\right) \\
&+ \cdots + \frac{1}{3}\left(A_{S_2^K S_2^K} + \frac{1}{2} A_{S_2^K S_1^K}\right) + \frac{1}{2}\left(A_{S_1^K S_1^K}\right),
\end{aligned}$$

where $A_{S_i S_j}$ represents the mean number of jobs in S_j seen by a job just prior to arrival at S_i.

Proof. Given in Appendix D. □

The exact response time equation given in the above lemma is complicated due to the term AQ_K which refers to parameters in the state-dependent model. The next lemma addresses this issue.

Lemma 2.4. $AQ_K \leq A_{P_K}$.

Proof. Given in Appendix E. □

The main result of the paper follows directly from Lemmas 2.3 and 2.4.

Theorem 2.1. *For parallel subsystems with homogeneous, exponential servers in open and closed networks,*

$$R_{P_K} \leq s[H_K + A_{P_K}],$$

where R_{P_K} is the mean response time of P_K, s is the mean service time of a server within P_K, H_K is the Kth harmonic number, and A_{P_K} is the mean number of jobs in P_K seen by an arriving job.

Corollary 2.1. *For a closed network containing a stand-alone P_2, $R_{P_2} = s[H_2 + A_{P_2}]$.*

Proof. Given in Appendix F. □

3 Extension of the Fork-Join Model

In the model considered so far, each job arriving at P_K is constrained to fork into K tasks, which are assigned to the K service centers of P_K. This constraint is removed here. A job arriving at P_K could fork into an integral number of tasks less than or equal to K and these tasks are uniquely assigned to the K service centers (with at most one task of the same job assigned to a server) based on some known probability distribution. Thus, in this model, the probability that a server within P_K is assigned a task of an arriving job is less than 1.0. The advantage of this model is that it more accurately represents parallel systems like RAID disks, where an incoming request can be distributed across an arbitrary number of disks.[1] The next example is used to explain the model and the notation.

Example 4. Consider a network containing P_2.

Derivation of Task Visit Probability, v_c. v_c is the probability that a service center within P_K is assigned a task of a job arriving at P_K.

Case 1. Suppose a job arriving at P_2 always forks into two tasks that are assigned to the two servers within P_2. In this case, whenever P_2 is visited by a job, each of the service centers within P_2 is visited by a task of the job and $v_c = 1.0$.

Case 2. Now, consider the case when a job arriving at P_2 splits into one task (i.e., does not fork). Suppose this one task could be assigned to either of the two service centers within P_2 with equal probability. In this case, whenever a job visits P_2, there is only a 50 percent chance that its task will visit a service center within P_2 and $v_c = 0.5$.

Case 3. Finally, suppose a job arriving at P_2 splits into one or two tasks with equal probability. That is, there is a 50 percent chance that the job will split into two tasks and $v_c = 1.0$ in this case. There is also a 50 percent chance that the job will split into one task and $v_c = 0.5$ in this case. The overall task visit probability of a center within P_2 is given by

$$v_c = 0.5 * 1 + 0.5 * 0.5 = 0.75.$$

Derivation of Order-Statistic Scaling Factor, O_K, **for the Extended Model**. Let the mean service time of a server be s. Then, $s * O_K$ represents the mean time taken to execute all the tasks of a job arriving at P_K.

When a job always forks into two tasks, the time taken to execute both tasks of the job is given by sH_2. When a job always splits into one task, the time taken to execute this one task is given by $s = sH_1$.

When a job splits into one or two tasks with equal probability, the mean time to execute the tasks of this job is

1. We thank an anonymous reviewer for suggesting this extension to us.

given by $s[0.5 * H_1 + 0.5 * H_2] = s * 1.25$. Here, $O_K = 1.25$.

It can be proven that $R_{P_K} \leq s[O_K + A_{P_K}]$ for the extended model given here. (Note that the proof gets very "messy" due to the notational complexity.) An argument, similar to that provided in Section 2.1, for a much tighter response time equation, is presented here. Let A_{P_K} represent the mean number of jobs seen in P_K by an arriving job. Then, $v_c * A_{P_K}$ represents the mean number of tasks seen at the longest queue from among the K server queues. If the wait time of a job is defined to be the time taken for the last task of this job to start executing, the wait time must be at least equal to $s * v_c * A_{P_K}$. The mean service time of a job only begins when its last task starts service. Consequently, the job service time will be at most equal to $s * O_K$ since some tasks of the job may have finished execution and be at the join point when the job starts its service. This argument suggests that the response time expression (i.e., (1)) for the extended model can be written as:

$$R_{P_K} \approx= s[O_K + v_c * A_{P_K}].$$

A formal proof for this tighter approximation is required. In [30], this approximation is used to develop a performance model for RAID level 5 disk arrays.

4 Applications of Response Time Expression to Closed Networks

Equation (1) is simple and intuitive since it relates the response time of a job in a parallel system to its service time and wait time. However, there is one unknown argument in the equation, namely, A_{P_K}, the number of waiting jobs seen upon arrival. In this section, exact and approximate values for A_{P_K} are provided for P_K in closed networks.

4.1 Closed Networks Containing Stand-Alone P_K

The simplest case is a closed network containing just P_K. Let the number of jobs circulating in the closed network be m. In this case, $A_{P_K}(m)$, the mean number of jobs seen by a job just prior to arrival at P_K when there are m jobs circulating in the network, is equal to $m-1$. The next theorem follows immediately from Theorem 2.1 and Corollary 2.1.

Theorem 4.1. *For* $K = 2$, $R_{P_2}(m) = s[H_2 + (m-1)]$. *For* $K > 2$, $R_{P_2}(m) < R_{P_K}(m) \leq s[H_K + (m-1)]$.

The tightness of the optimistic (lower) and pessimistic (upper) bounds are studied by comparing them with response time values obtained using simulation. The response times are plotted for values of K ranging from $2, \cdots, 10, 15, 20, 25, 30, 35, 40, 45$ and m ranging from $1, \cdots, 10, 15, 20, 25, 30, 35, 40, 45, 50$. The mean service time is set at 1.00 time unit. The simulated mean response time estimate is accurate within 0.5 time units at 95 percent confidence. Fig. 7 plots the upper, lower, and simulated mean response times for fixed values of K as m varies. The graphs show that for a given K, the pessimistic and optimistic bounds grow at the same rate as R_{P_K} and the relative error (i.e., difference between the simulated and model values) remains approximately constant. This is further verified by graphs (c) and (d) of Fig. 8 which plot the relative error in response times for the pessimistic and optimistic bounds for fixed K as m varies. The relative error between the bounds and the simulated response time increases till $m \approx 5$ and then becomes constant. Graphs (a) and (b) of Fig. 8 plot the percentage error (where percentage error $= \frac{model-simulated}{simulated}$ percent) in the response time bounds. The maximum percentage error in the pessimistic and optimistic bound is around 9 and 30 percent, respectively, and occurs when $K = 45$ and $m = 5$. However, for the majority of values, the percentage error is less than 3 and 10 percent for the pessimistic and optimistic bounds, respectively. For fixed K, the percentage error increases sharply till $m \approx 5$ and then decreases as m increases. Fig. 9a plots the pessimistic and simulated response times for fixed values of m as K varies from 2 to 45.

Fig. 9b plots the model and simulated mean response times for a system based on the extended model given in Section 3. The graph given here is based on a system where a job arriving at P_K can fork into an arbitrary number of tasks with equal probability and these tasks can be assigned to any of the adjoining centers with equal probability. For the system analyzed, the response time values generated by the model are always an optimistic bound and the values are very close. This model will be potentially relevant in studying striping policies for RAID disk systems.

4.2 Closed Networks Modeling Several Devices

Computer systems contain a variety of devices (e.g., CPUs, memory, cache) connected together and all of these components must be represented in order to accurately model the system. These devices need not all be parallel. The service time distribution of centers in the nonparallel devices have the same restrictions imposed on them as do product-form networks [5]. The service time distributions of centers in the parallel devices are drawn from an exponential distribution (this restriction can be dropped as shown in the next subsection). Let R_i and Q_i represent the mean response time and the mean queue length of device i in the network. For such a network, it is observed that

$$A_i(m) \approx= Q_i(m-1).$$

That is, the mean arrival queue length of a subsystem within the network (whether parallel or nonparallel) is approximately equal to the mean queue length of the system when the multiprogramming level of the network is one less. This relationship has only been verified by solving small systems exactly and by running simulations [28]. Suppose there are N subsystems in the closed network. From the above equation, it follows that for such a network, the cycle time (i.e., sum of response times of all the subsystems) of the network is given by:

$$\sum_{i=1}^{N} R_i(m) \approx= \sum_{i=1}^{N} s_i[H_{K_i} + Q_i(m-1)].$$

(Note that for nonparallel devices, $K_i = 1$ and $H_{K_i} = 1$.) This equation can be used as the basis for the Mean Value Technique for computing approximate mean perfor-

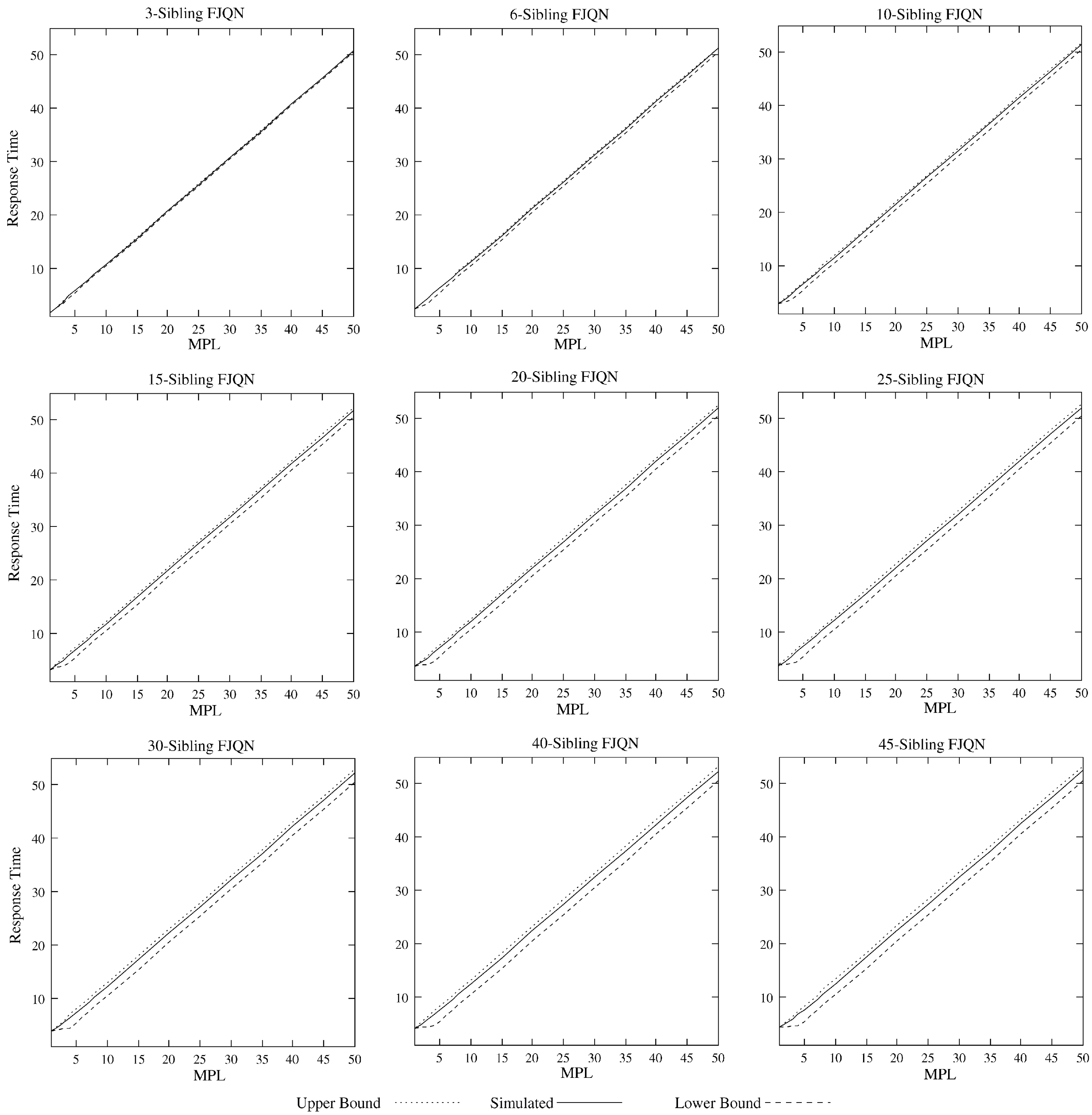


Fig. 7. Response times (upper, simulated, and lower) at varying MPL for P_K.

mance measures of a fork-join parallel network. The MVA technique for fork-join networks is given in [28] and graphs from that work are reproduced in Fig. 10. The technique is similar to the MVA technique for product-form networks [22] which is a computationally efficient technique that allows for easier parameterization of the devices being modeled [14].

Two quick bounding techniques based on (1) for fork-join parallel networks are given in [29]. These techniques are computationally simple and can be calculated by hand, even for large networks modeling several devices and jobs. Explaining these techniques is beyond the scope of this paper and an interested reader is referred to [29].

4.3 P_K with Nonexponential Service Times

While the main thrust of this paper has been to develop bounds/approximations for the specific functional form of exponential service time distribution, the intuitive argument given in Section 2.1 is independent of the functional form of the service type distribution. Here, this approximation is validated by simulations employing nonexponential service time distributions.

Fig. 11 plots the mean response times for the Erlang and Hyper-Exponential distributions (which have the property of coefficient of variation $(CV)^2$ lower and greater than one, respectively). The simulation results indicate that the bounds work very well for such distributions within the

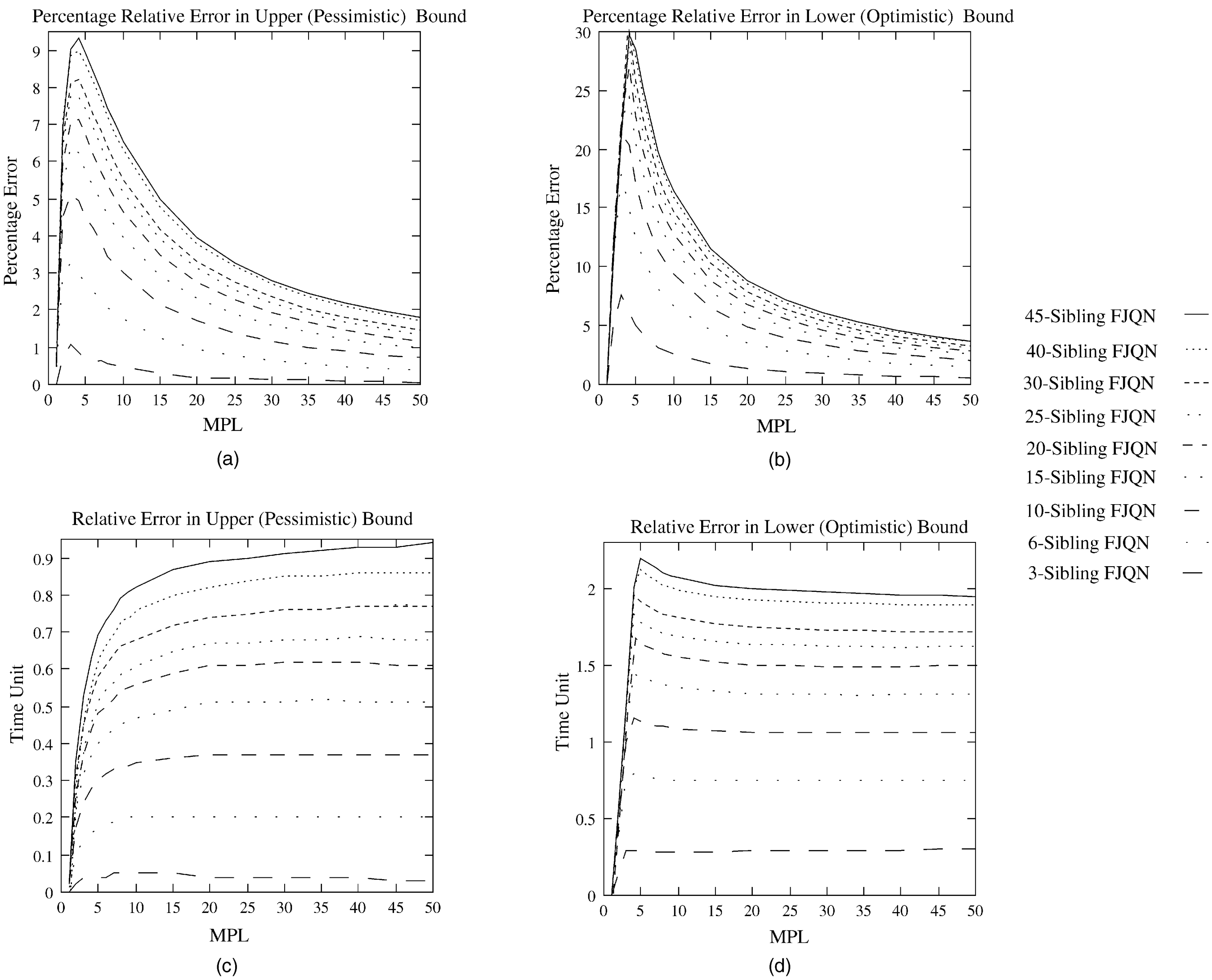


Fig. 8. Error bounds on response time at varying MPL for P_K.

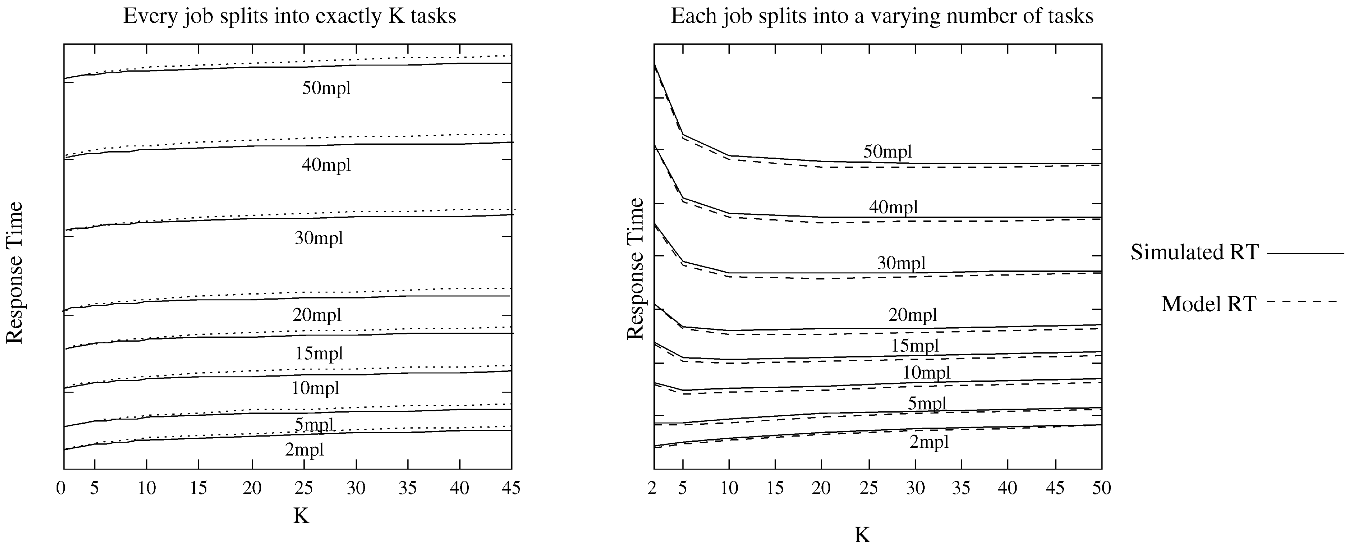
range of CV's simulated (0.5 and 1.5). The simulated mean response time is accurate within 0.5 time units at 90 percent confidence.

Summarizing, applications of (1) pertaining to different types of closed networks are given here. We are yet to find a good approximate value for the parameter A_{P_K} (i.e., the longest arrival queue length at a server) in the case of open networks.

5 Prior Work

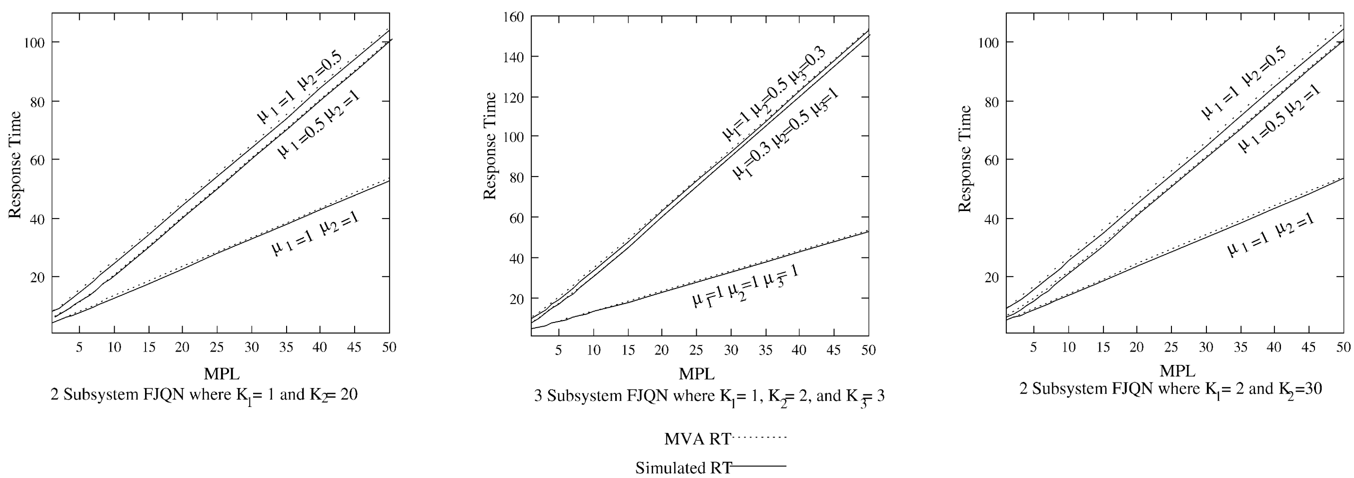
Several papers study fork-join queueing networks and propose tools for analyzing their performance. Exact solutions for the steady state distribution have been provided only for 2-sibling cases ([2], [8], [9]) in open networks. Due to the difficulty of analyzing fork-join models exactly, many studies on fork-join queues concentrate on approximation techniques. Heidelberger and Trivedi [10] consider a closed queueing network in which jobs divide into two or more asynchronous tasks. The join point is not modeled. The service centers are of a type described in the BCMP theorem [5]. They develop an iterative method for solving a sequence of product-form models. In [11], the model is expanded to include a join node. Nelson and Tantawi [21] consider a scaling approximation technique to analyze the mean response time of an open homogeneous fork-join queue with exponential service time distributions. They assume that the mean response time increases at the same rate as the number of sibling tasks. Closed-form approximation expressions for the mean response time are developed. An extension of this approximation to heavy traffic limit relying on a light traffic interpolation technique is developed by Makowski and Varma [20]. Kim and Agrawala [12] analyze waiting times for 2-sibling open, homogeneous fork-join systems with exponential and 2-stage Erlangian service time distributions. In [18], [19], Lui et al. present a bounding technique for an open, homogeneous fork-join network with a k-stage Erlang distribution. Liu and Perros [16], [17] propose an approximation procedure based on decomposition and aggregation for analyzing a closed queueing system with K-sibling fork-join queues. Their method provides an upper bound for mean response time. Response time bounds are obtained for acyclic fork-join queueing networks by Baccelli et al. [3]

2. CV = standard deviation/mean.



(a) (b)

Fig. 9. Response times (model and simulated) for the standard and extended model.



(a) (b) (c)

Fig. 10. Cycle times (simulated and MVA) for fork-join networks modeling several devices.

using stochastic ordering principles and association of random variables. Baccelli and Liu [4] propose a new class of queueing models for evaluating the performance of parallel systems. Using the concept of associated random variables, Kumar and Shorey [13] obtain response time bounds for an open fork-join model in which a job forks into a random number of tasks. Service times are drawn from a general distribution. Almeida and Dowdy [1] propose an iterative technique for obtaining lower performance bounds of closed fork-join networks with exponential service times. No proofs for the technique are presented. Varki and Dowdy [27] prove that the proportion of the number of jobs in the different subsystems of a closed, exponential, balanced fork-join network remains constant irrespective of the multiprogramming level. This property of balanced fork-join networks is used to bound the performance of arbitrary fork-join networks. The proof is limited to 2-server fork-join systems. In [28], Varki develops an approximate mean-value analysis technique for fork-join parallel networks. Balsamo et al. [6] propose a matrix-geometric algorithmic approach for computing performance bounds of open heterogeneous fork-join systems. The fork-join structure is studied with relation to parallel storage systems (RAID) in [15], [23], [24], [25].

6 Conclusion

The focus of this paper has been to derive a simple approximation for mean response time of a parallel system. This response time approximation is shown to be appropriate both when jobs arriving at a parallel system split into an arbitrary number of tasks and when several devices are contained in the network. While the paper formally derives the response time approximation only for the case of exponential service time distributions, simulation results indicate that the approximation is also valid for other distributions such as the Erlang and Hyper-Exponential distributions.

Though the response time expression derived here applies to both closed and open networks, one of the

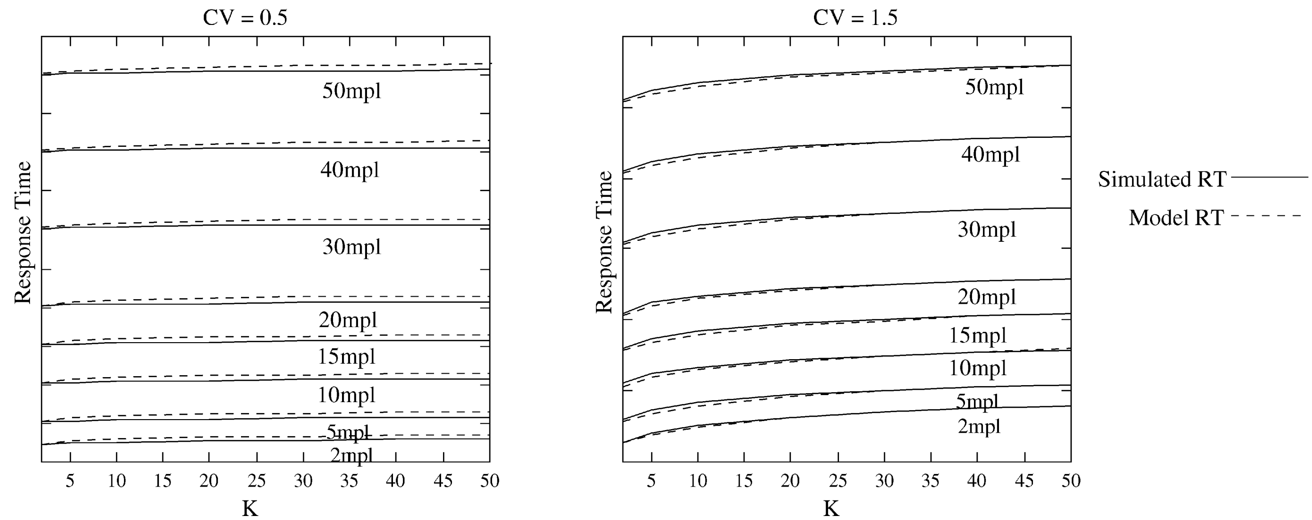


Fig. 11. Response times (simulated and model) for parallel systems with nonexponential service time distributions.

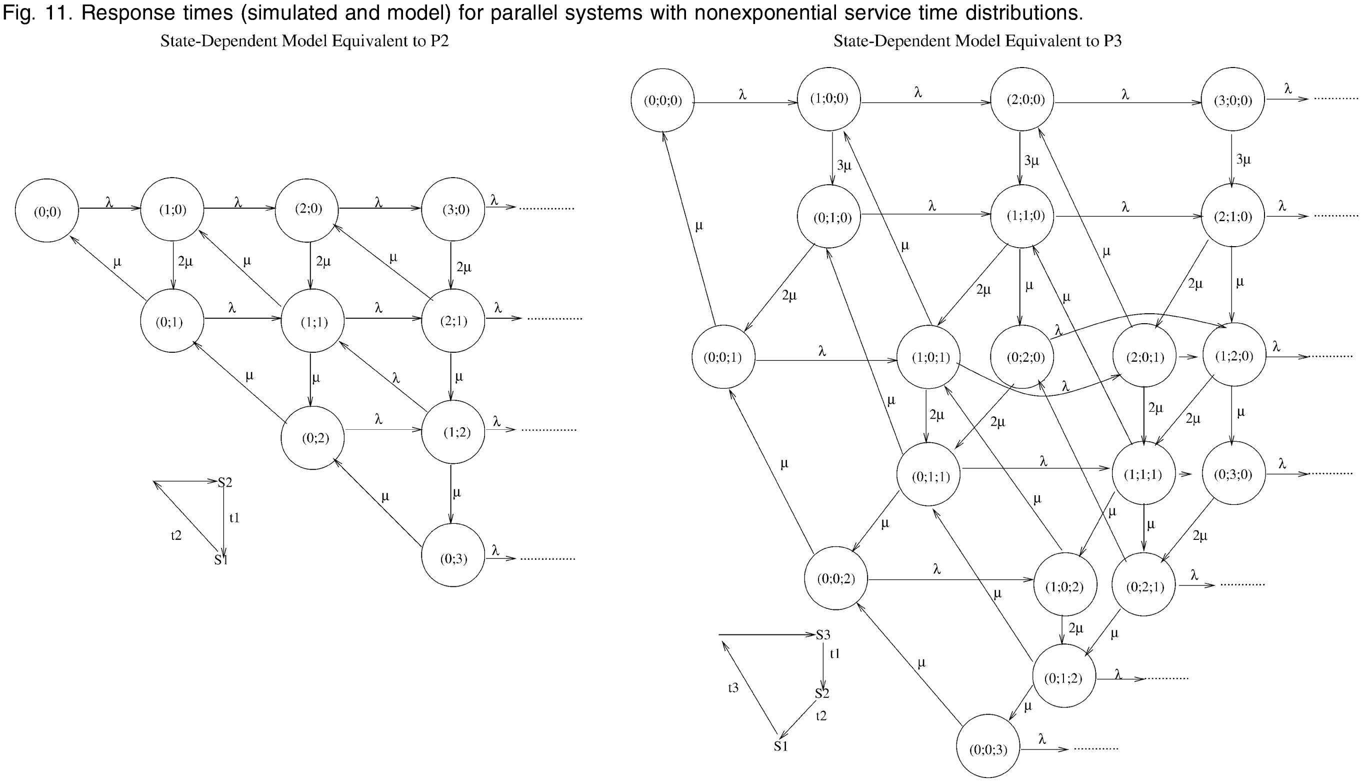


Fig. 12. Markov diagram of the state-dependent model.

arguments A_{P_K}, the longest arrival queue length at the K servers, is unknown in the case of open-networks and, hence, no performance techniques for open networks have been provided in this paper. Future research that addresses this limitation of the paper is needed. Still, there are several applications of this work that could be of value to performance engineers. For example, in storage systems, the performance of disk arrays under synchronous I/O workloads (represented by closed networks) can be analyzed using the approximation. Further, the performance of parallel programs on multiprocessor systems with fixed multiprogramming levels can be analyzed using the response time approximation derived here.

APPENDIX A

Mean service rate at a service center S_k in the state-dependent model can be found by analyzing the Markov diagram of P_K.

The mean service rate at service center S_k varies according to the number of customers at service centers S_i, $i = k-1, k-2, \cdots, 1$ as follows: If there is at least one customer in service center S_{k-1}, the mean service rate at S_k equals μ, else, if there is at least one customer in service center S_{k-2}, the mean service rate at S_k equals 2μ, else, if there is at least one customer in service center S_{k-3}, the mean service rate at S_k equals 3μ, $\cdots$, else, if there is at least one customer in service center S_1, the mean service rate at S_k equals $(k-1)\mu$, else (if there are no customers in $S_{k-1}, S_{k-2}, \cdots, S_1$), the mean service rate at S_k equals $k\mu$. The service rate at server S_1 is equal to μ. Thus, service rates

TABLE 1
The Notation

Description of P_K	P_K	The K-sibling parallel system
	μ^{-1}, s	Mean service time per visit to a center in P_K
	A_{P_K}	Mean number of jobs seen upon arrival at P_K. Also equal to the number of tasks seen at the longest queue from amongst the K server queues.
	n_i	Number of tasks at service center i
	$(n_1, \cdots, n_K)$	State of P_K. Since all centers are identical, the n_i's are ordered such that $n_1 \leq n_2 \leq \cdots \leq n_K$.
	$task_k$	k^{th} task of a job to complete execution and move to the join point.
Description of Equivalent Models	P_k^K	A k-sibling parallel subsystem in a hybrid network equivalent to P_K
	S_i, S_i^K	A non-parallel subsystem i in an equivalent model of P_K
	n_{S_k}	Number of jobs at state-dependent server S_k.
	$(n_{S_K}; \cdots; n_{S_1})$	State of state-dependent model $S_K, S_{K-1}, \cdots S_1$.
	AQ_k^K	The sum of mean arrival queue lengths at P_k^K during the various phases of a parallel job's response time
Performance Measures	R_i	Mean response time of subsystem i
	Q_i	Mean queue length of subsystem i
	A_i	Mean queue length of system i seen by a job just prior to arrival at i
	A_{ij}	Mean queue length of system j seen by a job just prior to arrival at i
General Parameters	m	Multiprogramming level of a closed network
	H_k	The kth harmonic number defined as $\sum_{i=1}^{k} \frac{1}{i}$

at all servers S_k, $k = K, \cdots, 2$ are dependent on the states at the service centers S_i $(i = k-1, \cdots, 1)$. Only server S_1 is state independent.

Appendix B
Proof for Lemma 2.1

Proof. Since S_a is a nonparallel subsystem with mean service time s, its response time is given by $R_{S_a} = s[1 + A_{S_a}]$ where A_{S_a} represents the mean number of jobs in S_a seen by a job just prior to arrival at S_a.

Let $task_k$ represent the kth task of a job to complete execution and move to the join point. The average performance measure of S_a is equal to the average performance measures of the service centers serving $task_1, task_2, \cdots, task_K$. Thus, A_{S_a} equals the mean arrival queue lengths at service centers serving $task_1, task_2, \cdots, task_K$ seen by a job just prior to arrival at P_K.

In the state-dependent model, server S_k services $phase_k$ of a parallel job's response time. Recall that during $phase_k$ of a parallel job's response time, k tasks of the job are at the service center queues while the remaining $(K-k)$ tasks wait at the join point. Therefore, $task_k$ will execute only at service centers $S_K, S_{K-1}, \cdots, S_{K-k+1}$. Let $A_{P_K task_k}$ represent the mean arrival queue length at the service center serving $task_k$ and let $A_{S_K S_i}$ represent the mean number of jobs in S_i seen by a job just prior to arrival at S_K. Then, $A_{P_K task_k}$ is equal to the sum $A_{S_K S_K} + A_{S_K S_{K-1}} + \cdots + A_{S_K S_{K-k+1}}$. (For example, the average arrival queue lengths at the service centers that service the first and last task to complete execution are equal to $A_{S_K S_K}$ and

$$A_{P_K} = A_{S_K S_K} + A_{S_K S_{K-1}} + \cdots + A_{S_K S_1},$$

respectively.) Thus,

$$\begin{aligned} A_{S_a} =& \frac{1}{K}[A_{P_K task_1} + A_{P_K task_2} + \cdots + A_{P_K task_K}] \\ =& \frac{1}{K}[(A_{S_K S_K}) + (A_{S_K S_K} + A_{S_K S_{K-1}}) \\ &+ \cdots + (A_{S_K S_K} + \cdots + A_{S_K S_1})] \\ =& A_{S_K S_K} + \frac{K-1}{K} A_{S_K S_{K-1}} + \cdots + \frac{2}{K} A_{S_K S_2} + \frac{1}{K} A_{S_K S_1}. \end{aligned}$$

□

Appendix C
Proof for Lemma 2.2

The average time spent by a job in the join point is equal to the sum of the average times spent by each of its tasks in the join point. This gives:

$$\begin{aligned} R_{S_j} =& \frac{1}{K}[(R_{task_K} - R_{task_1}) + (R_{task_K} - R_{task_2}) \\ &+ \cdots + (R_{task_K} - R_{task_K})], \end{aligned}$$

where R_{task_k} represents the mean response time of the kth task of a job to complete execution.

By construction, $R_{task_k} = R_{S_K} + R_{S_{K-1}} + \cdots + R_{S_{K-k+1}}$. Thus,

$$\begin{aligned} R_{S_j} =& \frac{1}{K} R_{S_{K-1}} + \frac{2}{K} R_{S_{K-2}} + \cdots + \frac{K-2}{K} R_{S_2} + \frac{K-1}{K} R_{S_1}, \\ &\forall K > 2 \\ =& \frac{1}{K}[(R_{S_{K-1}} + \cdots + R_{S_1}) + (R_{S_{K-2}} + \cdots + R_{S_1}) \\ &+ \cdots + (R_{S_2} + R_{S_1}) + R_{S_1}] \\ =& \frac{1}{K}\left[R_{P_{K-1}^K} + R_{P_{K-2}^K} + \cdots + R_{P_2^K} + R_{S_1^K}\right]. \end{aligned}$$

□

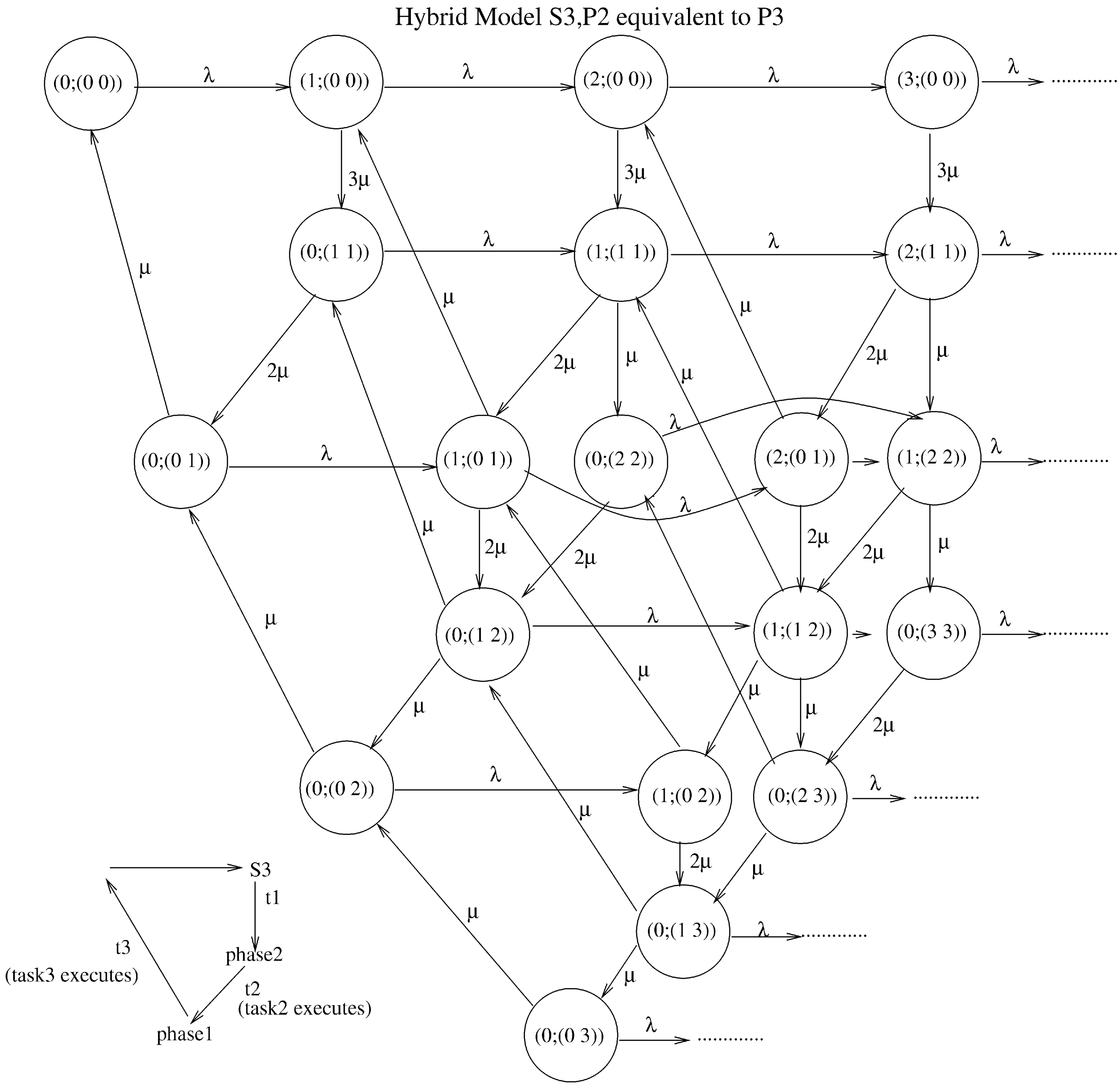


Fig. 13. Markov diagram of the hybrid model.

Appendix D

Proof for Lemma 2.3

Proof.

$$\begin{aligned} R_{P_K} =& R_{S_a} + R_{S_j} \\ =& s[1 + A_{S_a}] + \frac{1}{K}\Big[R_{P^K_{K-1}} + R_{P^K_{K-2}} \\ &+ \cdots + R_{P^K_2} + R_{S^K_1}\Big] \end{aligned}$$

from Lemmas 2.1 and 2.2.

Induction on P_K is used to prove the lemma.
Induction Basis $\boldsymbol{P^K_2 (\forall K \geq 2)}$.
In this case, $R_{S_j} = \frac{1}{2} R_{S^K_1} = \frac{1}{2\mu}\Big[1 + A_{S^K_1 S^K_1}\Big]$ and

$$R_{S_a} = \frac{1}{\mu}\left[1 + A_{S^K_2 S^K_2} + \frac{1}{2} A_{S^K_2 S^K_1}\right],$$

where $A_{S_i S_j}$ represents the mean number of jobs in S_j seen by a job just prior to arrival at S_i. It follows that

$$\begin{aligned} R_{P^K_2} &= \frac{1}{\mu}\left[1 + A_{S^K_2 S^K_2} + \frac{1}{2} A_{S^K_2 S^K_1}\right] + \frac{1}{2\mu}\Big[1 + A_{S^K_1 S^K_1}\Big] \\ &= H_2 + \frac{1}{\mu}\left[\left(A_{S^K_2 S^K_2} + \frac{1}{2} A_{S^K_2 S^K_1}\right) + \frac{1}{2} A_{S^K_1 S^K_1}\right] \\ &= \frac{1}{\mu}\left[H_2 + AQ^K_2\right]. \end{aligned}$$

Induction Hypothesis. Assume that the theorem holds for all $P^K_k, k = 3, \cdots, K-1$.

Induction Step. To show that the theorem holds for P_K, by hypothesis,

$$R_{S_j} = \frac{1}{K}\left[R_{P^K_{K-1}} + R_{P^K_{K-2}} + \cdots + R_{P^K_2} + R_{S^K_1}\right] = \frac{1}{K\mu}\left[\left(H_{K-1} + AQ^K_{K-1}\right) + \cdots + \left(H_2 + AQ^K_2\right) + \left(1 + A_{S^K_1 S^K_1}\right)\right].$$

This gives

$$R_{P_K} = \frac{1}{\mu}[1 + A_{S_a}] + \frac{1}{K\mu}\left[\left(H_{K-1} + AQ^K_{K-1}\right) + \cdots + \left(H_2 + AQ^K_2\right) + \left(1 + A_{S^K_1 S^K_1}\right)\right] = \frac{1}{\mu}\left[\left(1 + \frac{1}{K}\left(H_{K-1} + \cdots + H - 2 + 1\right)\right) + \left(A_{S_a} + \frac{1}{K}\left(AQ^K_{K-1} + \cdots + AQ^K_2 + A_{S^K_1 S^K_1}\right)\right)\right].$$

To show that $H_K = 1 + \frac{1}{K}(H_{K-1} + H_{K-2} + \cdots + H_2 + 1)$ and $AQ^K_K = A_{S^K_a S^K_a} + \frac{1}{K}(AQ^K_{K-1} + \cdots + AQ^K_2 + A_{S^K_1 S^K_1})$. Now,

$$H_K = 1 + H_{K-1} - \frac{K-1}{K} = 1 + \left(1 - \frac{1}{K}\right) + \left(\frac{1}{2} - \frac{1}{K}\right) + \cdots \left(\frac{1}{K-1} - \frac{1}{K}\right) = 1 + \frac{1}{K}(H_{K-1} + H_{K-2} + \cdots + H_2 + 1).$$

By a rearrangement of the elements of the series AQ^K_K, it can be show that

$$AQ^K_K = A_{S_a} + \frac{1}{K}(AQ^K_{K-1} + \cdots + AQ^K_2 + A_{S^K_1 S^K_1}.$$

Thus, $R_{P_K} = \frac{1}{\mu}\left[H_K + AQ^K_K\right]$. □

Appendix E

Proof for Lemma 2.4

Proof. Rearranging of the terms of AQ_K gives:

$$AQ_K = \frac{1}{K}(A_{S_K S_K} + A_{S_K S_{K-1}} + \cdots + A_{S_K S_1}) + \frac{1}{K}(A_{S_K S_K} + A_{S_{K-1} S_{K-1}} + A_{S_{K-1} S_{K-2}} + \cdots + A_{S_{K-1} S_1}) + \cdots + \frac{1}{K}(A_{S_K S_K} + A_{S_{K-1} S_{K-1}} + \cdots A_{S_3 S_3} + A_{S_2 S_2} + A_{S_2 S_1}) + \frac{1}{K}(A_{S_K S_K} + A_{S_{K-1} S_{K-1}} + \cdots + A_{S_2 S_2} + A_{S_1 S_1}).$$

By construction, $A_{P_K} = \sum_{i=1}^{K} A_{S_K S_i}$ where $A_{S_K S_j}$ represents the mean number of jobs in S_j seen by a job just prior to arrival at S_K. Thus, if it is shown for $k \geq 2$, $\sum_{i=1}^{k-1} A_{S_k S_i} \geq \sum_{i=1}^{k-1} A_{S_{k-1} S_i}$, the lemma is proven. This relationship becomes intuitively obvious when one observes that at any point in time the total number of arrivals to the service station S_k is always greater than or equal to the total number of arrivals to any of the service centers to its right (refer to Fig. 5). The formal proof given here is an adaptation of a corresponding result by P.J. Burke [7] (Chapter 5).

Let $T_{S_k}(\alpha)$ be the time instants at which the αth arrival occurs at S_k after some time $t = 0$. Let $N_{S_k S_j}(\alpha)$ be the total number of jobs at service stations $S_j, S_{j-1}, \cdots S_1$ (where $j \leq k$) just prior to the αth arrival instant at S_k. It is shown that

$$\lim_{n\to\infty} P\{N_{S_k S_{k-1}}(n) = x\} \geq \lim_{n\to\infty} P\{N_{S_{k-1} S_{k-1}}(n) = x\},$$

where $P\{x\}$ represents the probability of being in state x. Suppose $N_{S_k S_k}(n) = y$ and $N_{S_k S_{k-1}}(n) = x$. This implies that the number of jobs at S_k equals $y - x$ just prior to the nth arrival instant and there are $n + x - y - 1$ of the $T_{S_{k-1}}(\alpha)$ preceding $T_{S_k}(n)$. Thus, $N_{S_{k-1} S_{k-1}}(n + x - y) \leq x$, since some of the x jobs may have departed the set of service stations, $S_{k-1}, \cdots S_1$, by time instant $T_{S_{k-1}}(n + x - y)$. Hence, for any $y \geq x$,

$$\lim_{n\to\infty} P\{N_{S_k S_{k-1}}(n) = x\} \geq \lim_{n\to\infty} P\{N_{S_{k-1} S_{k-1}}(n + x - y) = x\};$$

that is,

$$\lim_{n\to\infty} P\{N_{S_k S_{k-1}}(n) = x\} \geq \lim_{n\to\infty} P\{N_{S_{k-1} S_{k-1}}(n) = x\}$$

and the proof is complete. □

Appendix F

Proof for Corollary 2.1

Proof. From Lemma 2.3, $R_{P_2} = s[H_2 + AQ_2]$, where

$$AQ_2 = A_{S_2 S_2} + \frac{1}{2}[A_{S_2 S_1} + A_{S_1 S_1}]$$

and $A_{S_i S_j}$ represents the mean number of jobs in S_j seen by a job just prior to arrival at S_i. There is a very general theorem in queuing theory (by P.J. Burke, 1968)[3] which states that: In any ''system'' (the actual nature of which is unimportant), provided that the number of ''customers'' it contains varies by at most one at a time, the probability distribution of the number of customers in the system is the same just prior to an arrival and just after a departure [7] (Chapter 5).

From this theorem, it follows that for a closed network containing stand-alone P_K, $A_{S_1 S_1} = A_{S_K S_1}$ (i.e., the arrival queue length at $S_1(A_{S_1 S_1})$ is equal to the departure queue length at $S_1(A_{S_K S_1})$). For a network containing just P_2, $A_{S_1 S_1} = A_{S_2 S_1}$, which gives

$$AQ_2 = A_{S_2 S_2} + A_{S_2 S_1}.$$

By construction,

$$A_{P_2} = A_{S_2 S_2} + A_{S_2 S_1}.$$

Hence, $R_{P_2} = s[H_2 + A_{P_2}]$. □

3. The theorem employed here is not *Burke's Theorem* which refers to the property that Poisson arrivals implies Poisson departures for a class of feed-forward product-form networks.

ACKNOWLEDGMENTS

The author thanks Larry Dowdy, Manish Madhukar, Sajeev Varki, the associate editor, and three anonymous reviewers for their helpful comments. This work was supported in part by the US National Science Foundation under grants ITR-0082399, CCR-0093111, and EIA-9974992.

REFERENCES

[1] V.A.F. Almeida and L.W. Dowdy, "A Reduction Technique for Solving Queueing Network Models of Programs with Internal Concurrency," *Proc. Third Int'l Conf. Supercomputing,* May 1988.

[2] F. Baccelli, "Two Parallel Queues Created by Arrivals with Two Demands: The M/G/2 Symmetrical Case," Technical Report 426, Inst. Nat'l de Recherche en Informatique et en Automatique, July 1985.

[3] F. Baccelli, W.A. Massey, and D. Towsley, "Acyclic Fork-Join Queueing Networks," *J. ACM,* vol. 36, no. 3, pp. 615-642, July 1989.

[4] F. Baccelli and Z. Liu, "On the Execution of Parallel Programs on Multiprocessor Systems—A Queueing Theory Approach," *J. ACM,* vol. 37, no. 2, pp. 373-414, Apr. 1990.

[5] F. Baskett, K.M. Chandy, R.R. Muntz, and F.G. Palacios, "Open, Closed, and Mixed Networks of Queues with Different Classes of Customers," *J. ACM,* vol. 22, no. 2, pp. 248-260, Apr. 1975.

[6] S. Balsamo, L. Donatiello, and N.M. Van Dijk, "Bound Performance Models of Heterogeneous Parallel Processing Systems," *IEEE Trans. Parallel and Distributed Systems,* vol. 9, no. 10, Oct. 1998.

[7] R.B. Cooper, *Introduction to Queueing Theory,* third edition, Maryland: Mercury Press/Fairchild, 1990.

[8] L. Flatto and S. Hahn, "Two Parallel Queues Created by Arrivals with Two Demands I," *SIAM J. Applied Math.,* vol. 44, pp. 1041-1053, Oct. 1984.

[9] L. Flatto, "Two Parallel Queues Created by Arrivals with Two Demands II," *SIAM J. Applied Math.,* vol. 45, pp. 861-878, Oct. 1985.

[10] P. Heidelberger and S.K. Trivedi, "Queueing Network Models for Parallel Processing with Asynchronous Tasks," *IEEE Trans. Computers,* vol. 31, no. 11, pp. 1099-1109, Nov. 1982.

[11] P. Heidelberger and S.K. Trivedi, "Analytic Queueing Models for Programs with Internal Concurrency," *IEEE Trans. Computers,* vol. 32, pp. 73-82, Jan. 1983.

[12] C. Kim and A.K. Agrawala, "Analysis of the Fork-Join Queue," *IEEE Trans. Computers,* vol. 38, no. 2, pp. 250-255, Feb. 1989.

[13] A. Kumar and R. Shorey, "Performance Analysis and Scheduling of Stochastic Jobs in a Multicomputer System," *IEEE Trans. Parallel and Distributed Systems,* vol. 4, no. 10, pp. 1147-1164, Oct. 1993.

[14] E.D. Lazowska, J. Zahorjan, G.C. Graham, and K.C. Sevcik, *Quantitative System Performance—Computer System Analysis Using Queueing Network Models.* Prentice-Hall, 1984.

[15] E.K. Lee and R.H. Katz, "An Analytic Performance Model of Disk Arrays" *Proc. ACM SIGMETRICS,* 1993.

[16] Y.C. Liu and H.G. Perros, "Approximate Analysis of a Closed Fork/Join Model," *European J. Operational Research,* vol. 53, pp. 382-392, 1991.

[17] Y.C. Liu and H.G. Perros, "A Decomposition Procedure for the Analysis of a Closed Fork/Join Queueing System," *IEEE Trans. Computers,* vol. 40, no. 3, pp. 365-370, Mar. 1991.

[18] J.C.S. Lui, R.R. Muntz, and D. Towsley, "Bounding the Response Time of a Minimum Expected Delay Routing System: An Algorithmic Approach," *IEEE Trans. Computers,* vol. 44, no. 5, pp. 1371-1382, May 1995.

[19] J.C.S. Lui, R.R. Muntz, and D. Towsley, "Computing Performance Bounds of Fork-Join Parallel Programs under a Multiprocessing Environment," *IEEE Trans. Parallel and Distributed Systems,* vol. 9, no. 3, pp. 295-311, Mar. 1998.

[20] A. Makowski and S. Varma, "Interpolation Approximations for Symmetric Fork-Join Queues," *Performance Evaluation,* vol. 20, pp. 145-165, 1994.

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

[22] M. Reiser and S.S. Lavenberg, "Mean-Value Analysis of Closed Multichain Queueing Networks," *J. ACM,* vol. 27, no. 2, pp. 313-322, Apr. 1980.

[23] A. Thomasian and A.N. Tantawi, "Approximate Solutions for M/G/1 Fork-Join Synchronization," *Proc. 1994 Winter Simulation Conf.,* pp. 361-368, Dec. 1994.

[24] A. Thomasian and J. Menon, "Approximate Analysis for Fork-Join Synchronization in RAID5," *Computer Systems: Science and Eng.,* 1997.

[25] A. Thomasian and J. Menon, "RAID5 Performance with Distributed Sparing," *IEEE Trans. Parallel and Distributed Systems,* vol. 8, no. 6, pp. 640-657, June 1997.

[26] K.S. Trivedi, *Probability and Statistics with Reliability, Queuing, and Computer Science Applications.* Prentice-Hall, 1982.

[27] E. Varki and L.W. Dowdy, "Analysis of Closed Balanced Fork-Join Queueing Networks," *Proc. ACM/SIGMETRICS,* vol. 24, no. 1, pp. 232-241, May 1996.

[28] E. Varki, "Mean Value Technique for Closed Fork-Join Networks," *Proc. ACM SIGMETRICS Conf. Measurement and Modeling of Computer Systems,* pp. 103-112, May 1999.

[29] E. Varki, L.W. Dowdy, and C. Zhang, "Quick Performance Bounds for Parallel Networks," technical report, Univ. of New Hampshire, Feb. 2000.

[30] E. Varki and X.S. Wang, "An Approximate Performance Model of Disk Array Storage Systems," *Proc. Computer Measurement Group's 2000 Int'l Conf.,* Dec. 2000.

Elizabeth Varki received the BS degree in mathematics from Delhi University, the MS degree in computer science from Villanova University, and the PhD degree in computer science from Vanderbilt University. She is an assistant professor in the Department of Computer Science at the University of New Hampshire. Her research interests include performance evaluation of computer and storage systems, management of storage systems, and parallel input/output. She is a member of the ACM, the IEEE, and the IEEE Computer Society.