

Quick Performance Bounds for Computer and Storage Systems with Parallel Resources

Elizabeth Varki

Chang Zhang

Abstract

This paper derives balanced job bounds and asymptotic bounds for parallel computer and storage systems. The computation of these performance bounds is so simple that it can even be performed by hand. These bounds are the parallel system counterparts to previously developed bounds for non-parallel, product-form systems. The contribution of this paper is showing how these simple bounds for non-parallel systems can be applied to parallel systems. Since these bounds are computed via single-step formulae, the performance technique is useful in the early phases of design projects or capacity planning studies when several candidate configurations must be evaluated quickly.

Index Terms: performance evaluation, parallel computer and storage systems, fork-join networks, queueing systems, RAID, disk arrays.

1 Introduction

The performance of computing and storage systems can be improved by parallelism. In computing systems, the execution time of a program can be improved by splitting the program into sub-programs which then execute in parallel on different processors. In storage systems, the service time of an I/O request can be improved by dividing the request into sub-requests which then execute in parallel on multiple disks. From a performance analysis perspective, such parallel resources are modeled by *fork-join* queues, since a job *forks* into its various sub-tasks and then *joins* (exits) after all these sub-tasks complete service. For ease of exposition, we will henceforth use the term parallel networks to refer to systems with parallel computing or parallel storage resources.

This paper presents a simple, yet powerful technique for computing the mean performance bounds of parallel networks with batch and terminal closed workloads (see Figure 1). The technique uses the fact that the performance of any network, parallel or non-parallel, is bounded above and below by the performance of balanced networks. A network is balanced if the service demands at all service centers of the network are equal. This idea was first proposed by Zahorjan et.al. [34] for non-parallel product-form networks. Simple expressions for computing the performance of balanced product-form networks are provided [34], but unfortunately these results do not carry over to balanced parallel networks. In this

paper, we develop mean performance measures of balanced parallel networks by using a response time result derived by Varki [32] and Theorem 4.1 that is proven in this paper. The performance measures of balanced parallel networks are then used to generate balanced job bounds for parallel networks. Since the response time result [32] is proven only for parallel resources with exponential service time distributions, the balanced job bounds are valid for closed networks with exponential servers. We also derive asymptotic performance bounds for parallel networks using the idea proposed by Muntz and Wong [25] that the performance of a network is bounded above and below by the performance of the network under light and heavy workloads. Thus, performance engineers can compute both types of bounds (balanced and asymptotic) and choose the tighter of these two bounds.

The advantage of this work is that the performance bounds of parallel networks presented here can be derived using a single-step computation, regardless of the degree of parallelism, the load on the system, and the number of modeled resources. Thus, this bounding technique is simple and efficient, with the added advantage of requiring a single-step to calculate (as opposed to the iterative calculations required by alternative approaches). A disadvantage of the bounding technique is that it is valid for only single-class workloads. Another disadvantage of the bounding technique is a lack of guarantee on the tightness of the bounds. Thus, the bounding technique is useful for applications where the speed and simplicity of these bounds override their disadvantages. One such application is system evaluation during the early stages of a design or capacity planning project when the inputs are not completely known and/or when a large number of candidate configurations must be evaluated - as in automated storage management systems such as Minerva [2] that must evaluate thousands of candidate storage configurations quickly. Subsequent to the elimination of non-suitable configurations, it is recommended that performance engineers use more precise, but computationally difficult bounds/approximations (*e.g.*, detailed simulations).

The remainder of this paper is organized as follows. Section 2 summarizes related work on performance analysis of fork-join systems. Section 3 presents a prior result on response time of parallel networks that is used here to derive the quick bounds. Section 4 presents a single-step performance computation technique for balanced parallel networks. Sections 5 and 6 develop the balanced job bounds and the asymptotic bounds, respectively. Section 7 illustrates the application of the bounds to disk arrays. Finally, conclusions are presented in Section 8.

2 Related Work

Several papers study parallel (fork-join) queues and propose tools for analyzing their performance. Exact performance measures have been derived only for fork-join queues with two servers [3, 12, 32]. Bac-

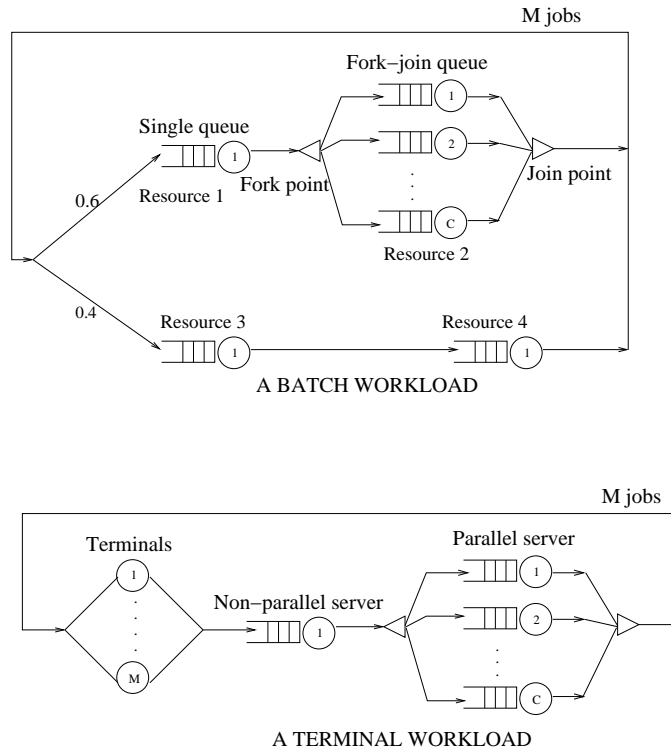


Figure 1: Examples of parallel networks

celli [3] derives exact steady state distribution for two server fork-join queues with general service time distribution. Flatto and Hahn [12] derive exact steady state distribution for two server fork-join queues with exponential arrival and service times. Varki [32] derives exact mean performance measures of two server fork-join queues in closed networks. Due to the difficulty of analyzing fork-join queues exactly, all studies on fork-join queues with three or more servers concentrate on approximation or bounding techniques. Heidelberger and Trivedi [13] 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 [9]. They develop an iterative method for solving a sequence of product-form models. The model is expanded to include a join node in a later paper [14]. Nelson and Tantawi [26] 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, relying on a light traffic interpolation technique, is developed by Makowski and Varma [24]. Kim and Agrawala [16] analyze waiting times for two server open, homogeneous fork-join systems with exponential and 2-stage Erlang service time distributions. Lui, Muntz, and Towsley [22, 23] present a bounding technique for an open, homogeneous fork-join network with a k-stage Erlang distribution. Liu and Perros [20, 21]

propose an approximation procedure based on decomposition and aggregation for analyzing a closed queuing 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 queuing networks by Baccelli et. al. [4] using stochastic ordering principles and association of random variables. Baccelli and Liu [5] propose a new class of queuing models for evaluating the performance of parallel systems. Using the concept of associated random variables, Kumar and Shorey [17] 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 [31] develops an approximate mean-value analysis technique for fork-join parallel networks. Balsamo, Donatiello, and Van Dijk [8] 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 several papers[19, 28, 29, 30].

All the above performance techniques grow in complexity as the degree of parallelism increases (i.e., as the number of servers in the fork-join queue increases) and as the number of jobs accessing the fork-join queue increases. The contribution of this work is the computational speed and the simplicity of the bounding technique presented. Regardless of the complexity of the parallel network or the load on the network, the bounds can be computed by simple non-iterative formulae. These bounds accurately quantify the effects of bottleneck devices, a critical factor in capacity planning studies. So, these bounds are potentially useful in the early phases of a design or capacity planning project.

3 Response Time of Parallel Resources

Consider closed parallel networks with batch or terminal workloads, as shown in Figure 1. Let K represent the number of queueing resources, both parallel and non-parallel, in a network. It is assumed that all jobs in the networks are identical (i.e., the workload is single-class and jobs cannot be distinguished by their service demands). It is also assumed that all service centers of a parallel resource are identical and that every job that arrives at a parallel resource with C service centers divides into exactly C sub-tasks that are assigned to the C load-independent service centers of the parallel resource. It is assumed that the network is routing homogeneous. The jobs cycle through the network moving from resource to resource depending on the network layout and the visit probability of the resources. For each resource k , the visit probability V_k represents the probability that a job will visit (access) resource k during the job's cycle through the network. For example, there are 4 resources (i.e., $K = 4$) in the first queueing network in Figure 1. For the top branch, $V_1 = 0.6$, $V_2 = 0.6$. For the two single service centers in the lower branch,

$$V_3 = 0.4, V_4 = 0.4.$$

It is assumed that the service times at the service centers are not dependent on the number or placement of jobs within the network. If S_k is the mean service time at resource k , the service demand at resource k , D_k , equals $V_k \times S_k$. Similarly, the service demand, d_k , at a service center within a parallel resource is given by $V_k \times s_k$, where s_k is the mean service time at the center. If the service times at the C service centers of the fork-join queue are exponentially distributed, then the mean service time S_k of a fork-join queue with exponential service times is given by

$$S_k = H_C \times s_k$$

where H_C is the C^{th} harmonic number (*i.e.*, $H_C = \sum_{i=1}^C 1/i$) and s_k is the mean service time at a center in the fork-join queue [32]. For example, consider the first queueing network in Figure 1 again. Assume that mean service time at the four resources in the network are all 1 (*i.e.*, $s_1 = s_2 = s_3 = s_4 = 1$), $C = 3$, and the service times are exponentially distributed. Then, $D_1 = d_1 = V_1 \times s_1 = 0.6$, $D_2 = V_2 \times H_3 \times s_2 = 0.6 \times 1.833 \times 1 = 1.099$, $d_2 = V_2 \times s_2 = 0.6$, $D_3 = d_3 = V_3 \times s_3 = 0.4$, $D_4 = d_4 = V_4 \times s_4 = 0.4$.

Performance techniques typically compute the mean performance measures (throughput, response time, queue length) of each resource (X_k, R_k, Q_k) and of the overall network (X, R, Q). Let M represent the number of jobs cycling through the network (*i.e.*, M represents the multiprogramming level of the network). To refer to a performance parameter at a specific multiprogramming level M , the symbol (M) is attached to the parameter.

Again, consider fork-join queueing resources with C identical service centers with exponential service times where every arriving job forks into exactly C sub-tasks. Let A_k represent the arrival state of the fork-join queue. That is, A_k represents the number of jobs in the fork-join queue (both waiting for service and receiving service) that are seen by a job just prior to arrival at the fork-join queue. Let $E(A_k)$ represent the expected value of A_k . Then, the mean response time, R_k , of the fork-join queue is bounded by [32]

$$\boxed{R_k \leq D_k + d_k \times E(A_k)} \quad (1)$$

Here, $D_k = V_k \times S_k = V_k \times H_C \times s_k$ is the mean service demand at the fork-join queue, and $d_k = V_k \times s_k$ represents the mean service demand at a center within the fork-join queue. The relationship is a strict equality in the case of closed fork-join networks with a single resource where this single queueing resource is a fork-join queue with two service centers (*i.e.*, $K = 1, C = 2$). In other cases, the computed response times are greater than the actual response times. Simulation results show that the computed values are very close to the simulated actual values [32].

Equation 1 shows how the response time at a resource is related to the sum of the service time (D_k) and the wait time ($d_k \times E(A_k)$) at the resource. Note that for non-parallel resources, $D_k = d_k$ and the above equation reduces to $R_k = d_k \times (1 + E(A_k))$. In the next section, we show how Equation 1 derived in an earlier paper [32] can be used to derive a simple performance bound for parallel networks.

4 Performance Measures of Balanced Parallel Networks

A balanced network is one where the demands, d_k , at all the service centers in the network are equal (i.e., $d_1 = d_2 = d_3 = \dots = d_K$). Here, we explain how Equation 1 can be used to derive quick single-step performance bounds of balanced parallel networks. Let d ($= d_1 = d_2 = \dots = d_K$) represent the service demand at a service center in a balanced parallel network. Then using Equation 1, the response time, R , of a balanced parallel network is bounded by:

$$\begin{aligned} R &\leq (D_1 + D_2 + \dots + D_K) + d \times (E(A_1) + E(A_2) + \dots + E(A_K)) \\ &= D + d \times (E(A_1) + E(A_2) + \dots + E(A_K)) \end{aligned} \quad (2)$$

where $D = D_1 + D_2 + \dots + D_K$. The values of D (the sum of mean service demands at each resource) and d (the mean service demand at a service center of a resource) can be computed if the mean service times at the various centers are known and the configuration of the network is known. The issue is to compute $(E(A_1) + E(A_2) + \dots + E(A_K))$, where A_k is the number of jobs that are seen in resource k just prior to an arrival at resource k , and $E(A_k)$ is the expected value of A_k .

For closed product-form networks with M jobs, $(E(A_1) + E(A_2) + \dots + E(A_K)) = M - 1$, since the mean arrival queue length at a resource k equals the steady-state queue length at the resource when the network has one less job (i.e., $E(A_k) = Q_k(M - 1)$) [18]. However, this result need not hold in the case of non product-form closed networks. For more general closed networks (in steady-state), it is easy to show that $(E(A_1) + E(A_2) + \dots + E(A_K)) \leq K \times (M - 1)$, since in the worst case, a job could see $(M - 1)$ jobs ahead of it upon arrival at each of the K resources. Here, we prove a much tighter bound, namely, $(E(A_1) + E(A_2) + \dots + E(A_K)) \leq M - 1$.

We now present the intuition behind the tighter bound. In a closed single-class network, a fixed number of identical jobs continuously cycle through the network moving between the K resources depending on the layout of the network and the visit probabilities and service times of the resources. Let random variable Y_k represent the number of jobs seen in resource k just prior to an arrival of a job at any of the K resources of the network. For example, suppose there are $K = 3$ resources in a network. Then, Y_3 represents the number of jobs seen in resource 3 just prior to an arrival at any of the three

resources 1, 2, and 3. Now, suppose there are 7 jobs in a network and an arrival state is (3, 2, 1). Then, $Y_1 = 3, Y_2 = 2, Y_3 = 1$. The random variable A_k represents the number of jobs seen in resource k just prior to an arrival at resource k , so $Y_k = A_k$ if the next arrival is to resource k . In the above example, suppose the arrival state (3, 2, 1) is the result of the movement of a job from resource 1 to resource 3. (That is, the state immediately before arrival equals (4, 2, 1) and the state immediately after arrival equals (3, 2, 2).) The random variable A_3 represents the number of jobs seen in resource 3 just prior to an arrival at resource 3. So, $Y_3 = A_3 = 1$.

Observe the arrival states of the network during some time period $(0, T)$. Let Y_k^n represent the number of jobs in resource k just prior to the n^{th} arrival at any of the K resources of the network during the observed time period. For this n^{th} arrival, let $Y_k^{n'} (n' \geq n)$ represent the state of resource k just prior to the **next** arrival at resource k . That is, $Y_k^{n'} = A_k^i (i \geq 1)$ where A_k^i represents the number of jobs seen in resource k just prior to the i^{th} arrival at resource k . The only arrival that results in an increase to the number of jobs at resource k is an arrival at resource k . Between the n^{th} and the $(n' - 1)^{\text{th}}$ arrivals, there are no arrivals at resource k but there could be departures from resource k . Hence, $Y_k^n \geq Y_k^{n'} = A_k^i$.

If a probability experiment is defined to be the next observed value of Y_k and the next observed value of A_k , then for every possible case, the outcome of A_k is less than or equal to the outcome of Y_k . The experiment is valid for every observation of Y_k and A_k . Thus, A_k is stochastically smaller than Y_k , and therefore, if it is assumed that the means of A_k and Y_k exist, then $E(A_k) \leq E(Y_k)$ [27]. Since the network is closed with a fixed number of M circulating jobs, at all arrival instants, there is a distribution of jobs (m_1, m_2, \dots, m_K) at the K resources, where $m_1 + m_2 + \dots + m_K = M - 1$. That is, $Y_1^n + Y_2^n + \dots + Y_K^n = M - 1$. Since this equation is valid for all arrivals, $Y_1 + Y_2 + \dots + Y_K = M - 1$. Hence, $E(Y_1) + E(Y_2) + \dots + E(Y_K) = (M - 1)$. Since $E(A_k) \leq E(Y_k)$, it follows that $E(A_1) + E(A_2) + \dots + E(A_K) \leq (M - 1)$. We formalize this result in the following theorem.

Theorem 4.1 *Consider a closed queuing network with M jobs circulating among K parallel/non-parallel resources. Let random variable A_k represent the state (i.e., number of jobs) at resource k just prior to an arrival at resource k . If it is assumed that the mean of A_k , $E(A_k)$, exists, then*

$$\sum_{k=1}^K E(A_k) \leq M - 1$$

Proof: Presented in Appendix A. □

Using Theorem 4.1, Equation 2 (presented at the start of this section) can be written as

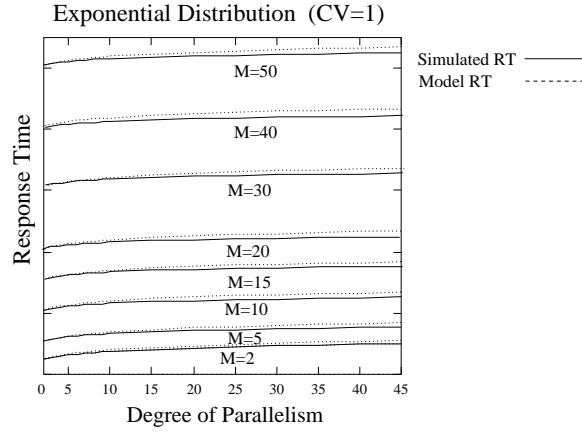


Figure 2: Response time of balanced parallel networks

$$\boxed{R(M) \leq D + d \times (M - 1)} \quad (3)$$

We test the tightness of Equation 3 via simulation. All assumptions such as the exponential service time distribution are adhered to in the simulations. Figure 2 plots the response time of balanced parallel networks containing two resources, one parallel and the other non-parallel. The computed response times are a tight pessimistic bound of the simulated response times.

The mean throughput, X , of a closed balanced parallel network is derived from the response time equation using Little's Law.

$$\boxed{X(M) \geq \frac{M}{D + d \times (M - 1)}} \quad (4)$$

The mean pessimistic performance bounds of balanced parallel networks can be computed in a single step using Equations 3 and 4. Since these equations are derived from Equation 1 (Section 3), they are valid if the assumptions for the validity of Equation 1 hold, namely, that all fork-join queues in the parallel network have exponential service times.

5 Balanced Job Bounds for Parallel Networks

The balanced job bounding technique is based on the fact that the performance of a network is bounded above and below by the performance of balanced networks. The technique developed here is the parallel network counterpart of the balanced job bounding technique for product-form networks [34]. For product-form networks, the balanced job bounding technique is derived by using the well known fact that

the mean queue lengths at all the resources of a balanced product-form network are equal at all multi-programming levels. Thus, the mean throughput and response times of balanced product-form networks can be computed trivially using Little's Law. Unfortunately, for balanced parallel networks, this property of equality of mean queue lengths does not hold since the mean queue length at resource k is dependent not only on the service demand d_k at the resource but also on the degree of parallelism at the resource. However, using Theorem 4.1, we have shown that pessimistic performance bounds of balanced parallel networks can be computed in a single step. We use this property of balanced networks to compute the balanced job bounds for parallel networks.

Consider a parallel network whose performance measures have to be evaluated. Let d_{max} and d_{min} denote the maximum and minimum demands at the service centers of the network to be evaluated. Further, let

$$d_{avg} = \frac{\sum_{k=1}^{k=K} d_k}{K}$$

So, d_{avg} represents the average demand at the centers of the network. The key idea underlying the balanced job bounding technique is that the performance of the given parallel network is bounded by the performance of two balanced networks: (1) pessimistic bounds are obtained when the demands at all service centers in the parallel network are raised to d_{max} , and (2) optimistic bounds are obtained by converting all parallel resources in the network to non-parallel resources and reducing the demand at all service centers in this non-parallel product-form network to d_{min} . Tighter optimistic bounds can be obtained by changing the demand at all service centers in this product-form network to d_{avg} [34]. The reason for optimistically bounding the parallel network by a balanced product-form network instead of a balanced parallel network is that exact performance measures of balanced product-form networks can be found in a single step, whereas only pessimistic bounds of balanced parallel networks can be found in a single step. Thus, a balanced parallel network with the demands at all service centers changed to d_{min} forms as optimistic bound, but the solution of such a network can only be approximated quickly by the pessimistic bound from Section 4.

Balanced parallel networks, unlike balanced product-form networks, do not satisfy the property of equality of server queue lengths. Also, exact performance measures of balanced product-form networks can be computed, while only pessimistic performance bounds of balanced parallel networks can be computed. Hence, as the presentation below shows, the balanced job bounds for parallel networks is not an exact counterpart of balanced job bounds for product-form networks.

5.1 Balanced Job Bounds for Batch Workloads

The performance measures of a parallel network are bounded by the performance of two related balanced networks: (1) a pessimistic bound formed from a parallel network in which the demands at all service centers are increased to d_{max} , the maximum demand at any service center of the parallel network, and (2) an optimistic bound formed from a non-parallel network in which the demands at all the service centers are set to d_{avg} , the average demand at the service centers of the network to be evaluated. Thus,

$$\frac{M}{D + d_{max} \times (M - 1)} \leq X(M) \leq \frac{M}{d + d_{avg} \times (M - 1)}$$

where $D = D_1 + D_2 + \dots + D_K$ is the sum of the service demand at all resources of the parallel network and $d = d_1 + d_2 + \dots + d_K$ is the sum of the service demand at all service centers of the network to be evaluated.

Tighter optimistic bounds can be obtained for networks under heavy load as explained here. At high multiprogramming levels, the first resource to saturate is the resource containing the bottleneck center with demand d_{max} . The utilization, u_{max} , of this center approaches the maximum limit of 1. Using the Utilization Law ($U = X \times D$), an optimistic throughput bound for the parallel network under heavy load is given by

$$X(M) \leq \frac{u_{max}}{d_{max}} \leq \frac{1}{d_{max}}$$

At lower multiprogramming levels, the optimistic balanced bound computed using d_{avg} is tighter. The optimistic balanced bound intersects the heavy load bound as the multiprogramming level increases to some point, say, m^* . After this point, the heavy load bound is tighter. This gives

$$\boxed{\frac{M}{D + d_{max} \times (M - 1)} \leq X(M) \leq \text{minimum} \left\{ \frac{M}{d + d_{avg} \times (M - 1)}, \frac{1}{d_{max}} \right\}}$$

The response time bounds are computed from the throughput bounds by using Little's Law.

$$\boxed{\text{maximum} \{d + d_{avg} \times (M - 1), M \times d_{max}\} \leq R(M) \leq D + d_{max} \times (M - 1)}$$

5.2 Balanced Job Bounds for Terminal Workloads

For terminal workloads, a job in the network spends time Z at a terminal (modeled by a delay server) during each cycle. The cycle time of the balanced network is given by

$$Z + R(M) \leq Z + (D + d \times (E(A_1) + E(A_2) + \dots + E(A_K)))$$

The issue is to compute $(E(A_1) + E(A_2) + \dots + E(A_K))$. For batch workloads, the number of waiting jobs $(E(A_1) + E(A_2) + \dots + E(A_K))$ is less than or equal to $(M - 1)$ since jobs spend all their time at queuing resources. For terminal workloads, the number of waiting jobs at the queuing resources $(E(A_1) + E(A_2) + \dots + E(A_K))$ would be less than $(M - 1)$ since some jobs would be at the terminals. The degree by which $(E(A_1) + E(A_2) + \dots + E(A_K))$ varies from $(M - 1)$ depends on the relative time that each job spends at the queuing resources as opposed to the time it spends at the terminal server [18]. This gives:

$$E(A_1) + E(A_2) + \dots + E(A_K) \approx \frac{R(M)}{R(M) + Z} \times (M - 1)$$

Bounds for $R(M)$ are computed as follows:

- By Theorem 4.1, $E(A_1) + E(A_2) + \dots + E(A_K) \leq M - 1$. Thus, the response time of the queuing resources is pessimistically bounded by $R(M) \leq D + d \times (M - 1)$. Therefore, the relative time spent by a job in the queuing resources is given by $\frac{D + d \times (M - 1)}{D + d \times (M - 1) + Z}$ and $E(A_1) + E(A_2) + \dots + E(A_K) \approx \frac{M - 1}{1 + Z / (D + d \times (M - 1))}$.
- In the best-case scenario, each arriving job receives service immediately (i.e., there are no waiting jobs). The job spends D time units in service and Z time units thinking. Therefore, the relative time spent by a job at the queuing resources is given by $\frac{D}{D + Z}$ and $E(A_1) + E(A_2) + \dots + E(A_K) \approx \frac{M - 1}{1 + Z / D}$.

The throughput of a parallel network under a terminal workload is given by $X(M) = M / (Z + R(M))$. Let $D = D_1 + D_2 + \dots + D_K$ and let $d = d_1 + d_2 + \dots + d_K$ for the network to be evaluated. The balanced job bounds for a parallel network under a terminal workload are then computed as:

$$\boxed{\frac{M}{Z + D + \frac{d_{max} \times (M - 1)}{1 + Z / (D + d_{max} \times (M - 1))}} \leq X(M) \leq \text{minimum} \left\{ \frac{M}{Z + d + \frac{d_{avg} \times (M - 1)}{1 + Z / d}}, \frac{1}{d_{max}} \right\}}$$

The response time bounds are computed from the throughput bounds by using Little's Law.

$$\boxed{\text{maximum} \left\{ d + \frac{d_{avg} \times (M - 1)}{1 + Z / d}, M \times d_{max} - Z \right\} \leq R(M) \leq D + \frac{d_{max} \times (M - 1)}{1 + Z / (D + d_{max} \times (M - 1))}}$$

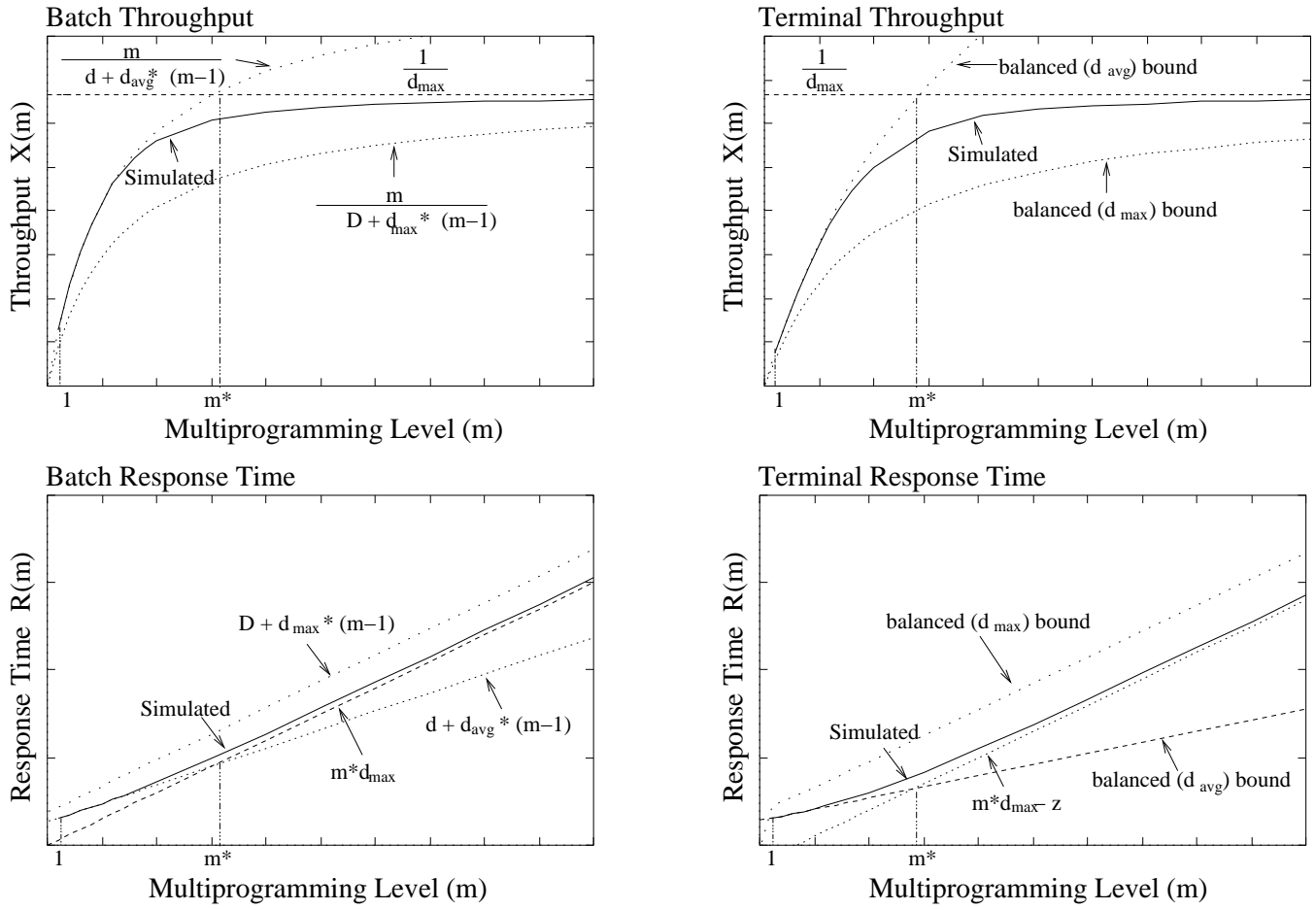


Figure 3: Balanced job bounds for exponential servers

5.3 Validation

Figure 3 plots the balanced job bounds for networks with exponential servers. The performance measures of parallel networks are generated via simulation. The simulated network for batch workloads has 4 resources - resource 1 is non-parallel with mean service time of 1 time unit, resource 2 is a parallel queue with 2 service centers and mean service time of 2 time units, resource 3 is a parallel queue with 3 service centers and mean service time of 3 time units, and resource 4 is a parallel queue with 10 service centers and mean service time of 2 time units. In addition to the above resources, the simulated network for terminal workloads has a delay server with mean think time of 10 time units. The multiprogramming levels of the networks are varied from 1 to 50. All simulated values are accurate within 0.5 time units at 90% confidence level.

6 Asymptotic Bounds for Parallel Networks

The asymptotic bounding technique for parallel networks is a straight-forward extension of the corresponding technique for non-parallel networks [6, 25]. The technique is based on the fact that performance bounds for any network can be easily computed under the extreme conditions of very light or very heavy loads. The asymptotic bounding technique is valid for a large class of networks since the only assumption that must be satisfied by networks is that the service demand of a job at a service center does not depend on how many other jobs are currently in the network or at which service centers they are located. We first consider bounds for terminal workloads with M circulating jobs. Bounds for batch workloads are derived by setting the think time, Z , to zero. Under heavy load, the throughput of the system approaches $1/d_{max}$, the demand at the bottleneck center. Under light load, the bounds are computed as follows:

- In the worst-case scenario, each arriving job has to wait behind $(M - 1)$ jobs at each of the resources before receiving service. The job spends Z time units thinking, $D \times (M - 1)$ time units waiting for service, and D time units receiving service. ($D = D_1 + D_2 + \dots + D_K$.) The network's throughput then equals $\frac{M}{Z + M \times D}$
- In the best-case scenario, each arriving job receives service immediately (*i.e.*, there are no waiting jobs). The job spends Z time units thinking and D time units in service. The network's throughput then equals $\frac{M}{Z + D}$.

The asymptotic bounds on throughput are then given by:

$$\boxed{\frac{M}{Z + M \times D} \leq X(M) \leq \text{minimum} \left\{ \frac{M}{Z + D}, \frac{1}{d_{max}} \right\}}$$

The bounds on response time are derived from the throughput bounds by using Little’s law:

$$\boxed{\text{maximum}\{D, M \times d_{max} - Z\} \leq R(M) \leq M \times D}$$

The optimistic light load bound intersects the heavy load bound as the multiprogramming level increases to some point, say, m^+ ($= (D + Z)/d_{max}$). After this point, the heavy load bound is tighter.

The graphs in Figure 4 plot the asymptotic bounds for a terminal and batch workload. The simulated networks have the same configuration as the networks used in Figure 3. All the bounds are straight lines (except for the pessimistic throughput/response time bounds for terminal workloads) and can be calculated manually. As the graphs in Figure 4 indicate, the bounds can be very loose. However, these bounds are simpler than balanced job bounds and hold for a larger class of networks. Asymptotic bounds are also useful in analyzing the effects of primary and secondary bottleneck devices [6, 7] since these bounds can be used to quickly get a rough understanding of any system. In fact, asymptotic bounds and balanced job bounds are typically used together to quickly bound the performance of networks. That is, both asymptotic bounds and balanced job bounds are plotted in the same graph and the tightest bounded area is used to analyze the performance bounds of the corresponding network.

7 Illustrative Example

The balanced job bounds assume that the distribution of service times at parallel resources is exponential. The reason for this assumption is that Equation 1 (presented in Section 3), which forms the basis of the balanced job bounds is valid only for exponential service time distribution. However, simulation results imply that the response time result is valid for other service time distributions such as the Erlang distribution and the hyper-exponential distribution [32]. This would imply that the balanced job bounds also hold for these distributions. We tested this theory out using simulations.

Figure 5 shows two graphs that plot the response time of balanced parallel networks containing two resources, one parallel and the other non-parallel. The first graph relates to Erlang servers with coefficient of variation (CV)¹ = 0.5, and the second graph relates to hyper-exponential servers with coefficient of variation = 1.5. Comparison of the response time approximations against simulations show an average difference of 3% at 95% confidence level for Erlang, and hyper-exponential distributions with multiprogramming levels ranging from 1 to 50 and degree of parallelism ranging from 1 to 50. Next, Figure 6 plots the balanced job bounds for networks with all Erlang servers and for networks with all hyper-exponential servers. The simulated networks contain 3 resources each - resource 1 is non-parallel with mean service time of 1 time unit, resource 2 is a parallel queue with 2 service centers and mean

¹CV = standard deviation/mean

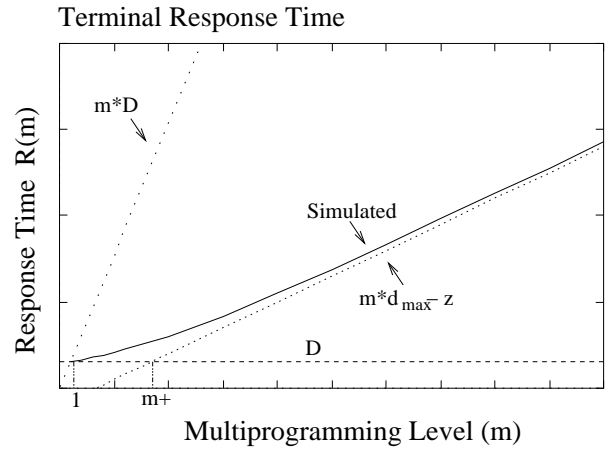
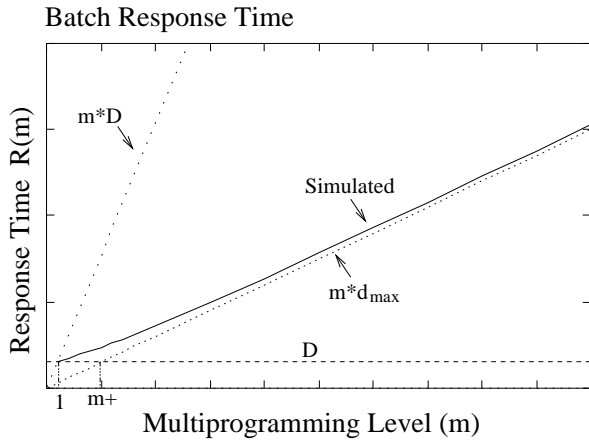
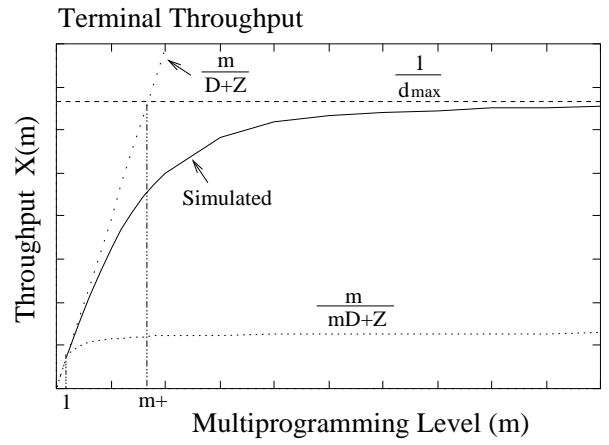
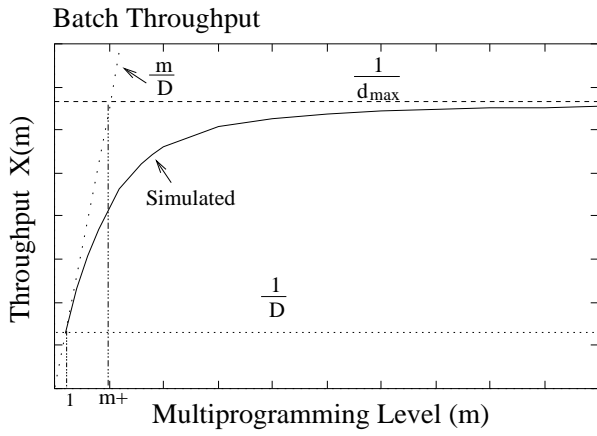


Figure 4: Asymptotic bounds for batch and terminal workloads

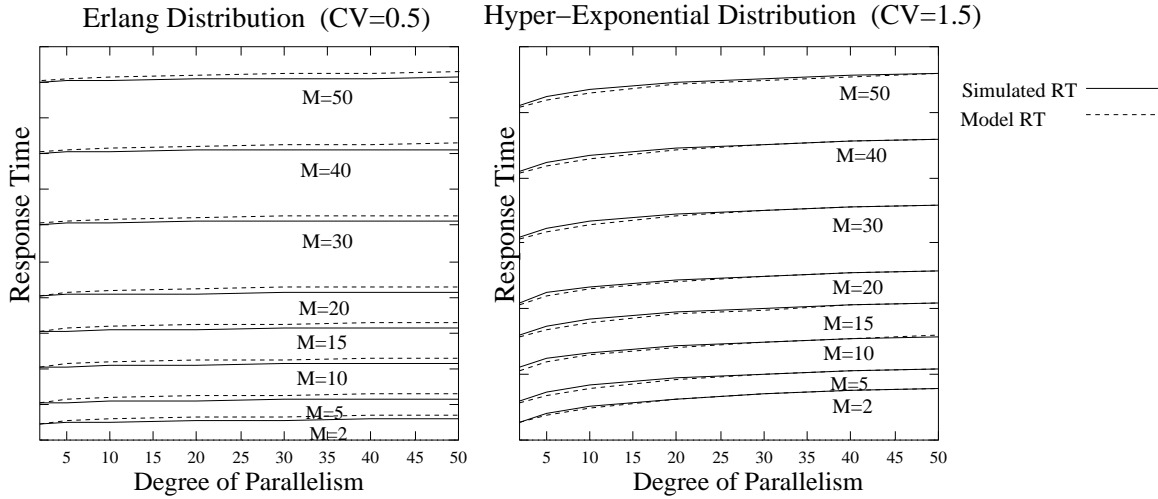


Figure 5: Response time of balanced parallel networks

service time of 2 time units, and resource 3 is a parallel queue with 10 service centers and mean service time of 3 time units. The simulation results show that the bounds work well for such distributions within the range of CV's simulated (0.5 and 1.5).

Next consider fork-join queues where there is no restriction on the service time distribution and where jobs can split into any number of sub-tasks (*i.e.*, jobs are not restricted to forking into exactly C sub-tasks; the number of sub-tasks can be either $< C$ or $= C$). So, V_k , the probability that a job visits a fork-join queue, is not necessarily equal to v_k , the probability that a sub-task of a job visits a center within the fork-join queue during each cycle of the job through the network. For example, consider parallel resource 2 in the first network of Figure 1 with C service centers. Suppose every job that arrives at resource 2 divides into $C/2$ sub-tasks that are assigned to any $C/2$ of the service centers with equal probability. In this case, $V_k = 0.6$ but $v_k = V_k \times 0.5 = 0.3$. It is conjectured via simulations and an informal argument that [32]

$$R_k \approx D_k + d_k \times E(A_k)$$

We illustrate the application of the bounds by generating one-step performance bounds of a disk array under a synchronous I/O workload. The material pertaining to disk arrays described here is summarized from an earlier paper [33]. In that paper, key performance parameters for disk arrays are identified and then performance techniques for computing the values of these disk array parameters are described. In this paper, we use those disk array parameter values to compute balanced job bounds for a real disk array.

A disk array is a parallel device since requests submitted to a disk array are divided into one or more disk I/Os that are submitted to separate disks of the array. Figure 7 (reproduced from Varki et. al. [33]) shows a representation of the Hewlett-Packard SureStore E disk array FC-60 [15] that is used to

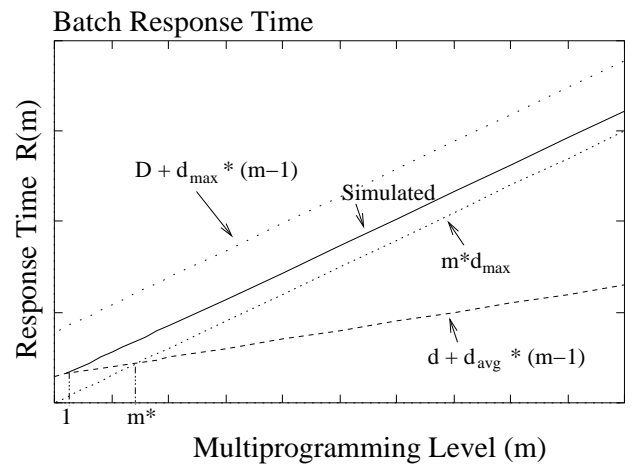
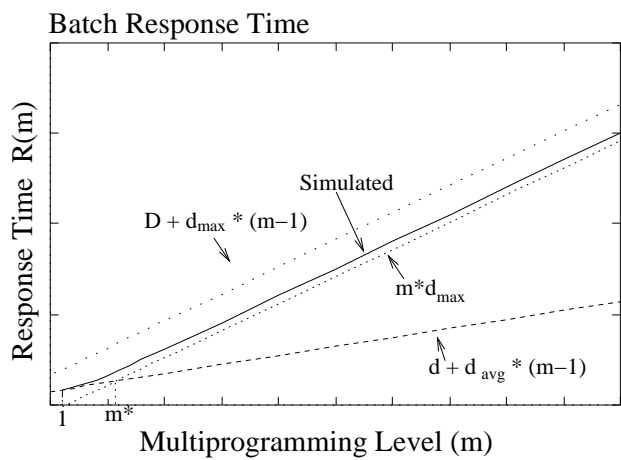
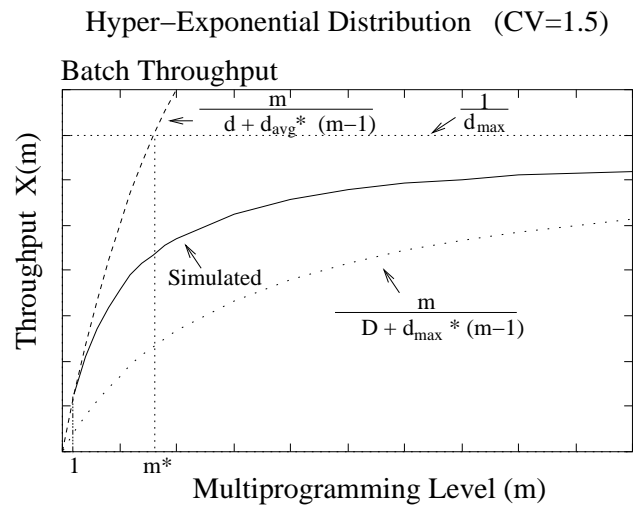
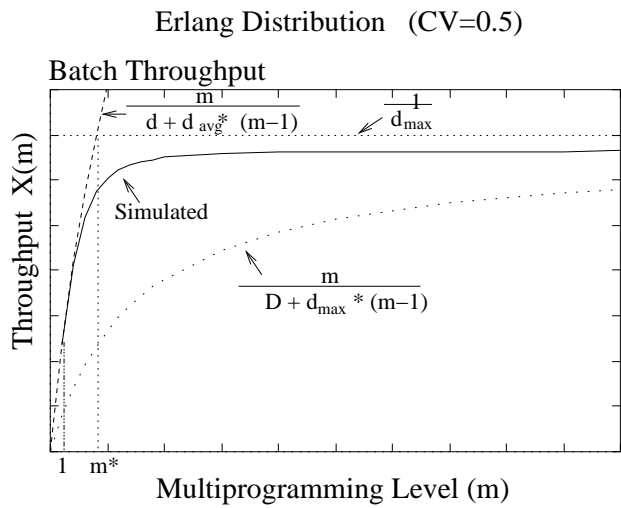


Figure 6: Balanced job bounds for non-exponential servers

Parameter	Description	Value/Units
cache_size	size of the array cache at each controller	256 MB
cache_transfer_rate	mean array cache transfer rate	62 MB/sec
stripe_unit_size	size of a stripe unit	16 KB
stripe_width	number of stripe units in a stripe (logical row)	6
disk_type	type of disks in the FC-60 disk array	Cheetah73
disk_capacity	total formatted capacity of a disk	73.4 GB
disk_read_seek_time	mean disk read seek time	6.05 ms
disk_max_read_seek_time	maximum disk read seek time	14.2 ms
disk_revolutions	revolutions per minute	10000 rpm
disk_transfer_rate	mean disk transfer rate	31 MB/sec

Table 1: Disk array characterization (reproduced from Varki et. al. [33])

validate the bounds. The FC-60 has 2 array controllers and each controller has 256 MB of battery backed array cache memory. Both array controllers of the FC-60 are connected to a single backplane bus. The backplane bus has 6 ultra-wide SCSI buses each connected to a separate tray. Each of the 6 trays has 2 SCSI disk controllers and up to 10 disks. Thus, the FC-60 holds up to 60 disks - at 73 GB per disk drive, the total raw capacity is about 4.3 terabytes. Table 1 (reproduced from Varki et. al. [33]) presents the array parameters of significance to the model. All our experiments are run on 6 disks of the FC 60. These disks are configured using RAID 1/0 with a stripe unit size of 16 KB.

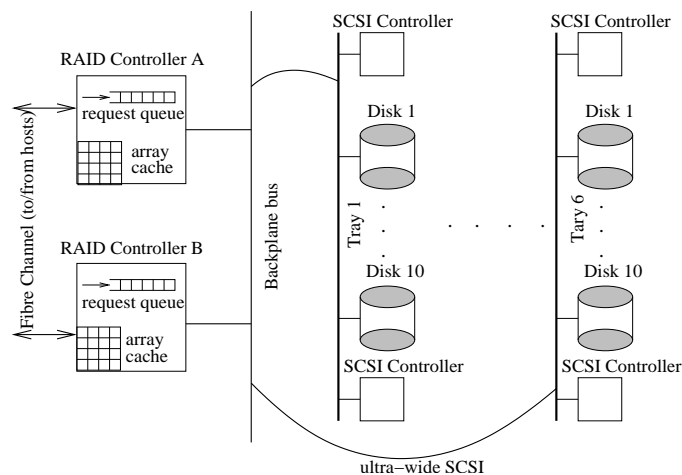


Figure 7: The HP FC-60 Disk Array (reproduced from Varki et. al. [33])

The workload submitted to the disk array consists of M I/O streams with similar characteristics. The key characteristics of an I/O stream are the request type, the request size, and the degree of sequentiality. We assume that the jobs accessing the disk array generate synchronous I/O requests. This implies that a job must wait until its I/O request completes and there is at most one request from each stream at the

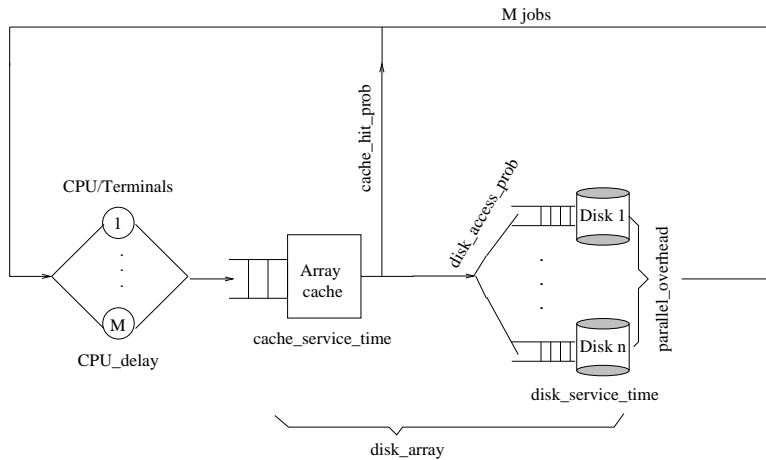


Figure 8: Queueing network model of disk arrays with read workloads (reproduced from Varki et. al. [33])

disk array. Each of the M jobs spends some time at the CPU/terminal before submitting an I/O request to the disk array. A job waits until the disk array completes processing its I/O request, and then the job spends time at the CPU/terminal before issuing another I/O request. Such cyclical behavior is modeled using a closed queueing network with jobs cycling between the CPU/terminals and the disk array. Let CPU_delay represent the mean time spent by a job at its CPU/terminal.

We analyze the behavior of disk arrays when the submitted workload consists of read-only I/O requests. Read requests are first submitted to the disk array cache. With probability $cache_hit_probability$ a read request's data are found in the cache, and the disk array controller signals service completion. With probability $cache_miss_probability (= 1 - cache_hit_probability)$ a request is forwarded to the disks, and the disk array controller signals service completion after all disk I/Os for the request complete service. From a performance perspective, the key components of a disk array are the array cache and the disks. The array cache is modeled by a single queueing server with mean service time $cache_service_time$. The disks are modeled by a parallel queueing system since a request submitted to the disks is divided into one or more disk I/Os issued in parallel to some (or all) of the disks, and the request completes service only after all its disk I/Os complete. The parameter $disk_access_probability$ represents the probability that a request accesses a disk in the array. The parameter $disk_service_time$ represents the mean time to service a disk I/O access. Even if all disk I/Os for a request are issued simultaneously, these disk I/Os need not all start and finish at the same time since the disks are independent devices. The parameter $parallel_overhead$ represents the additional time (over $disk_service_time$) taken to execute all the disk I/Os of a request. Figure 8 (reproduced from Varki et. al. [33]) presents the queueing network model of the disk array system with read workloads. In the figure, the input parameters of the disk array model are labeled at the appropriate places.

Bound parameters	Equivalent disk array model input parameter/value	Description
S_1, s_1	cache_service_time[M]	mean cache service time per request
V_1, v_1	1 (all requests submitted to cache)	probability that a request is submitted to the cache
s_2	disk_service_time[M]	mean disk service time
v_2	disk_access_probability[M]	probability that a request accesses a disk in the array
S_2	disk_service_time[M]+ parallel_overhead[M]	time taken to execute all disk I/Os of a request
V_2	cache_miss_probability[M] = (1 - cache_hit_probability[M])	array cache miss probability
Z	CPU_delay	CPU delay time

Table 2: Input parameters to bounding technique when there are M I/O streams accessing the disk array

The bounding technique is validated by comparing throughput bounds against measured throughput values from the FC-60 array for synthetic read-only workloads with request sizes ranging from 4 KB to 256 KB. The number of jobs generating requests range from 1 to 12 and the CPU delay ranges from 0 ms to 10 ms. For this synthetic workload, we compute values for the input parameters specified in Table 2 using model parameterization techniques for real disk arrays presented in an earlier paper [33]. For details of these techniques, an interested reader is referred to that paper.

Figure 9 presents the throughput balanced job bound for a random read workload consisting of workload streams with mean request size of 32KB. The first graph pertains to a batch workload while the second graph pertains to a terminal workload with a mean delay (think time) of 10ms. Figure 10 presents the throughput balanced job bounds for two sequential read workloads. The first graph pertains to workload streams with mean request size of 64KB, while the second graph pertains to workload streams with request size of 256KB. Both graphs in Figure 10 pertain to terminal workloads. In the graphs, the legend “Actual” refers to measured throughput values from the HP SureStore E disk array FC-60, and the legend “disk array model” refers to performance measures computed using a computationally intensive performance modeling approach [33]. The rest of the plotted lines pertain to balanced job bounds. The performance of a disk array is very dependent on the degree of sequentiality of its workload. As these graphs show, the balanced job bounds are fairly tight for the disk array under both random and sequential workloads.

8 Conclusions

This paper presents a single-step bounding technique for parallel networks that is based on the idea that the performance of any parallel network is bounded above and below by the performance of suitably

Request size = 32K

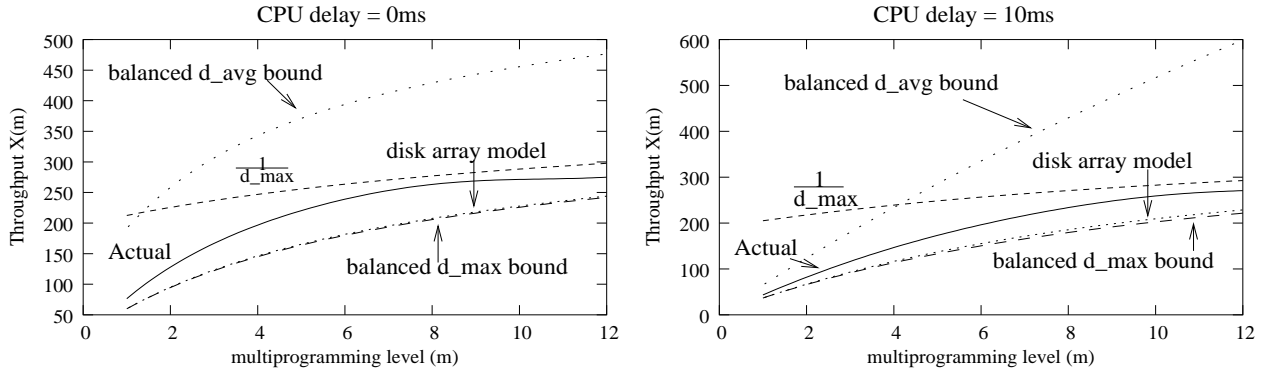


Figure 9: Random read workload

CPU delay = 10ms

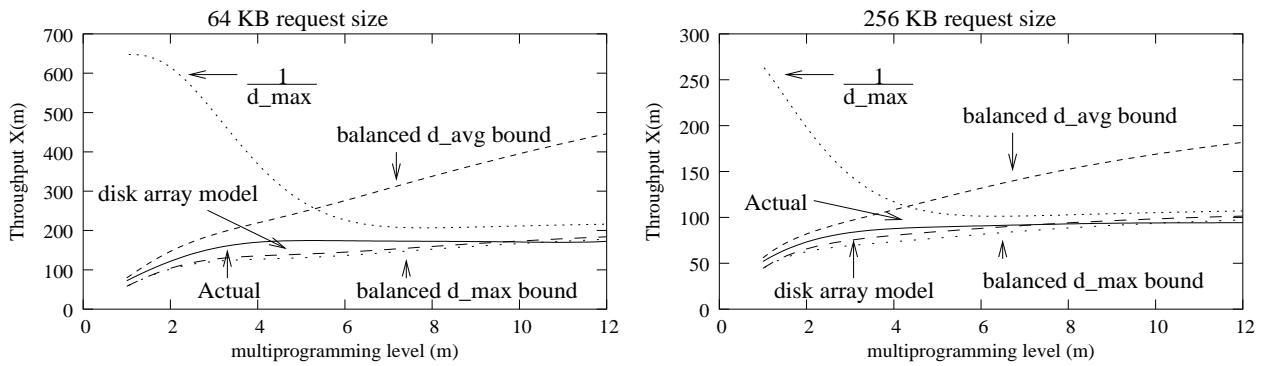


Figure 10: Sequential read workload

balanced parallel networks (balanced job bounds) and by the performance of the network under extreme workloads (asymptotic bounds). While the balanced job bounds are tighter than the asymptotic bounds, the asymptotic bounds are computationally simpler and are applicable for a larger class of parallel networks. The idea behind balanced bounds and asymptotic bounds was first presented for product-form networks. The contribution of this paper is showing how this single-step bounding technique developed for tractable product-form networks can be applied to parallel networks. General fork-join queueing networks are considered to be intractable. Consequently, most performance evaluation techniques of such networks are approximations that are computationally expensive, and the complexity of these techniques increases with the size of the network. The bounding technique presented here is computationally simple regardless of the size of the network and the number of jobs accessing the network.

Bounding techniques are the simplest useful approach to computer system performance evaluation using queueing network models. A trade-off paid for the simplicity of bounding techniques is the loss of accuracy resulting from making assumptions such as single-class workloads, and a lack of guarantee on the tightness of the bounds. Hence, the balanced job bounds and the asymptotic bounds are more useful in situations where the errors resulting from making simplifying assumptions do not override the benefits from getting quick performance estimates. One such situation is the early stages of a design and planning study when the system estimates are preliminary and imprecise. Another situation where the speed of bounding techniques is critical is when a large number of candidate configurations must be evaluated, as in automated storage management systems such as Minerva [2] that must evaluate thousands of candidate storage configurations quickly. Yet another application where simplifying assumptions and loose bounds are not very restrictive is the analysis of bottleneck centers on performance.

As future work, one could use insights gained from the fork-join analysis presented here to develop more accurate performance bounding techniques. One such technique is the performance bound hierarchies by Eager and Sevcik [10] for product-form networks that provides increasingly tighter bounds at the expense of more computational cost. Bounding techniques for product-form networks under multiple-class workloads [6, 7, 11] could also be extended to parallel networks. The balanced job bounds are proven for networks where the parallel resources have service times that are drawn from an exponential distribution. Simulation results and measurements from a disk-array, however, provide limited evidence that the balanced job bounds are valid for networks with other service time distributions; more research is needed to fully investigate this proposition.

A Proof of Theorem 4.1

Consider a closed queueing network with K resources and M jobs. Let A_k represent the number of jobs at resource k just prior to an arrival at resource k . If it is assumed that the mean value of A_k , $E(A_k)$, exists then

$$\sum_{k=1}^K E(A_k) \leq M - 1$$

Proof: Let random variable Y_k represent the number of jobs at resource k just prior to an arrival of a job at any of the K resources of the network.

Observe the network for some period of time $(0, T)$. Suppose there are N arrivals at the K resources of the network during this period. Suppose N_k of these arrivals are to resource k ($N_k \leq N$).

Let A_k^n represent the arrival state of resource k just prior to the n^{th} arrival at resource k during the observed time period.

Let Y_k^n represent the state of resource k just prior to the n^{th} arrival at any resource of the network during the observed time period.

Since N_k of the N arrivals in the observed time period are to resource k ,

$$Y_k^{n_i} = A_k^i \quad i = 1, \dots, N_k, 1 \leq n_1 < n_2 < \dots < n_{N_k} \leq N \quad (5)$$

That is, if the next arrival is to resource k , then the next observed value of Y_k equals the next observed value of A_k .

Since the number of jobs at resource k increases only during arrival instants at resource k , while the number of jobs at resource k could decrease between two arrivals because of departures from resource k ,

$$Y_k^j \geq A_k^i \quad i = 1, \dots, N_k, j = n_{i-1} + 1, \dots, n_i \quad (6)$$

Since Equation 6 is valid for all arrivals

$$Y_k \geq A_k \quad k = 1, \dots, K \quad (7)$$

If it is assumed that the means of A_k and Y_k exist, then it follows that [27]

$$E(Y_k) \geq E(A_k) \quad k = 1, \dots, K \quad (8)$$

Since there are M jobs in the network, one of these M jobs arriving at a resource in the network sees $(M - 1)$ jobs in the network. That is,

$$\sum_{k=1}^K Y_k^n = M - 1 \quad \forall n \quad (9)$$

Since Equation 9 is valid for all arrivals n ,

$$\sum_{k=1}^K Y_k = M - 1 \quad (10)$$

Hence, it follows that [27],

$$\sum_{k=1}^K E(Y_k) = M - 1 \quad (11)$$

From Equations 8 and 11 it follows that

$$\sum_{k=1}^K E(A_k) \leq M - 1$$

□

References

- [1] Almeida, V.A.F., Dowdy, L.W., ‘A reduction technique for solving queueing network models of programs with internal concurrency’, Proceedings of the 3rd International Conference on Supercomputing, Boston, May 1988.
- [2] G. A. Alvarez, E. Borowsky, S. Go, T. H. Romer, R. Becker-Szendy, R. Golding, A. Merchant, M. Spasojevic, A. Veitch, J. Wilkes, ‘MINERVA: An automated resource provisioning tool for large-scale storage systems’, to appear in ACM Transactions on Computer Systems.
- [3] Baccelli, F. ‘Two parallel queues created by arrivals with two demands: The M/G/2 symmetrical case’, Report INRIA, 426, July 1985.
- [4] Baccelli, F., Massey, W. A., Towsley, D. ‘Acyclic fork-join queueing networks’, Journal of the ACM, 36, 3, July 1989, pp. 615 – 642.
- [5] Baccelli, F., Liu, Z. ‘On the execution of parallel programs on multiprocessor systems – A queueing theory approach’, Journal of the ACM, 37, 2, April 1990, pp. 373 – 414.
- [6] Balbo, G., Serazzi, G., ‘Asymptotic analysis of multiclass closed queueing networks: Common bottleneck’, Performance Evaluation 26, 1996, pp. 51 – 72.

- [7] Balbo, G., Serazzi, G., “Asymptotic analysis of multiclass closed queueing networks: Multiple bottleneck”, *Performance Evaluation* 30, 1997, pp. 115 – 152.
- [8] Balsamo, S., Donatiello, L., Van Dijk, N.M., ‘Bound performance models of heterogeneous parallel processing systems”, *IEEE Trans. on Parallel and Distributed Systems*, Vol. 9, No. 10, October 1998.
- [9] Baskett, F., Chandy, K.M., Muntz, R.R., Palacios, F.G. ‘Open, closed, and mixed networks of queues with different classes of customers”, *Journal of the Association of Computing Machinery*, Vol. 22, No. 2, April 1975, pp. 248 – 260.
- [10] Eager, D.L., Sevcik, K.C., Performance bound hierarchies for queueing networks, *ACM Transactions on Computer Systems*, Vol 1, No. 2, May 1983, pp. 99 – 115.
- [11] Eager, D.L., Sevcik, K.C., Bound hierarchies for multiple-class queueing networks, *Journal of the ACM*, vol. 33, 1986, pp. 179 – 206.
- [12] Flatto, L., Hahn, S. ‘Two parallel queues created by arrivals with two demands”, *SIAM J. Appl. Math.*, 44, Oct. 1984, pp. 1041 – 1053.
- [13] Heidelberger, P., Trivedi, S. K. ‘Queueing network models for parallel processing with asynchronous tasks”, *IEEE Trans. on Computers*, 31, Nov. 1982, pp. 1099 – 1109.
- [14] Heidelberger, P., Trivedi, S. K. ‘Analytic queueing models for programs with internal concurrency”, *IEEE Trans. on Computers*, 32, Jan. 1983, pp. 73 – 82.
- [15] Hewlett-Packard Company, *HP SureStore E Disk Array FC60 User’s guide*, December 2000, Pub. No. A5277-90001.
- [16] Kim, C., Agrawala, A.K. ‘Analysis of the fork-join queue”, *IEEE Transactions on Computers*, 38, 2, Feb. 1989, pp. 250 – 255.
- [17] Kumar, A., Shorey, R. ‘Performance analysis and scheduling of stochastic jobs in a multicomputer system”, *IEEE Trans. on Parallel and Distributed Systems*, 4, 10, Oct. 1993, pp. 1147 – 1164.
- [18] Lazowska, E.D., Zahorjan, J., Graham, G.C., Sevcik, K.C., *Quantitative System Performance – Computer System Analysis Using Queueing Network Models*, Prentice-Hall, 1984.
- [19] Lee, E.K., Katz, R.H., ‘An analytic performance model of disk arrays”, *Proceedings of ACM SIGMETRICS*, 1993.
- [20] Liu, Y. C., Perros, H. G. ‘Approximate analysis of a closed fork/join model”, *European Journal of Operational Research*, 53, 1991, pp. 382 – 392.
- [21] Liu, Y. C., Perros, H. G. ‘A decomposition procedure for the analysis of a closed fork/join queueing system”, *IEEE Transactions on Computers*, 40, 3, Mar. 1991, pp. 365 – 370.
- [22] Lui, J.C.S., Muntz, R.R., Towsley, D, ‘Bounding the response time of a minimum expected delay routing system: an algorithmic approach”, *IEEE Trans. on Computers*, vol 44, no. 5, May 1995, pp. 1371 – 1382.
- [23] Lui, J.C.S., Muntz, R.R., Towsley, D, ‘Computing performance bounds of fork-join parallel programs under a multiprocessing environment”, *IEEE Trans. on Parallel and Distributed Systems*, vol. 9, no. 3, March 1998, pp. 295-311.
- [24] Makowski, A., Varma, S., ‘Interpolation approximations for symmetric fork-join queues”, *Performance Evaluation* 20, 1994, pp. 145 – 165.

- [25] Muntz, R.R., Wong, J.W., "Asymptotic properties of closed queueing network models", Proc. 8th Princeton Conference on Information Sciences and Systems, 1974.
- [26] Nelson, R., Tantawi, N., "Approximate analysis of fork/join synchronization in parallel queues", IEEE Transaction on Computers, 37, 6, June 1988, pp. 739 – 743.
- [27] Ross, S.M., *A first course in probability - 6th ed.*, Prentice Hall, Inc., 2002.
- [28] Thomasian, A., Tantawi, A.N., "Approximate solutions for M/G/1 fork-join synchronization", Proc. 1994 Winter Simulation conf., Orlando, Fla., Dec 1994, pp. 361 – 368.
- [29] Thomasian, A., Menon, J., "Approximate analysis for fork-join synchronization in RAID5", Computer Systems: Science and Eng., 1997.
- [30] Thomasian, A., Menon, J., "RAID5 performance with distributed sparing", IEEE Transactions on Parallel and Distributed Systems, Vol. 8, No. 6, June 1997, pp. 640 – 657.
- [31] Varki, E., "Mean value analysis of fork-join parallel networks", Proceedings of ACM/SIGMETRICS. Also, Performance Evaluation Review, May 1999.
- [32] Varki, E., "Response time analysis of parallel computer and storage systems", IEEE Transactions on Parallel and Distributed Systems, Vol. 22(11), pp. 1146 – 1161, November 2001.
- [33] Varki, E., Merchant, A., Xu, J., Qiu, X., "Issues and challenges in the performance analysis of real disk arrays", IEEE Transactions on Parallel and Distributed Systems, Vol. 15(6), pp. 559 – 574, June 2004.
- [34] Zahorjan, J., Sevcik, K. C., Eager, D. L., Galler, B. "Balanced job bound analysis of queueing networks", Communications of the ACM, 25, 2, Feb. 1982, pp. 134 –141.