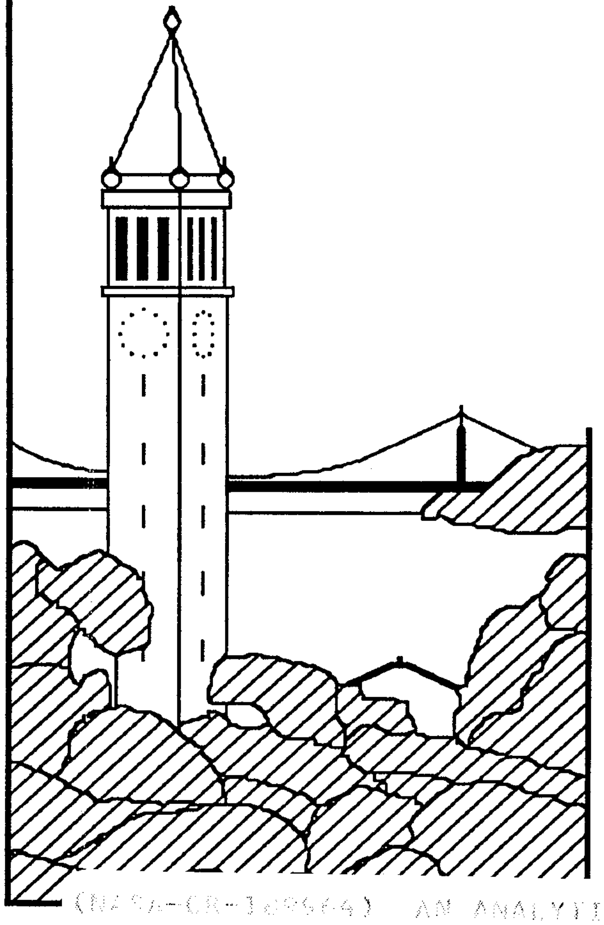
NAG2-591
IN-60-CR
73847

P-22

An Analytic Performance Model of Disk Arrays and its Application

Edward K. Lee and Randy H. Katz



Report No. UCB/CSD 91/660

November 1991

**Computer Science Division (EECS)
University of California, Berkeley
Berkeley, California 94720**

NAG 2-(591)

An Analytic Performance Model of Disk Arrays and its Application

Edward K. Lee   Randy H. Katz

November 21, 1991

Abstract

As disk arrays become widely used, tools for understanding and analyzing their performance become increasingly important. In particular, performance models can be invaluable in both configuring and designing disk arrays. Accurate analytic performance models are desirable over other types of models because they can be quickly evaluated, are applicable under a wide range of system and workload parameters, and can be manipulated by a range of mathematical techniques. Unfortunately, analytic performance models of disk arrays are difficult to formulate due to the presence of *queuing* and *fork-join synchronization*; a disk array request is broken up into independent disk requests which must all complete to satisfy the original request. In this paper, we develop, validate and apply an analytic performance model for disk arrays. We derive simple equations for approximating their utilization, response time and throughput. We then validate the analytic model via simulation and investigate the accuracy of each approximation used in deriving the analytic model. Finally, we apply the analytic model to derive an equation for the optimal unit of data striping in disk arrays.

1 Introduction

In recent years, improvements in microprocessor performance has greatly outpaced improvements in I/O performance. If the trend continues, future improvements in microprocessor performance will be wasted as computer systems become increasingly I/O bound. To overcome the impending I/O crisis, several researchers [7,8,10,12,13,15] have proposed the use of disk arrays that stripe data across multiple disks and provide improved I/O performance by using parallelism to increase data transfer rates and by servicing multiple I/O requests concurrently.

Given the important role disk arrays will play in the I/O systems of tomorrow, tools for understanding their performance become increasingly important. In particular, performance models, combined with a thorough understanding of an installation's workload, will be invaluable in both configuring and designing disk arrays. In general, accurate analytic performance models are desirable over other types of models, such as empirical and simulation, because they can be quickly evaluated, are applicable under a wide range of system and workload parameters, and can be manipulated by a range of mathematical techniques. Even when analytic models are not directly applicable to a particular system or workload, they are frequently useful for quickly analyzing general properties of the system, stimulating intuition and furthering understanding.

Unfortunately, analytic performance models of disk arrays are difficult to formulate due to the presence of queuing and fork-join synchronization; a disk array request is broken up into independent disk requests which must all complete to satisfy the disk array request. Exact analytic solutions for the two server fork-join queue given Poisson arrivals and independent service times currently exist [1,5] but the k-server fork-join queue remains unsolved. Other related work in the field falls into four primary categories: (1) simulation studies, (2) analytic models that ignore queueing effects, (3) analytic models that ignore fork-join synchronization and (4) restricted queueing models that deal with fork-join synchronization using specialized techniques not easily extended to modeling disk arrays. Most analytic queueing studies deal with general queueing systems rather than disk arrays in particular. The following lists previous work that is representative of the field.

- Kim [8] investigates the performance of n independent disks without data striping versus n synchronized disks with data striping; the n disks are essentially equivalent to a single disk with n times higher data transfer rate. She derives equations for response time assuming each disk is an M/G/1 system. Because the disks are completely synchronized, she avoids fork-join synchronization altogether.
- Livny [10] investigates the performance of *declustering*, where data is striped in 26KB units, versus *clustering*, where data is not striped, over a range of transaction workloads via simulation.
- Reddy [14] investigates the performance tradeoff between synchronized fine-grained data striping versus asynchronous coarse-grained data striping via simulation. He also proposes and investigates hybrid schemes that combine aspects of synchronized fine-grained data striping and asynchronous coarse-grained data striping.
- Chen [4] derives empirical rules for optimally selecting the unit of data striping in disk arrays over a range of workloads via simulation.
- Salem and Garcia-Molina [15]; Kim and Tantawi [9]; Bitton and Gray [2] derive minimum response time formulas (no queueing) for asynchronous disk arrays.
- Patterson, Gibson and Katz [13] derive analytic formulas for maximum throughput in RAID's (Redundant Array of Inexpensive Disks) which are subsequently verified by Chen [3] via measurement.
- Heidelberger and Trivedi [6] formulates an analytic model for systems with forks but no joins.

In this paper, we develop, validate and apply an analytic performance model for disk arrays. Our model is different from previous analytic models of disk arrays mentioned above for the following reasons. First, we use a closed queueing model with a fixed number of processes whereas previous analytic models of disk arrays have used open queueing models with Poisson arrivals. A closed model more accurately models the synchronous I/O behavior of scientific, time-sharing and distributed systems. In such systems, processes tends to wait for previous I/O requests to complete before issuing new I/O requests, whereas in transaction based systems, I/O requests are issued at random points in time regardless of whether the previous I/O requests have completed. Second, to the best of our knowledge, this is the first analytic model for disk arrays that handles both the queueing at individual disks and the fork-join synchronization introduced by data striping. Previous analytic models that handle both queueing and fork-join synchronization cannot easily be applied to disk arrays because they assume service times across servers (disks) are independent whereas in disk arrays, they are very much dependent.

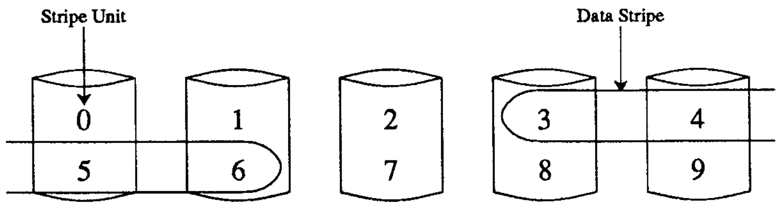


Figure 1: Data Striping.

In the following sections, we first derive an exact expression for the utilization of the model system. Because the exact expression contains parameters that are difficult or impossible to compute, we either analytically approximate or empirically calibrate the difficult parameters to make the expression more tractable. From the resulting approximate equation for utilization, we derive equations for response time and throughput. We then validate the analytic model via simulation and investigate the accuracy and sensitivity of each approximation used in deriving the analytic model. Finally, we apply the analytic model to derive an equation for determining the optimal unit of data striping in disk arrays.

2 Definitions

Disk arrays provide high I/O performance by striping data over multiple disks. High performance is achieved by servicing multiple I/O requests *concurrently* and by using more than one disk to service a single request in a *parallel* manner.

Figure 1 illustrates the basic disk array of interest and illustrates the terms *stripe unit* and *data stripe* which we formally define as follows:

Stripe unit is the unit of data interleaving, that is, the amount of data that is placed on a disk before data is placed on the next disk. Stripe units typically range from a sector to a track in size (512 bytes to 64 kilobytes). Figure 1 illustrates a disk array with five disks with the first ten stripe units labeled.

Data stripe is a sequence of logically consecutive stripe units. A logical I/O request to a disk array corresponds to a data stripe. Figure 1 illustrates a data stripe consisting of four stripe units spanning stripe units three through six.

3 The Analytic Model

In this section, we derive equations to approximate the performance of disk arrays. Our approach is to derive the expected utilization of a given disk in the disk array. Because we are modeling a closed system where each disk plays a symmetric role with respect to each other, knowing the expected utilization of a given disk in the system will allow us to compute the system's throughput and response time.

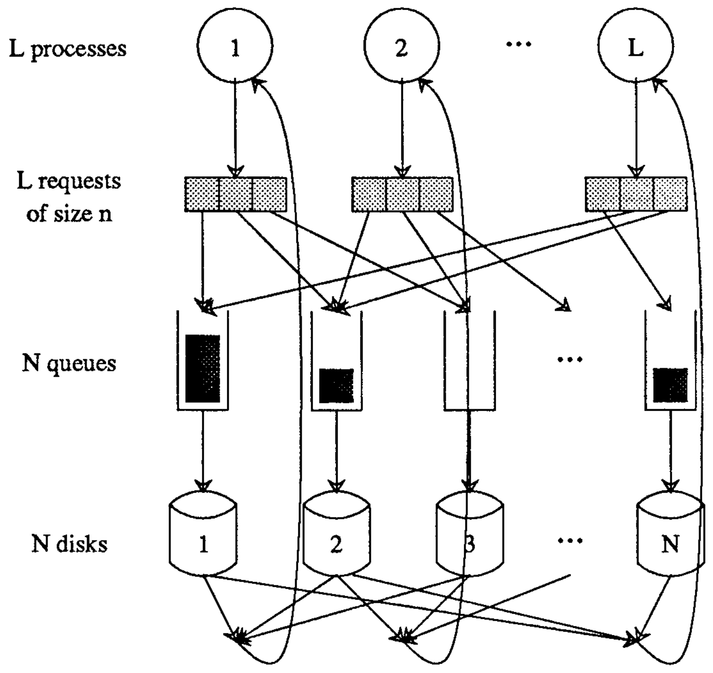


Figure 2: Closed Queuing Model for Disk Arrays.

3.1 The Model System

Consider the *closed* queueing system illustrated by Figure 2. The system consists of L processes, each of which issues, one at a time, an *array request* of size n stripe units. Each array request is broken up into n *disk requests* and the disk requests are queued round-robin starting from a randomly chosen disk. Each disk services a single disk request at a time in a FIFO manner. When all of the disk requests corresponding to an array request are serviced, the process that issued the array request issues another array request, repeating the cycle. Note that two array requests may partially overlap on some of the disks, resulting in complex interactions. We sometimes refer to array requests simply as *requests*. The parameters of the above system are as follows:

L	$\equiv$	Number of processes issuing requests.
N	$\equiv$	Number of disks.
n	$\equiv$	Request size (number of disks/stripe-units accessed per request); $n \leq N$.
S	$\equiv$	Service time of a given disk request.

In the derivation of the analytic model, we will assume that L and n are fixed. We will also assume that the processes do nothing but issue I/O requests.

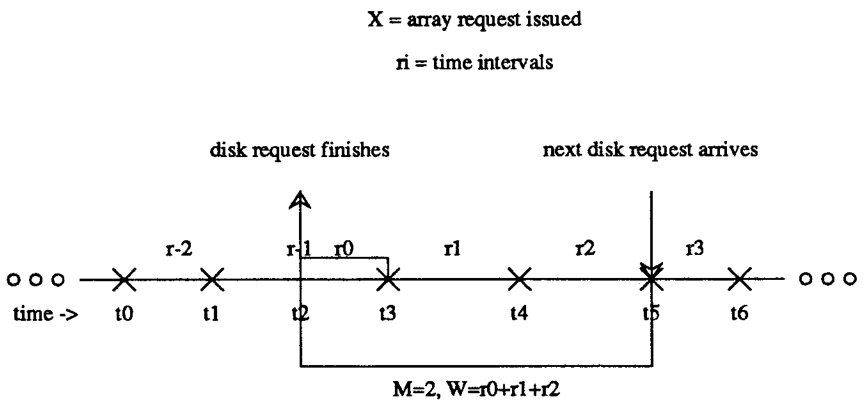


Figure 3: Time-line of Events at a Given Disk. *After the disk request finishes service at time t_2, $M = 2$ array request that do not access the given disk are issued at times t_3 and t_4 before an array request that accesses the given disk is issued at time t_5. The disk remains idle for a time period of $W = r_0 + r_1 + r_2$.*

3.2 The Expected Utilization

In derivating the expected utilization of the model system, the following definitions will prove useful:

U	$\equiv$	Expected utilization of a given disk.
R	$\equiv$	Response time of a given array request.
W	$\equiv$	Disk idle (wait) time between disk request servicings.
Q	$\equiv$	Queue length at a given disk.
p_0	$\equiv$	Probability that the queue at a given disk is empty when the disk finishes servicing a disk request.
p	$\equiv$	Probability that a request will access a given disk; n/N.

If we visualize the activity at a given disk as an alternating sequence of busy periods of length S and idle periods of length W, the expected utilization of a given disk is,

$$U = \frac{E(S)}{E(S) + E(W)}. \tag{1}$$

Idle periods of length zero can occur and imply that another disk request is already waiting for service, $Q > 0$, when the current disk request finishes service.

Let r_0 denote the time between the end of service of a given disk request and the issuing of a new array request into the system. Let $r_i, i \in \{1, 2, \ldots\}$ denote the successive time intervals between successive issues of array requests numbered relative to r_0. Let M denote the number of array requests that are issued after a given disk finishes a disk request until, but excluding, the array request that accesses the given disk. Since each array request has probability p of accessing a given disk, M is geometrically distributed and $E(M) = 1/p - 1$. Figure 3 illustrates the above terms.

By conditioning on the queue length at the time a disk request finishes service, we can write,

$$\begin{aligned}
E(W) &= P(Q > 0)E(W|Q > 0) + P(Q = 0)E(W|Q = 0), \\
E(W) &= (1 - p_0)0 + p_0 E(\textstyle\sum_{i=0}^{M} r_i), \\
E(W) &= p_0(E(r_0) + E(\textstyle\sum_{i=1}^{M} r_i)).
\end{aligned}$$

Substituting into Equation 1 we have,

$$U = \frac{E(S)}{E(S) + p_0(E(r_0) + E(\sum_{i=1}^{M} r_i))} \tag{2}$$

Equation 2 is an exact equation for the expected utilization of the model system.

3.3 Approximating the Expected Utilization

In the previous section, we formulated an exact equation, Equation 2, for the expected utilization of the model system. Unfortunately, the exact equation consists of terms which are very difficult if not impossible to compute. In this section, we approximate components of Equation 2 to make it analytically tractable.

To simplify Equation 2, we make the following assumption: $E(\sum_{i=1}^{M} r_i) \simeq E(M)E(r) = (1/p - 1)E(R)/L$. From Little's Law, we know that the average time between successive issues of array requests is $E(R)/L$; thus, the above approximation would be exact if $r_i, i \in \{1, 2, \ldots\}$ were independently distributed with a common mean of $E(R)/L$. For the moment, we will take the above approximation as given, but later show via simulation that the above is an extremely good approximation. Thus, we can write,

$$U \simeq \frac{E(S)}{E(S) + p_0(E(r_0) + (1/p - 1)E(R)/L)}. \tag{3}$$

Given the above approximation, it is natural to assume the following restriction on $E(r_0)$ solely for the purpose of providing an intuitive feel for the range of r_0:

$$0 \leq E(r_0) \leq E(r). \tag{4}$$

The first inequality must hold since $r_0 \geq 0$ whereas the second is just an intuitive, almost arbitrary, restriction. The following observations concerning $E(r_0)$ are evident:

- $E(r_0) = 0$ implies that disk requests associated with the same array request finish at the same time and thus an array request is issued immediately whenever *any* disk request finishes.
- $E(r_0) = 0$ when $n = 1$, that is, when each array request consists of a single disk request. In this case, the completion of each disk request corresponds to the completion of the corresponding array request and, thus, the process that issued the disk request will immediately issue another array request.
- $E(r_0) \simeq 0$ when $n = N$, that is, when an array request always uses all the disks. In this case, disk requests associated with the same array request will tend to finish at close to the same time because all of the disks will be in very similar states and operate in a lock step fashion since disk service times are deterministic and disk requests across disks will be almost identical.

We will find it convenient to express $E(r_0)$ as a multiple of $E(R)/L$; thus, we introduce the parameter γ as $E(r_0) = \gamma E(R)/L$, where Restriction 4 implies $0 \leq \gamma \leq 1$. Later, we will empirical calibrate γ. For now, we know that $\gamma = 0$ when $n = 1$ and $\gamma \simeq 0$ when $n = N$. Rewriting Equation 3 in terms of γ we have,

$$U \simeq \frac{E(S)}{E(S) + \frac{p_0 E(R)}{L}((1/p - 1) + \gamma)}. \tag{5}$$

The following is the key approximation:

$$p_0 E(R)/E(S) = 1. \tag{6}$$

The above equation is true for $M/M/1$ systems but is unlikely to be completely accurate for the model system. We will later examine, via simulation, the accuracy, sensitivity and error introduced by this approximation. We can now rewrite Equation 5 as,

$$U \simeq \frac{1}{1 + \frac{1}{L}(1/p - 1 + \gamma)}. \tag{7}$$

Note that under the approximations we have made, *the expected utilization is insensitive to the disk service time distribution, S.*

Since this is a closed system, the expected response time can be directly calculated from the expected utilization:

$$E(R) = \frac{E(S)Ln}{UN}. \tag{8}$$

The expected throughput in megabytes per second can be written as,

$$MBS = \frac{UNSU}{E(S)}, \tag{9}$$

where SU is the size of the stripe unit. Future references to a specific analytic model will refer to the above equations and to Equation 7 in particular.

3.4 Summary

In this section we have derived a simple analytic model for disk arrays, $U \simeq \frac{1}{1+\frac{1}{L}(1/p-1+\gamma)}$, based upon two approximations:

- $E(\sum_{i=1}^{M} r_i) = (1/p - 1)E(R)/L$,
- $p_0 E(R)/E(S) = 1$.

With regard to the first approximation, we will show that it is very accurate and introduces very small errors. The second approximation is more difficult to justify. While it is not an accurate approximation for certain workloads, the error introduced into the analytic model is insensitive to the accuracy of the approximation under those same workloads. The approximation introduces errors on the order of $\pm 10\%$.

The model also contains an undefined parameter γ, a complex function of the model system's parameters. We will empirically calibrate the value of γ to a constant and show that this introduces only small errors to the analytic model.

4 Validation of the Analytic Model

In this section, we calibrate and validate the analytic model developed in the previous section via simulation. We show that the parameter γ can be calibrated to a constant. The resulting analytic model closely approximates the simulation results over the range of system and workload parameters investigated.

4.1 The Disk Model

The disk model is based upon the IBM 0661 3.5 inch 320 MB SCSI disk drive. Figure 4 tabulates the parameters and plots the seek profile of the simulated disk.

4.2 Simulation Parameters

The simulation parameters of interest are as follows:

- Input Variables.

  N Number of disks in array.

  SU Size of the stripe unit.

  L Load, that is, the number of processes generating array requests.

  SZ Array request size.

  S Disk service time distribution (implicit in the disk model).

- Output Variable.

  U Utilization. (Note that for the model system of interest, throughput is proportional to U and response time is inversely proportional to U.)

Recall that,

$$\begin{array}{rcl} n & \equiv & \text{Request size (number of disks/stripe-units accessed per request);} \\ & & SZ/SU. \\ p & \equiv & \text{Probability that a request will access a given disk;} \\ & & n/N. \end{array}$$

4.3 Simulation Results

Figure 5 plots the utilization from a representative simulation run versus the utilization predicted by Equation 7 for a disk array consisting of 17 disks and a stripe unit size of 32KB for four values of $\gamma \in \{0, 1, p(1-p), 0.15\}$. As previously mentioned, $\gamma \simeq 0$ when $n \in \{1, N\}$, that is, when $SZ \in \{SU, NSU\}$; thus, we first try $\gamma = 0$. As expected, Equation 7 with $\gamma = 0$ models the utilization of the system fairly well when $n \in \{1, N\}$; however, the analytic model with $\gamma = 0$ overestimates the utilization at other values of n. This is because $\gamma = 0$ underestimates $E(r_0)$ when n is between 1 and N. To get an idea for the sensitivity of the analytic model to γ, Figure 5 also plots the utilization of the simulation run versus the utilization predicted by Equation 7 over the same range of input parameters with $\gamma = 1$. The resulting analytic model is highly inaccurate.

cylinders per disk	949
tracks per cylinder	14
sectors per track	48
bytes per sector	512
track skew in sectors	4
revolution time	13.9 ms
single cylinder seek time	2.0 ms
average seek time	12.5 ms
max stroke seek time	25.0 ms
max sustained transfer rate	1.7 MB/s

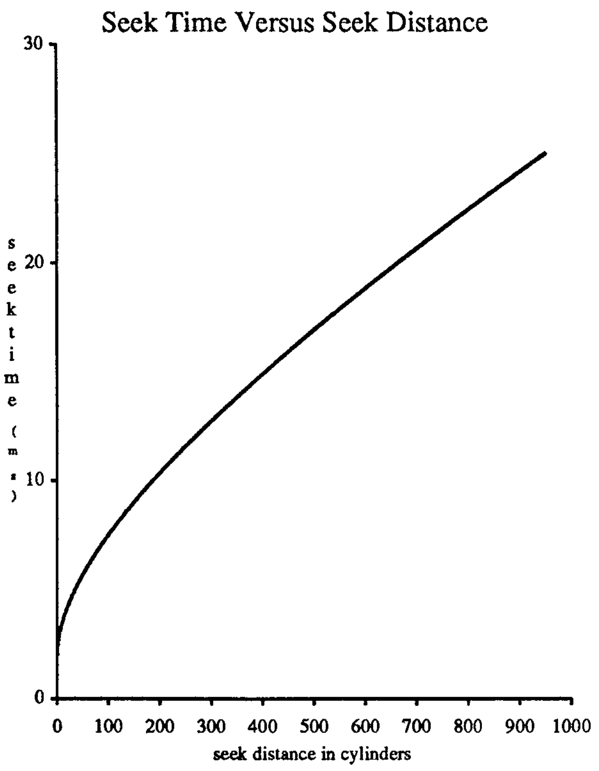


Figure 4: Disk Characteristics. *The graph plots the seek time in milliseconds versus the seek distance in cylinders. The curve is derived from the following formula:*

$$seekTime = \begin{cases} 0, & \text{if } x = 0 \\ a(x-1)^{0.5} + b(x-1) + c, & \text{if } x > 0 \end{cases}$$

where x is the seek distance in cylinders and a, b and c are constants chosen to satisfy the single cylinder, max stroke and average seek times. For the simulated disk, $a = 0.4623, b = 0.0092$ and $c = 2$.

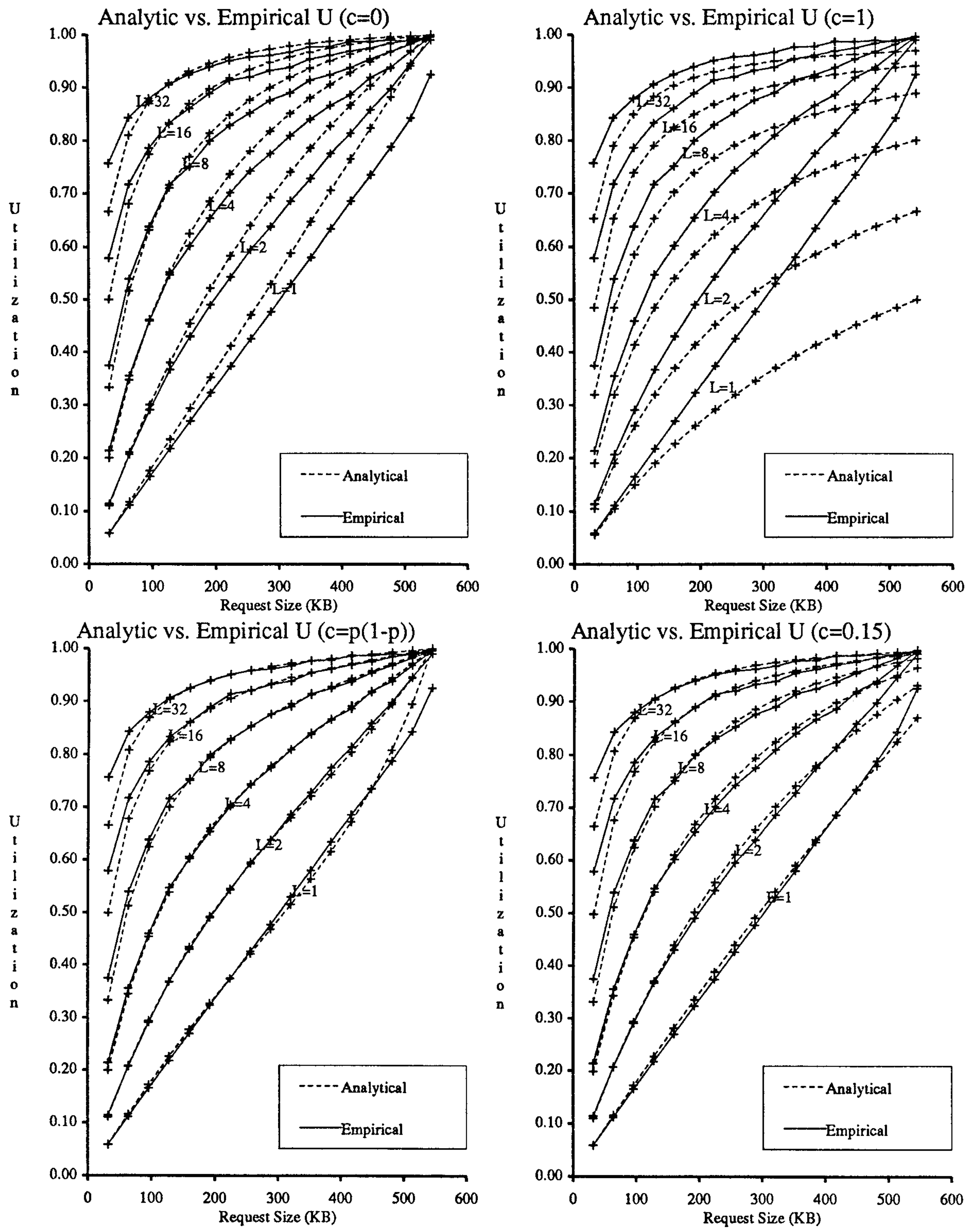


Figure 5: Analytic vs. Empirical Models. $c \equiv \gamma$; $N = 17$; $SU = 32KB$; $L \in \{1, 2, 4, 8, 16, 32\}$; $SZ \in \{32KB, 64KB, \ldots, 544KB = N \times SU\}$. *Each pair of lines is labeled with its corresponding value of L. Each graph illustrates the accuracy of the analytic model for the given value of γ which is denoted by c in the title of each graph.*

The best result is achieved when $\gamma = p(1-p)$. This will make $\gamma \simeq 0$ at the boundaries when $n \in \{1, N\}$ and somewhat positive elsewhere. In this case, 0.25 is the maximum value for γ reached at $p = 0.5$. The resulting correspondence between the analytic and simulated utilization is very good. Unfortunately, using $\gamma = p(1-p)$ introduces higher order dependencies with respect to p into Equation 7. Consequently, we will try $\gamma = 0.15$ to see if we can improve over our original choice of $\gamma = 0$ without introducing higher order dependencies. The resulting correspondence between the analytic and simulated utilization, while not as good as $\gamma = p(1-p)$, is still good. Henceforth, we will assume that γ can be accurately modeled as a constant and where a specific value for γ is needed, we will assume that $\gamma = 0.15$.

5 Error Analysis

In the derivation of the analytic model, we have made the following approximations:

- $p_0 E(R)/E(S) \simeq 1$,
- $E(\sum_{i=1}^{M} r_i) \simeq (1/p - 1)E(R)/L$,
- $\gamma \simeq 0.15$.

A previous section has already shown that the above approximations result in an accurate analytic model over a range of system and workload parameters. In this section, we examine the accuracy, sensitivity and the error introduced by each approximation. Our methodology is to rewrite the exact equation for utilization, Equation 2, in terms of variables α, β and γ. If we could calculate the values of these variables exactly, we would have an exact analytic model. We show that the approximate analytic model, Equation 7, can be derived by substituting specific estimates for the true values of the variables α, β and γ. Thus, we can study how the approximations affect the error in the analytic model by determining how inaccuracies in the estimated values for α, β and γ contribute to the error in the analytic model.

5.1 Exact and Approximate Models

Let

$$\begin{aligned} \alpha &\equiv p_0 E(R)/E(S), \\ \beta &\equiv E(\sum_{i=1}^{M} r_i)L/E(R), \\ \gamma &\equiv E(r_0)L/E(R). \end{aligned}$$

Note that $\beta \simeq E(M)$, the average number of array requests that are issued until an array request that accesses a given disk is issued. Rewritting Equation 2 in terms of α, β and γ we have,

$$U = \frac{1}{1 + \frac{\alpha}{L}(\beta + \gamma)}. \tag{10}$$

Note that Equation 10 is an exact equation for the expected utilization of the model system and does not utilize any of the approximations used in deriving the analytic model.

Let

$$\begin{aligned}\hat{\alpha} &\equiv 1,\\ \hat{\beta} &\equiv 1/p-1,\\ \hat{\gamma} &\equiv 0.15.\end{aligned}$$

The above definitions for $\hat{\alpha}, \hat{\beta}$ and $\hat{\gamma}$, estimating the true values α, β and γ respectively, directly correspond to the three approximations we have made in deriving the analytic model. Substituting $\hat{\alpha}, \hat{\beta}$ and $\hat{\gamma}$ for α, β and γ into Equation 10 results in the approximate analytic model given by Equation 7 and denoted below as $\hat{U}$:

$$\hat{U} \equiv \frac{1}{1+\frac{\hat{\alpha}}{L}(\hat{\beta}+\hat{\gamma})}. \tag{11}$$

The primary question of interest in the following section is how errors in the estimators $\hat{\alpha}, \hat{\beta}$ and $\hat{\gamma}$ affect the error of the analytic model, $\hat{U}$.

5.2 Propagation of Error

The previous section has shown that the approximate equation for utilization, Equation 7, can be viewed as derived from the exact equation for utilization, Equation 2, by estimating the variables α, β and γ. This section looks at how inaccuracies in the estimates of α, β and γ affect the error of the analytic model.

We know that,

$$dU = \frac{\partial U}{\partial \alpha}\, d\alpha + \frac{\partial U}{\partial \beta}\, d\beta + \frac{\partial U}{\partial \gamma}\, d\gamma. \tag{12}$$

The above equation shows how small changes in α, β and γ affects U. Analogously,

$$\delta\hat{U} \simeq \frac{\partial \hat{U}}{\partial \hat{\alpha}}\, \delta\hat{\alpha} + \frac{\partial \hat{U}}{\partial \hat{\beta}}\, \delta\hat{\beta} + \frac{\partial \hat{U}}{\partial \hat{\gamma}}\, \delta\hat{\gamma}. \tag{13}$$

The above equation shows how small inaccuracies in $\hat{\alpha}, \hat{\beta}$ and $\hat{\gamma}$ affect the error in $\hat{U}$. For example, the first term of Equation 13 shows how small inaccuracies in $\hat{\alpha}$ affects the accuracy of $\hat{U}$; unfortunately, the first term of Equation 13 also depends on $\hat{U}$ which depends on $\hat{\beta}$ and $\hat{\gamma}$. This means that we may incorrectly calculate the error contributed by $\hat{\alpha}$ due to inaccuracies in the other two variables.

Because of the above drawback to using Equation 13 directly, we will instead use the following equations as the basis of error analysis.

$$\hat{U}_{\alpha} \equiv \frac{1}{1+\frac{\hat{\alpha}}{L}(\beta+\gamma)}, \tag{14}$$

$$\hat{U}_{\beta} \equiv \frac{1}{1+\frac{\alpha}{L}(\hat{\beta}+\gamma)}, \tag{15}$$

$$\hat{U}_{\gamma} \equiv \frac{1}{1+\frac{\alpha}{L}(\beta+\hat{\gamma})}. \tag{16}$$

The main advantage to using the these equations rather than Equation 11 is that errors in $\hat{U}_{\alpha}$, $\hat{U}_{\beta}$ and $\hat{U}_{\gamma}$ can be directly attributed to the inaccuracy in $\hat{\alpha}$, $\hat{\beta}$ and $\hat{\gamma}$ respectively. Table 1 formally defines the terms *error, sensitivity, accuracy, relative error* and *relative accuracy.*

Estimator	Error	Sensitivity	Accuracy	Rel Err	Rel Acc
$\hat{\alpha}$	$\hat{U}_\alpha - U$	$\frac{\partial U}{\partial \alpha}$	$\hat{\alpha} - \alpha$	$\frac{\hat{U}_\alpha - U}{U}$	$\frac{\hat{\alpha}-\alpha}{U}$
$\hat{\beta}$	$\hat{U}_\beta - U$	$\frac{\partial U}{\partial \beta}$	$\hat{\beta} - \beta$	$\frac{\hat{U}_\beta - U}{U}$	$\frac{\hat{\beta}-\beta}{U}$
$\hat{\gamma}$	$\hat{U}_\gamma - U$	$\frac{\partial U}{\partial \gamma}$	$\hat{\gamma} - \gamma$	$\frac{\hat{U}_\gamma - U}{U}$	$\frac{\hat{\gamma}-\gamma}{U}$

Table 1: Definition of Error, Sensitivity and Accuracy. *The above definitions use the true values U, α, β and γ which are unknown. We will use the simulated values for U, α, β and γ whenever the true values are required for computations. Although the simulated values will never equal the true values, the simulated values can be made to approximate the true values arbitrarily closely. For comparison purposes, we will find it convenient to use relative error and relative accuracy rather than error and accuracy directly. Note that just as the error is approximately equal to the accuracy times the sensitivity, the relative error is approximately equal to the relative accuracy times the sensitivity.*

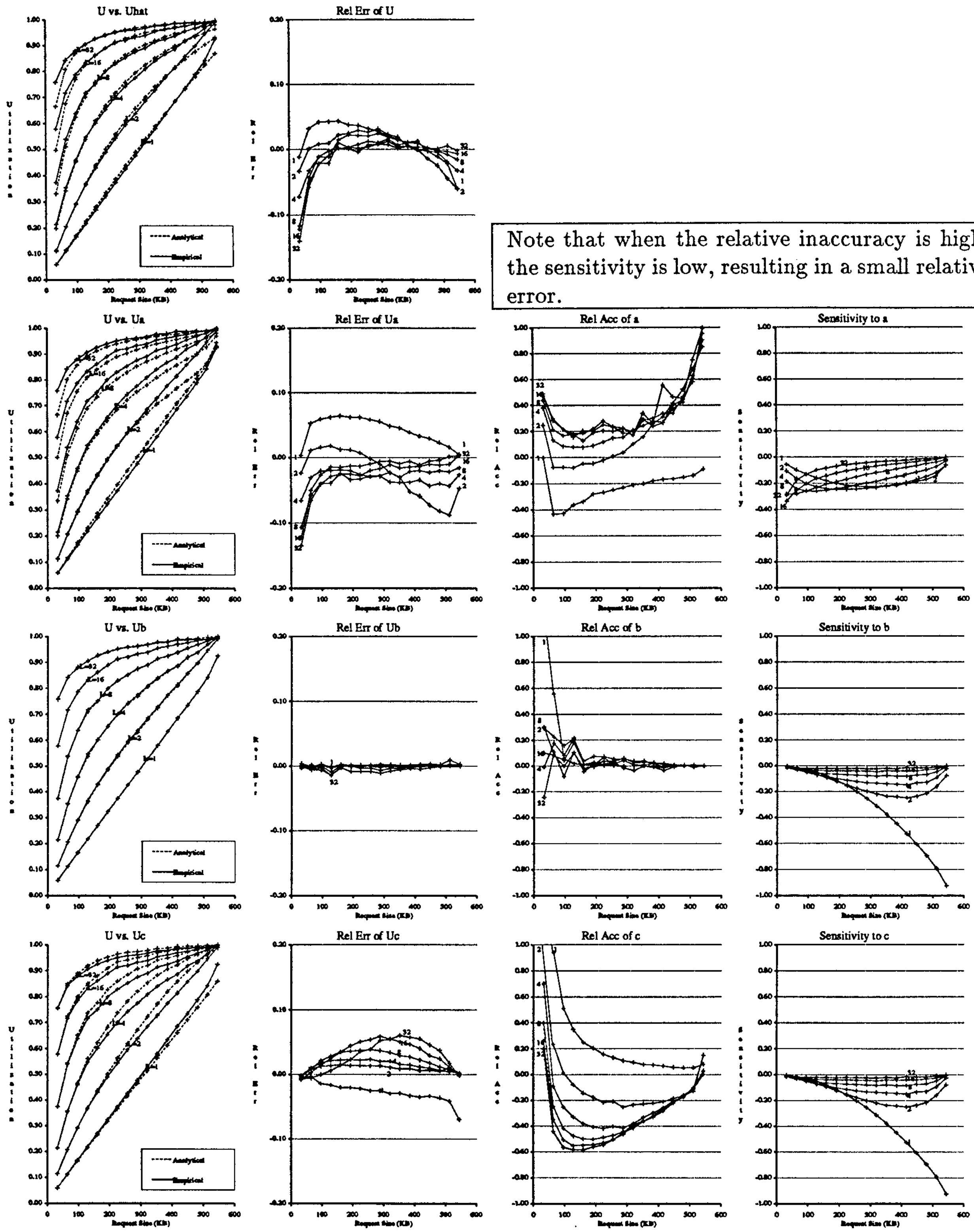
5.3 Simulation Results

Figure 6 plots the simulated and estimated U, relative error, relative accuracy and sensitivity corresponding to each of the estimated parameters $\hat{\alpha}, \hat{\beta}$ and $\hat{\gamma}$. The first row of graphs in Figure 6 illustrates the overall relative error in the analytic model. This is roughly equal to the sum of the relative errors due to $\hat{\alpha}, \hat{\beta}$ and $\hat{\gamma}$ illustrated in the succeeding rows. Note that the overall relative error of the analytic model is generally smaller than $\pm 5\%$. Before discussing the relative error, relative accuracy and sensitivity of $\hat{\alpha}, \hat{\beta}$ and $\hat{\gamma}$ individually, we make several general comments concerning all three variables. First, the relative errors due to $\hat{\alpha}, \hat{\beta}$ and $\hat{\gamma}$ rarely exceeds $\pm 10\%$. Second, the relative inaccuracy rarely exceeds ± 1. Third, for the simulated parameters, the absolute value of the sensitivity of all three variables is always less than one and is small in general. This implies that the model is fairly robust and is insensitive to inaccuracies in the approximations used to derive the model. Fourth, when the relative inaccuracy is high, the sensitivity tends to be low, resulting in a small relative error. Fifth, the sensitivity of all three variables tends to decrease as L increases.

The second row of Figure 6 illustrates the relative error, relative accuracy and sensitivity of $\hat{\alpha}$ and corresponds to the approximation $p_0 E(R)/E(S) \simeq 1$. Simulation shows that this is a good approximation for small request sizes but is inaccurate for large request sizes; as request sizes become large, $p_0 E(R)/E(S)$ approaches zero. Fortunately, the sensitivity also decreases with increasing request size resulting in small errors. Note from Figure 6 that at the smallest request size, the approximation becomes less accurate with increasing load. The absolute value of the sensitivity also increases with increasing load but reaches a maximum of approximately 0.35 then decreases. This leads us to believe that the analytic model will continue to display relatively small errors at higher loads.

The third row illustrates $\hat{\beta}$ and corresponds to the approximation $E(\sum_{i=1}^{M} r_i) \simeq (1/p-1)E(R)/L$. As previously stated, we believe this to be a very good approximation which evidence now confirms. In the graph plotting U versus $\hat{U}_\alpha$, the two sets of lines are almost indistinguishable. The relative error is generally less than $\pm 1\%$.

Finally, the fourth row corresponds to the approximation $\gamma \simeq 0.15$. In addition to the general comments already made, we note that the error introduce by $\hat{\gamma}$ tends to cancel out the error



Note that when the relative inaccuracy is high, the sensitivity is low, resulting in a small relative error.

Figure 6: Error, Accuracy and Sensitivity.

introduced by $\hat{\alpha}$. This is not surprising given that $\hat{\gamma}$ was empirically calibrated to reduce the overall error in the analytic model.

5.4 Summary

In this section we examined the error introduced by the approximations made in deriving the analytic model. We examined the accuracy and sensitivity of each approximation. While some of the approximations are grossly inaccurate for certain workloads, this does not introduce large errors because the model is insensitive to the approximations at such workloads. Finally, because the model is generally insensitive to inaccuracies in the approximations, it is reasonably robust.

6 The Optimal Stripe Unit Size

In this section, we will use the analytic model to derive an equation for the *optimal stripe unit size*, the stripe unit size that maximizes throughput in megabytes per second. The equation for the optimal stripe unit size is useful as a rule of thumb in configuring disk arrays and also provides valuable insights into the factors that influence the optimal stripe unit size. Given today's disk technology, the optimal stripe unit equation is most useful for workloads consisting of I/O requests that are a couple of hundred or more kilobytes in size. Miller [11] has shown that such workloads are typical of scientific applications. For such workloads, we have found that there is typically a 10-20% degradation in performance when the stripe unit is a factor of two smaller or larger than the optimal size.

In addition to deriving the equation for the optimal stripe unit size, we will show that the stripe unit size that maximizes throughput also minimizes response time. Note, however, that maximizing throughput is *not* the same as maximizing utilization; just because a disk is busy does not mean that it is doing useful work. The fundamental tradeoff in selecting a stripe unit size is one of parallelism versus concurrency. Small stripe unit sizes increase the parallelism available for servicing a single request by mapping a request over a larger number of disks but reduce concurrency because each request uses a greater number of disks [4].

6.1 Derivation

We will derive the equation for the optimal stripe unit size from Equation 9. But first, because the disk service time, S, is dependent on the stripe unit size, SU, we must formulate a simple model that makes this dependency explicit. Recall the definition for the following disk parameters:

- **P** is the average positioning time (seek + rotational latency).
- **X** is the sustained data transfer rate (this is the rate that the disk head reads data off of the disk platter).

Then,

$$E(S) = P + SU/X. \tag{17}$$

Note that n, the number of stripe units per request can be calculated as follows:

$$n = SZ/SU. \tag{18}$$

Substituting equations 7, 17 and 18 into Equation 9 and simplifying we can write the throughput in megabytes second as,

$$MBS = \frac{LNX\,SU\,SZ}{(PX + SU)(N\,SU + SZ(L - 1 + \gamma))}. \tag{19}$$

Solving for the local maxima in the above equation as a function of SU (γ is assumed to be a constant) we get the following equation for the optimal stripe unit size:

$$optSU = \sqrt{\frac{PX(L - 1 + \gamma)SZ}{N}}. \tag{20}$$

Repeating the above procedure to minimize response time starting from Equation 8 results in the same equation for the optimal stripe size; thus, the stripe unit size that maximizes throughput also minimizes response time and is given by Equation 20.

The following remarks can be made about Equation 20:

- Changes to the system that increase the effective load, that is, an increase in L, an increase in SZ, or a decrease in N, favor larger optimal stripe units. The opposite is true for changes that decrease the effective load.

- In our model system, the optimal stripe unit size is dependent only on the product PX, the relative rate at which a disk can position and transfer data, and not on P or X independently. If you replace the disks with those that position and transfer data twice as quickly, the optimal stripe unit size remains unchanged [4]. In this respect, the selection of an optimal stripe unit size is a trade-off between the disk positioning time and the data transfer time.

6.2 Validation

As a further validation of the analytic model and of Equation 20 in particular, we compare the analytic values for the optimal stripe unit size with empirically determined values. Figure 7 plots the analytically determined optimal stripe unit sizes versus the empirically determined optimal stripe unit sizes on a log-log scale. The shaded regions on the figure represent optimal stripe unit sizes that can be ruled out for the following reasons. First, throughput when $SU < SZ/N$ for fixed SZ is less than or equal to the throughput when $SU = SZ/N$. At this stripe unit size, requests are being distributed uniformly across all disks and it is not possible to increase parallelism or concurrency by reducing the stripe unit size. Second, throughput when $SU > SZ$ for fixed SZ is identical to when $SU = SZ$. In this case, the request already fits completely within a single disk and there is no advantage or disadvantage to increasing the stripe unit size. We have empirically verified the above two facts. Thus, $SZ/N \leq SU \leq SZ$. For comparison purposes, Figure 8 adds the optimal stripe unit sizes predicted by Chen [4]. Note that Chen's model assumes that the optimal stripe unit size is independent of the request size.

To get a feel for the sensitivity of performance to the choice of stripe unit size, Figure 9 individually plots each group of lines from Figure 8 with vertical bars to indicate the range of stripe unit sizes providing 95% of the throughput of the optimal stripe unit size. Note that there is fairly good correlation between the analytically and empirically determined values for $optSU$. In all cases except when $L = 1$ and request sizes are small, the optimal stripe unit sizes determined

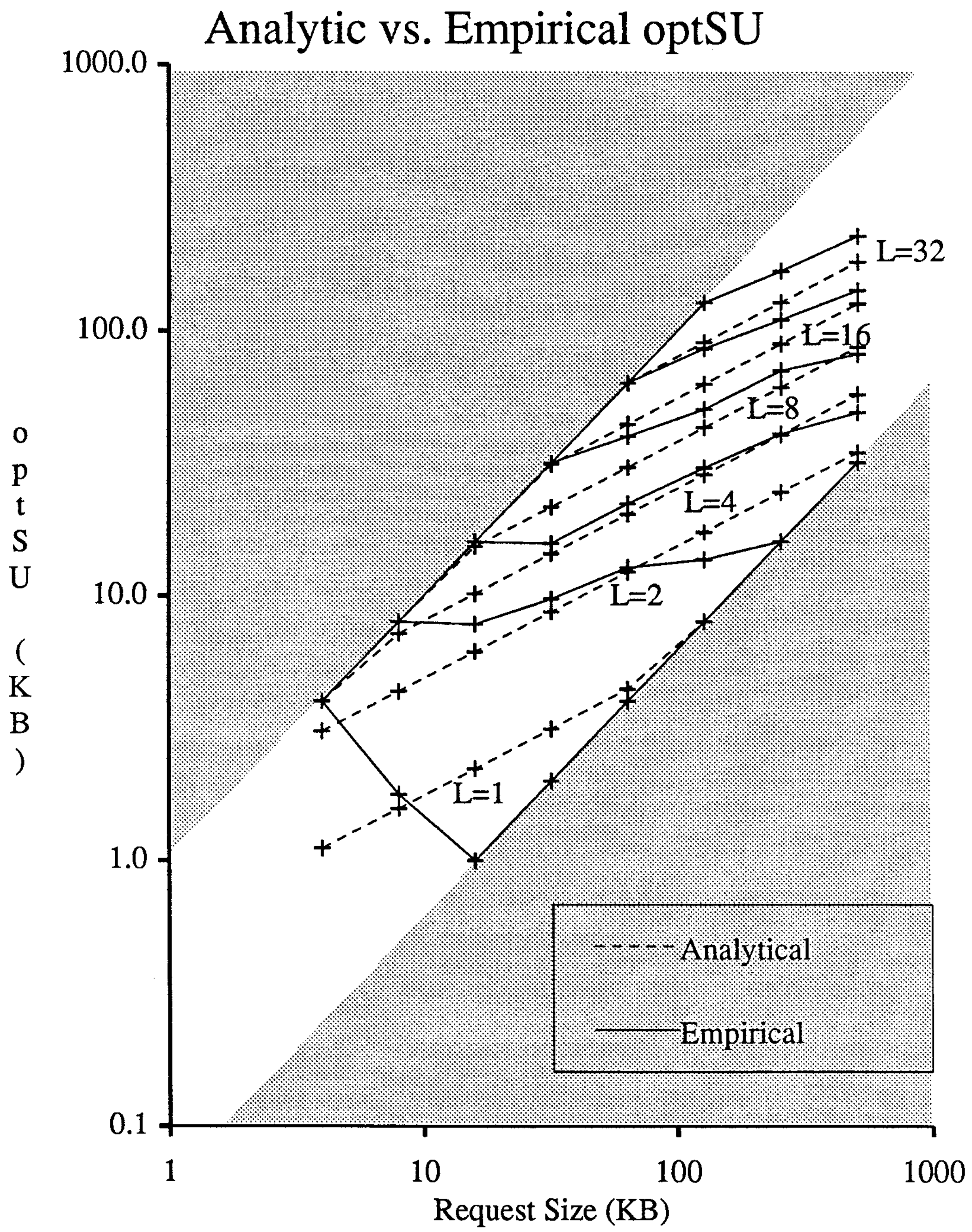


Figure 7: Analytic vs. Empirical Optimal Stripe Unit Sizes. $N = 16$; $SU = 32KB$; $L \in \{1, 2, 4, 8, 16, 32\}$; *Each pair of lines is labeled with its corresponding value of L. Optsu = Optimal stripe unit size in kilobytes. The graph may appear odd at first because the beginning and end of the individual lines overlap at the edges of the shaded regions.*

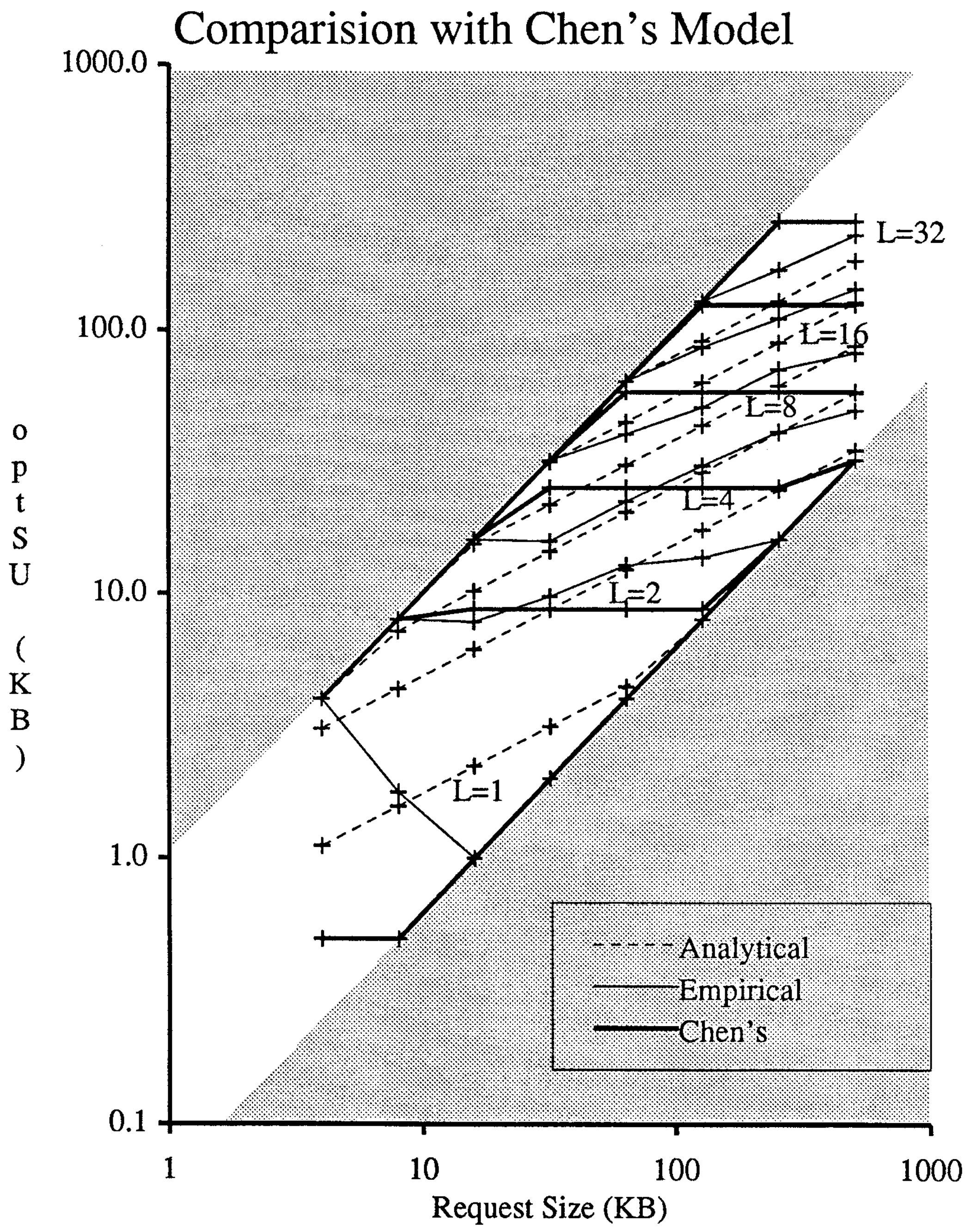


Figure 8: Comparison with Chen's Model. *$N = 16$; $SU = 32KB$; $L \in \{1, 2, 4, 8, 16, 32\}$; Each pair of lines is labeled with its corresponding value of L. Optsu = Optimal stripe unit size in kilobytes.*

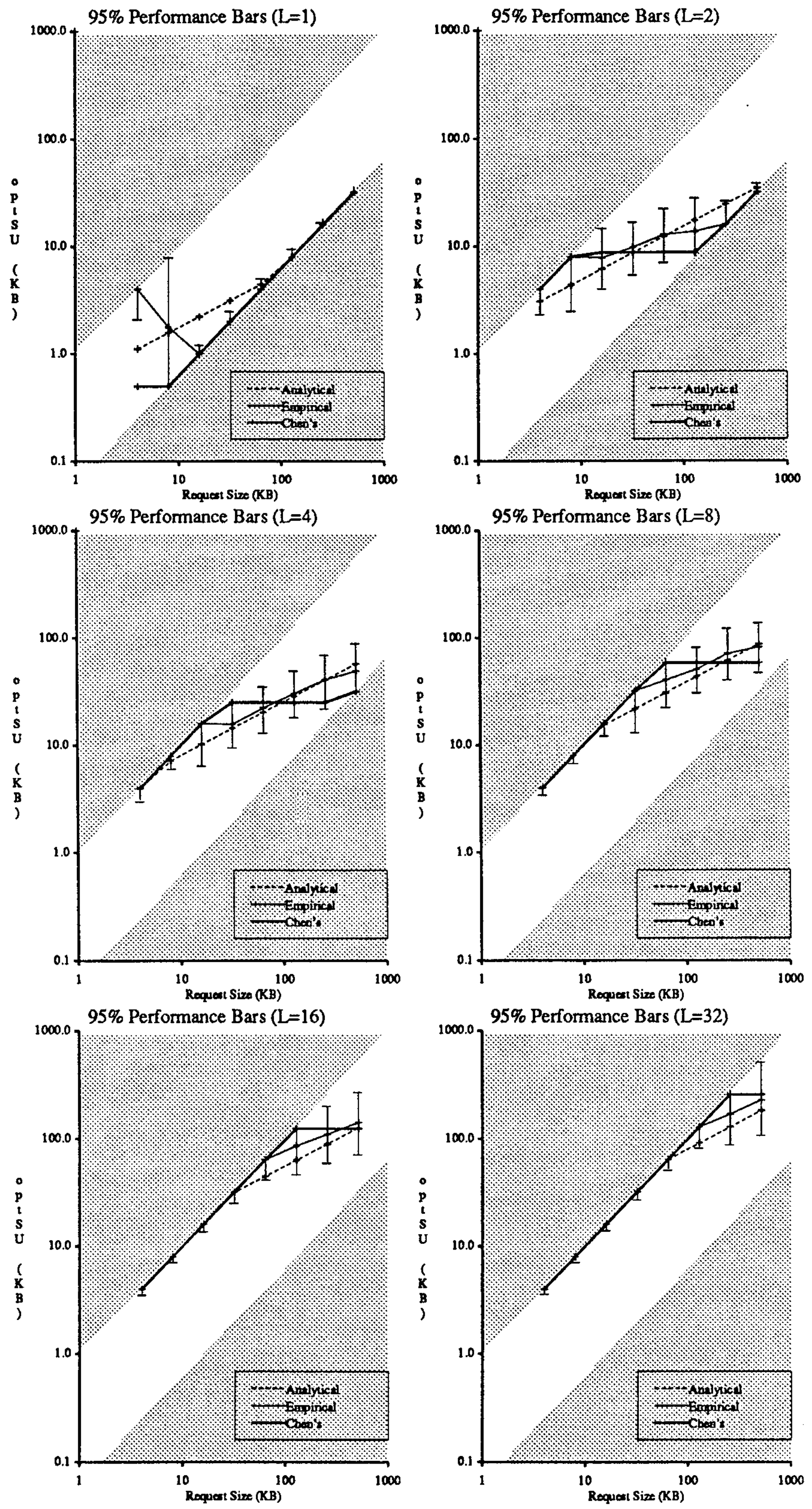


Figure 9: Optimal Stripe Unit Sizes with 95% Performance Intervals. $N = 16$; $SU = 32KB$; *Optsu = Optimal stripe unit size. Stripe unit sizes within the solid vertical bars provide 95% of the performance of the optimal stripe unit size indicated by the oblique thin solid line.*

by both Chen's model and our model lie within the 95% performance intervals. This is remarkable given the different simulation methodologies and criteria used in selecting the optimal stripe unit size and indicates that the optimal stripe unit size is a robust property.

6.3 Summary

In this section, we derived an equation for the optimal stripe unit size and validated it via simulation. The stripe unit size that maximizes throughput also minimizes response time and is given by $optSU = \sqrt{\frac{PX(L-1+\gamma)SZ}{N}}$. We showed that the optimal stripe unit size is dependent only on the relative rates at which a disk can position and transfer data, PX. Our equation for the optimal stripe unit size agrees well with Chen's [4] equation.

7 Summary and Future Work

We have derived, validated and applied an analytic performance model for disk arrays. We modeled disk arrays as a closed queueing system consisting of a fixed number, L, of processes continuously issuing requests of a fixed size, n, to a disk array consisting of N disks. The expected utilization of the model system, U, is approximately $\frac{1}{1+\frac{1}{L}(1/p-1+0.15)}$ where $p = n/N$ is the size of the request as a fraction of the number of disks in the disk array. We directly derived the expected response time and throughput in megabytes per second as $\frac{E(S)Ln}{UN}$ and $\frac{UNSU}{E(S)}$ respectively where $E(S)$ is the expected service time of a disk request. We showed via simulation that the utilization predicted by the analytic model is generally within $\pm 5\%$ of the simulated values. We examined the error, accuracy and sensitivity of each approximation made in the derivation of the analytic model to better understand the validity and limits of the model. Finally, we applied the analytic model to show that the optimal unit of data striping simultaneously maximizes throughput and minimizes response time and is equal to $\sqrt{\frac{PX(L-1+0.15)SZ}{N}}$ where P is the average disk positioning time, X is the average disk transfer rate and SZ is the request size.

There are several major areas for future work with respect to the analytic model presented here. First, one can extend the workload model to handle non-constant distributions of request sizes and something similar to CPU think time, where the processes, instead of simply issuing I/O requests, would alternate between computation and I/O. Second, one can extend the types of disk arrays to which the analytic model can be applied. In particular, it would be highly desirable to model RAID, Redundant Arrays of Inexpensive Disks [13], systems. Finally, the analytic model can be applied to other problems in the design and configuration of disk arrays.

8 Acknowledgements

We would like to thank Peter Chen for taking the time to answer specific questions about his work and Garth Gibson for the disk model we used in our simulations. We would also like to thank our government and industrial affiliates, Array Technologies, DARPA/NASA (NAG2-591), DEC, Hewlett-Packard, IBM, Intel Scientific Computers, California MICRO, NSF (MIP 8715235), Seagate, Storage Tek, Sun Microsystems and Thinking Machines Corporation for their financial support.

References

[1] Francois Baccelli. Two parallel queues created by arrivals with two demands. Technical Report 426, INRIA–Rocquencourt France, 1985.

[2] Dina Bitton and Jim Gray. Disk shadowing. In *Proc. Very Large Data Bases*, August 1988.

[3] Peter M. Chen, Garth A. Gibson, Randy H. Katz, and David A. Patterson. An evaluation of redundant arrays of disks using an amdahl 5890. In *Proc. SIGMETRICS*, pages 74–85, May 1990.

[4] Peter M. Chen and David A. Patterson. Maximizing performance in a striped disk array. In *Proc. International Symposium on Computer Architecture*, May 1990.

[5] L. Flatto and S. Hahn. Two parallel queues created by arrivals with two demands i. *SIAM J. Appl. Math.*, October 1984.

[6] Philip Heidelberger and Kishor S. Trivedi. Queueing network models for parallel processing with asynchronous tasks. *IEEE Trans. on Computers*, January 1982.

[7] R. H. Katz, G. A. Gibson, and D. A. Patterson. Disk system architectures for high performance computing. In *Proc. IEEE*, pages 1842–1858, December 1989.

[8] M. Y. Kim. Synchronized disk interleaving. *IEEE Trans. on Computers*, C-35:978–988, November 1986.

[9] M. Y. Kim and A. N. Tantawi. Asynchronous disk interleaving. Technical Report RC12497, IBM, January 1987.

[10] M. Livny, S. Khoshafian, and H. Boral. Multi-disk management algorithms. In *Proc. SIGMETRICS*, pages 69–77, May 1987.

[11] Ethan L. Miller. Input/output behavior of supercomputing applications. Technical Report UCB/CSD 91/616, University of California at Berkeley, 1991.

[12] David A. Patterson, Peter M. Chen, Garth Gibson, and Randy H. Katz. Introduction to redundant arrays of inexpensive disks (RAID). In *Proc. IEEE COMPCON*, pages 112–117, Spring 1989.

[13] David A. Patterson, Garth Gibson, and Randy H. Katz. A case for redundant arrays of inexpensive disks (RAID). In *Proc. ACM SIGMOD*, pages 109–116, June 1988.

[14] A. L. Narasimha Reddy and Prithviraj Banerjee. An evaluation of multiple-disk I/O systems. *IEEE Trans. on Computers*, December 1989.

[15] K. Salem and H. Garcia-Molina. Disk striping. In *Proc. IEEE Data Engineering*, pages 336–342, February 1986.