From *IEEE International Computer Performance and Dependability Symposium (IPDS '95)*, Erlangen, Germany, April 24-26, 1995, pp. 236-245

PERFORMANCE ANALYSIS OF THE RAID 5 DISK ARRAY

Anand Kuratti and William H. Sanders
Center for Reliable and High-Performance Computing
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
1308 W. Main St., Urbana, IL 61801
(kuratti,whs)@crhc.uiuc.edu

Abstract

While the processor and memory performance of computers continues to improve, I/O performance has remained relatively constant. Strategies to achieve better I/O performance than current disk systems have been investigated to address the growing I/O bottleneck. One effort is the RAID (Redundant Arrays of Inexpensive Disks) Level 5 disk array. RAID 5 offers increased parallelism of I/O requests through the disk array architecture and fault tolerance through rotated parity. Although RAID 5 offers the promise of improved I/O performance, such gains must be quantified. Accurate analytical modeling can be used to quantify and predict the I/O request response time for different workloads. While previous analytical models to calculate the I/O request response time have been constructed, they often rely upon simplifying assumptions or computational bounds, which are accurate for certain values of the possible workload parameters. This paper presents an analytical performance model to compute the mean steady state response time of a RAID 5 I/O request under a transaction-processing workload. By carefully considering individual disk accesses, the arrival process of disk requests, and correlation between disk requests, an accurate model for a wide range of the workload parameters is developed.

1 Introduction

Over the past decade, it has been recognized that as CPU and memory performance of computers continue to dramatically increase, I/O performance has improved at a much slower pace. To address the growing gap between processor/memory and I/O performance, new strategies for mass data storage have begun to be developed. One way to increase I/O performance is to use an array of disks. By interleaving data across many disks, both throughput (megabytes per second) and the I/O rate (requests per second) are improved. However with more disks, the reliability and availability of disk arrays is lower than for single disks. Because disk arrays hold the promise of improved performance, different ways to design and organize disk array architectures have been investigated. One effort is Redundant Arrays of Inexpensive Disks (RAID).

Introduced in [1, 2], Patterson, Gibson, and Katz present five ways to introduce redundancy into an array of disks: RAID Level 1 to RAID Level 5. For each level, data is interleaved across multiple disks, but the incorporated redundancy ranges from traditional mirroring to rotated parity. Using a simple model of maximum throughput, the authors suggest that RAID 5 with rotated parity offers the best performance potential of the organizations considered.

Although it appears that RAID 5 can achieve lower I/O request response times than single disks, tools for modeling and analyzing response time under different operating conditions are important to be able to compare RAID 5 and current disk systems. In particular, analytical models combined with a realistic assessment of workload would allow a system architect to accurately design and predict the performance of a RAID 5 disk array. But like many parallel systems, disk arrays are difficult to model because of the effect of *queuing and fork-join synchronization*. Since data is placed on multiple disks, each I/O request requires several disk requests. Each disk request waits to access a disk then waits for the other disk requests to complete before the I/O request can complete. Under general conditions, exact analysis of the effect of these non-independent delays on the I/O request response time is not possible. However, approximate analysis such as computation of upper and lower bounds of the mean I/O request response time is possible by careful consideration of the characteristics of the system.

This paper presents an analytical model to calculate the mean steady state response time for a RAID 5 I/O request under a transaction-processing workload. By systematically deriving the distribution of time for a disk to service a request, the arrival process of requests to individual disks in the array, and the time for all disk requests in an I/O request to complete, a model which considers both queuing and fork-join synchronization is developed. Based on this model, analytical values for the mean overall, mean read, and mean write response times for I/O requests are computed over a wide range of the workload parameters and compared to results obtained from a detailed simulation model.

Previous analytical models of I/O request response time include [3, 4, 5, 6]. In particular, Lee and Katz [3] constructed a closed queuing model and compared several disk utilization approximations to results from a detailed simulation model. Although a formula for response time based on disk utilization is derived, it is based on the mean service time for a disk request which is not determined. Chen and Towsley [4, 5] modeled RAID 5 performance using queuing analysis and determined the mean overall, mean read and mean write I/O request response times. They considered write synchronization, the effect of different request size distributions, and disk access skewing. Thomasian and Menon [6] developed a model for I/O request response time under normal, degraded, and rebuild modes based on an $M/G/1$ queue with server vacations.

In [4, 5, 6], the authors assume that arrivals to each disk in the array can be approximated as independent Poisson processes with a uniform rate. Although both report accurate results for the mean I/O request response time based on the workloads considered, we show how such assumptions can lead to inaccurate results when a larger range of the workload is considered. In our model, we illustrate how the workload influences the arrival and service of disk requests. As a result, we can analytically determine the regions of the workload where such assumptions are appropriate and verify these conclusions by detailed simulation.

The organization of this paper is as follows: section 2 briefly describes the RAID 5 architecture, including data and parity assignment and I/O methods. Using this description, the workload and assumptions used in developing the analytical models are presented in sections 3 and 4. Section 5 develops a model for individual disk accesses, including derivation and approximation of the time to service a disk request. In section 6, we show how disk requests are distributed to individual disks given Poisson arrivals of I/O requests and a transaction processing workload. Based on the arrival process and disk service time for disk requests, section 7 demonstrates that response time for an I/O request is the maximum of the completion times of the resulting disk requests. Using the completion time for an I/O request, formulas for the mean overall, mean read, and mean write response times are derived. Section 8 presents graphs of the analytical and simulated mean I/O request response times. Finally, section 9 gives conclusions and directions for future research.

2 RAID 5 Architecture

2.1 Data and Parity Placement

A RAID 5 disk array consists of N identical disks on which data is interleaved. The unit of data interleaving, or amount of data that is placed on one disk before data is placed on the next disk, is a *stripe unit.* Since disks are organized into rows and columns, the set of stripe units with the same physical location on each disk in a row is a *stripe.* Each stripe contains data stripe units and a *parity stripe unit.* A parity stripe unit is the *exclusive-or* (XOR) of all the data stripe units in the stripe. The presence of a parity stripe unit in each stripe allows a RAID 5 disk array to tolerate single disk failures in each row of disks. When a single disk in a row fails, a data stripe unit can be reconstructed by reading the corresponding data and parity stripe units from the other disks in the stripe. The number of stripe units in a stripe is defined as the *stripe width* (W_s), where the number of data stripe units in each stripe is $W_s - 1$.

Parity stripe units in a RAID 5 disk array are rotated, i.e. distributed uniformly across all disks in a RAID 5 disk array. This helps to balance the increased load at each disk caused by requests that update parity. Another advantage of rotated parity is that data is also distributed more evenly. A more uniform distribution of data increases the probability that a disk will participate in an I/O operation, which increases the throughput and I/O rate of the array.

2.2 I/O Methods

Given a description of how data and parity are placed on a RAID 5 disk array, operations to read data from and write to the array can be defined. A read request for data results in read requests for data stripe units at individual disks. When all stripe units have been accessed, the I/O request is complete. For a write request, data stripe units must not only be written, but the corresponding parity stripe units must be updated. Depending on how much of a stripe is written and how many stripes are accessed, three different cases can occur.

1. If the request starts at the first data disk in a stripe and the request size is $W_s - 1$ data stripe units, all data stripe units in the stripe are written. This is called a *full stripe write.* Since the parity stripe unit is the exclusive-or of all data stripe units in the stripe and all data stripe units are written, the new parity stripe unit is generated entirely from new data. The request completes when all data and parity stripe units are written.

2. If the request consists of less than $W_s - 1$ data stripe units and all stripe units belong to the same stripe, the stripe is partially modified. Since the new parity stripe unit is a combination of the new

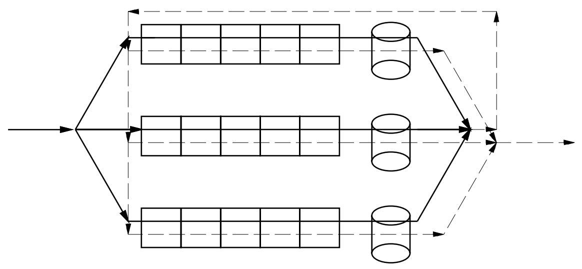

Figure 1: Write Synchronization

and old data stripe units, the new parity is computed by reading the old data and parity stripe units, then XORing the old and new data. The request completes after the new data and parity stripe units have been written.

3. If the request accesses two or more stripes, i.e. data stripe units requested are allocated across stripe boundaries, two or more partial and/or full stripes must be updated. Because an operation on one stripe does not effect the data and parity stripe units in another stripe, a write I/O request that accesses multiple stripes is divided into several partial and/or full stripe writes. The I/O request is complete when all operations finish.

A potential problem that arises during a partial stripe write is how the parity stripe unit is updated. If the old data stripe units are overwritten by new data before the new parity stripe unit is calculated, the parity stripe unit will be incorrect. To ensure that old data and parity stripe units are read before new data is written, the write request must be *synchronized* between read and write phases. Several methods have been proposed to ensure synchronization [7]. For example, Chen and Towsley [4] model the disk-first with priority scheme, in which parity requests are issued when the data accesses reach the head of the disk queues, but the parity access is given higher priority than non-parity accesses at the same disk. In our model shown in Figure 1, synchronization is ensured by waiting for all reads for old data and parity, then issuing the write requests for the new data and parity. As opposed to [4], this method incurs higher response time for the parity update requests by not using priority, but reduces the response times for other requests at the same disk.

3 System Workload

The workload we consider is determined by the size and arrival pattern of I/O requests. Since the arrivals of I/O requests to the disk array depend on the characteristics of the application(s) which read from and write data to the array, it is impossible to give a general model for the arrival of I/O requests. However, for many applications, the stream of I/O requests can be approximated by a Poisson process. In [4, 5, 6], they also assume that the arrival of I/O requests is Poisson but further assume that arrivals to each disk are also Poisson and independent with a uniform rate. In this paper, we assume that the arrival of I/O requests is Poisson but compute the arrival process of requests to individual disks in section 6.

The second component of the system workload is the size of an I/O request. For many applications, request sizes can be classified as either *supercomputer-based*, where requests are large and infrequent, or *transaction-based*, where small amounts of data are frequently accessed. For this work, it is assumed that requests are transaction-based, where the number of data stripe units requested is less than or equal to the number of data stripe units in a stripe, $W_s - 1$. A distribution which reflects this type of workload is a quasi-geometric distribution [4, 5]:

$$P\{N = n\} = \begin{cases} \sigma & n = 1, \\ (1-\sigma)\frac{\rho(1-\rho)^{n-1}}{(1-\rho)-(1-\rho)^{W_s-1}} & n = 2, \ldots, W_s - 1 \end{cases},$$

where N is a random variable representing the request size ($1 \leq n \leq W_s - 1$), σ is the probability that one data stripe unit is requested ($0 \leq \sigma \leq 1$), and ρ is the parameter of the geometric distribution ($0 \leq \rho < 1$). Since we assume that the maximum number of data stripe units in an I/O request is $W_s - 1$ and a request for data can overlap stripe boundaries, at *most* two stripes can be accessed during any I/O request.

4 Model Overview

Using a description of a RAID 5 disk array, including data and parity mapping, I/O methods and system workload, we can develop a model to compute the mean response time of a RAID 5 I/O request. In doing so, the following assumptions are made:

1. Although our approach works with any RAID 5 configuration, we assume that the array contains 20 disks with a stripe width of 5 disks and a stripe unit size of 4 KB to obtain numerical results.

2. For many transaction-processing workloads, such as scientific databases, a majority of requests are queries to read data. The ratio of reads to writes for such systems is typically 2 or 3 to 1. In this paper, the probabilities for read and write requests are assumed to be 0.7 and 0.3.

3. Each disk can service only one request at a time. Other requests wait and are serviced in first come-first served (FCFS) order.

4. The arrival of I/O requests to the system is Poisson with rate λ.

5. I/O requests access data throughout the disk array in a uniform pattern. Since an I/O request requires multiple stripe units, this means that the starting stripe unit is assumed to be random and that each disk in the array is equally likely to contain the starting stripe unit in a request.

6. Since the focus of this paper is the performance of the array, we assume that the response time for I/O requests is only affected by the disk subsystem, i.e. memory and data paths are fast enough to have little impact on the response time of a request.

Based on a description of the data and parity placement, I/O methods, workload, and model assumptions, we can construct a model to calculate the response time for a RAID 5 I/O request. Recall that an I/O request generates several disk requests. Each request may wait for several disk requests to complete before service for that request begins. Once the data for a request has been located and transfered to memory, it must wait for the remaining disk requests in the I/O request to complete before the I/O request can complete. In the following sections, we will analyze each of the non-independent delays that contribute to the response time of an I/O request.

To do this, we first develop a model of the time for a disk to service a disk request. Second, based on the Poisson arrival of I/O requests, we derive the arrival process of requests to individual disks and the probability that a disk is accessed by an I/O request, subject to the request size distribution. Third, using the arrival process and service time for a disk, the completion time for a disk request is viewed as the virtual waiting time of a queue with the same arrival and service time distribution. Finally, we compute the response time for an I/O request as the maximum of the dependent completion times of the accessed disks.

5 Disk Model

Since an I/O request is composed of multiple disk requests, a model of the I/O request response time must consider the time for individual disk accesses. Although a disk access involves many complex electrical and mechanical interactions, the seek time, rotational latency, and transfer time dominate the time to service a disk request. At a high level, the time for a disk to service a request can be viewed as the time to locate the correct track and sector where the data is located and the time to transfer the data to memory. To precisely define the time needed for disk service, let S, R, T, and Y be a set of random variables representing seek time, rotational latency, transfer time and disk service time. As stated above, a disk request involves locating the correct track and sector, then transferring the data to memory, or $Y = S + R + T$. Because the track location is independent of sector placement, density of Y, $f_Y(y)$, can be written as a convolution $f_S(s) * f_R(r) * f_T(t)$. Using previous results for the probability densities (pdf) for seek time, rotational latency, and transfer time, we can evaluate the convolution to determine the pdf of the disk service time.

maximum disk rotation time (R_{max})	16.7 ms
number of disk cylinders (C)	1200
total disk storage	500 MB
single cylinder seek time (a, S_{min})	3 ms
average seek time ($\bar{S}$)	11 ms
maximum stroke time (S_{max})	18 ms
sustained transfer rate (T_r)	3 MB/s

Table 1: Assumed Disk Parameters

5.1 Seek Time

In considering a model for seek time, there exists a possibility that the disk arm does not move during a disk access. This is called the *sequential access probability* p_s [4, 5]. This probability changes according to the data access pattern for disk requests and the disk scheduling policy. When requests access data in uniformly and the disk scheduling is first-come, first-served, the disk arm tends to move to any other cylinder with equal probability during a request. Given these assumptions, the pdf of seek distance can be expressed as [4, 5]

$$P\{D = i\} = \begin{cases} p_s & i = 0 \\ (1 - p_s)\left\{\frac{2(C-i)}{C(C-1)}\right\} & i = 1, 2, \ldots, C-1\,, \end{cases}$$

where C is the total number of disk cylinders.

To determine a relationship of seek time to seek distance, Chen and Katz [8] empirically derive a formula for the disk profile, $s \approx a + b\sqrt{i}$, $i > 0$, where s is the seek time, i is the seek distance in cylinders, a is the arm acceleration time, and b is the seek factor of the disk. In terms of the parameters listed in table 1, a is equivalent to the single cylinder seek time. The seek factor b can be expressed as $(-10a + 15\bar{S} - 5S_{max})/3\sqrt{C}$ where a is the single cylinder seek time, $\bar{S}$ is the average seek time, and S_{max} is the maximum seek time [3]. Because seek time is a function of seek distance, the seek time pdf can be written as a transformation of the seek distance pdf

$$f_S(s) = \begin{cases} 0 & s = 0 \\ \frac{(1-p_s)2[C-((s-a)/b)^2]}{C(C-1)} & 0 < s \leq S_{max}\,. \end{cases}$$

5.2 Rotational Latency

Under a variety of workloads and disk scheduling policies, rotational latency is commonly assumed to be uniformly distributed in $[0, R_{max}]$, where R_{max} is the time for a full disk rotation. However, in [9], the authors note that the uniform distribution is an accurate approximation for rotational latency when requests at a disk are independent. Since we assume that the starting data stripe unit of each I/O request is random, the locations of data for successive requests at a disk will tend be scattered across the disk. Thus, we use the uniform distribution to model the rotational latency of a disk.

5.3 Transfer Time

Once the location for the data has been determined, the data is transferred to memory. Because each disk request involves a request for a stripe unit of fixed size, the transfer time is constant. The transfer time is $T = bT_r$, where b is the stripe unit size in bytes and T_r is the sustained transfer rate. Using the parameters in table 1, the transfer time for a 4 KB stripe unit is approximately 1.3 ms.

To determine the probability density for the disk service time, we convolve the each of the densities. Because the transfer time is constant, the disk service density depends on the convolution of the seek time and rotational latency. The transfer time will shift the disk service time, but will not change the shape of the probability density. Since seek time is based on the number of cylinders that the disk arm moves, S is not a continuous random variable. Due to the discrete nature of seek time, the regions of integration for $f_Y(Y)$ depend on a, S_{max}, and R_{max}. For the case where $R_{max} \leq b\sqrt{C-1}$, which corresponds to the parameters in table 1, the density of Y can be written as a set of polynomial equations. This set of equations is similar to those derived in [4], but different behavior is observed at the boundaries of integration as p_s increases.

Figure 2 shows a graph of $f_Y(Y)$ using the parameters listed in table 1. To interpret this graph, we consider how seek time, rotational latency, and transfer time affect the disk service time. Note that seek time and rotational latency are both larger than the transfer time of a disk request. As p_s increases, the seek time becomes smaller and rotational latency dominates the disk service time of the request. If the disk arm does not move, rotational latency is effectively the only component of time for a disk request, and the density of disk service time equals the uniform density of the rotational latency. When p_s=$1/C$, where C is the number of disk cylinders, the probability that the disk arm does not move equals the probability of moving to any other cylinder, making the function continuous. To determine which of the curves in Figure 2 reflects the service time of a disk in our RAID 5 disk array model, we consider the locations of data accessed by successive requests. Since we assume that the starting stripe unit of an I/O request is random and requests for a disk are serviced in first come, first served order, the data locations accessed between successive requests will tend to be scattered across the disk. Therefore, during a request, the disk

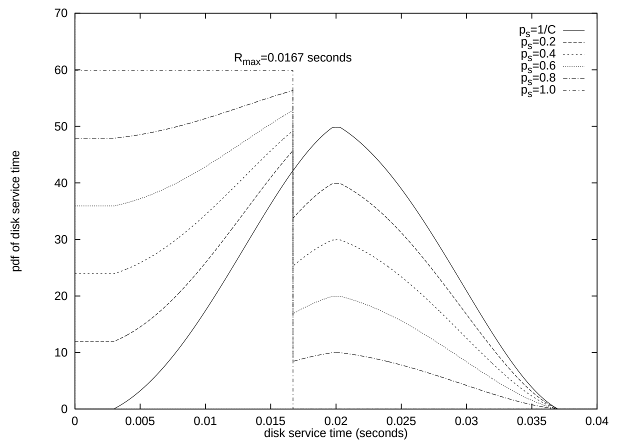

Figure 2: Disk Service Time for Different Values of p_s

arm will tend to move to any other cylinder (including not move) with equal probability. This is equivalent to the case where $p_s = 1/C$. When the sequential access probability p_s is 1/C, the pdf of disk service time can be approximated by an Erlang density of order k and mean μ. Figure 3 shows an optimized fit of the Erlang pdf to the actual pdf using the least squares curve fitting method. By shifting the mean slightly, the peak

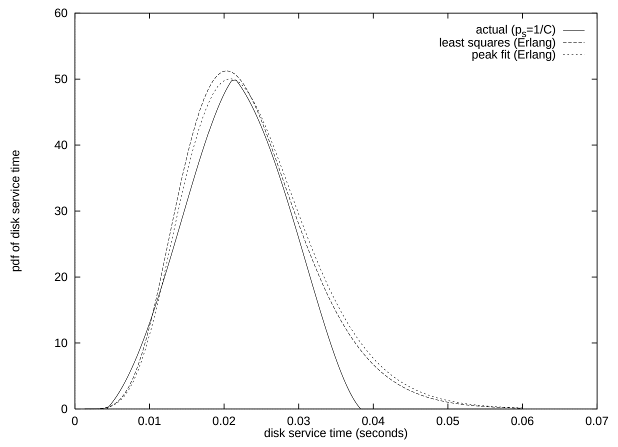

Figure 3: Erlang Approximation for Disk Service Time Density

can be more closely matched (peak fit), while sacrificing the error on the right side of the curve. The peak fit parameters of the Erlang density are order (k) 8 and mean (μ) 42. We will use the peak fit parameters for the Erlang density as an approximation to the disk service time pdf. Using this approximation, we will derive analytical expressions for the mean steady state I/O request response time in section 7.

6 Disk Arrival Process

Since the response time for an I/O request depends on completion times for disk requests, it is important to characterize the arrival process of disk requests. In this section, we will show how requests arrive at individual disks given a Poisson stream of I/O requests in a RAID 5 disk array.

Let $\{N(t)|t \geq 0\}$ be the Poisson stream of I/O requests. Let $\{N_k(t)|t \geq 0\}, 1 \leq k \leq n$, be n output processes, where each output process is a group of k of disks accessed by an I/O request. p_k is the probability that a group of k disks is accessed by an I/O request. Since only one out of a possible n groups of disks may be accessed by each I/O request, selections of groups of disks during successive I/O requests form a sequence of generalized Bernoulli trials.

Multiplying by the probability that there are m requests in $(0,t]$, the probability mass function that m_1 requests access group 1, m_2 requests access group 2,... m_n requests access group n in $(0,t]$ is a multinomial partitioning of I/O requests that results in mutually independent Poisson arrivals at groups of disks. Thus, the rate of arrivals for group 1, group 2,... group n are $p_1\lambda, p_2\lambda, \ldots p_n\lambda$. Because an individual disk is part of different groups of disks that can be accessed during a request, the superposition of these Poisson group requests results in Poisson arrivals at an individual disk. The arrival rate of requests at a disk is $p_j\lambda$, where p_j is the probability that disk j, $1 \leq j \leq N$, is accessed during an I/O request and N is the number of disks in the array. Although the arrival process to each disk is Poisson, it is important to note that these processes are not independent, since the probability that an individual disk is accessed depends upon the group of disks that is accessed. The effect of the dependence between arrivals to individual disks on the I/O request response time will be addressed in section 7.

The probability p_j that disk j is accessed depends on the number of data stripe units requested and whether the request is a read or write. p_j is expressed as

$$p_j = \sum_{n=1}^{W_s-1} P\{N=n\}p_r q_{r_n,j} + \sum_{n=1}^{W_s-1} P\{N=n\}p_w q_{w_n,j}$$

where $P\{N=n\}$ is the probability that an I/O request accesses n data stripe units, p_r and p_w are the probabilities that an I/O request is a read or write, and $q_{r_n,j}$ and $q_{w_n,j}$ are the probabilities that disk j is accessed during a read/write request for n data stripe units. Both $q_{r_n,j}$ and $q_{w_n,j}$ depend on how a disk is accessed during an I/O request. A simple approach to compute p_j would be to enumerate all possible I/O requests that access disk j, then divide by the total number of I/O requests. However, for even small arrays, the number of possible I/O requests becomes large and calculations become complicated. A better approach is to look at how I/O requests access stripes of data.

Because disks are organized into rows and columns, any disk can be referenced by the row r and the disk k within that row. Recall that at most two stripes can be accessed by an I/O request. We can separate the probability that disk j is accessed during an I/O request into the probabilities that row r is accessed, q_r, the probability that n data stripe units are requested, $P\{N=n\}$, and the probability that disk k of row r is accessed during a request for n data stripe units, q_{k_n},

$$p_{k,r} = q_r \left\{ \sum_{i=1}^{W_s-1} P\{N=n\}q_{k_n} \right\}.$$

To evaluate q_{k_n}, recall that an I/O request can access one or two stripes. By enumerating all possible read and write requests that directly access a stripe and those that overlap from the adjacent stripe, we can determine the requests that access a specific disk of a row. Multiplying by the probability that a specific row is accessed gives the probability that disk j is accessed, p_j,

$$p_j = q_r P\{N=n\} \sum_{n=1}^{W_s-1} \{p_r q_{r_n,k} + p_w q_{w_n,k}\}$$

where $q_{r_n,k}$ is the probability that disk k of row r is accessed during a read I/O request for n data stripe units and $q_{w_n,k}$ is the probability that disk k of row r is accessed during a write I/O request for n data stripe units. Since the probability that a specific row is accessed by an I/O request is assumed to be uniform, the set of q_r probabilities are equal and the access probability of a disk k in a row is equal to the probability for disk k of any row. By considering how I/O requests access stripes of data, we are able to compute disk access probabilities based on a row of disks rather than all disks in the array.

A graph of the disk access probabilities, for a representative row of disks, is shown in Figure 4. The parameters of the request size distribution, σ and ρ of the quasi-geometric distribution in Chapter 2, are placed on the x-axis. For each σ from 0.1 to 0.9 in increments of 0.1, ρ varies from 0.1 to 0.9 in increments of 0.1. To verify the accuracy of the analytical computations, disk access probabilities for all disks in the array are also obtained from a detailed simulation model of I/O requests (Figure 5) with a confidence level of 99% and relative confidence interval width of $\pm 1\%$. As σ and ρ are varied, the graph separates into nine regions where σ is fixed and ρ varies within each region. As σ increases, the probability that one data stripe unit is accessed by an I/O request increases. Since a disk is less likely to be accessed during smaller sized requests, the disk access probabilities for each region are higher when σ is lower. Within a region for a given σ, as the parameter of the geometric distribution ρ increases, the probability that one data stripe unit is requested increases and the disk access probabilities decrease. Therefore, as reflected in the graphs,

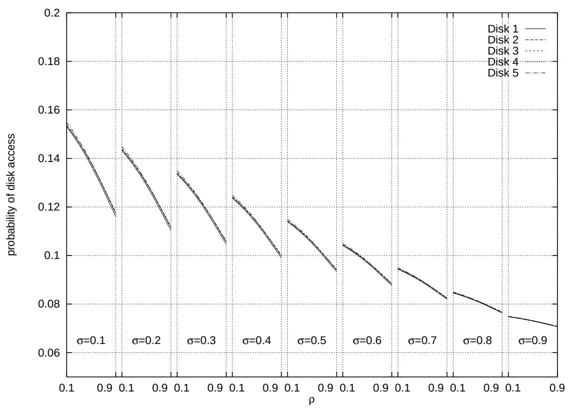

Figure 4: Analytical Disk Access Probabilities

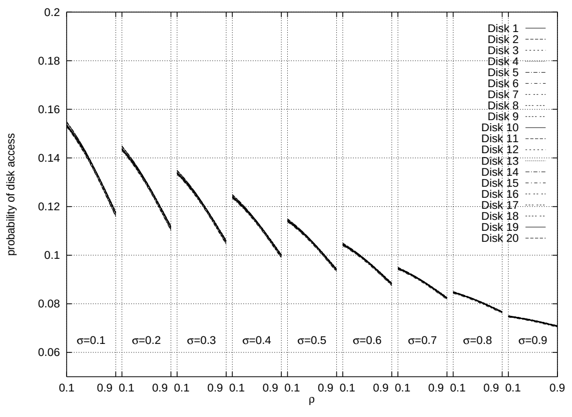

Figure 5: Simulated Disk Access Probabilities

disk access probabilities are highest when $< \sigma, \rho >$ is lowest. An important point is that only at the smallest request sizes are the disk access probabilities close to a uniform value. Disk access probabilities are more than three times larger than a uniform value when the request sizes are largest.

Comparing analytical and simulation results, probabilities computed analytically are extremely close to the the simulated probabilities for all disks. It is clear that since the probability that a particular row is accessed is uniform, access probabilities for the ith $(1 \leq i \leq W_s)$ disk in each row are equal and the access probabilities for each disk in the array are approximately equal. However, as shown in the graph and calculated in [10], there are slight differences in the disk access probabilities for disks of the same row since all combinations of n data stripe units are not possible in an I/O request for n data stripe units.

7 Response Time

7.1 Analytic Computation

Given how requests arrive and are serviced by a disk, we can calculate the response time for an I/O request. Based on the number of disks accessed, the type of I/O request, and completion time for a disk request, we will determine expressions for the mean overall, mean read, and mean write response times of an I/O request.

Depending on the number of data stripe units accessed and whether a request is a read or write, a request for n data stripe units results in l disk requests, where $l \geq n$. For a read request, a request for n data stripe units results in n disk requests. The request completes when all n disk requests are complete. A write of n data stripe units may be a partial stripe write, a full stripe write, or a partial write of two stripes. A partial write of a single stripe requires $n+1$ data and parity stripe units to be read and written in two separate operations. A full stripe write requires W_s requests to write data and parity since the new parity is completely determined from new data. When two stripes are accessed, the operation can be separated into two independent partial stripe operations because stripe units in one stripe do not depend on those in another stripe.

If we represent a disk as a queue, the completion time for a disk request is equivalent to the *virtual waiting time* of a queue. The virtual waiting time is defined as the time for both the requests ahead of and a request itself to be serviced [11]. Since the disk service time can be approximated by an Erlang distribution and arrivals to each disk are Poisson, the steady state distribution of the completion time for a disk request can be viewed as the virtual waiting time distribution of an $M/E_k/1$ queue in steady state ($V_{M/E_k/1}$). Furthermore, since the arrival rates of requests and disk service time for each disk are approximately equal, the virtual waiting times of each disk are also approximately equal. Based on the fact that the virtual waiting times of each disk are approximately equal, the response time for both read and full stripe write requests can be formulated as the maximum of l identical virtual waiting times.

Although the calculation is similar for a partial write of a single stripe, the request requires data and parity to be read then written in two separate operations. Thus, the response for the I/O request is the maximum of the virtual waiting times for the disk requests to read old data and parity and the maximum of the virtual waiting times for disk requests to write the new data and parity. An important consideration is the distribution of virtual waiting times for requests to read old data and parity and the distribution to write new data and parity.

Because requests for a partial stripe write are fed back to write new data and parity, the combined arrival process of I/O requests to the array of new and feedback arrivals is not Poisson. However, we conjecture that since the percentage of partial write requests is small compared to read and full stripe write requests and that disk requests are fed back once after synchronization, the overall arrival process will remain approximately Poisson. This differs from analyses of single server queues in which jobs are fed back with a fixed probability. When jobs are fed back multiple times for the same arriving task, the percentage of customers that are fed back over steady state is large enough to destroy the Poisson process. However, for RAID 5 partial write, disk requests are fed back once after synchronization. Furthermore, this synchronization step tends to smooth out the feedback arrival process and preserve the overall Poisson arrival process of requests to groups and individual disks in the array.

Assuming that the arrival process remains Poisson, the mean time to write the new data and parity will be the same as to read the old data and parity because of the *PASTA (Poisson Arrivals See Time Averages)* property of Poisson arrivals. Therefore, the mean response time for a partial stripe write request is 2×(maximum l $V_{M/E_k/1}$). By comparing analytical and simulated response times for write I/O requests in section 8, we will investigate the assumption that the feedback of requests caused by a partial write I/O request does not markedly change the overall Poisson arrival process to disks in the array.

To determine the distribution of the virtual waiting time for the steady state $M/E_k/1$ queue, we refer to [11]. If W is a random variable denoting the virtual waiting time of the $M/E_k/1$ queue in steady state, where k is the order and μ is the mean of the Erlang distribution and λ is the disk request rate, distribution of W is written as a series expansion

$$F_W(t) = 1 - e^{-k\mu t} \sum_{i=0}^{\nu} \frac{\beta_i [k\mu t]^i}{i!}$$

where the coefficients β_i have the form

$$\beta_i = \begin{cases} 1 & \text{if } i = 0, 1, \ldots k-1 \\ \frac{\lambda}{k\mu} \sum_{j=1}^{r} \beta_{i-j} & \text{if } i = r, r+1, \ldots \end{cases}$$

and ν is chosen to be large relative to k. Since $k = 8$ as described in the Erlang approximation in section 2, ν was chosen as 60, sufficiently accurate for all numerical calculations [11].

As stated above, calculation of the response time of an I/O request is based on the maximum of the virtual waiting times of the l disks accessed during a request for n data stripe units. However, for the general case, computation of the maximum of l non-independent random variables becomes difficult if the distribution is complicated or l is large. As an approximation, we use the method in [12] to compute an upper bound for the expected value of the maximum of l identical random variables. Using the expression for the virtual waiting time for the steady state $M/E_k/1$ queue, the expected maximum of l $M/E_k/1$ virtual waiting times is

$$M_l = \frac{\nu+1}{k\mu} \left\{ 1 + \frac{[\nu+1] k\mu m_l \beta_{\nu+1}/(\nu+1)!}{\sum_{i=o}^{\nu} (k\mu m_l)^i \beta_i/(i)!} \right\}$$

where m_l is such that

$$l e^{-k\mu m_l} \sum_{i=0}^{\nu} \frac{\beta_i [k\mu m_l]^i}{i!} = 1$$

Based on the upper bound for the mean time for l disk requests to complete, the mean overall, mean read, and mean write response times for an I/O request can be written as

$$\begin{aligned} \overline{t_r} &= \textstyle\sum_{l=1}^{W_s-1} P_r\{A=l\} M_l \\ \overline{t_w} &= \textstyle\sum_{l=2}^{W_s+1} P_w\{A=l\} 2 \times M_l \\ \overline{t} &= p_r \overline{t_r} + p_w \overline{t_w} \end{aligned}$$

where $P_r\{A=l\}$ and $P_w\{A=l\}$ are the probabilities that l stripe units are accessed during a read request and write request, and M_l is an upper bound for the expected maximum of the virtual waiting times for l disk requests.

Note, that since parity must be updated during a write request, the minimum number of stripe units that can be accessed is a data stripe unit and the corresponding parity stripe unit, or two stripe units. Since $W_s - 1$ is the maximum number of data stripe units in a request and at most two stripes can be accessed given the transaction-processing workload based on the quasi-geometric distribution, two parity stripe units must also be updated in addition to the W_s data stripe units, making the maximum number of stripe units accessed in a write request, $W_s + 1$.

7.2 Model Validation

In order to compare and validate analytical results, a detailed event-driven simulator was constructed. The simulator is designed to mean the average steady state throughput and response time of a RAID 5 I/O request. We will describe how the simulator models I/O requests and point out key differences from the analytical model.

Although an I/O request is generated according to the same workload as in the analytical model, the simulator and analytical model differ in how requests are assigned to a disk and how disks service a request. Once the number of data stripe units in the request has been determined from the quasi-geometric distribution, the starting data stripe unit for the request is randomly chosen from all possible data stripe units. For the number of data stripe units in the request, requests are assigned to successive disks. If the request is a read, disks where a parity stripe unit is located are skipped; requests to read the old parity are assigned if the request is a write. This differs from the probabilistic assignment of requests used in the analytical model.

Each disk services requests in first-come, first-served order and maintains the current disk position (track, sector). When a new request is received, the new sector and track are computed and the seek time and rotational latency to position the read/write head at this new position is calculated. Then the time to transfer the stripe unit to memory is added to the completion time. When the disk request completion time becomes the minimum next event time, the request is removed from the disk queue and the next disk request is serviced; otherwise the disk remains idle. By explicitly maintaining disk position, we check the Erlang disk service time approximation used in the analytical model.

In the same manner, each disk request of an I/O request is serviced. Once all requests have completed, the time at which the last disk request finished is recorded. The time from when the I/O request was generated to this time is recorded. If the request is a read, this is the response time of the I/O request. If the request is a write, the disk requests are reassigned to the same disks and the time until all disk requests complete again is the response time for a write I/O request. When a sufficient number of I/O requests have been generated, the mean overall I/O request response time is calculated by summing the completion times of all the I/O requests and dividing by the total number of I/O requests. In a similar manner, the mean read and write response times are the sum of the completion times of the read and write requests divided by the number of read and write requests.

The *batch means* method [13] was used to statistically determine whether a sufficient number of I/O requests had completed for the steady state response time to lie within a specified confidence interval and relative confidence interval width. Using an initial batch of 1000 I/O requests to complete the transient period and a batch size of 1000, the simulation was sequentially lengthened by increasing the number of batches until a confidence level of 99% and relative confidence interval width of 1% was reached. Since we are careful to achieve a high degree of accuracy with the simulator, differences between the analytical results and simulation results are due to the probabilistic assumptions of the analytical model rather than statistical inaccuracies in the simulation.

8 Results

Graphs of the analytical and simulated values of the steady state mean overall I/O request response time are shown in Figures 6 and 7; graphs for the steady state mean read and write I/O request response times are given in the appendix. In addition, graphs of the percent difference are shown for each set of graphs of the mean overall, mean read, and mean write I/O request response times. For each graph, the x-axis represents the parameters of the request size distribution, σ and ρ. For each σ from 0.1 to 0.9, ρ varies from 0.1

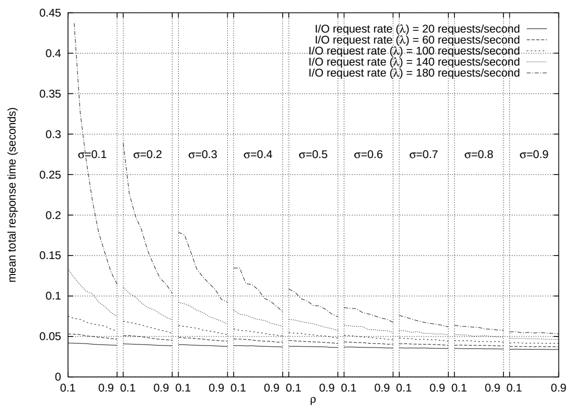

Figure 6: Mean Overall Response Time - Analytical

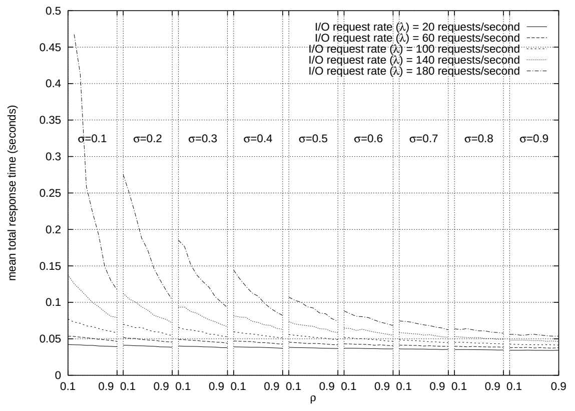

Figure 7: Mean Overall Response Time - Simulation

to 0.9. The y-axis represents the mean steady state response time for an I/O request. The separate curves on each graph represent the I/O request rate, λ, ranging from 20 requests/second to 180 requests/second.

It can be seen that as the rate of I/O requests increases, response time increases and jumps caused by the disk access probabilities illustrated in Figures 4-5 become more pronounced. As σ and ρ increase, the number of stripe units for an I/O request decreases, causing the mean response time of the I/O request to decrease. For the range of I/O request rates and sizes, the percent difference between the analytical and simulation response times is less than 10%, where the percent difference is highest when the request rate is highest and request size is largest. In addition, we point out that the analytical values are slightly larger than the simulation results, since their computation is based on an upper bound for the expected maximum.

An important point is the relationship between the read and write I/O request times. Since a write request requires two phases to read then write data and parity, we would expect that the write response time would be approximately two times greater than the read response time for an I/O request of n data stripe

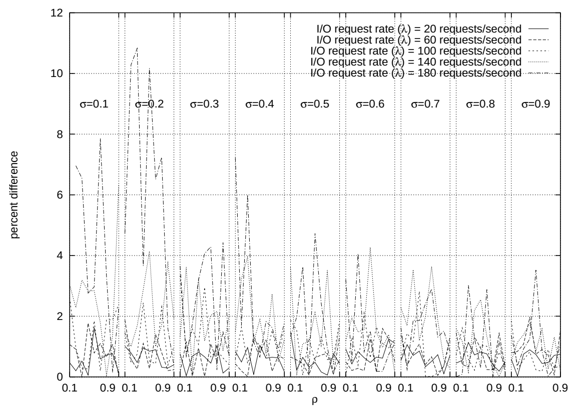

Figure 8: Mean Overall Response Time - Percent Difference

units. Although this is true at low I/O request rates, the write response time is approximately three times greater at the highest request rates and sizes [10]. Due to additional queuing at a disk at higher request rates, additional disk accesses for parity update(s) cause the response time for an I/O request to increase dramatically. This illustrates why it is important to consider the actual I/O request rate and disk access probabilities for read and write requests rather than assuming a uniform distribution of arrivals, as was done in previous studies [4, 5, 6].

9 Conclusions

This paper has presented an accurate analytical model to compute the mean response time of an I/O request during steady state. By analyzing and deriving the components needed for an I/O request to access data, a higher degree of accuracy when compared to measurement and detailed simulation than previous studies was obtained.

First, given previous work for different components of a disk access, the probability density of the time to locate and transfer data during a disk request was derived. Second, given that I/O requests are Poisson, it was determined that requests to groups and individual disks were also Poisson with a rate equal to the probability that the group or disk is accessed during a request times the I/O request rate, where disk access probabilities were calculated from the behavior of a representative row of disks. Third, since an I/O request resulted in multiple disk requests, it was observed that response time for l disk requests can be expressed as the maximum of l virtual waiting times of a queue. Since requests were shown to be Poisson and disk service time distribution was approximated by an Erlang density, an upper bound for the expected maximum of l virtual waiting times of the steady state $M/E_k/1$ queue was determined and used to calculate the completion time of l dependent disk requests.

Based on this approach, we can analyze other aspects of RAID 5 performance. Although we do not discuss degraded and rebuild modes in this paper, RAID 5 can tolerate single disk failures in each row. An important consideration of RAID 5 design is how the response time for an I/O request is affected when one or more disks have failed and are undergoing reconstruction. We are currently extending this work to investigate different ways to balance response time and total reconstruction time for the RAID 5 disk array.

References

[1] D. Patterson, G. Gibson, and R. Katz, "A Case for Redundant Arrays of Inexpensive Disks (RAID)," in *Proceedings of the 1988 ACM SIGMOD Conference on Management of Data*, (Chicago, IL), pp. 109–116, June 1988.

[2] D. Patterson, G. G. P. Chen, and R. Katz, "Introduction to Redundant Arrays of Inexpensive Disks (RAID)," in *Proceedings of the Spring 1989 COMPCON*, (San Francisco, CA), pp. 112–117, April 1989.

[3] E. Lee and R. Katz, "An Analytic Performance Model of Disk Arrays," in *Proceedings of the 1993 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, (Santa Clara, CA), pp. 98–109, May 1993.

[4] S. Chen and D. Towsley, "The Design and Evaluation of RAID 5 and Parity Striping Disk Array Architectures," *Journal of Parallel and Distributed Computing*, vol. 17, January 1993.

[5] S. Chen and D. Towsley, "A Performance Evaluation of RAID Architectures," UM-CS-1992 067, University of Massachusetts, Amherst, MA, September 1992.

[6] A. Thomasian and J. Menon, "Performance analysis of RAID 5 disk arrays with a vacationing server model for rebuild mode operation," in *Proceedings of the IEEE International Conference on Data Engineering*, (Houston), pp. 111–119, February 1994.

[7] A. Mourad, W. Fuchs, and D. Saab, "Performance of Redundant Disk Array Organizations in Transaction Processing Environments," in *Proceedings of the 1993 International Conference on Parallel Processing*, (Boca Raton, FL), August 1993.

[8] P. Chen, E. Lee, G. Gibson, R. Katz, and D. Patterson, "RAID: High-Performance, Reliable Secondary Storage," *ACM Computing Surveys*, vol. 26, June 1994.

[9] C. Ruemmler and J. Wilkes, "An Introduction to Disk Drive Modeling," *IEEE Computer*, vol. 27, March 1985.

[10] A. Kuratti, “Analytical Evaluation of the RAID 5 Disk Array,” Master’s thesis, University of Arizona, Tucson, AZ, August 1994.

[11] C. Kim and A. Agrawala, “Virtual Waiting Time of an Erlangian Single Server Queuing System,” UMIACS-TR-1986 6, University of Maryland, College Park, MA, December 1985.

[12] A. Gravey, “A Simple Construction of an Upper Bound for the Mean of the Maximum of N Identically Distributed Random variables,” *Journal of Applied Probability*, December 1985.

[13] A. Law and W. Kelton, *Simulation Modeling and Analysis.* New York: McGraw Hill, second ed., 1991.