# RAID 5 vs. Parity Striping: Their Design and Evaluation

Shenze Chen and Don Towsley

**COINS Technical Report 92-13**
February 1992

# RAID 5 vs. Parity Striping: Their Design and Evaluation *

*Shenze Chen* *Don Towsley*

*Department of Computer and Information Science*
*University of Massachusetts*
*Amherst, MA 01003*

**Abstract**

In this paper, we examine the design issues and performances of two disk array architectures, *RAID 5* and *Parity Striping*, for small I/O applications such as transaction processing systems and UNIX environments.

Our work differs from previous work by explicitly modeling the *write synchronization procedure*, which has to be followed by both of these architectures. Specifically, we propose two synchronized I/O scheduling policies, which provide practical solutions to the problem of providing write synchronization. A priority queueing model is developed for describing the behavior of RAID 5 and Parity Striping coupled with these two policies. The performance (mean I/O response times and maximum I/O throughput) of these two architectures is examined under a wide variety of workloads, where each request may access multiple data blocks. Arbitrary distributions for the request size and skewed access patterns are considered. The results show that the performance of RAID 5 is sensitive to the mean request size but not to the skew in the access pattern, whereas the Parity Striping architecture is sensitive to the skew in the access pattern. We identify workloads for which RAID 5 and Parity Striping each provide the best performance. This allows the designer the ability to choose between the RAID 5 and Parity Striping architectures for their applications.

Last, simulation results are provided to substantiate our analyses.

*This work is supported, in part, by the Office of Naval Research under contract N00014-87-K-796, by NSF under contract IRI-8908693, and by an NSF equipment grant CERDCR 8500332.

# 1 Introduction

In many computing systems, the disk I/O subsystem is often identified as the major bottleneck to system performance. Because of the rapid increase in CPU speed and slow increase in disk speed, this speed gap is expected to increase in the near future. Currently, disks are at least four orders of magnitude slower than main memory [22]. Therefore, the problem of how to build high performance disk I/O systems based on current or foreseeable future disk technologies has become a hot topic among researchers.

One attractive idea is the so called *disk array*, where the disk I/O subsystem consists of multiple disks with data spread over these disks. The ideas of *disk interleaving* and *disk striping* were first introduced by Kim [11] and Salem et al [30], respectively. Since then a great deal of work has focused on various design issues related to the performance of disk arrays [22, 23, 12, 15, 28, 1, 33, 4] and to their reliability [32, 5, 20]. Disk array architectures fall into one of five different classes proposed in [26, 27, 9] referred to as *Redundant Arrays of Inexpensive Disks (RAID)*. Among the five, the two most promising candidates for high performance computing systems appear to be the *mirrored disk array (RAID 1)* and the *rotated parity array (RAID 5)*. In [8], Gray further suggested a Parity Striping architecture and claimed it to be better than RAID 5 for applications with predominantly small I/O's, such as transaction processing systems. A nice survey on disk array architectures and issues related to disk array reliability can be found in [7].

The most significant advantage of RAID 5 and Parity Striping over RAID 1 is the lower cost, since a mirrored array requires nearly double the number of disks in order to provide the same capacity. Previous work on the analysis and experimental study of RAID 5 can be found in [3, 18, 2], and of Parity Striping in [8]. These studies ignored the effects of *write synchronization*, which is required by both RAID 5 and Parity Striping, when obtaining the performance results. In addition, in the analysis of [18, 2], all requests were restricted to access only one block of data. The analysis of [8] assumed "zero-load", i.e., each request sees an idle disk system and can start service immediately without queueing delay. Also, only a non-skewed access pattern was assumed in [8]. The mirrored array architecture (RAID 1) and its variations have been studied in the literature [2, 3, 19].

In this paper we focus on the analysis of RAID 5 and Parity Striping for the above mentioned small I/O environments (an analytic model of RAID 5 for large I/O applications can be found in [2]). Our work differs from previous work by explicitly modeling *write synchronization*. Specifically, we first propose two synchronized I/O scheduling policies suitable for both RAID 5 and Parity Striping, which take care of this problem. These two policies provide practical solutions to the problem of solving write synchronization in RAID 5 and Parity Striping. We provide accurate mathematical models for estimating the mean I/O response times and the maximum throughput of the RAID 5 and Parity Striping architectures coupled with these two synchronized scheduling policies. Using these models, we compare the performance of RAID 5 and Parity Striping with each other. The results show that the performance of RAID 5 is sensitive to the increase of mean request size but not to the skew in the access pattern. On the other hand, the Parity Striping architecture is sensitive to skew in the access pattern. Therefore, depending on applications, RAID 5 outperforms Parity Striping in some cases but is outperformed by Parity Striping in other cases. We identify workloads for which each architecture provides the best performance. This allows the designer the ability to

choose between the RAID 5 and Parity Striping architectures for their applications.

Finally, the queueing model used here itself is also of interest. While previous analyses on disk arrays all assumed FIFO queueing discipline, a priority queueing model is used in our analysis.

Other related work on performance evaluation of disk arrays can be found in [16, 29, 25]. However, the disk arrays examined in these work do not include redundant information in order to provide reliable I/O subsystems.

I/O performance can also be improved by introducing a disk cache [31, 21]. The effectiveness of this cache depends on the I/O access pattern as well as the cache size. For applications where disk accesses show a high locality, disk caching may satisfy most of the read requests and therefore reduce the I/O traffic to the disk. For transaction processing, however, disk caching may be less effective, because I/O requests randomly access the disks and it is impractical to cache the entire database. In any case, disk caches can be combined with disk arrays and the results of our study remain valid for such systems as well.

The remainder of this paper is organized as follows. Section 2 describes the RAID 5 and Parity Striping architectures, as well as the *write synchronization problem* inherent in these two architectures. Section 3 introduces the two new synchronized I/O scheduling policies. The analyses of RAID 5 and Parity Striping for single block accesses are given in Section 4. Section 5 extends the analyses of RAID 5 and Parity Striping to multiple block accesses. The performance of these two architectures is compared with each other in Section 6. Last, Section 7 summarizes this paper.

# 2 Background Knowledge on Disk Arrays

## 2.1 The RAID 5 Architecture

The RAID 5 array consists of $N$ identical disks, where data is block interleaved across the $N$ disks. The set of blocks on each disk with the same location constitutes a strip. For reliability purposes, each strip contains a parity block, which is the XOR of all of the other data blocks in the same strip. The strip width, $W_s$, defined as the number of disks which can concurrently contribute to serve a request, is $N-1$. If any one disk fails, the array is still operational, since the failed data can be restored by XORing data from all of the other disks. The price paid for the reliability is that each write operation is required to update the desired data block(s) as well as the parity block(s). This creates additional workload to the array. A new parity block is calculated by

$$new\_parity = new\_data \oplus old\_data \oplus old\_parity. \quad (1)$$

where $X \oplus Y$ corresponds to the XOR of $X$ and $Y$. Thus, prior to updating the new data and parity, the old data and parity have to be read out first in order to calculate the new parity. This is referred to as the additional "read-update procedure" for a write request. In order to avoid the operation of updating the parity blocks from becoming a bottleneck, they are rotated among the $N$ disks in RAID 5.

The RAID 5 architecture is attractive because only 1 block of redundant information is required for every $N-1$ data blocks in order to achieve reliable service. This is contrasted with RAID 1 (mirrored array) which doubles the number of disks. Another advantage of RAID 5 is its inherent load balancing property. Since all data files are spread over multiple disks, each disk basically faces the same workload.

### 2.2 The Parity Striping Architecture

Gray *et al.* [8] argued that RAID 5 may suffer a performance loss in small I/O environments, such as encountered in transaction processing systems, because multiple seeks and rotations are needed to serve a multiple block request. Instead, they recommend the *Parity Striping* architecture, which in fact is a variation of RAID 5. Parity Striping only strips parity blocks across the disks, but does not strip data. In addition, parity blocks occupy a contiguous area on each disk, called the parity area. By using this architecture, a read request can be satisfied by one disk. For most transaction systems, since a request size is typically small (less than 10K bytes) compared to the parity area on each disk (more than $10^5$K bytes [1] ), Gray *et al.* [8] claimed that 99.9% of the writes can be satisfied by accessing a data disk and a disk containing its parity. While uniform access is assumed to each disk in [8], we find that the Parity Striping architecture may suffer when the access pattern is nonuniform (skewed).

### 2.3 The Write Synchronization Problem

The *write synchronization problem* exists for both RAID 5 and Parity Striping. That is, the parity update cannot proceed before the corresponding data block(s) has(have) been read out, since the calculation of a new parity block is based on the old data information (see equation (1)). We will address this problem in the next section.

## 3 Two Synchronized I/O Scheduling Policies

In this section, we propose two I/O scheduling policies for RAID 5 and Parity Striping, which synchronize parity update operations for write requests. In all cases, a write request completes only when both the data and the parity have been updated.

- **The Before Service (BS) Policy**

  With this policy, two queues are maintained for each disk in the array, one for arriving read and write requests (*RW queue*) and the other for parity update requests (*P queue*). When a read or write request arrives at the disk subsystem, the dispatcher sends it to the RW queue of the target disk. When it reaches the head of the queue and is scheduled for service, if it is a write, it generates a parity update request to the P queue of the corresponding disk containing its parity block(s). Since the parity request is generated prior to the beginning of service, we call it the *before service* policy. The requests in the P queue are given higher priority than requests in the RW queue to ensure that an outstanding parity request begins service as soon as possible since its corresponding data update operation has already started. The disk operation is assumed to be non-preemptive; therefore a new parity request cannot commence service until the current I/O finishes.

  Under the BS policy, if the disk containing the parity block(s) is idle, the parity update request can start service immediately. To serve a write request, both the data and parity update requests will experience the following 5 service stages: seek, rotation, read (for the

[1] Note: the parity area on each disk depends on the capacity of each disk and the total number of disks in the array.

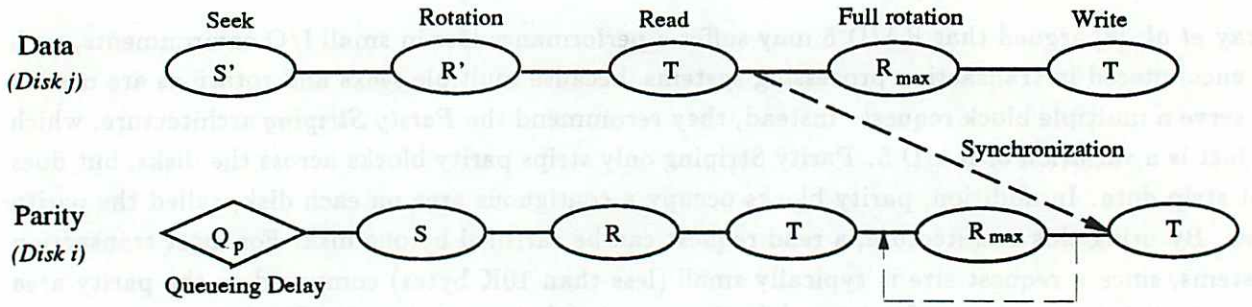



Figure 1: Scenario for Serving a Write Request

calculation of the new parity), full rotation, and write. Hence, if the parity update request finishes the 4-th stage prior to the completion of the 3-rd stage by the data update request, then one or more additional full rotations are necessary. Figure 1 shows the scenario for serving a write request. However, the probability that the sum of the seek ($S'$) and rotational delay ($R'$) on the data disk exceeds the sum of queueing delay in the P queue ($Q_p$), seek ($S$), rotation ($R$), and a full rotation ($R_{max}$) on the parity disk is very small (the transfer times ($T$) on both disks are the same) as we shall observe later from the analysis.

- **The After Service (AS) Policy**

  The AS policy is the same as the BS policy, except that the parity request is generated after the data required to calculate the new parity has been read out. This guarantees that the data required to calculate the new parity is available whenever needed, and therefore case (c) in Figure 1 can never occur.

*Remark:* The performance of the AS policy is expected to be worse than the BS policy since it delays the generation of parity update requests. However, it may be useful in a network environment or a distributed file system where the disk containing the parity block is located on another site, since it may reduce the network traffic and simplify the file system.

# 4 Analytic Models for Single Block Accesses

In this section, we assume that each I/O request accesses one block of data. Multiple block access models are discussed in the next section.

## 4.1 The Disk Model and Disk Service Time Distribution

We consider a disk array containing $N$ identical disks. In our model, we assume that each disk in the array has its own data path so that the channel is not a bottleneck, and therefore the channel delay is ignored. For the disk seek pattern, since it is observed that in reality, most of the time the disk arm doesn't move [17, 12], we introduce a *sequential access probability*, $p_s$, which is defined as the probability that the seek distance is equal to zero. We identify two kinds of access patterns, the sequential access pattern and non-sequential access pattern. Under the *sequential access* pattern,

the disk arm does not move, whereas under the *non-sequential access* pattern, the arm has to move to serve a request. In the case of a *non-sequential* access, the arm is assumed to move to any other cylinder with equal probability.

Define,

$N$: the number of disks in a array;

$W_s$: the strip width of an array ($W_s = N - 1$);

$C$: the number of cylinders for each disk;

$N_b$: the number of blocks in a track;

$\alpha$: the transfer time for a single block;

$S_{max}$: the maximum seek time;

$R_{max}$: the maximum rotation time (16.7 ms);

$D$: a *r.v.* for the disk seek distance;

$p_s$: the probability that the seek distance is equal to zero, $p_s = P\{D = 0\}$;

$S$: a *r.v.* for disk seek time;

$R$: a *r.v.* for rotational latency;

$X$: a *r.v.* for the sum of seek and rotation time, $X = S + R$;

$B$: a *r.v.* for the number of blocks in a request;

$T$: the data transfer time, $T = \alpha B$;

$Y_r, Y_w$: *r.v.*s for disk service times of read and write requests, respectively.

In [2], we derived the following expression for the distribution of seek distance $D$,

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

We also assumed the following expression for the seek time, $S$, as a function of the seek distance $D$,

$$S = \begin{cases} 0, & D = 0, \\ a + b\sqrt{D}, & D > 0, \end{cases}$$

where $a$ is the arm acceleration time (3ms), and $b$ is the seek factor (0.5) [25].

For ease of analysis, we approximate the seek distance $D$ as a continuous *r.v.* with probability density function (*pdf*)

$$f_D(x) = \begin{cases} p_s u_0(x), & x = 0, \\ (1 - p_s)\frac{2(C-x)}{C^2}, & 0 < x \leq C, \end{cases}$$

where $u_0(x)$ is the unit impulse function defined by (see [13] pp. 342)

$$u_0(x) = \begin{cases} \infty, & x = 0, \\ 0, & x \neq 0, \end{cases}$$

$$\int_{-\infty}^{\infty} u_0(x)\, dx = 1.$$

The cumulative distribution function $(CDF)$ of $D$, $F_D(x) = P\{D < x\}$, is

$$F_D(x) = \begin{cases} p_s, & x = 0, \\ (1 - p_s)\left(\frac{2x}{C} - \frac{x^2}{C^2}\right), & 0 < x \leq C, \\ 1, & C < x. \end{cases}$$

Therefore, the seek time $S$ has the following $CDF$,

$$F_S(x) = \begin{cases} F_D(0), & 0 \leq x \leq a, \\ F_D\left((\frac{x-a}{b})^2\right), & a < x \leq S_{max}, \\ 1, & S_{max} < x, \end{cases} \tag{2}$$

where the maximum seek time $S_{max} = a + b\sqrt{C}$.

The rotation time is assumed to be uniformly distributed in $[0, R_{max}]$ with *pdf*

$$f_R(x) = \frac{1}{R_{max}}, \qquad 0 \leq x \leq R_{max}. \tag{3}$$

Let $X = S + R$ be the sum of seek and rotation time, then the *pdf* of $X$ is

$$f_X(x) = \int_0^{S_{max}} f_R(x - s)\, dF_S(s).$$

Specifically, since the seek time $S$ is not a pure continuous $r.v.$, $f_X(x)$ has one of the following forms, depending on the disk parameters,
case 1: $R_{max} \leq b\sqrt{C}$,

$$f_X(x) = \begin{cases} f_R(x)p_s + \int_{a^+}^{x} f_R(x - s)\, dF_S(x), & 0 \leq x \leq R_{max}, \\ \int_{a^+}^{x} f_R(x - s)\, dF_S(s), & R_{max} < x \leq R_{max} + a, \\ \int_{x-R_{max}}^{x} f_R(x - s)\, dF_S(s), & R_{max} + a < x \leq S_{max}, \\ \int_{x-R_{max}}^{S_{max}} f_R(x - s)\, dF_S(s), & S_{max} < x \leq S_{max} + R_{max}; \end{cases} \tag{4}$$

case 2: $R_{max} > b\sqrt{C}$,

$$f_X(x) = \begin{cases} f_R(x)p_s + \int_{a^+}^{x} f_R(x - s)\, dF_S(s), & 0 \leq x \leq R_{max}, \\ \int_{a^+}^{x} f_R(x - s)\, dF_S(s), & R_{max} < x \leq R_{max} + a, \\ \int_{x-R_{max}}^{S_{max}} f_R(x - s)\, dF_S(s), & R_{max} + a < x \leq S_{max} + R_{max}. \end{cases}$$

For our particular disk parameter settings ($R_{max} = 16.7, b = 0.5$,and $C = 1200$), $f_X(x)$ is obtained from case 1, where each of the integrals is a $4th$-order polynomial. Other disk parameters include: transfer rate= 3Mb/sec, block size = 4096 bytes. Thus, the time required to transfer a single block is $\alpha = 1.3ms$.

Last, the disk service time for a read request is

$$Y_r = X + T \tag{5}$$

and for a write request

$$Y_w = X + T + R_{max} + T \tag{6}$$

where $T$ is the transfer time. In this section, since we assumed single block access, the $T = \alpha$, the single block transfer time. In the next section, we will extend our discussions to multiple block access by assuming the number of blocks in a request, $B$, to be a *r.v.* with an arbitrary distribution. In that case, $T = \alpha B$.

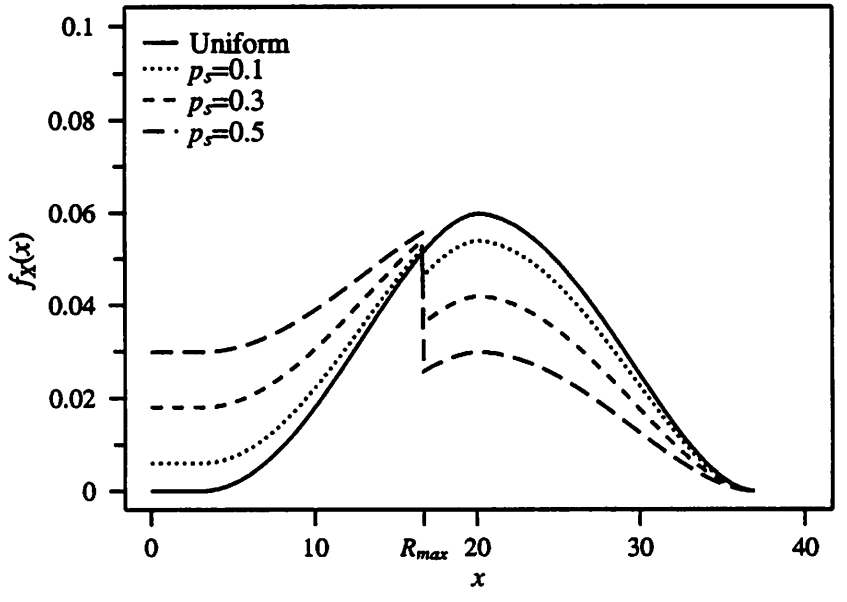



Figure 2: Probability Density Function for the Sum of Disk Seek and Rotational Latency

It is interesting to observe that, according to some real disk traces, in [12], Kim and Tantawi suggested that the distribution of $X$, the sum of seek and rotation time, can be approximated by a normal distribution. According to our derivation, the curve for the *pdf* of $X$ matches the bell shape of a normal distribution only when $p_s = 1/C$ (Figure 2), i.e., the arm moves to any cylinder with equal probability, including the current cylinder. As $p_s$ increases, there is a discontinuity at $R_{max}$, which becomes more pronounced as $p_s$ increases. This is because when $X$ is greater than $R_{max}$, the possibility for a zero seek time disappears (notice the difference between the first and second formula in equation (4)). In fact, similar behavior can also be observed from the histograms presented in [12].

## 4.2 A Queueing Analysis of RAID 5

In our model, we assume that I/O requests arrive at the disk I/O subsystem according to a Poisson process with parameter $\lambda$. As shown in Figure 3, a dispatcher sends each request to its target disk. Since RAID 5 spreads data across multiple disks, it is reasonable to assume that requests

go to each disk with the same probability, i.e., $p_i = 1/N$, $i = 1, 2, \cdots, N$. Therefore the arrivals to the RW queue of disk $i$ is described by a Poisson process with parameter $\lambda_{rw} = \lambda/N$, where a request is a read with probability $p_r$ and a write with probability $p_w$. The arrivals at the P queue of disk $i$ is a superposition of $N-1$ parity request generating processes which direct requests to disk $i$ with probability $1/(N-1)$. When $N$ is large, it can be approximated by a Poisson process with parameter $\lambda_p = p_w\lambda/N$ [10]. Strictly speaking, the $N-1$ parity generating processes are not independent of each other. However, when $N$ is large, the correlations between these processes become small, and therefore we conjecture that they become independent in the limit as $N \to \infty$. In fact, when $N = 16$, our simulation results yield a close match to those of analysis. Last, the FIFO discipline is used within both the RW and P queues.

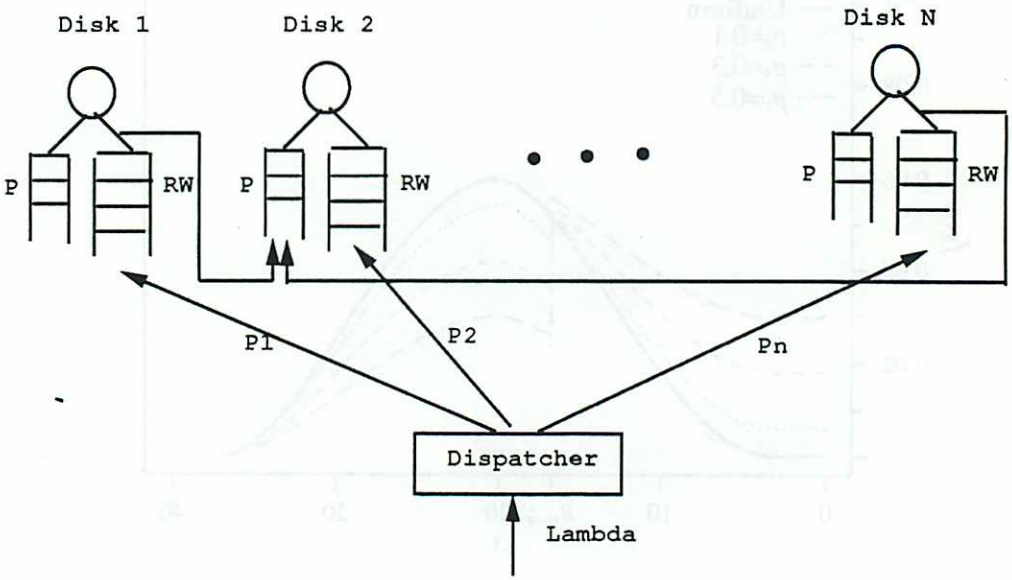



Figure 3: A Queueing Model for the PS Policy

Thus, disk $i$ ($i = 1, \cdots, N-1$) can be modeled as a two priority class non-preemptive priority queue, where the service time for low priority customers (read and write requests in the RW queue) is obtained by equations (5) and (6). The service time of a high priority customer (parity update request in the P queue) depends on the service time of its corresponding data update request on disk $j$ and the time that it waits in the P queue. Specifically, let $X$ and $X'$ be the sum of seek and rotation times on disks $i$ and $j$ (which have the same distribution), and $Q_p$ be the queueing time of a parity update request in queue P of disk $i$, then the service time for this parity update request [2] is

$$Y_p = \begin{cases} X + 2\alpha + R_{max}, & X' < Q_p + X + R_{max}, \\ X + 2\alpha + 2R_{max}, & X' \geq Q_p + X + R_{max}. \end{cases} \quad (7)$$

As shown in Figure 1, the first term on the right hand side corresponds to the case where the old

[2]Note: since the probability $P\{X' \geq Q_p + X + 2Rmax\}$ is extremely small (less than 0.00015), we ignore the cases where two or more additional full rotations are required in serving a parity update request in order to achieve the write synchronization.

data required to calculate the new parity block has become available prior to disk $i$ becoming ready to update the parity block. The second term corresponds to the opposite situation, i.e., the old data has not been read out by disk $j$ by the time disk $i$ completes a seek, a rotation, a read of the old parity, and a full rotation. Therefore, one more full rotation is needed before the old data becomes available.

Since the service time for high priority requests, $Y_p$, depends on the queueing time $Q_p$, there is no easy way to solve this model. As an alternative, we develop bounds on the service time $Y_p$, denoted by $U_p$ and $L_p$ for upper and lower bounds respectively, and then obtain estimates of the mean I/O response times based on these bounds.

For ease of reference, we introduce the following additional notation in our analysis:

$\lambda$: the arrival rate to the disk I/O subsystem;

$\lambda_p, \lambda_{rw}$: the arrival rates for the high priority parity update requests and the low priority read, write requests to a disk, respectively;

$Q_p, Q_{rw}$: $r.v.$'s for queueing times of high and low priority requests;

$Y_p, Y_{rw}$: $r.v.$'s for service times of high and low priority requests;

$Z_r, Z_w, Z$: $r.v.$'s for read, write, and overall I/O response times.

In addition, when it is necessary to distinguish between the disks, we will use the notation $Q_p(i), Q_{rw}(i), \cdots,$ and $Z(i)$ for the queueing delays, service times, and response times on disk $i$, $i = 1, 2, \cdots, N$. This is convenient when we consider the case that the access patterns are nonuniform. We also use superscripts $l$ and $u$ to denote quantities obtained based on the upper and lower bounds of $Y_p$.

### 4.2.1 The PS policy

As described before, under the BS policy, a parity request is generated whenever a write request is scheduled for service.

**Lower Bound on the Service Time $Y_p$:**

The following is a lower bound on $Y_p$,

$$Y_p \geq L_p \equiv X + 2\alpha + R_{max} \tag{8}$$

having mean

$$\overline{L_p} = \overline{X} + 2\alpha + R_{max}. \tag{9}$$

Compared to equation (7), the second full rotation never occurs. $L_p$ matches well to $Y_p$ when the system is highly loaded. This is because when the workload increases, the queueing delay $Q_p$ increases, and the probability $P\{X' \geq Q_p + X + R_{max}\}$ approaches zero.

The density function for $L_p$ is

$$f_{L_p}(x) = f_X(x - 2\alpha - R_{max}), \qquad 2\alpha + R_{max} \leq x \leq 2\alpha + 2R_{max} + S_{max}. \tag{10}$$

Let $Y_{rw}$ be a $r.v.$ for the service time of low priority read and write requests,

$$Y_{rw} = \theta Y_r + (1-\theta)Y_w$$

where $P\{\theta = 1\} = p_r$ and $P\{\theta = 0\} = p_w$. The density function for $Y_{rw}$ is given by

$$f_{Y_{rw}}(y) = \begin{cases} p_r f_X(y-\alpha), & \alpha \le y \le 2\alpha + R_{max}, \\ p_r f_X(y-\alpha) + p_w f_X(y-2\alpha-R_{max}), & 2\alpha + R_{max} < y \le \alpha + R_{max} + S_{max}, \\ p_w f_X(y-2\alpha-R_{max}), & \alpha + R_{max} + S_{max} < y \le 2\alpha + 2R_{max} + S_{max}, \end{cases} \tag{11}$$

and its mean is

$$\overline{Y_{rw}} = p_r\left(\overline{X} + \alpha\right) + p_w\left(\overline{X} + 2\alpha + R_{max}\right). \tag{12}$$

We then obtain the Laplace transforms of the service times of high and low priority requests by

$$B_p^*(s) = \int_{2\alpha+R_{max}}^{2\alpha+3R_{max}+S_{max}} f_{L_p}(y)e^{-sy}\,dy \tag{13}$$

and

$$B_{rw}^*(s) = \int_{\alpha}^{2\alpha+2R_{max}+S_{max}} f_{Y_{rw}}(y)e^{-sy}\,dy. \tag{14}$$

We now use these service time distributions to develop an approximate analysis of the system under the assumption of Poisson arrivals. Based on results on non-preemptive priority queues, the Laplace transforms for the queueing delays of the P and RW queues are ([14] pp. 121)

$$W_p^*(s) = \frac{(1-\rho)s + \lambda_{rw}[1 - B_{rw}^*(s)]}{s - \lambda_p + \lambda_p B_p^*(s)} \tag{15}$$

where $\rho$ is the device utilization, $\rho = \lambda_p\overline{L_p} + \lambda_{rw}\overline{Y_{rw}}$, and

$$W_{rw}^*(s) = \frac{(1-\rho)[s + \lambda_p - \lambda_p G_p^*(s)]}{s - \lambda_{rw} + \lambda_{rw}B_{rw}^*\left(s + \lambda_p - \lambda_p G_p^*(s)\right)} \tag{16}$$

where $G_p^*(s) = B_p^*\left(s + \lambda_p - \lambda_p G_p^*(s)\right)$.

The mean read response time is given by

$$\overline{Z_r^l} = \overline{Q_{rw}^l} + \overline{X} + \alpha \tag{17}$$

where $\overline{Q_{rw}^l} = -W_{rw}^{*\prime}(0)$.

From the scenarios shown in Figure 1, it is clear that the mean write response time, $\overline{Z_w^l}$. is calculated by

$$\overline{Z_w^l} = \overline{Q_{rw}^l} + E\left[\max\{X', Q_p^l + X\}\right] + 2\alpha + R_{max}. \tag{18}$$

Notice that the $Q_{rw}^l$ above is the queueing time that a write request waits in the RW queue of disk $j$ which contains its target data, whereas the $Q_p^l$ is the queueing time that its corresponding parity request waits in the P queue of disk $i$ which contains its parity block. Since all of the disks are statistically identical, we use the notation $Q_{rw}^l$ and $Q_p^l$ instead of $Q_{rw}^l(j)$ and $Q_p^l(i)$ in (18). Since we only have the Laplace transform for the distribution of $Q_p^l$ and in our case, there is no easy way to obtain a closed form for the distribution of $Q_p^l$ by inverting the Laplace transform,

we approximate the distribution of $Q_p^l$ by a Coxian distribution [6] with identical first and second moments.

The first and second moments of $Q_p^l$ are derived from equation (15),

$$E[Q_p^l] = \frac{\lambda_{rw}\overline{Y_{rw}^2} + \lambda_p\overline{L_p^2}}{2(1-\lambda_p\overline{L_p})}, \tag{19}$$

$$E\left[(Q_p^l)^2\right] = \frac{\lambda_{rw}\overline{Y_{rw}^3} + \lambda_p\overline{L_p^3}}{3(1-\lambda_p\overline{L_p})} + \frac{(\lambda_{rw}\overline{Y_{rw}^2} + \lambda_p\overline{L_p^2})\lambda_p\overline{L_p^2}}{2(1-\lambda_p\overline{L_p})^2} \tag{20}$$

where the moments of $L_p$ and $Y_{rw}$ are given in the Appendix.

Let $\xi_k$ be a $K$-stage Coxian *r.v.* with parameters $q$ and $\mu$, and *pdf*

$$g(x) = (1-q)u_0(x) + (1-q)\sum_{i=1}^{K-1}\frac{q^i\mu(\mu x)^{i-1}e^{-\mu x}}{(i-1)!} + \frac{q^K\mu(\mu x)^{K-1}e^{-\mu x}}{(K-1)!} \tag{21}$$

where $u_0(x)$ is the unit impulse function (see Section 4.1).

The first and second moments of $\xi_k$ are

$$\overline{\xi_k} = \frac{\sum_{i=1}^K q^i}{\mu},$$

$$\overline{\xi_k^2} = \frac{2\sum_{i=1}^K iq^i}{\mu^2}.$$

The parameters $q, \mu$, and $K$ are calculated in such a way that the above two moments match exactly to those of $Q_p^l$. Let $C_x$ be the coefficient of variance of $Q_p^l$, $C_x^2 = E\left[(Q_p^l)^2\right]/E\left[Q_p^l\right]^2 - 1$, then the number of stages satisfies the equation

$$\frac{1}{C_x^2} \le K < \frac{1}{C_x^2} + 1$$

and $q$ and $\mu$ are obtained by solving the following equations

$$E[Q_p^l] = \frac{\sum_{i=1}^K q^i}{\mu},$$

$$E\left[(Q_p^l)^2\right] = \frac{2\sum_{i=1}^K iq^i}{\mu^2}.$$

Define *r.v.* $V = Q_p^l + X$, having *pdf*

$$f_V(x) = \int_0^x g(x-t)f_X(t)\,dt.$$

Since $X'$ is independent of $Q_p^l$ and $X$, the cumulative distribution function ($CDF$) of $\max\{X', Q_p^l + X\}$ is $F_{X'}(x)F_V(x)$, where $F_{X'}(x)$ and $F_V(x)$ are the $CDF$'s of $X'$ and $V$ respectively. The mean is given by

$$E\left[\max\{X', Q_p^l + X\}\right] = E[X'] + E[Q_p^l + X] - \int_0^{R_{max}+S_{max}} [1 - F_{X'}(x)]\,[1 - F_V(x)]\,dx.$$

Last, the mean I/O response time is given by

$$\overline{Z^l} = p_r \overline{Z_r^l} + p_w \overline{Z_w^l}.$$

we shall refer to this as the *Lower Bound (LB) Approximation.*

**Upper Bound on the Service Time $Y_p$:**

The upper bound on $Y_p$ is obtained by

$$Y_p \leq U_p \equiv X + 2\alpha + R_{max} + \mathbf{1}(X' \geq X + R_{max})R_{max} \tag{22}$$

where $\mathbf{1}(A)$ is the indicator function defined by

$$\mathbf{1}(A) = \begin{cases} 1, & \text{if A is true,} \\ 0, & \text{if A is not true.} \end{cases}$$

We expect that $U_p$ better matches $Y_p$ when the system is lightly loaded since, as workload decreases, the queueing delay $Q_p$ approaches zero.

The density function for $U_p$ is

$$f_{U_p}(y) = \begin{cases} f_X(y - 2\alpha - R_{max})F_{X'}(y - 2\alpha), & 2\alpha + R_{max} \leq y \leq 2\alpha + 2R_{max}, \\ f_X(y - 2\alpha - 2R_{max})[1 - F_{X'}(y - 2\alpha - R_{max})] + \\ \quad f_X(y - 2\alpha - R_{max})F_{X'}(y - 2\alpha), & 2\alpha + 2R_{max} < y \leq 2\alpha + R_{max} + S_{max}, \\ f_X(y - 2\alpha - 2R_{max})[1 - F_{X'}(y - 2\alpha - R_{max})] + \\ \quad f_X(y - 2\alpha - R_{max}), & 2\alpha + R_{max} + S_{max} < y \leq 2\alpha + 2R_{max} + S_{max}, \end{cases} \tag{23}$$

where $F_{X'}(y)$ is the $CDF$ of $X'$ and $X' =_{st} X$, i.e., $X'$ and $X$ have the same $CDF$.

The mean read, write and overall response time can be obtained in terms of the upper bound on $Y_p$ by using the same methodology as described in obtaining the LB Approximation. We shall refer to this as the *Upper Bound (UB) Approximation.*

**Simulation Validations:**

To verify our models above, we have run several simulation experiments on a disk array of 16 disks. The simulation results are obtained by averaging over 40 runs, each run including 50,000 I/O requests. We obtained 95% confidence intervals whose widths are less than 2.5% of the estimated point values. The results obtained from the upper and lower bound on the service time $Y_p$, as well as simulations are shown in Table 1 and 2 for the cases of read probability $p_r = 0.25$ and $p_r = 0.75$, respectively. The sequential access probability $p_s$ is set to 0.3 in these experiments. The device utilization varies from 0.11 to 0.92 in these tables.

From these tables, we observe that the LB approximation, which is based on the lower bound of $Y_p$, estimates the system performance of the true system very well. Therefore, in the following discussions, we will only consider the LB approximation. Whenever there is no confusion, we will drop the superscript $l$ and use $Y_p$ instead of $L_p$ for ease of discussions.

| $\lambda$ | Read Resp. Time (sec) | | | Write Resp. Time (sec) | | |
|---|---|---|---|---|---|---|
| | UB Approx. | LB Approx. | Sim. | UB Approx. | LB Approx. | Sim. |
| 32.0 | 0.02090 | 0.02081 | 0.02087 | 0.04537 | 0.04523 | 0.04530 |
| 59.2 | 0.02412 | 0.02391 | 0.02399 | 0.05041 | 0.05007 | 0.05016 |
| 86.4 | 0.02868 | 0.02826 | 0.02847 | 0.05703 | 0.05639 | 0.05665 |
| 113.6 | 0.03538 | 0.03457 | 0.03469 | 0.06607 | 0.06495 | 0.06516 |
| 140.8 | 0.04569 | 0.04418 | 0.04453 | 0.07909 | 0.07712 | 0.07737 |
| 168.0 | 0.06279 | 0.05985 | 0.06022 | 0.09932 | 0.09574 | 0.09595 |
| 195.2 | 0.09467 | 0.08831 | 0.08837 | 0.13484 | 0.12761 | 0.12763 |
| 222.4 | 0.16880 | 0.15128 | 0.15192 | 0.21324 | 0.19456 | 0.19526 |
| 249.6 | 0.48015 | 0.37764 | 0.37772 | 0.53010 | 0.42600 | 0.42694 |

Table 1: Compare the Results from Upper and Lower Bound on Service Time $Y_p$, and Simulations. ($p_r = 0.25$)

| $\lambda$ | Read Resp. Time (sec) | | | Write Resp. Time (sec) | | |
|---|---|---|---|---|---|---|
| | UB Approx. | LB Approx. | Sim. | UB Approx. | LB Approx. | Sim. |
| 32.0 | 0.01931 | 0.01929 | 0.01931 | 0.04272 | 0.04267 | 0.04272 |
| 84.8 | 0.02160 | 0.02152 | 0.02158 | 0.04639 | 0.04625 | 0.04632 |
| 137.6 | 0.02472 | 0.02454 | 0.02466 | 0.05101 | 0.05074 | 0.05089 |
| 190.4 | 0.02912 | 0.02879 | 0.02893 | 0.05705 | 0.05657 | 0.05678 |
| 243.2 | 0.03563 | 0.03502 | 0.03515 | 0.06534 | 0.06452 | 0.06464 |
| 296.0 | 0.04593 | 0.04479 | 0.04505 | 0.07758 | 0.07616 | 0.07644 |
| 348.8 | 0.06408 | 0.06177 | 0.06208 | 0.09786 | 0.09518 | 0.09535 |
| 401.6 | 0.10302 | 0.09726 | 0.09818 | 0.13913 | 0.13290 | 0.13368 |
| 454.4 | 0.23732 | 0.21107 | 0.21049 | 0.27614 | 0.24929 | 0.24806 |

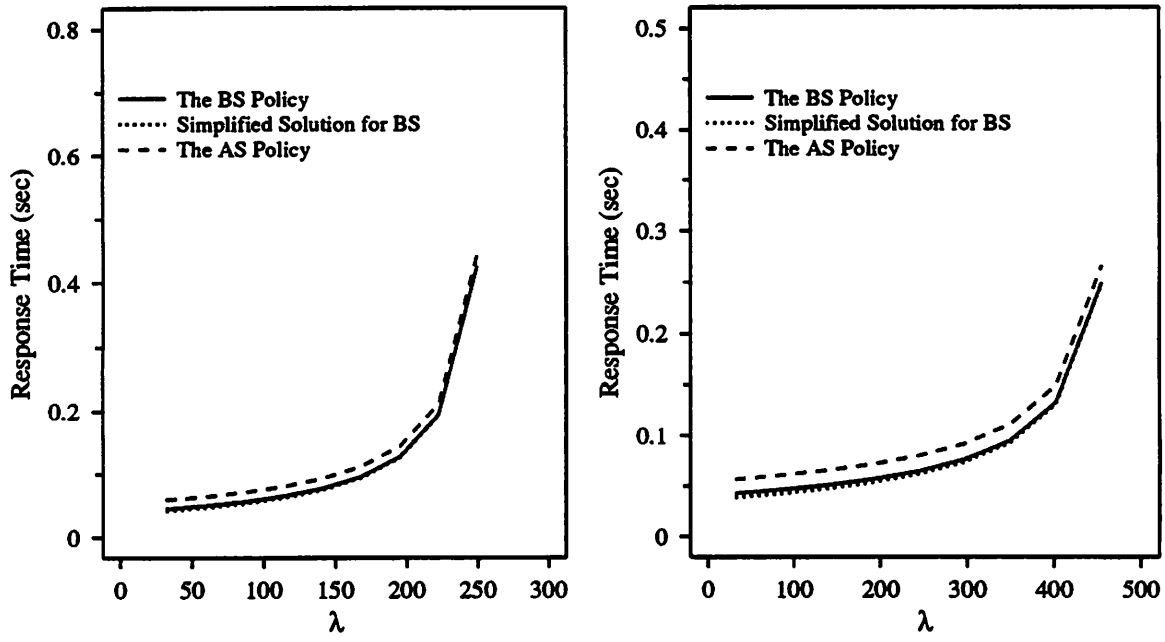
Table 2: Compare the Results from Upper and Lower Bound on Service Time $Y_p$, and Simulations. ($p_r = 0.75$)

### 4.2.2 The AS Policy

As described in Section 3.2, the AS policy delays generating a parity update request until the corresponding old data, which is required for the calculation of new parity, has been read out. In this case, whenever the disk containing the parity is ready to update the parity block, the information for the calculation of new parity block is always available.

Since the only difference between the AS and the BS policies is the time at which parity update requests are generated, we conclude that the analytic model for the BS policy is also applicable to the AS policy, except that the write response time is now calculated by

$$\overline{Z_w} = \overline{Q_{rw}} + \overline{X'} + \overline{Q_p} + \overline{X} + R_{max} + 3\alpha. \tag{24}$$




(a) Write Response Time. ($p_r$= 0.25)

(b) Write Response Time. ($p_r$= 0.75)

Figure 4: Compare the BS and AS policies, and the Simplified Solution for BS. ($p_s = 0.3$)

In Figure 4, we show the performance degradation of the AS policy compared to the BS policy, where only the curves obtained from the lower bound of $Y_p$ are illustrated. The mean response times for read requests are the same under both policies. In the following discussions, we will focus on the BS policy. Readers can easily generate results for the AS policy using the same approach as described earlier for the BS policy.

### 4.2.3 A Simplified Solution

In this subsection, we investigate a simplified analytic model and examine the deviation of its results from those obtained in the previous analysis. Specifically, we replace the second term

$E[\max\{X', Q_p + X\}]$ in equation (18) by $E[Q_p + X]$ when calculating the mean write response time, i.e.,

$$\overline{Z_w} = \overline{Q_{rw}} + E[Q_p + X] + 2\alpha + R_{max}. \tag{25}$$

This simplification causes little error when the load is high. In this case, the queueing time $Q_p$ is large and therefore $Q_p + X$ dominates $X'$. However, when the system is lightly loaded, $Q_p$ tends to be small, and some error may appear in the estimate for the mean write response time due to the simplification. Fortunately, as shown in Figure 4, it turns out that the error is small and the simplified solution provides a good approximation to the previous analysis.

Based on these observations, we will use the simplified analytic model in the next section when we direct our attention to multiple block accesses.

### 4.3 An Analysis of the Parity Striping Architecture

For a single block access, if each I/O request is assumed to be directed to any disk with equal probability, then there is no difference between the RAID 5 and the Parity Striping architectures. However, if the I/O access pattern is skewed, i.e., some files are "hotspots", then the Parity Striping architecture will suffer a performance degradation, since files are allocated on a single disk. Under skewed accesses, a large fraction of I/O requests are directed to a small fraction of disks. Hence these disks are likely to become a bottleneck which will limit the throughput of the I/O subsystem.

In order to examine the effect of skewed accesses, we use the "$s$-th degree of skew" model which is a common mathematical way to characterize skewing [24]. In this model, if the least busy disk has an arrival rate of $\lambda$, then the $k$-th least busy disk has an arrival rate of $k^s\lambda$. A 0-degree skew produces a uniform distribution. Suppose the skewed access is of $s$-degree, and let disk $i$ be the $i$-th least busy device in the array, then a request goes to disk $i$ with probability

$$p_i = \frac{i^s}{\sum_{j=1}^{N} j^s}, \qquad i = 1, 2, \cdots, N \tag{26}$$

and the arrival rates of data and parity requests to disk $i$ are

$$\begin{aligned} \lambda_{rw}(i) &= p_i\lambda, \\ \lambda_p(i) &= \frac{p_w(1-p_i)\lambda}{N-1}, \qquad i = 1, 2, \cdots, N. \end{aligned} \tag{27}$$

Here, we assume that for any write request to disk $i$, its corresponding parity block resides on any other $N-1$ disks with the same probability, i.e., its parity update request might go to disk $j$ with probability $1/(N-1)$, $j \neq i$. Substituting (27) into equations (16) yields $\overline{Q_{rw}(i)}$, and the mean read response time on disk $i$, $Z_r(i)$, can be obtained immediately from (17). Substituting $\lambda_{rw}(j)$ and $\lambda_p(j)$ into equations (19) and (20), and then following the same steps yields $E[\max\{X', Q_p(j) + X\}]$. Thus the mean write response time on disk $i$, $\overline{Z_w(i)}$, is

$$\overline{Z_w(i)} = \overline{Q_{rw}(i)} + \frac{1}{N-1} \sum_{j=1; j\neq i}^{N} E[\max\{X', Q_p(j) + X\}] + 2\alpha + R_{max}$$

Finally, the mean response times are averaged over all disks

$$\overline{Z_r} = \sum_{i=1}^{N} p_i \overline{Z_r(i)},$$

$$\overline{Z_w} = \sum_{i=1}^{N} p_i \overline{Z_w(i)},$$
$$\overline{Z} = p_r \overline{Z_r} + p_w \overline{Z_w}. \quad (28)$$

# 5 Analytic Models for Multiple Block Accesses

In this section, we extend our previous discussions from single block access to multiple block access. Specifically, we assume that each arriving I/O request needs to access $B$ blocks of data, where $B$ is a $r.v.$ having an arbitrary distribution $P\{B = k\}$. Since this paper focuses on applications that predominantly generate requests for small numbers of blocks, we assume that the number of blocks in each request is less than the strip width $W_s$ for ease of discussion. As stated before, the simplified model discussed in the last section will be used in the following analyses, and only the BS policy is considered.

## 5.1 The RAID 5 Architecture

In RAID 5, data is interleaved in such a way that contiguous blocks are allocated on contiguous disks in the array. When a $k$-block request arrives at the I/O subsystem, the dispatcher generates $k$ tasks (referred to as RW tasks) and sends them to the $k$ corresponding RW queues. Let $L$ be a $r.v.$ denoting the disk where the first block of each request is located. We assume that $L$ is uniformly distributed in $[1, W_s]$. Again, this assumption is reasonable because of the inherent load balancing property of RAID 5. If the $k$-request is a read, then it finishes as soon as all of the $k$ tasks complete.

If it is a write, we distinguish three cases:

**case 1**: the request starts at disk 1 and the request size is $k = W_s$. In this case, a *full strip write* occurs and there is no need to read the old data and parity before updating, since the new parity block can be obtained by XORing all of the $W_s$ new data blocks. The mean response time for such a full strip write is calculated in the same way as for a read;

**case 2**: the request accesses a partial strip, i.e., $k < W_s$, and all of the $k$ blocks belongs to the same strip. In this case, a parity task (referred to as P task) is generated to update the parity block when the last of the $k$ RW tasks is scheduled for service. The write request completes when all of the $k$ RW tasks and the P task finish.

**case 3**: the request accesses two partial strips, i.e., the $k$ blocks are allocated across the strip boundary. In this case, two parity blocks need to be updated and the write is divided into two separate partial strip operations. The write completes only when the two partial strip operations finish.

To analyze the system, the priority queueing model developed in the last section can still be used but with different arrival rates. The arrival rate to each RW queue is now

$$\lambda_{rw} = \lambda \overline{B}/N \quad (29)$$

and to each P queue,

$$\lambda_p = p_w \lambda (q_1 + 2q_2)/N \quad (30)$$

where $q_1, q_2$ are the probabilities that a write is to access one or two partial strips, respectively,

$$\begin{aligned} q_1 &= \sum_{k=1}^{W_s} \sum_{i=1}^{W_s-k+1} P\{B=k\}P\{L=i\}, \\ &= \frac{W_s - \overline{B} + 1}{W_s}, \\ q_2 &= 1 - q_1, \\ &= \frac{\overline{B}-1}{W_s}. \end{aligned} \tag{31}$$

The response times of write requests can be expressed in term of the maximum of as many as $B+2$ $r.v.$'s corresponding to the times that $B$ RW tasks complete and two P tasks complete. These $r.v.$'s are not necessarily independent. However, we will assume that they are of each other (see below) in order to complete the analysis. In order to simplify our derivation and results, we will express the results in terms of the $CDF$'s of different $r.v.$'s rather than the $r.v.$'s themselves. This allows us to ignore any dependences.

We introduce the following notations: let $F_X(t)$ and $F_Y(t)$ be $CDF$'s such that $F_X(t) = F_Y(t) = 0, t \leq 0$,

$$\begin{aligned} (F_X * F_Y)(t) &= \int_0^t F_X(t-y) f_Y(y)\, dy \\ E_k[F_X(t)] &= \int_0^\infty x^k\, dF_X(x) \end{aligned}$$

These correspond to the convolution operation and to the operation for obtaining the $k$-th moment, respectively. Last, $(F_X\ F_Y)(t)$ corresponds to the $CDF$ associated with the max of two independent $r.v.$'s having $CDF$'s $F_X(t)$ and $F_Y(t)$. To simplify subsequent expressions, whenever there is no confusion, we will omit the parameter $(t)$ for $CDF$'s, e.g., $(F_X * F_Y)(t)$ will be abbreviated as $F_X * F_Y$.

**Mean Read Response Time:**
The mean read response time is

$$E[Z_r] = \sum_{k=1}^{W_s} P\{B=k\}E[Z_r|B=k]. \tag{32}$$

where

$$E[Z_r|B=k] = E\left[(F_{Q_{rw}} * F_X)^k\right] + \alpha.$$

Since all the disks are statistically identical, it does not matter which $k$ disks are involved in the calculation of maximum of $k$ random variables.

Again, a Coxian $r.v.$ $\xi$ is used to approximate $Q_{rw} + X$ so that the first and second moments are identical. The moments for $Q_{rw}$ are

$$\overline{Q_{rw}} = \frac{\lambda_{rw}\overline{Y_{rw}^2} + \lambda_p\overline{Y_p^2}}{2(1-\rho_p)(1-\rho)}, \tag{33}$$

$$\overline{Q_{rw}^2} = \frac{\lambda_{rw}\overline{Y_{rw}^3} + \lambda_p\overline{Y_p^3}}{3(1-\rho_p)^2(1-\rho)} + \frac{(\lambda_{rw}\overline{Y_{rw}^2} + \lambda_p\overline{Y_p^2})\lambda_p\overline{Y_p^2}}{2(1-\rho_p)^3(1-\rho)} + \frac{(\lambda_{rw}\overline{Y_{rw}^2} + \lambda_p\overline{Y_p^2})^2}{2(1-\rho_p)^2(1-\rho)^2} \tag{34}$$

where $\rho_p = \lambda_p \overline{Y_p}$ and the first three moments for service times $Y_{rw}$ and $Y_p$ are found in the Appendix. Hence

$$\begin{aligned} \overline{\xi} &= \overline{Q_{rw}} + \overline{X} \\ \overline{\xi^2} &= \overline{Q_{rw}^2} + 2\overline{Q_{rw}}\,\overline{X} + \overline{X^2}. \end{aligned}$$

Let $F_\xi(x)$ and $f_\xi(x)$ be the $CDF$ and $pdf$ of $\xi$ (equation (21)). Since the $k$ $\xi$'s are independent of each other, we have

$$\begin{aligned} E[Z_r|B=k] &= E\left[(F_\xi)^k\right] + \alpha \\ &= \int_0^\infty x f_\xi(x) F_\xi^{k-1}(x)\, dx + \alpha. \end{aligned}$$

**Mean Write Response Time:**
The mean write response time is given by

$$E[Z_w] = \sum_{k=1}^{W_s} \sum_{i=1}^{W_s} P\{B=k\} P\{L=i\} E[Z_w|B=k, L=i] \tag{35}$$

where $P\{L=i\} = 1/W_s$, and

$$E[Z_w|B=k, L=i] = \begin{cases} E\left[(F_{Q_{rw}} * F_X)^{W_s}\right] + \alpha, & k = W_s;\ i = 1, \\ E\left[(F_{Q_{rw}})^k\right] + E[F_{Q_p}] + F[F_X] + 2\alpha + R_{max}, & \\ & 1 \le k < W_s;\ 1 \le i \le W_s - k + 1, \\ E\left[\left((F_{Q_{rw}})^{W_s-i+1} * F_{Q_p} * F_X\right),\ \left((F_{Q_{rw}})^{k-W_s+i-1} * F_{Q_p} * F_X\right)\right] + & \\ 2\alpha + R_{max}, & 1 \le k \le W_s;\ W_s - k + 2 \le i \le W_s. \end{cases}$$

Approximations are again made in the first two cases by replacing the $CDF$'s $F_{Q_{rw}} * F_X$ and $F_{Q_{rw}}$ with Coxian $CDF$'s such that the first two moments match. In the third case, this type of approximation is applied twice, first to approximate $F_{Q_{rw}}$ by a Coxian distribution $\tilde{F}_{Q_{rw}}$, and second to approximate both $\left((\tilde{F}_{Q_{rw}})^{W_s-i+1} * F_{Q_p} * F_X\right)$ and $\left((\tilde{F}_{Q_{rw}})^{k-W_s+i-1} * F_{Q_p} * F_X\right)$ by Coxian distributions.

Finally, the mean I/O response time is

$$\overline{Z} = p_r \overline{Z_r} + p_w \overline{Z_w}.$$

Observe that the queueing times and service times of all tasks (RW and P) are assumed too be independent in this approximate analysis. While this independence assumption is valid for single block accesses, it is violated in the case that requests are allowed to access multiple blocks of data. First, arrivals to the different disks are not independent since all of the RW tasks associated with a read or write request arrive simultaneously at different disks. Second, the tasks corresponding to a single request leave the disk arms on the same cylinder. Hence a subsequent request may generate RW tasks that will exhibit the same seek times at different disks. Last, in the case that a write request is to separate strips (i.e., across strip boundary), although the RW tasks are routed to separate disks, it is possible that the P task associates with one strip will be routed to one of the disks that is executing a RW task associated with the other strip.

Based on these observations, we have compared the predictions made by this model with those obtained through simulation. The results show that the analytic results match that of simulations very well when the request size follows a quasi-geometric distribution (to be described in Section 6.2) with mean 1.5 blocks. When the request sizes are uniformly distributed in $[1, W_s]$ (with $W_s = 15$), we observe that the approximate analysis produces estimates of the mean response time that are as much as 12% higher than those obtained from the simulation. The largest errors occur at high loads when most requests are write requests.

## 5.2 The Parity Striping Architecture

For the Parity Striping architecture, we assume that each read can be satisfied by a single disk, and that each write only needs to access two disks, the one containing the data to update and the one containing the associated parity block(s). Again, we model each disk as a 2-class non-preemptive priority system with arrivals described by a Poisson process. The arrival rates are $\lambda_{rw} = \lambda/N$ and $\lambda_p = p_w\lambda/N$ for non-skewed access patterns, i.e., each request is directed to any disk with the same probability.

As stated before, while a skewed access pattern has little effect on the RAID 5 architecture, it does produce a negative effect on the performance of the Parity Striping architecture. Suppose a skewed access pattern is of "$s$-th degree". Consider the $i$-th least busy device in the array. The arrival rates to the disk $i$, $\lambda_{rw}(i)$ and $\lambda_p(i)$, are given by (27). The disk service time, however, is independent of $i$, which for a read request is

$$Y_r = X + B\alpha.$$

For a write request, since it accesses multiple blocks, after reading out the $B$ blocks (for the purpose of a new parity block calculation), the disk will have rotated part way to the beginning of the $B$ data blocks. Therefore, instead of a full rotation, it only needs to rotate the remaining distance prior to writing back of the new data. Let $N_b$ denote the number of blocks on each track, then

$$Y_w = X + 2B\alpha + \left(1 - \frac{(B-1) \bmod N_b}{N_b}\right) R_{max}.$$

The service time for high priority parity requests, $Y_p$, is calculated in the same way as write requests $Y_w$.

The first and second moments for the service times of low and high priority requests are

$$\begin{aligned}
\overline{Y_{rw}} &= \overline{X} + (1+p_w)\alpha\overline{B} + p_w(1-\eta)R_{max}, && (36)\\
\overline{Y_{rw}^2} &= p_w\left(\overline{X^2} + 4\alpha^2\overline{B^2} + \left(\frac{R_{max}}{N_b}\right)^2 E\left[(N_b - (B-1) \bmod N_b)^2\right] + 4\overline{X}\alpha\overline{B} + \right.\\
&\quad \left.(2\overline{X} + 4\alpha\overline{B})(1-\eta)R_{max}\right) + p_r(\overline{X^2} + 2\overline{X}\alpha\overline{B} + \alpha^2\overline{B^2}),\\
\overline{Y_p} &= \overline{X} + 2\alpha\overline{B} + (1-\eta)R_{max}, && (37)\\
\overline{Y_p^2} &= \overline{X^2} + 4\alpha^2\overline{B^2} + \left(\frac{R_{max}}{N_b}\right)^2 E\left[(N_b - (B-1) \bmod N_b)^2\right] + 4\overline{X}\alpha\overline{B} + \\
&\quad (2\overline{X} + 4\alpha\overline{B})(1-\eta)R_{max}
\end{aligned}$$

where $\eta$ is the fraction of distance that has rotated when reading old data blocks,

$$\begin{aligned}\eta &= E\left[\frac{(B-1) \bmod N_b}{N_b}\right], \\ &= \sum_{k=0}^{N_b-1} k \sum_{m=0}^{\lfloor \frac{W_s-k-1}{N_b} \rfloor} P\{B = mN_b + k + 1\},\end{aligned}$$

and the second moment

$$\begin{aligned}E\left[(N_b - (B-1) \bmod N_b)^2\right] &= N_b^2 - 2N_b^2\eta + E\left[((B-1) \bmod N_b)^2\right], \\ &= N_b^2 - 2N_b^2\eta + \sum_{k=0}^{N_b-1} k^2 \sum_{m=0}^{\lfloor \frac{W_s-k-1}{N_b} \rfloor} P\{B = mN_b + k + 1\}.\end{aligned}$$

Substituting these moments and arrival rates $\lambda_{rw}(i)$ and $\lambda_p(i)$ into equation (33) yields the mean queueing delay $\overline{Q_{rw}(i)}$. Similar to the discussions in Section 4.2, by using these moments, $\lambda_{rw}(j)$ and $\lambda_p(j)$, and equation (19), we obtain $\overline{Q_p(j)}$. Thus, the mean response times on disk $i$ are

$$\begin{aligned}\overline{Z_r(i)} &= \overline{Q_{rw}(i)} + \overline{X} + \alpha\overline{B} \\ \overline{Z_w(i)} &= \overline{Q_{rw}(i)} + \frac{1}{N-1} \sum_{j=1;j\neq i}^{N} \overline{Q_p(j)} + \overline{X} + \alpha\overline{B} + (1-\eta)R_{max} \\ \overline{Z(i)} &= p_r\overline{Z_r(i)} + p_w\overline{Z_w(i)}.\end{aligned}$$

Last, the mean response times are obtained by averaging over all disks (equation (28)).

Unlike the analytic model for RAID 5 in the last subsection, which may over-estimate the system mean response time when the mean request size is large, here the analytic results for Parity Striping are quite accurate and match the simulation results well.

### 5.3 The Maximum Throughput

Since the database community is quite often interested in the throughput of I/O subsystems in terms of I/O requests per second, we examine the maximum throughput supported by the RAID 5 and Parity Striping architectures for multiple block access patterns. The single block access pattern is a special case with block size distribution $P\{B = 1\} = 1, P\{B \neq 1\} = 0$.

**RAID 5 :**

Let $\lambda$ be the I/O request arrival rate to the disk I/O subsystem. For RAID 5, since each disk stochastically behaves the same, we consider a randomly selected disk $j$. The arrival rates of high and low priority requests to this disk are given by equations (29) and (30). The equations (9) and (12) give the mean disk service times for the two classes of requests. Hence the disk utilization is

$$\begin{aligned}\rho &= \lambda_{rw}\overline{Y_{rw}} + \lambda_p\overline{Y_p}, \\ &= \left[\frac{\overline{B}}{N}\overline{Y_{rw}} + \frac{p_w}{N}(q_1 + 2q_2)\overline{Y_p}\right]\lambda\end{aligned}$$

where $q_1$ and $q_2$ are obtained from equation (31). The maximum throughput is achieved when $\rho = 1$, i.e.,

$$\lambda_{max} = \frac{N}{\overline{B}\ \overline{Y_{rw}} + p_w(q_1 + 2q_2)\overline{Y_p}}.$$

Since (9) only gives a lower bound on $Y_p$, we expect that the result derived here will provide an upper bound on the maximum throughput supported by RAID 5. Our simulation results yield a negligible difference from that of analysis. Therefore, this analytic model provides a very good approximation of the true system.

**Parity Striping** :

Unlike RAID 5, the throughput of the Parity Striping architecture is limited by the most heavily accessed disk in the array. The arrival rates to this disk are

$$\begin{aligned} \lambda_{rw} &= p_N\lambda, \\ \lambda_p &= \frac{p_w(1-p_N)}{N-1}\lambda \end{aligned}$$

where $p_N = N^s / \sum_{j=1}^{N} j^s$ is the probability that a request goes to the hottest disk (equation (26)). Thus,

$$\lambda_{max} = \frac{1}{p_N\overline{Y_{rw}} + \frac{p_w(1-p_N)}{N-1}\overline{Y_p}}$$

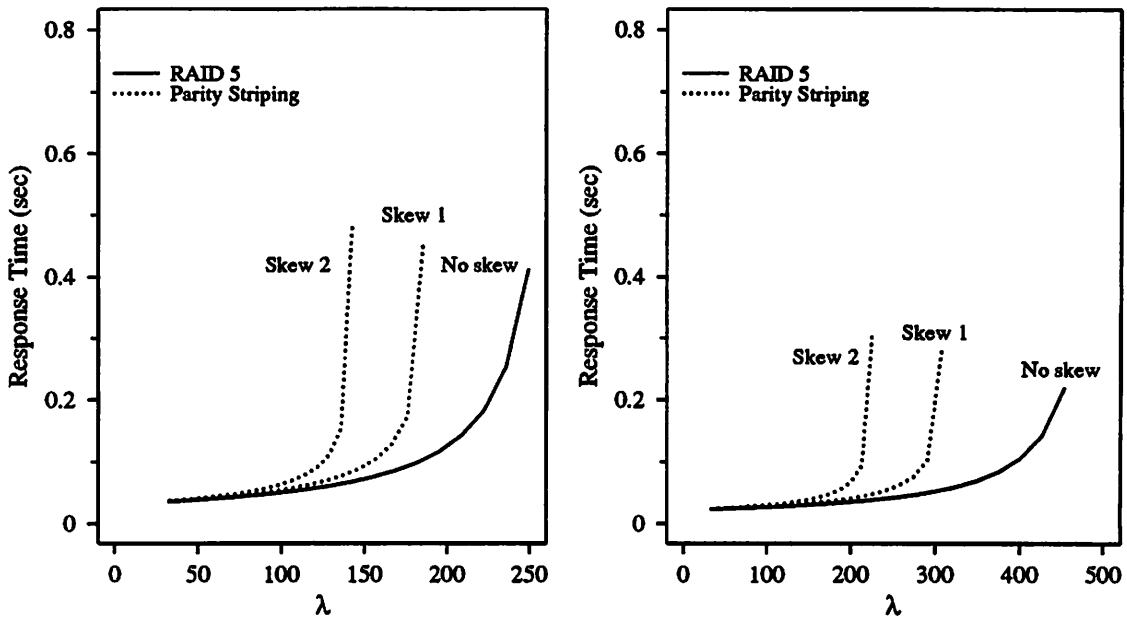
where the mean service times $\overline{Y_{rw}}$ and $\overline{Y_p}$ are given by equations (36) and (37).

# 6 Performance Comparisons

In this section, we compare the performance of the RAID 5 and Parity Striping architectures by using the analytic models developed in the last two sections. The number of disks, $N$, in the array is 16. Three kinds of workloads are used for the performance measurements, single block access, multiple block access with uniformly distributed request size, and multiple block access with request size having a quasi-geometric distribution. In all cases, I/O requests arrive at the disk subsystem according to a Poisson process with parameter $\lambda$. In addition, we also report the performance degradation of the Parity Striping architecture under skewed accesses. Finally, the maximum throughput supported by the RAID 5 and the Parity Striping architectures are examined. In all of our experiments, the sequential access probability, $p_s$, is set to 0.3.

## 6.1 Single Block Access

Figure 5 plots the mean I/O response time as a function of request arrival rate $\lambda$, where each request accesses one block of data (4K bytes). As discussed in Section 4, when each request is directed to any disk with the same probability, RAID 5 and Parity Striping are the same and therefore have the same mean I/O response times. However, when the access pattern is skewed, even for the first degree, the performance of Parity Striping degrades quickly and eventually exhibits a worse performance than RAID 5.




(a) Mean I/O Response Time.($p_r$=0.25)

(b) Mean I/O Response Time.($p_r$=0.75)

Figure 5: Performance of RAID 5 and Parity Striping with Single Block Accesses ($p_s = 0.3$)

## 6.2 Multiple Block Accesses

In this subsection, the performance of RAID 5 and Parity Striping is examined under two distributions of requests size $B$.

**Uniform Distribution** :

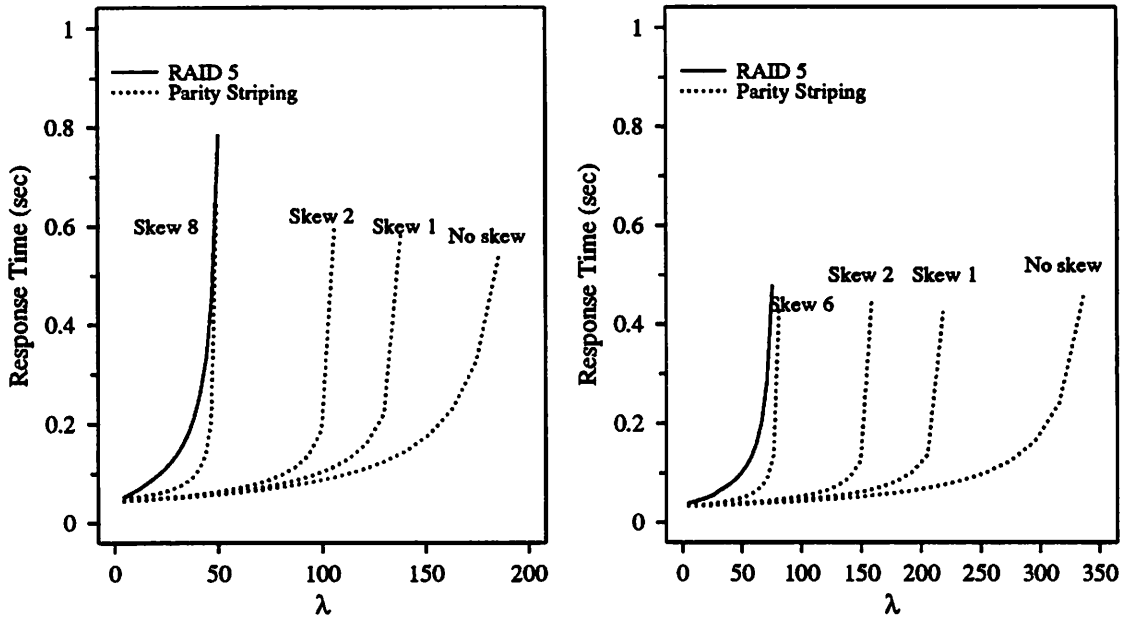
We first assume that request size is uniformly distributed in $[1, W_s]$, i.e.,

$$P\{B = k\} = \frac{1}{W_s}, \qquad i = 1, \cdots, W_s.$$

The performance results are shown in Figure 6. With this workload, we observe that Parity Striping performs much better than RAID 5 if there is no skew, since RAID 5 has to incur multiple seeks and rotations on several disks to read or update the desired data blocks whereas Parity Striping only incurs one. The advantage of the high transfer rate of RAID 5 does not show up here since only a small amount of data needs to be transferred for each request. When the access pattern is skewed, the performance of Parity Striping degrades. However, it can still achieve a performance comparable to or better than RAID 5 even for extremely skewed access patterns (it is unrealistic to believe that skew patterns with a degree of 6 or higher exist in most real world I/O subsystems). Based on these observations, we conclude that Parity Striping is better than RAID 5 when request sizes are uniformly distributed $r.v.$'s.

**Quasi-Geometric Distribution** :

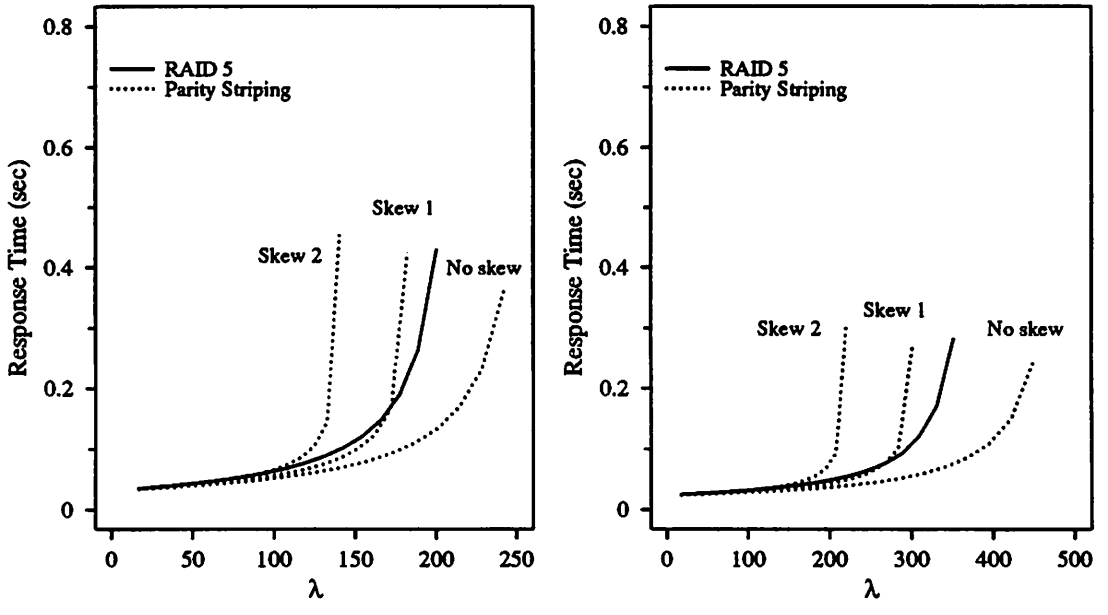
The second distribution for requests size, called quasi-geometric distribution, is more interesting, since it has been observed that in most database systems I/O's are predominated




(a) Mean I/O Response Time.($p_r$= 0.25)

(b) Mean I/O Response Time.($p_r$=0.75)

Figure 6: Multiple Block Accesses with Request Size Uniform in $[1, W_s]$. ($p_s = 0.3$)




(a) Mean I/O Response Time. ($p_r$=0.25)

(b) Mean I/O Response Time. ($p_r$=0.75)

Figure 7: Multiple Block Accesses with Request Size Quasi-Geometrically Distributed. ($p_s = 0.3, \sigma = p = 0.7$)

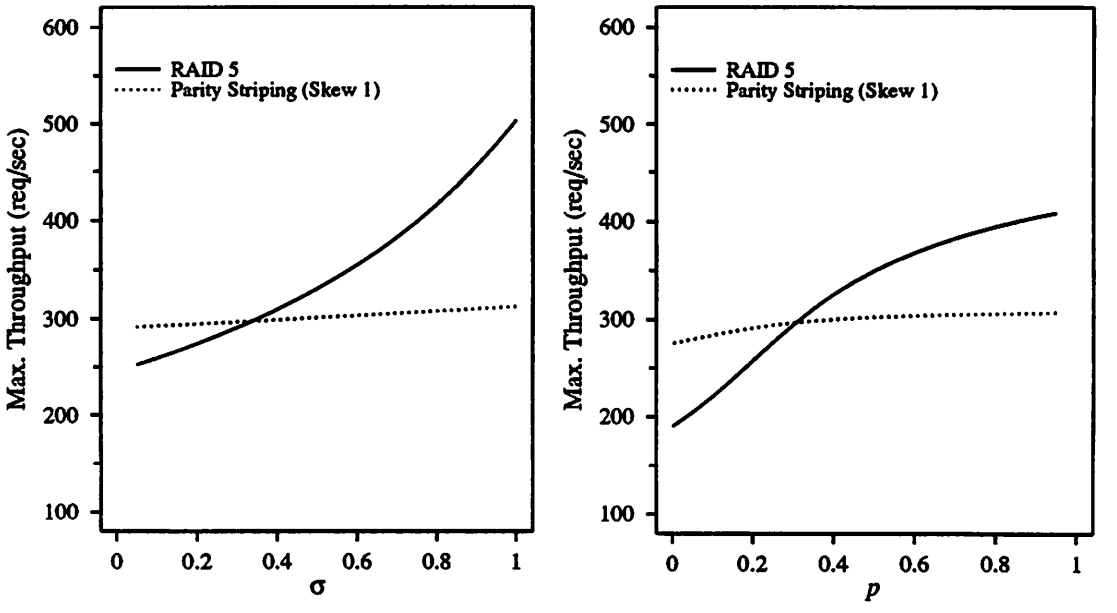
by requests which access only one block of data (4Kb) [24, 18]. Hence, we assume that the request size $B$ has the following distribution,

$$P\{B = i\} = \begin{cases} \sigma, & i = 1 \\ (1-\sigma)\frac{p(1-p)^{i-1}}{(1-p)-(1-p)^{W_s}}, & i = 2, \cdots, W_s. \end{cases}$$

Figure 7 illustrates the situation where 70% of requests accesses one block of data ($\sigma = p = 0.7$). The results show that Parity Striping still outperforms RAID 5 when there is no access skew. However, it is outperformed by RAID 5 even for the accesses of 1-degree of skew, which is likely to occur in most of the applications.

## 6.3 The Maximum Throughput

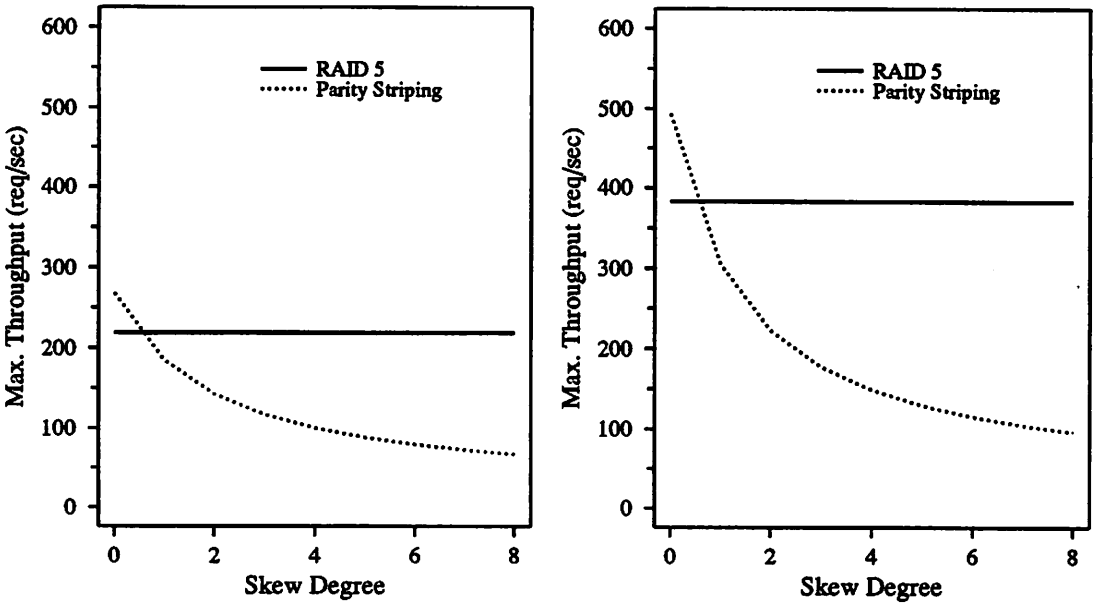
In this subsection, we examine the maximum throughput supported by RAID 5 and Parity Striping in terms of the number of I/O requests per second by using the formulas developed in Section 5.3. As discussed earlier, the performance of RAID 5 degrades quickly as the mean request size increases. On the other hand, the Parity Striping architecture is sensitive to the skewed access pattern. Figure 8 explores the sensitivity of the two architectures to the I/O request size. Here the request size is assumed to be quasi-geometrically distributed. In Figure 8 (a), we fix the parameter $p = 0.7$ and vary the single block probability $\sigma$, and in (b), we fix $\sigma = 0.7$ and vary $p$. Only the case $p_r = 0.75$ is plotted. The same behavior has been observed for the case $p_r = 0.25$. Figure 9 illustrates the sensitivity of Parity Striping to the skew in access patterns.




(a) Effect of Varying Req. Size. ($p$= 0.7)

(b) Effect of Varying Req. Size. ($\sigma$=0.7)

Figure 8: Sensitivity of RAID 5 and Parity Striping to the Request Size. ($p_s = 0.3, p_r = 0.75$, Request Size: Quasi-Geometrically Distributed )




(a) Effect of Skewing on Parity Striping. ($p_r$=0.25)

(b) Effect of Skewing on Parity Striping. ($p_r$=0.75)

Figure 9: Sensitivity of Parity Striping to the Skew in Access Patterns. ($p_s = 0.3$, Request Size: Quasi-Geometrically Distributed, $\sigma = p = 0.7$)

From the above discussions, it is clear that Parity Striping is advantageous only for a certain range of parameter values. Figure 10 shows the parameter ranges where one architecture may provide a higher throughput than the other. For example, if an application has a mean request size of 2 blocks and has a skew degree of one, then Figure 10 suggests that Parity Striping is better than RAID 5. On the other hand, if the access pattern has a skew degree of one and a mean request size less than 1.7, then RAID 5 may support a higher throughput than Parity Striping. Again, the request size is assumed to be quasi-geometrically distributed; the parameter $p$ is fixed to 0.7; and the various values for the mean request size are obtained by varying the single block probability $\sigma$.

# 7 Summary

In this paper, we examined the behavior of two cost effective disk array architectures, RAID 5 and Parity Striping. We first identified the "write synchronization problem" that exist with these architectures, which has been ignored in previous performance studies. We then proposed two synchronized scheduling policies suitable for both RAID 5 and Parity Striping. These two policies, we believe, provide a practical way for the implementation of RAID 5 and Parity Striping. A priority queueing model is developed for the analyses of the RAID 5 and Parity Striping architectures coupled with the two scheduling policies.

The performance of these two disk array architectures are compared with each other under different workloads, including single block accesses, multiple block accesses, and skewed accesses. The performance metrics of most interest are the mean I/O response time and the maximum

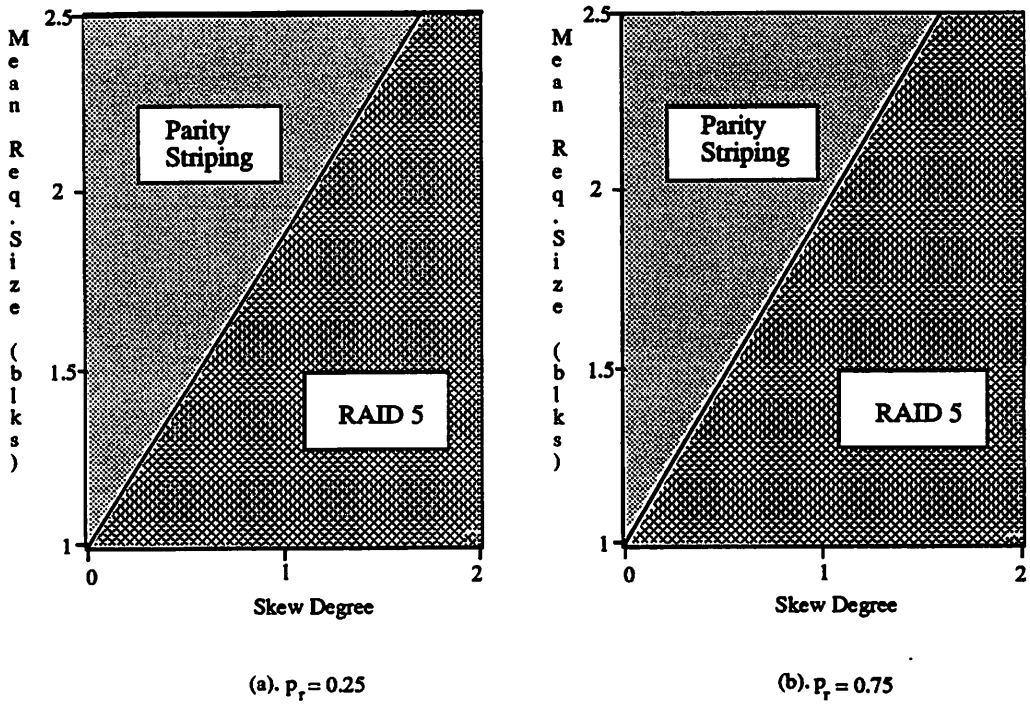



Figure 10: Favored Areas for the RAID 5 and Parity Striping Architectures. ($p_s = 0.3$, Request Size: Quasi-Geometric with $p = 0.7$)

throughput (req/sec) supported by the two architectures. As a result, RAID 5 and Parity Striping are equivalent when the access pattern is unskewed and I/O requests access only a single block of data. When the mean request size increases, the performance of RAID 5 degrades rapidly while the performance of Parity Striping degrades more slowly. When the access pattern is skewed, the performance of RAID 5 is not affected because of the load balancing property inherent in the RAID 5 architecture. However, the Parity Striping architecture is sensitive to the skewed accesses. Based on these results, we conclude that RAID 5 is most useful for those applications where most of the requests access only one block of data, no matter whether the access pattern is skewed. On the other hand, Parity Striping is suitable for those applications where most of the requests access more than one block of data and each data file is likely to be accessed with the same probability. These results also suggest that if an array is implemented as a RAID 5, then the block size, which is the unit of interleaving, should be large enough to cover the size of a large fraction of requests for the application. If the access pattern is skewed, the increase of block size should be limited. The best performance can only be achieved by carefully tuning up the block size.

# Appendix

In this Appendix, we present the moments for service times of low and high priority requests, $Y_{rw}$ and $Y_p$, which are used in calculating the moments of queueing delays $Q_{rw}$ and $Q_p$ in the text.

$$
\begin{aligned}
\overline{Y_{rw}} &= p_r\left(\overline{X}+\alpha\right)+p_w\left(\overline{X}+2\alpha+R_{max}\right)\\
\overline{Y_{rw}^2} &= p_r\left(\overline{X^2}+2\alpha\overline{X}+\alpha^2\right)+p_w\left(\overline{X^2}+2(2\alpha+R_{max})\overline{X}+(2\alpha+R_{max})^2\right)\\
\overline{Y_{rw}^3} &= p_r\left(\overline{X^3}+3\alpha\overline{X^2}+3\alpha^2\overline{X}+\alpha^3\right)+\\
&\quad p_w\left(\overline{X^3}+3(2\alpha+R_{max})\overline{X^2}+3(2\alpha+R_{mx})^2\overline{X}+(2\alpha+R_{max})^3\right).
\end{aligned}
$$

where

$$
\begin{aligned}
\overline{X} &= \overline{S}+\overline{R}\\
\overline{X^2} &= \overline{S^2}+2\overline{S}\,\overline{R}+\overline{R^2}\\
\overline{X^3} &= \overline{S^3}+3\overline{S^2}\,\overline{R}+3\overline{S}\,\overline{R^2}+\overline{R^3}
\end{aligned}
$$

and the moments for seek time $S$ and rotational latency $R$ can be easily calculated from their distributions provided in equation (2) and (3)

For $Y_p$, we distinguish two cases:

**Lower Bound on $Y_p$:**

$$
\begin{aligned}
\overline{L_p} &= \overline{X}+2\alpha+R_{max}\\
\overline{L_p^2} &= \overline{X^2}+2(2\alpha+R_{max})\overline{X}+(2\alpha+R_{max})^2\\
\overline{L_p^3} &= \overline{X^3}+3(2\alpha+R_{max})\overline{X^2}+3(2\alpha+R_{mx})^2\overline{X}+(2\alpha+R_{max})^3.
\end{aligned}
$$

**Upper Bound on $Y_p$:**

$$
\begin{aligned}
\overline{U_p} &= \overline{X}+2\alpha+R_{max}+R_{max}\int_0^{S_{max}} f_X(x)\left[1-F_{X'}(x+R_{max})\right]\,dx\\
\overline{U_p^2} &= \overline{X^2}+2(2\alpha+R_{max})\overline{X}+(2\alpha+R_{max})^2+2R_{max}\int_0^{S_{max}} x f_X(x)\,dx+\\
&\quad \left(4\alpha R_{max}+3R_{max}^2\right)F_{X'}(S_{max})-2R_{max}\int_0^{S_{max}} x f_X(x)F_{X'}(x+R_{max})\,dx-\\
&\quad \left((2\alpha+2R_{max})^2-(2\alpha+R_{max})^2\right)\int_0^{S_{max}} f_X(x)F_{X'}(x+R_{max})\,dx\\
\overline{U_p^3} &= \overline{X^3}+3(2\alpha+R_{max})\overline{X^2}+3(2\alpha+R_{max})^2\overline{X}+(2\alpha+R_{max})^3+\\
&\quad 3R_{max}\int_0^{S_{max}} x^2 f_X(x)\,dx+3\left(4\alpha R_{max}+3R_{max}^2\right)\int_0^{S_{max}} x f_X(x)\,dx+\\
&\quad \left((2\alpha+2R_{max})^3-(2\alpha+R_{mx})^3\right)F_{X'}(S_{max})-3R_{max}\int_0^{S_{max}} x^2 f_X(x)F_{X'}(x+R_{max})\,dx-\\
&\quad 3R_{max}\left(4\alpha+3R_{max}\right)\int_0^{S_{max}} x f_X(x)F_{X'}(x+R_{max})\,dx-\\
&\quad \left((2\alpha+2R_{max})^3-(2\alpha+R_{max})^3\right)\int_0^{S_{max}} f_X(x)F_{X'}(x+R_{max})\,dx.
\end{aligned}
$$

As indicated in Section 4.1, the *pdf* of $X$, $f_X(x)$, is expressed by 4th-order polynomial functions. Thus, since $X =_{st} X'$, the $CDF$ $F_{X'}(x)$ is expressed by 5th-order polynomials, and therefore all of the above integrals can be expressed in closed form.

## Acknowledgments:

The authors are gratefully acknowledge James F. Kurose for his contributions to this work, especially for the discussions on the ideas of this study.

## References

[1] D. L. Bultman, "High Performance SCSI Using Parallel Drive Technology," *Proc. BUSCON Conf.*, Anaheim, CA, pp. 40-44, Feb. 1988.

[2] S. Chen and D. Towsley, "A Queueing Analysis of RAID Architectures," *COINS Tech Report TR-91-71*, Univ. of Massachusetts, Oct. 1991.

[3] P. Chen, G. Gibson, R. H. Katz, and D. A. Patterson, "An Evaluation of Redundant Arrays of Disks using an Amdahl 5890," *Perf. Eval. Rev.*, Vol.18, No.1, pp. 74-85, May 1990.

[4] P. Chen, and D. A. Patterson, "Maximize Performance in a Stripped Disk Array," *Proc. Computer Architecture*, pp. 322-331, June 1990.

[5] G. Copeland and T. Keller, "A Comparison of High-Availability Media Algorithms," *Proc. ACM SIGMOD Conf.*, Portland, OR, pp. 98-109, May 1989.

[6] D. R. Cox, "A Use of Complex Probabilities in Theory of Stochastic Processes," *Proc. Combridge Phil. Soc.*, 51, pp. 313-319, 1955.

[7] G. A. Gibson, "Redundant Disk Arrays: Reliable, Parallel Secondary Storage," *PhD Dissertation*, Univ. of California/Berkeley, April, 1991.

[8] J. Gray, B. Host, and M. Walker, "Parity Striping of Disk Arrays: Low-Cost Reliable Storage with Acceptable Throughput," *Proc. 16th VLDB Conf.*, Brisbane, Australia, pp. 148-161, 1990.

[9] R. H. Katz, G. Gibson, and D. A. Patterson, "Disk System Architectures for High Performance Computing," *Proc. of the IEEE*, Vol.77, No.12, pp. 1842-1858, Dec. 1989.

[10] A. Y. Khinchine, *Mathematical Methods in the Theory of Queueing*, New York: Hafner Publishing Co., 1960.

[11] M. Y. Kim, "Synchronized Disk Interleaving," *IEEE Trans. Comp.*, Vol. C-35, pp. 978-988, Nov. 1986.

[12] M. Y. Kim and A. N. Tantawi, "Asynchronized Disk Interleaving: Approximating Access Delays," *IEEE Trans. Comp.*, Vol.40, No.7, pp. 801-810, July 1991.

[13] L. Kleinrock, *Queueing Systems* Vol. I, John Wiley & Sons, 1975.

[14] L. Kleinrock, *Queueing Systems* Vol. II, John Wiley & Sons, 1976.

[15] E. K. Lee and R. H. Katz, "Performance Consequences of Parity Placement in Disk Arrays," *Proc 4th Int'l Conf. on Archi. Supp. for Prog. Lang. and OS*, pp. 190-199, April, 1991.

[16] M. Livny, S. Khoshafian, and H. Boral, "Multi-Disk Management Algorithm," *Proc. SIGMETRICS*, pp. 69-77, May 1987.

[17] W. C. Lynch, "Do Disk Arms Move?" *Perform. Eval. Rev.*, Vol.1, pp. 3-16, Dec. 1972.

[18] J. Menon and D. Mattson, "Performance of Disk Arrays in Transaction Processing Environments," *IBM Tech. Report RJ 8230 (75424)*, July 15, 1991.

[19] A. Merchant and P. Yu, "Modeling and Comparisons of Striping Strategies in Data Replicated Architectures," *IBM Tech. Report*, Sept. 1991.

[20] R. R. Muntz and J. C. Liu, "Performance Analysis of Disk Arrays under Failure," *Proc. 16th VLDB Conf.*, Brisbane, Australia, pp. 162-173, 1990.

[21] M. Nelson, B. Welch, and J. Ousterhout, "Caching in the Sprite Network File System," *ACM Trans. on Comp. Sys.*, Vol.6, No.1, pp. 134-154, Feb. 1988.

[22] S. Ng, D. Lang, and R. Selinger, "Trade-offs Between Devices and Paths in Achieving Disk Interleaving," *Proc. 15th Int'l Sym. on Comp. Archit.*, Hawaii, pp. 196-201, May 1988.

[23] S. Ng, "Some Design Issues of Disk Array," *IEEE Spring ComCon,*, pp. 137-142, 1989.

[24] S. Ng, "Improving Disk Performance via Latency Reduction," *IEEE Trans. on Computers,*, Vol.40, No.1, pp. 22-30, Jan. 1991.

[25] M. Ogata and M. Flynn, "A Queueing Analysis for Disk Array Systems," *Stanford Tech. Rep. CSL-TR-90-443*, Aug. 1990.

[26] D. A. Patterson, G. Gibson, and R. H. Katz, "A Case for Redundant Arrays of Inexpensive Disks (RAID)," *Proc. ACM SIGMOD Conf.*, pp. 109-116, July, 1988.

[27] D. A. Patterson, P. Chen, G. Gibson, and R. H. Katz, "Introduction to Redundant Arrays of Inexpensive Disks (RAID)," *IEEE Spring ComCon,*, pp. 112-117, 1989.

[28] A. Poston, "A High Performance File System for UNIX," *USENIX Proc.*, pp. 215-226, Sept. 1988.

[29] A. L. N. Reedy and P. Banerjee, "An Evaluation of Multiple-Disk I/O Systems," *IEEE Trans. Comp.*, Vol.38, No.12, pp. 1680-1690, Dec. 1989.

[30] K. Salem and H. Garcia-Molina, "Disk Striping," *Proc. IEEE Data Engineering Conf.*, Los Angeles, CA, pp. 336-342, Feb. 1986.

[31] M. Schroeder, D. Gifford, and R. Needham, "A Caching File System for a Programmer's Workstation," *Proc. 10th Sym. on Operating System Principles*, pp. 25-34, Dec. 1985.

[32] M. Schulze, G. Gibson, R. Katz, D. Patterson, "How Reliable is a RAID?" *Proc. IEEE Spring ComCon Conf*, San Francisco, CA, pp. 118-123, Feb. 1989.

[33] M. Stonebraker and G. A. Schloss, "Distributed RAID - a New Multiple Copy Algorithm," *Proc. 6th Int'l. Conf. on Data Engineering*, pp. 430-437, Feb., 1990.