A Response Time Distribution Model for Zoned RAID

Abigail S. Lebrecht*, Nicholas J. Dingle, and William J. Knottenbelt

Department of Computing, Imperial College London,
South Kensington Campus, SW7 2AZ, United Kingdom
{asl102,njd200,wjk}@doc.ic.ac.uk

Abstract. RAID systems are widely deployed, both as standalone storage solutions and as the building blocks of modern virtualised storage platforms. An accurate model of RAID system performance is therefore critical to understanding storage system performance. To this end, this paper presents a queueing network-based model of RAID systems comprised of zoned disks and operating at RAID level 0-1 or 5. The contribution over previous work is twofold. Firstly, our analysis approximates full I/O request response time distributions rather than just mean values. This provides the ability to reason about response time quantiles and higher moments of response time – both of which are useful in the context of modern quality of service requirements. Secondly, we validate our model against measurements from a real RAID system rather than a software simulation. The close agreement between predicted and observed response time distributions gives a high level of confidence in the validity of our model.

1 Introduction

There is an unrelenting business demand for fast, reliable storage. For example, the CIO of Chevron Corporation, the world's fifth largest energy company, claims that Chevron accumulates data at a rate of 2TB per day [1]. Similarly, the news agency Reuters estimates that it manages 1.4PB of data, a figure which is growing at an annual rate of 50%.

Much of this data is ultimately stored on RAID systems, which are deployed either as standalone storage solutions or as the building blocks of virtualised storage infrastructures. The detailed understanding of RAID system performance is therefore critical to determining whether or not application-level quality of service demands will be met by a given storage infrastructure.

* Corresponding author. Telephone: +44 20 7594 8385.

In this paper, we propose a novel queueing model for the analysis of RAID systems comprised of zoned disks[1]. The inputs to this analysis are a specified I/O request arrival rate, an I/O request access profile, a given RAID configuration and physical disk parameters. The primary output of this analysis is an approximation to the cumulative distribution function of I/O request response time. From this, it is straightforward to calculate response time quantiles, as well as the mean, variance and higher moments of I/O request response time. This improves on the state-of-the-art in RAID performance models [2–6], all of which yield approximations to the mean response time only.

The first step in building a good RAID model is to derive an accurate model of the behaviour of a single disk drive. To do this, we abstract a disk drive as an M/G/1 queue and model the service time as the sum of the random variables of seek time, rotational latency and data transfer time. In doing so, we take into account the properties of zoned disks. Section 2 describes this zoned disk drive queueing model in detail.

In line with previous work [7], we then abstract a RAID system as a fork-join queueing network. This comprises several queues, each of which represents one disk drive in the array. Incoming I/O requests *fork* into subtasks which are processed across the drives; upon completion of all subtasks, they *join*, signalling completion of the I/O request. Section 3 presents our analytical approximation of such a fork-join queueing network.

We tailor our basic fork-join approximation to account for the I/O request patterns associated with particular request types and request sizes under different RAID levels. We focus on RAID 0-1 (mirrored stripes) and RAID 5 (distributed parity), which are the two most commonly used RAID levels. Section 4 derives the resulting novel cumulative distribution functions of I/O request response time for RAID 0-1 and 5. Models for RAID 0 (striping without redundancy) and RAID 6 (double distributed parity) can also be derived from these results.

To test the accuracy of our resulting models, we validate them against RAID device measurements in Section 5. Section 6 concludes and considers directions for future work.

[1] On modern hard drives there are more blocks on cylinders on the outside of the platter than those closer to the centre. Cylinders with the same number of blocks are grouped together in zones. Disks rotate with a constant angular velocity and therefore data throughput is higher for outer zones than for inner ones.

2 Disk Model

In making performance predictions for a disk array or storage system, it is fundamental to model disk service time accurately. To this end, we model a disk drive as an M/G/1 queue where the service time density is the convolution of seek time, rotational latency and data transfer time densities. Defining random variables for seek time, S, rotational latency, R, and block transfer time, T, we describe their distributions below.

2.1 Seek Time

A seek, S, is the time taken for the disk head to move from the cylinder where it is currently located, C_2, to the cylinder containing a target sector, C_1. We define a random variable, $D = |C_1 - C_2|$, as the seek distance. Seek time can then be defined in terms of seek distance. Specifically [8],

$$S(D) = \begin{cases} 0 & \text{if } D = 0 \\ a + b\sqrt{D} & \text{otherwise} \end{cases}$$

where a and b are constants defined in terms of the disk geometry, and are given by:

$$a = \frac{minseek\ \sqrt{Cyls - 1} - maxseek}{\sqrt{Cyls - 1} - 1}$$
$$b = \frac{maxseek - minseek}{\sqrt{Cyls - 1} - 1}$$

Here *Cyls* is the total number of cylinders on the disk, *minseek* is the track-to-track seek time and *maxseek* is the full-stroke seek time.

The disk model must reflect the layout of a zoned disk accurately. As cylinders get closer to the disk edge, their circumference increases and the number of sectors per cylinder increases. Therefore, a random request has an increased probability of being directed to a sector on an outer cylinder. Let C be a random variable representing the cylinder number of a randomly selected disk sector. Then the probability distribution of C can be approximated by assuming that the number of sectors per track increases linearly [9]. That is,

$$f_C(x) = \frac{\alpha + \beta x}{\gamma} \qquad x = 0, 1, \ldots, Cyls - 1$$

with constants α, β and γ defined as:

$$\alpha = \frac{SEC[0]}{spb}$$
$$\beta = \frac{SEC[Cyls-1] - SEC[0]}{(Cyls-1)\ spb}$$
$$\gamma = \alpha(Cyls-1) + \frac{\beta}{2}(Cyls-1)^2$$

where $SEC[0]$ and $SEC[Cyls-1]$ are the number of sectors on the innermost and outermost tracks respectively and *spb* is the number of physical sectors per logical block. α represents the number of logical blocks on the innermost track and β charts the rate of increase in blocks per cylinder.

The probability density function (pdf) of seek distance is calculated by assuming the two random variables, C_1 and C_2, as two distinct cylinder numbers, and calculating the seek distance between all possible cylinder numbers. This is split into two terms, one for the case when $C_1 \leq C_2$ and one for the case where $C_1 > C_2$:

$$f_D(x) = \int_0^{Cyls-1-x} f_C(y) f_C(x+y) dy + \int_x^{Cyls-1} f_C(y) f_C(y-x) dy \quad (1)$$

The full expansion of Equation (1) appears in [9]. The cumulative distribution function (cdf) of seek time, $F_S(t)$, can be defined in terms of the cdf of $f_D(x)$, $F_D(x)$, as [8]:

$$F_S(t) = F_D\left(\left(\frac{t-a}{b}\right)^2\right)$$

2.2 Rotational Latency

Rotational latency, R, is the time to rotate to the angle of a target sector. R has a uniform distribution with a range between 0 and the time for a full disk revolution, R_{max} [3].

2.3 Data Transfer Time

The time to transfer k logical blocks on cylinder x of a zoned disk can be approximated as [9]:

$$t(x) = \frac{k\ spb\ R_{max}}{\alpha + \beta x}$$

Denoting T_k as the random variable of the time to transfer k blocks of data, its cdf is:

$$\begin{aligned} F_{T_k}(t) &= \int P(T_k \leq t \mid C = x) f_C(x) dx \\ &= \int_{\max(\phi_k(t),0)}^{Cyls-1} f_C(x) dx \end{aligned} \tag{2}$$

where

$$\phi_k(t) = \frac{k\ spb\ R_{max}}{\beta t} - \frac{\alpha}{\beta}$$

calculates the minimum cylinder number it is possible to transfer k logical blocks of data in less than t ms. The solution of the integral in Equation (2) is a function of t with a domain bounded between the minimum and maximum possible k-block transfer times. If t_{min} and t_{max} are the times to transfer to a single physical sector on the outermost and innermost tracks respectively, Equation (2) expands to:

$$F_{T_k}(t) = \begin{cases} 0 & \text{if } t < k\ spb\ t_{min} \\ \frac{1}{2(t_{max}-t_{min})^2\gamma}\left(p + \frac{q}{t} + \frac{r}{t^2}\right) & \text{if } k\ spb\ t_{min} \leq t \leq k\ spb\ t_{max} \\ 1 & \text{otherwise} \end{cases} \tag{3}$$

with

$$\begin{aligned} p &= (Cyls-1)t_{max}(2(t_{max}-t_{min})\alpha + (Cyls-1)(t_{max}-2t_{min})\beta) \\ q &= ((Cyls-1)t_{max}(-2\ k\ spb(t_{max}-t_{min})t_{min}\alpha + (Cyls-1)k\ spb\ t_{max}t_{min}\beta \\ &\quad +(1-Cyls)k\ spb(t_{max}-2t_{min})t_{min}\beta)) \\ r &= (1-Cyls)(Cyls-1)k^2 spb^2 t_{max}^2 t_{min}^2 \beta \end{aligned}$$

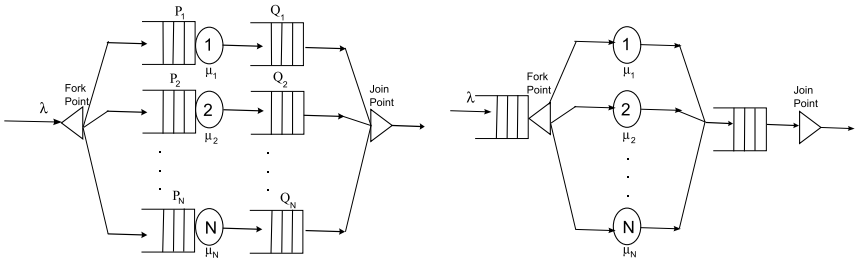
The Laplace transform of the disk's response time pdf is derived using the Pollaczek-Khintchine transform equation for M/G/1 queues [10]:

$$W^*(\theta) = \frac{(1-\rho)\theta X^*(\theta)}{\lambda X^*(\theta) - \lambda + \theta}$$

Here $X^*(\theta)$ is the Laplace transform of the service time pdf, which in our case is the product of the Laplace transforms of the pdfs of S, R and T, i.e. $S^*(\theta)R^*(\theta)T_k^*(\theta)$. Also, $\rho = \frac{\lambda}{\mu}$, where λ is the I/O request arrival rate to the disk and μ is the mean service rate, which in our case is given by $\frac{1}{E[R]+E[S]+E[T_k]}$. As $W^*(\theta)$ is unlikely to have an analytical inversion, we invert it numerically using the Euler method [11] to obtain the response time pdf $f_W(t)$. The cumulative distribution function $W(t)$ is also easily obtained by inverting $W^*(\theta)/\theta$.

3 The Fork-Join Queue

Fork-join queues have been widely employed as an appropriate queueing abstraction of the operation of disk arrays [2]. Given N queues, (see Figure 1(a)), each incoming job is split into N subtasks at the fork point. Each of these subtasks queues for service at a parallel service node before joining a queue for the join point. When all N subtasks in the job are at the head of their respective join queues, they rejoin (synchronise) at the join point.



(a) Fork-join queueing model (b) Split-merge queueing model

Fig. 1. Fork-join vs. split-merge queueing models

It is difficult to model job response times in a fork-join synchronisation analytically. Indeed, to date, exact analytical results exist only for the mean response time of a two server system consisting of homogeneous M/M/1 queues [12]. Approximate results for mean response times for M/M/1 and M/G/1 fork-join queues are more abundant [12–15, 5]. However, they all have limitations in the context of our present application. In particular, none of these results have yet been extended to find higher moments or full response time distributions. Furthermore, some are not applicable to M/G/1 queues (necessary to support our disk service time model) or heterogeneous servers (necessary to support the modelling of heterogeneous disks), and some can be very computationally intensive for a large number of queues (necessary when modelling very large disk arrays). Our present work addresses these issues.

Our approach is inspired by Harrison and Zertal's method for approximating the mean of the maximum of multiple random variables [16]. This gives an approximation to a fork-join synchronisation by assuming it to be a similar queueing network, the split-merge queue [17] (see Figure 1(b)).

In the split-merge queue, a job splits into N subtasks which are serviced in parallel. Only when all the subtasks finish servicing and rejoin can the next job split into subtasks and start servicing. This will lead to a slower mean response time than its fork-join equivalent.

An exact solution for the cumulative distribution function of job response time in a split-merge queue can be found by utilising properties of Order Statistics [18, 19]. Any random variables, $X_1, X_2, \ldots, X_n$ can be reordered as $X_{(1)}, X_{(2)}, \ldots, X_{(n)}$, where $X_{(1)} \leq X_{(2)} \leq \ldots \leq X_{(n)}$. Then $X_{(1)}, X_{(2)}, \ldots, X_{(n)}$ are the order statistics of $X_1, X_2, \ldots, X_n$.

The cdf of $X_{(n)}$, the maximum order statistic (corresponding to job response time in a split-merge queue), is calculated exactly as:

$$\begin{aligned} F_{X_{(n)}}(x) &= P(X_{(n)} \leq x) \\ &= P(X_{(i)} \leq x \ \forall i) \end{aligned}$$

Thus, if $X_1, X_2, \ldots, X_n$ are independent with cdfs $F_i(x)$,

$$F_{X_{(n)}}(x) = \prod_{i=1}^{n} F_i(x) \tag{4}$$

Applying this to a disk array, consider an n-block I/O request sent to an array of n homogeneous disks. If each disk processes a 1-block request and has a response time cdf of $W(t)$, then the approximate response time cdf of the I/O request is $(W(t))^n$. The next section generalises this to deal with variable size I/O requests and I/O request sizes under various RAID levels.

4 RAID Model

The results derived in Equation (4) above suffice to calculate the response time cdf for read or write requests to an n-disk RAID 0 system in which each request consists of a multiple of n blocks. However, not every I/O request leads to an access to all disks, being influenced by I/O request size and type, and also by RAID level. Below we deal with RAID levels 0-1 and 5, for both read and write requests.

Our model is designed to accept a homogeneous stream of I/O requests of a given size and type. We further assume that all the service time distributions on all disks are identically distributed. For the sake of notational simplicity, let $W(t, \gamma, \mu)$ define the cdf of the response time distribution of a single M/G/1 queue (disk), γ is the arrival rate at an individual disk and μ is the mean service rate. We assume there are n disks in the array and that the arrival rate of logical I/O requests to the disk array as a whole is λ.

4.1 RAID 0-1

Read Requests Assuming an efficient RAID controller, a b-block read on RAID 0-1 can read data from either primary or mirror disks. With $b \geq n$, we thus utilise all n disks of the array (and not $\frac{n}{2}$ disks) to give better performance results for medium and large sized requests. However, if $b < n$ only b disks are utilised at any time. To account for this, we view the system as a b-queue fork-join queue. The arrival rate at the disks needs to be modified since each request only arrives at b of the n disks.

Therefore the cdf of the response time distribution for a read on a RAID 0-1 system is:

$$\begin{array}{l} \left(W\left(t, \frac{\lambda b}{n}, \frac{1}{E[R]+E[S]+E[T_1]}\right)\right)^b \text{ if } b < n \\ \left(W\left(t, \lambda, \frac{1}{E[R]+E[S]+E[T_{\frac{b}{n}}]}\right)\right)^n \text{ otherwise} \end{array}$$

Write Requests A b-block write must account for each request being written on both a primary and mirror disks. The corresponding response time cdf is defined as:

$$\begin{array}{l} \left(W\left(t, \frac{2\lambda b}{n}, \frac{1}{E[R]+E[S]+E[T_1]}\right)\right)^{2b} \text{ if } 2b < n \\ \left(W\left(t, \lambda, \frac{1}{E[R]+E[S]+E[T_{\frac{2b}{n}}]}\right)\right)^n \text{ otherwise} \end{array}$$

4.2 RAID 5

Read Requests A read request under RAID 5 is modelled in the same way as the equivalent read request under RAID 0-1. Note that, since RAID 5 distributes data (and parity) across all disks, a $b \geq n$ read request will access all n disks, despite the stripe size of $n-1$ disks. The response time cdf is:

$$\begin{array}{l} \left(W\left(t, \frac{\lambda b}{n}, \frac{1}{E[R]+E[S]+E[T_1]}\right)\right)^b \text{ if } b < n \\ \left(W\left(t, \lambda, \frac{1}{E[R]+E[S]+E[T_{\frac{b}{n}}]}\right)\right)^n \text{ otherwise} \end{array}$$

Write Requests The behaviour of a RAID 5 write depends on the size of the request, with different methods used to update the parity.

Small Partial Stripe Write If a request consists of $b < \frac{n-1}{2}$ blocks (i.e. a small partial stripe write), then parity is calculated using [20]:

$$new\_parity = new\_data \oplus old\_data \oplus old\_parity$$

where $\oplus$ is the exclusive-or (XOR) operator. This is a read-modify-write operation. Each of the b blocks and parity must be transferred twice, first to read the old data and parity, then to write the new data and parity. When the old data and parity have been read from all disks, a new request will be issued to write the new data and parity to the same disks. This request is given priority in the queue, so at least one disk (the last to complete the pre-read) will just have completed reading a data or parity block that now needs to be re-written. Therefore we add a full disk rotation into the service time distribution. However, it is likely that by the time the last disk has completed its pre-read, the remaining disks will have started servicing the next I/O request in their queues. These disks will need to re-seek to write the new data and parity. Therefore, we assume that b disks seek again on the second request, while one disk needs a complete rotation only.

The request to pre-read will have a mean service time of $E[R]+E[S]+E[T_1]$. The arrival rate at each of $b+1$ disks for both requests (i.e. the pre-read and data transfer operations) is $\lambda(b+1)/n$. Combining both arrival streams, we approximate the cdf of the response time as:

$$\left(W\left(\frac{t}{2}, \frac{2\lambda(b+1)}{n}, \frac{1}{\frac{(2b+1)(E[R]+E[S])+R_{max}}{2(b+1)} + E[T_1]} \right) \right)^{b+1}$$

The mean service time in the above is calculated by averaging the mean services times of the pre-read and data transfer operations. Thus, the pdfs of seek time and rotational latency are altered to:

$$f'(t) = \begin{cases} \frac{1}{2(b+1)} & \text{if } x = 0 \\ \frac{2b+1}{2(b+1)} f(t) & \text{otherwise} \end{cases}$$

where $f(t)$ represents the probability density function of seek time or rotational latency.

Large Partial Stripe Write If $\frac{n-1}{2} \leq b < n-1$ (i.e. a large partial stripe write), then to minimise disk accesses the parity is calculated by reading only from the disks that are not being written to. The new parity is calculated by XOR-ing the data that will be written with the data from the disks that will remain unchanged. This is a reconstruct-write operation.

The first request pre-reads $n-1-b$ blocks of data for the calculation of the new parity. When all $n-1-b$ disks complete their pre-read, a new request is sent to the other $b+1$ disks to write the new data and parity. The arrival rate for the pre-read will be $\lambda(n-1-b)/n$, and we compute the time to complete this phase as the slowest of the $n-1-b$ queues. The arrival rate of the write request will be $\lambda(b+1)/n$ to $b+1$ queues. Both requests will have the same mean service time of $E[R]+E[S]+E[T_1]$. Averaging the number of queues we are finding the maximum of $(n/2)$, we approximate the response time cdf of the two requests required (pre-read and data transfer) as:

$$\left(W\left(\frac{t}{2},\lambda,\frac{1}{E[R]+E[S]+E[T_1]}\right)\right)^{n/2}$$

Full Stripe Write If a request consists of a number of complete stripes (i.e. $b \bmod (n-1)=0$), no pre-reads are needed to calculate the parity. All the disks are utilised, with either the new data block or the new parity block written to each disk. The response time cdf is:

$$\left(W\left(t,\lambda,\frac{1}{E[R]+E[S]+E[T_{\frac{b}{n-1}}]}\right)\right)^{n}$$

Full Stripe followed by Small Partial Stripe Write If $b > n-1$ and $0 < b \bmod (n-1) < \frac{n-1}{2}$, at least one full stripe write will occur followed by a small partial stripe write. Let $k=\lfloor\frac{b}{n-1}\rfloor$ and $b_{mod}=b \bmod (n-1)$. We assume that there are two types of requests to be averaged. The first request involves k full stripe writes, followed by a parity pre-read to $b_{mod}+1$ disks. The second request writes the new data and parity to $b_{mod}+1$ disks. The response time cdf is then approximated as:

$$\left(W\left(\frac{t}{2},\frac{\lambda(n+b_{mod}+1)}{n},\frac{1}{\frac{(n+b_{mod})(E[R]+E[S])+R_{max}}{n+b_{mod}+1}+E[T_{\frac{k}{2}+\frac{b_{mod}+1}{n}}]}\right)\right)^{\frac{n+b_{mod}+1}{2}}$$

Full Stripe followed by Large Partial Stripe Write If $\frac{n-1}{2} \leq b \bmod (n-1) < n-1$, at least one full stripe write will occur followed by a large partial stripe write. The initial request will be to write k blocks to all disks and then pre-read an additional block on $n-b_{mod}-1$ disks. The second request, issued upon the completion of the first, writes the new data and parity to the remaining $b_{mod}+1$ disks. If one of the $n-b_{mod}-1$ pre-reading disks complete service last, then all the disks to be written to will

need to seek to write the new data. However, it is possible that, despite the smaller mean service time, one of the $b_{mod}+1$ disks will complete last. This disk will not need to seek or wait for rotation at all, and will be in position to write the new data immediately. To account for these possibilities, we assume that $b_{mod}+0.5$ disks will need to seek again in the second request. The cdf is:

$$\left(W\left(\frac{t}{2}, \frac{\lambda(n+b_{mod}+1)}{n}, \frac{1}{\frac{(n+b_{mod}+0.5)(E[R]+E[S])}{2n}+E[T_{\frac{k+1}{2}}]}\right)\right)^{\frac{n+b_{mod}+1}{2}}$$

5 Validation

To provide confidence in our analytical models, we validate them against device measurements. Our experimental platform consists of an Infortrend A16F-G2430 RAID system containing 4 Seagate ST3500630NS disks. Each disk has 60801 cylinders. A sector is 512 bytes and we have derived from measurement that the time to write a single physical sector on the innermost and outermost tracks are 0.012064ms (t_{max}) and 0.005976ms (t_{min}) respectively. The stripe width on the array is configured as 128KB, which we define as the block size. Therefore there are 256 sectors per block. The time for a full disk revolution is 8.33ms. A track to track seek takes 0.8ms and a full-stroke seek requires 17ms for a read request; the same measurements are 1ms and 18ms respectively for write requests [21].

To obtain response time measurements, we implemented a benchmarking program that issues read and write requests using a master process and a number of child processes. The master process constructs a list of arrival times according to a specified distribution (here a negative exponential distribution to conform to the model assumptions) and then monitors the system clock until the first generated arrival time occurs. When it does, a child process is spawned which performs the read/write operation and measures the time taken. This leaves the master process free to spawn further child processes at the calculated arrival times without the need for it to wait for previously-issued operations to complete.

Throughout, it was necessary to minimise the effects of buffering and caching as these are not represented in the model. We therefore disabled the write-back cache on the RAID system and set the read-ahead buffer to 0KB. Furthermore, devices are opened with the `O_DIRECT` flag set. For each of the experiments presented below, 100 000 requests were issued with an arrival rate of $\lambda = 0.02$ requests per millisecond to random logical locations. The resulting cumulative distribution functions of the response times were calculated using the statistical package `R`.

5.1 Data Transfer Model

The data transfer model in Equation (3) can be validated by setting the number of sectors to write to a large enough number that the seek and rotation time will be insignificant in comparison to the transfer time. We thus write 100MB in each transfer to a single ST3500630NS disk connected directly to a separate test machine. This ensures that any overheads imposed by the RAID controller are bypassed. We also ensure that no queueing occurs by waiting until a request completes before issuing another. Figure 2 compares the cumulative distribution function of the analytical data transfer time model with device measurements. The effects of disk zoning are clearly evident in the measurements, which are a close match to the analytical model.

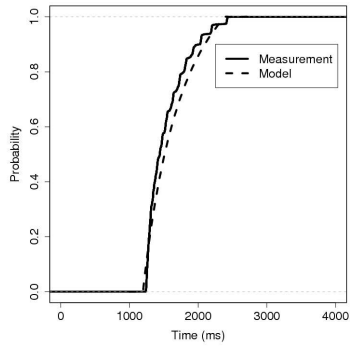


Fig. 2. Data transfer time of 100MB requests on a Seagate ST3500630NS disk, compared with analytical model of zoned data transfer time

5.2 RAID 0-1

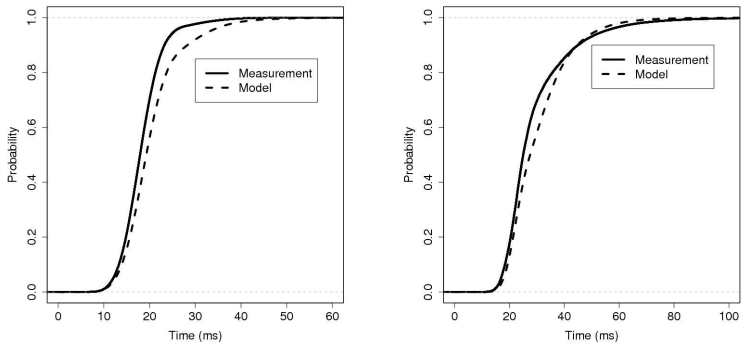
Figure 3(a) displays the measured and modelled cdfs for read requests on a four disk RAID 0-1 system, and Figure 3(b) shows the corresponding cdf for write requests. We observe good agreement between model and measurement. Table 1 further illustrates the accuracy of the model, comparing mean and variance for the model and measured results.

5.3 RAID 5

The analytical model assumes that all writes begin with a full stripe write, or, if the request is for fewer blocks than a full stripe, then all blocks in

		Blocks	Mean measured (ms)	Mean model (ms)	Variance measured (ms^2)	Variance model (ms^2)
RAID 0-1	read	2	18.3	20.4	20.3	40.1
	write	2	29.0	30.3	164.9	119.8
RAID 5	read	3	22.6	24.2	58.9	66.4
	write	1	51.9	48.4	286.2	466.8
		2	50.0	50.1	257.9	411.5
		3	31.2	30.3	186.3	119.8
		4	70.9	69.0	2445.8	975.1
		5	65.5	66.3	2256.3	848.0

Table 1. Response time mean and variance comparison for read and write requests on RAID 0-1 and RAID 5 ($\lambda = 0.02$ requests/ms)



(a) 2-block read request (b) 2-block write request

Fig. 3. Cumulative distribution functions of RAID 0-1 I/O request time on a 4 disk RAID system ($\lambda = 0.02$ requests/ms)

the request are written to the same stripe. Consequently, for the measurements presented here, the alignment of I/O request starting locations on the RAID system was constrained to ensure that this assumption held. We note that more general alignments will occur in practice; extending the model to account for this is part of our future work.

Figure 4 displays the measured and modelled cdfs of I/O request response time on a four disk RAID 5 system. Figure 4(a) compares the distributions for 3-block read requests. Figure 4(b) is a small partial stripe write and Figure 4(c) is a large partial stripe write. Figures 4(d), 4(e) and 4(f) are full stripe requests, full stripes followed by a small partial stripe, and full stripes followed by a large partial stripe, respectively.

Table 1 compares means and variances in all the above cases, and again we note good agreement between the measured and modelled results. The slight discrepancies between measured and modelled variances are caused by the model's simple abstraction of the array's behaviour. Specifically, for partial stripe request sizes, each request will not transfer to all disks in the array in a single request. To approximate this, the model transfers a fraction of a single block to each disk. This enables accurate mean response time results, but the variance suffers.

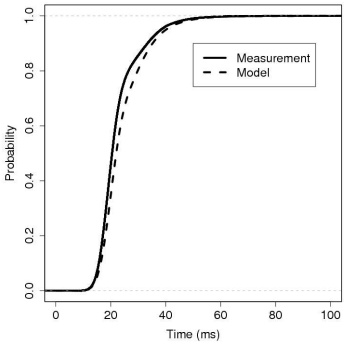
6 Conclusion and Future Work

In this paper we have developed new methods for modelling the performance of RAID systems. Our analytical queueing models enable, for the first time, the calculation of an approximation to the response time distribution of I/O request response time in these systems. These results are validated against device measurements from a real RAID system, demonstrating the accuracy of the analytical models.

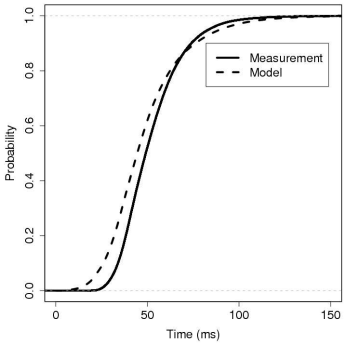
There are three ways in which we will seek to relax constraints on our model, making it more applicable in the context of real I/O workloads and systems. Firstly, caching is not yet supported in our model, and therefore in our measurements all caching (read-ahead and write-through) was disabled both on the disk and on the RAID controller. However, we appreciate the important role that caching at both these levels plays in I/O performance and we will therefore seek to incorporate it. Secondly, as discussed in Section 5.3, we currently constrain the alignment of RAID 5 write requests. In the future, we intend to extend the RAID 5 write model to describe requests that start with a partial stripe, followed by further data. Finally, we have assumed Markovian arrivals in our model, and have generated request streams that conform to this assumption for our measurements. We intend to compare the model response times with response times generated from real I/O traces. With these constraints removed, we can then extend the model to allow I/O request streams consisting of mixed request types and sizes. This will involve converting our analytical model into a multi-class queueing system.

Acknowledgements

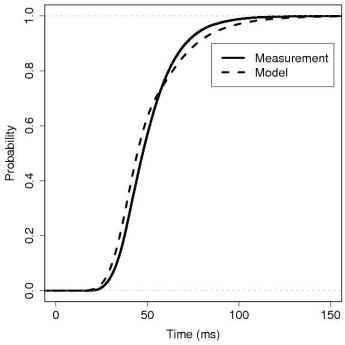
We are grateful to Peter Harrison and Soraya Zertal for helpful discussions. This work is supported by EPSRC research grant EP/F010192/1.



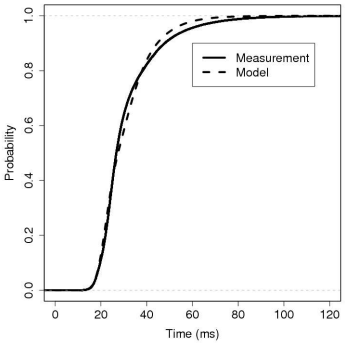
(a) 3-block read request



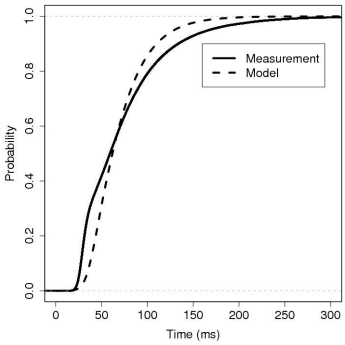
(b) 1-block write request



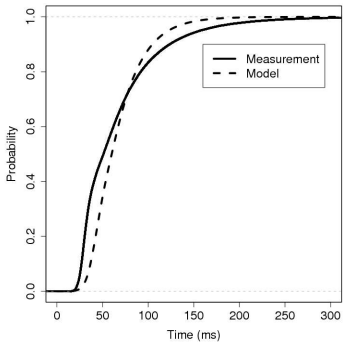
(c) 2-block write request



(d) 3-block write request



(e) 4-block write request



(f) 5-block write request

Fig. 4. Cumulative distribution functions of RAID 5 I/O request time on a 4 disk RAID system ($\lambda = 0.02$ requests/ms)

References

1. Gantz, J.F.: The expanding digital universe: A forecast of worldwide information growth through 2010. White paper, IDC (2007)
2. Lee, E.K.: Performance Modeling and Analysis of Disk Arrays. PhD thesis, University of California at Berkeley (1993)
3. Chen, S., Towsley, D.: A performance evaluation of RAID architectures. IEEE Transactions on Computers **45** (1996) 1116–1130
4. Harrison, P.G., Zertal, S.: Queueing models of RAID systems with maxima of waiting times. Performance Evaluation **64** (2007) 664–689
5. Varki, E.: Response time analysis of parallel computer and storage systems. IEEE Transactions on Parallel and Distributed Systems **12** (2001) 1146–1161
6. Varki, E., Merchant, A., Xu, J., Qiu, X.: Issues and challenges in the performance analysis of real disk arrays. IEEE Transactions on Parallel and Distributed Systems **15** (2004) 559–574
7. Chen, P.M., Lee, E.K., Gibson, G.A., Katz, R.H., Patterson, D.A.: RAID: High-Performance, Reliable Secondary Storage. ACM Computing Surveys **26** (1994) 145–185
8. Chen, S., Towsley, D.: The design and evaluation of RAID 5 and parity striping disk array architectures. IEEE Transactions on Parallel and Distributed Systems **17** (1993) 58–74
9. Zertal, S., Harrison, P.G.: Multi-RAID queueing model with zoned disks. In: High Performance Computing and Simulation Conference (HPCS'07). (2007)
10. Harrison, P.G., Patel, N.M.: Performance Modelling of Communication Networks and Computer Architectures. Addison-Wesley (1993)
11. Abate, J., Whitt, W.: The Fourier-series method for inverting transforms of probability distributions. Queueing Systems Theory and Applications **10** (1992) 5–88
12. Nelson, R., Tantawi, A.N.: Approximate analysis of fork/join synchronization in parallel queues. IEEE Transactions on Computers **37** (1988) 739–743
13. Varma, S., Makowski, A.M.: Interpolation approximations for symmetric fork-join queues. In: Proc. Performance '93. (1994) 245–265
14. Thomasian, A., Tantawi, A.N.: Approximate solutions for M/G/1 fork/join synchronization. In: Proc. WSC '94. (1994) 361–368
15. Varki, E.: Mean value technique for closed fork-join networks. In: Proc. ACM SIGMETRICS. (1999) 103–112
16. Harrison, P.G., Zertal, S.: Queueing models with maxima of service times. In: Proc. TOOLS Conference. (2003) 152–168
17. Duda, A., Czachórski, T.: Performance evaluation of fork and join synchronization primitives. Acta Informatica **24** (1987) 525–553
18. David, H.A.: Order Statistics. John Wiley and Sons, Inc (1981)
19. Lebrecht, A.S., Knottenbelt, W.J.: Response time approximations in fork-join queues. In: 23rd UK Performance Engineering Workshop (UKPEW). (2007)
20. Patterson, D.A., Gibson, G., Katz, R.H.: A case for redundant arrays of inexpensive disks (RAID). In: Proc. International Conference on Management of Data (SIGMOD). (1988)
21. Seagate: Barracuda ES Data Sheet (2007) http://www.seagate.com/docs/pdf/datasheet/ disc/ds_barracuda_es.pdf.