# Modelling and Validation of Response Times in Zoned RAID

Abigail S. Lebrecht Nicholas J. Dingle William J. Knottenbelt

Department of Computing, Imperial College London,
180 Queen's Gate, London SW7 2BZ, United Kingdom.
{asl102,njd200,wjk}@doc.ic.ac.uk

## Abstract

*We present and validate an enhanced analytical queueing network model of zoned RAID. The model focuses on RAID levels 01 and 5, and yields the distribution of I/O request response time. Whereas our previous work could only support arrival streams of I/O requests of the same type, the model presented here supports heterogeneous streams with a mixture of read and write requests. This improved realism is made possible through multiclass extensions to our existing model. When combined with priority queueing, this development also enables more accurate modelling of the way subtasks of RAID 5 write requests are scheduled. In all cases we derive analytical results for calculating not only the mean but also higher moments and the full distribution of I/O request response time. We validate our model against measurements from a real RAID system.*

## 1. Introduction

RAID systems are fundamental components of almost all modern data storage systems due to their ability to increase storage infrastructure performance and reliability in a cost-effective manner. As a result they are now widely deployed at every level from personal home storage devices to enterprise-scale storage area networks.

Choice of RAID level can critically affect the performance delivered by a storage system. It is therefore important to be able to predict performance of a given RAID configuration for various I/O workloads. The present paper aims to achieve this using an analytical queueing network-based model that extends our work in [10].

In the context of modern Service Level Agreements, effective performance prediction must provide the ability to reason not only about mean response times, but also higher moments and percentiles of response time. Therefore, our target in this work is the full cumulative distribution function of I/O request response time, from which all of the previous measures can be easily derived. Analytical queueing network models of RAID performance [4, 9, 12, 18, 19] developed prior to [10] approximate only the mean response time of the system. We note that RAID performance can also be modelled using other techniques including simulation [4, 12], table-based [2] and black-box modelling [13].

Our RAID model is developed in a bottom-up hierarchical fashion. We begin by modelling each disk drive in the array as a single M/G/1 queue. We then abstract the RAID as a fork-join queueing network [3] in which each disk in the array is represented by an M/G/1 queue. In an $N$-queue fork-join network (see Fig. 1) 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.

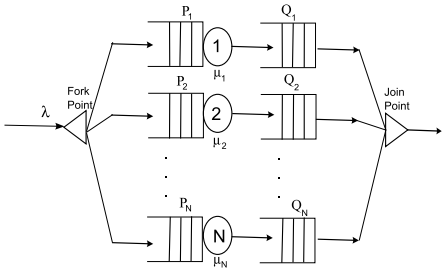



**Figure 1. Fork-join queueing model**

The standard fork-join network directly models the behaviour of a RAID system in only a small number of cases (e.g. full stripe I/O operations in RAID 0). Consequently, the fork-join model must be tailored to support the full range of I/O access patterns that occur when performing read or write operations of different sizes on different RAID levels. In [10] we used this approach to develop a preliminary analytical queueing model of RAID 01 and 5 for homogeneous Poisson arrival streams. By homogeneous we mean that all I/O requests are assumed to be ran-

dom accesses of the same type (read/write) and size. Our focus at present is modelling RAID 01 and RAID 5 as these are the two most commonly used RAID levels.

This paper presents a number of improvements to our initial work which take us closer to modelling real-life workloads. In particular, by introducing multiple classes into our model, we allow support for heterogeneous Poisson arrival streams in which I/O requests can have different types (read or write). By also introducing priority, we improve our RAID 5 write model to more authentically reflect subtask scheduling within each RAID 5 write request.

The remainder of this paper is organised as follows. Section 2 briefly summarises the zoned disk drive and fork-join queueing model previously presented in [10]. It also describes the mathematical background needed to introduce multiclass and priority queueing networks into our models. Section 3 presents our improved RAID 5 write model, as well as extensions for heterogeneous arrival streams. Section 4 validates all models against device measurements.

## 2. Background

### 2.1. Zoned Disk Model

The service time density of an access to a random location on a single disk drive is the convolution of the seek time, rotational latency and data transfer time probability density functions. An important subtlety that needs to be taken into account is that modern disks are *zoned*, with more sectors on the outer tracks than inner tracks. Therefore, a random request is more likely to be directed to a sector on an outer track. Similarly, zoning means that it is faster to transfer data on a track close to the circumference than the centre of the disk. The seek time and data transfer models must take these factors into account.

In our model we use the seek time and rotational latency probability distributions defined in [20] and the data transfer time distribution from [10]. We denote the random variables of seek time, rotational latency and $k$-block transfer time as $S$, $R$ and $T_k$ respectively. We represent a disk as an M/G/1 queue and, in keeping with our prior work, derive its response time distribution by numerically inverting its Pollaczek-Khintchine transform [1, 8].

### 2.2. Fork-Join Model

Our RAID model is based on a fork-join queueing network composed of M/G/1 queues. However, it is difficult to model job response times in a fork-join synchronisation analytically. Indeed, exact analytical results only exist for the mean response time of a two server system consisting of homogeneous M/M/1 queues [14]. Approximations for mean response times for M/M/1 and M/G/1 fork-join queues are more abundant [14, 17, 18] but such results do not permit higher moments or full response time distributions to be calculated. Therefore, we have previously presented [10] an approach using the maximum order statistic [6, 11] to derive an approximation to the cumulative distribution function of a fork-join queue's response time.

### 2.3. RAID Model

A fork-join queue does not model all the intricacies of a RAID system. The fork-join analysis defined above can 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. In [10], we tailor the fork-join approximation to model I/O operations on mirrored stripes (RAID 01) and distributed parity (RAID 5). We summarise these models in the Appendix.

### 2.4. Multiclass Queues

To extend our prior work to include multiclass queues, we define an expression for the cumulative distribution of the response time of a request in a multiclass model. In a system where arrivals have class $i$, $i = 1, \ldots, m$, let $W$ be a random variable representing a request's response time. The cumulative distribution function of $W$, $F_W(t)$, is:

$$\begin{aligned} F_W(t) &= P(W \leq t) \\ &= \sum_{i=1}^{m} P(W \leq t \mid \text{class}_i) P(\text{class}_i) \end{aligned} \tag{1}$$

### 2.5. Priority

We also need to introduce priorities for I/O requests for the specific purpose of improving our RAID 5 write model. This requires two priority levels, which we represent by two classes where class 0 has higher, non-preemptive priority than class 1. Each class has an arrival rate $\lambda_i$, a service time Laplace-Stieltjes transform (LST) $X_i^*(s)$ and a mean service rate $\mu_i$. The LST of the response time distribution for a job of class $i$, denoted $W_i^*(s)$, is therefore [5, 7]:

$$W_0^*(s) = \frac{((1-\rho)s + \lambda_1(1 - X_1^*(s)))X_0^*(s)}{\lambda_0(X_0^*(s) - 1) + s} \tag{2}$$

$$W_1^*(s) = \frac{(1-\rho)(s + \lambda_0(1 - M^*(s)))X_1^*(s)}{\lambda_1(X_1^*(s + \lambda_0(1 - M^*(s))) - 1) + s} \tag{3}$$

where $\rho = \frac{\lambda_0}{\mu_0} + \frac{\lambda_1}{\mu_1}$. $M^*(s)$ is the LST representing the sum of the service times of arriving class 0 jobs while a class 1 job is servicing. It is defined self-referentially by:

$$M^*(s) = X_0^*(s + \lambda_0(1 - M^*(s)))$$

## 3. Multiclass RAID Model

The ability to calculate the cumulative distribution function of a multiclass queue enables us to refine the original model of [10] to more accurately reflect the physical behaviour of the disk array and the mixed arrival streams it is likely to receive. In Section 3.1, we present an improved RAID 5 write model by introducing multiclass queues, and then extending this to include priority. In Section 3.2 we extend our models to support heterogeneous arrivals.

### 3.1. Improved RAID 5 Write Models

A RAID 5 partial stripe write is composed of two subtasks: a pre-read (for subsequent parity update) followed by a write of the partial stripe and new parity. If the partial stripe write follows some full stripe writes then the pre-read follows immediately after the full stripe writes. However, the array must wait for all the pre-reads to complete and the new parity to be calculated before the partial stripe writes can be issued to the disks. These partial stripe writes are then given priority over any other request in the disk queue.

The RAID 5 write model previously developed in [10] does not explicitly represent these two subtasks and instead computes the cdf of the overall response time based on the average of the service times of the pre-read and the partial stripe write. We therefore present a new RAID 5 partial stripe write model which employs two classes of request to separately model the pre-read and partial stripe write.

We assume that the arrival streams are composed of random access requests of homogeneous sizes and types. Further there are $n$ homogeneous disks in the array and the arrival rate of logical I/O requests to the array is $\lambda$.

We represent service time as the random variable $X = S + R + T_k$, where $S$, $R$ and $T_k$ are defined in Section 2.1, and denote the number of blocks (stripe units) accessed by a request as $b$. Let $W_d(t, \gamma, \mu)$ define the cumulative distribution function (cdf) of the response time of a single M/G/1 queue (disk), where $\gamma$ is the arrival rate at an individual disk and $\mu$ is the mean service rate.

**3.1.1. Multiclass Extension** We denote the cdfs of the response time of the pre-read subtask as $W_1(t)$ and of the partial stripe write subtask as $W_0(t)$. In a multiclass system, the total arrival rate to a queue (disk), $\gamma$, is the sum of the arrival rates to a queue for each class; thus $\gamma = \gamma_1 + \gamma_0$.

We assume that the time to complete a single pre-read and a single partial stripe write is equivalent to the weighted average of completing two pre-reads or two partial stripe writes. We note that these two subtasks are not independent. Indeed, we assume that they are highly dependent, giving:

$$W_{\text{write}}(t) = P(2W \leq t) = P\left(W \leq \left(\frac{t}{2}\right)\right)$$

Using these assumptions and Equation (1), the overall response time distribution of a single partial stripe write request $W_{\text{write}}(t)$ is:

$$W_{\text{write}}(t) = \frac{\gamma_1}{\gamma} W_1\left(\frac{t}{2}\right) + \frac{\gamma_0}{\gamma} W_0\left(\frac{t}{2}\right) \tag{4}$$

If $b < \frac{n-1}{2}$, a small partial stripe write, parity is calculated using [15]:

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

where $\oplus$ is the exclusive-or (XOR) operator. The first subtask pre-reads $b+1$ blocks of data for the parity update. The second subtask writes the new data to the same $b+1$ disks. This request is given priority in the queue, so at least one disk (the last to complete the pre-read) will have just completed reading a data or parity block that now needs to be re-written. Therefore we add a full disk rotation ($R_{max}$) into that disk's 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 only one disk needs a complete rotation. Since both sets of subtasks access the same number of disks, the arrival rates to each queue are $\gamma_1 = \gamma_0 = \frac{\lambda(b+1)}{n}$.

The cdfs of the response times of the subtasks are then:

$$\begin{aligned} W_1(t) &= \left(W_d\left(t, \tfrac{2\lambda(b+1)}{n}, \tfrac{1}{E[R]+E[S]+E[T_1]}\right)\right)^{b+1} \\ W_0(t) &= \left(W_d\left(t, \tfrac{2\lambda(b+1)}{n}, \tfrac{1}{\frac{b(E[R]+E[S])+R_{max}}{b+1}+E[T_1]}\right)\right)^{b+1} \end{aligned}$$

If $\frac{n-1}{2} \leq b < n-1$, 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. The first subtask 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. Thus, the arrival rates to each disk within each class are:

$$\gamma_1 = \frac{\lambda(n-b-1)}{n} \qquad \gamma_0 = \frac{\lambda(b+1)}{n}$$

The cdfs of the response times of the subtasks are:

$$\begin{aligned} W_1(t) &= \left(W_d\left(t, \lambda, \tfrac{1}{E[R]+E[S]+E[T_1]}\right)\right)^{n-b-1} \\ W_0(t) &= \left(W_d\left(t, \lambda, \tfrac{1}{E[R]+E[S]+E[T_1]}\right)\right)^{b+1} \end{aligned}$$

When a partial stripe write (either large or small) follows at least one full stripe write, the first subtask includes the

full stripe write and so will write to all $n$ disks. The second subtask is only the partial stripe write and hence will only write to $b_{mod}+1$ disks where $b_{mod} \equiv b \bmod (n-1)$. The arrival rates to each disk for each class are:

$$\gamma_1 = \lambda \qquad \gamma_0 = \frac{\lambda(b_{mod}+1)}{n}$$

In the case that a small partial stripe follows at least one full stripe write ($b > n-1$ and $0 < b_{mod} < \frac{n-1}{2}$), the first subtask is made up of $k = \lfloor \frac{b}{n-1} \rfloor$ block writes to each of the $n$ disks followed by pre-reads to $b_{mod}$ disks. The second subtask writes the new data and parity to $b_{mod}+1$ disks. The cdfs of the response times of the subtasks are:

$$\begin{aligned} W_1(t) &= \left(W_d\left(t, \tfrac{\lambda(n+b_{mod}+1)}{n}, \mu_1\right)\right)^n \\ W_0(t) &= \left(W_d\left(t, \tfrac{\lambda(n+b_{mod}+1)}{n}, \mu_0\right)\right)^{b_{mod}+1} \end{aligned}$$

where

$$\mu_1 = \frac{1}{E[R]+E[S]+E[T_{k+\frac{b_{mod}}{n}}]}$$

and

$$\mu_0 = \frac{1}{\frac{b_{mod}(E[R]+E[S])+R_{max}}{b_{mod}+1} + E[T_1]}$$

When a large partial stripe follows at least one full stripe write ($\frac{n-1}{2} \leq b_{mod} < n-1$), the first subtask consists of $k$ block writes to each of the $n$ disks followed by pre-reads to $n-b_{mod}-1$ disks. The second subtask writes the new data and parity to the remaining $b_{mod}+1$ disks. One of these disks will not have to seek again, as it will be the last disk to have finished transferring the full stripe. The cdfs of the response times of the subtasks are then:

$$\begin{aligned} W_1(t) &= \left(W_d\left(t, \tfrac{\lambda(n+b_{mod}+1)}{n}, \mu_1\right)\right)^n \\ W_0(t) &= \left(W_d\left(t, \tfrac{\lambda(n+b_{mod}+1)}{n}, \mu_0\right)\right)^{b_{mod}+1} \end{aligned}$$

where

$$\mu_1 = \frac{1}{E[R]+E[S]+E[T_{k+\frac{n-b_{mod}-1}{n}}]}$$

and

$$\mu_0 = \frac{1}{\frac{b_{mod}(E[R]+E[S])}{b_{mod}+1} + E[T_1]}$$

**3.1.2. Priority Extension** As soon as the new parity is calculated, the partial stripe write subtask is prioritised. Thus, we can give class 0 jobs high priority and all other jobs (including reads) low priority. Using the Laplace transforms defined in Section 2.5 instead of the Pollaczek-Khintchine transform equation, the response time distribution can be derived for high and low priority jobs.

### 3.2. Heterogeneous Arrival Streams

Thus far, our RAID models assume homogeneous arrival streams. Here we use multiclass queues to generalise these models for heterogeneous streams composed of both reads and writes. This is achieved using Equation (1) to calculate the request response time cdf:

$$W(t) = p_{read} W_{\text{read}}(t) + (1-p_{read}) W_{\text{write}}(t)$$

where $p_{read}$ is the probability that a request is a read.

We note that the arrival rate to the disk array used in [10] and Section 3 must be modified. For RAID 01 the arrival rate at each disk is:

$$\frac{\lambda(p_{read}\min(b,n) + (1-p_{read})\min(2b,n))}{n}$$

On RAID 5, the arrival rate at each disk is:

$$p_{read}\lambda\frac{\min(b,n)}{n} + (1-p_{read})\gamma$$

where $\gamma$ is the arrival rate at each disk in the array in the case that $p_{read} = 0$, defined for each size of write request in [10]. If the RAID 5 priority write model is used, then the arrival rate of each priority class must also be known. The arrival rate of low priority jobs is:

$$p_{read}\lambda\frac{\min(b,n)}{n} + (1-p_{read})\gamma_1$$

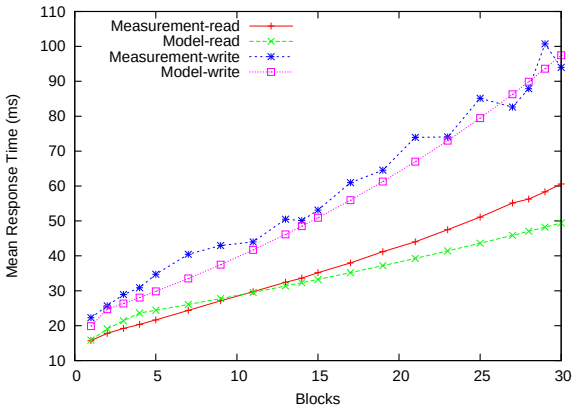
while high priority jobs have an arrival rate of:

$$(1-p_{read})\gamma_0$$

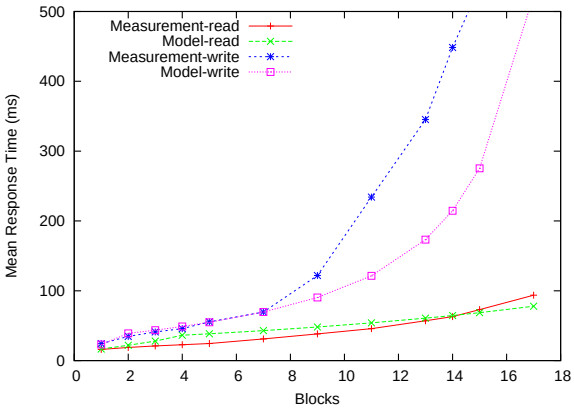
## 4. Validation

Our experimental platform consists of an Infortrend A16F-G2430 RAID system containing four Seagate ST3500630NS disks. Each disk has 60801 cylinders. A sector is 512 bytes and we have approximated, based on measurements from the disk drive, 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; the same measurements are 1ms and 18ms respectively for a write [16].

To obtain response time measurements from this system, we implemented a benchmarking program that issues read and write requests using a master process and a number of child processes. These child processes are responsible for issuing and timing I/O requests, leaving the master free to




(a) $\lambda = 0.01$ request/ms




(b) $\lambda = 0.03$ requests/ms

**Figure 2. Comparison of mean response time against block size for RAID 01 for different values of $\lambda$.**

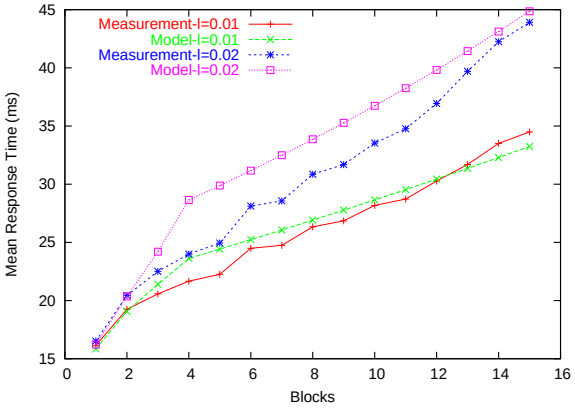



**Figure 3. Comparison of mean response time against block size for RAID 5 reads for different values of $\lambda$ (l).**

spawn further processes without the need for it to wait for previously-issued operations to complete.

In order to validate the analytical model effectively, it was necessary to minimise the effects of buffering and caching as these are not currently represented in the model. We therefore disabled the RAID system's write-back cache, set the read-ahead buffer to 0 and opened the device with the `O_DIRECT` flag set. For each of the experiments presented below, 100 000 requests were issued and the resulting means, variances and cumulative distribution functions of the response times were calculated using the statistical package R.

### 4.1. RAID 01

In [10], we conducted a limited validation of our RAID 01 model by comparing modelled and measured cdfs for the case of 2-block transfers only. Here, we aim to more fully test the accuracy of the model by comparing measurement and model for different load sizes and for block sizes ranging from 1 to 30.

Fig. 2 shows measured and modelled mean response times of reads and writes for RAID 01 for two different values of $\lambda$ – a light load of $\lambda = 0.01$ requests/ms (Fig. 2(a)) and a heavy load of $\lambda = 0.03$ requests/ms (Fig. 2(b)). For write requests under light load, agreement between model and measurement is excellent, even for large block sizes. Under heavy load, agreement is excellent up to 7 blocks which is when the system starts to saturate.

For read requests under both loads we observe good agreement for block sizes of less than 17, with a slight tendency for the model to overestimate for small block sizes. For larger block sizes, the model tends to increasingly underestimate the measurements. This behaviour is interesting because it does not occur with RAID 01 writes or RAID 5 reads (see Fig. 3); we speculate that this is possibly because of the drive selection policy (which controls whether to read from a primary disk or its mirror) implemented by the RAID controller. Our model assumes random choice of primary disk or mirror, but there are a number of other options; we intend to investigate further through measurements on RAID systems produced by other manufacturers.

| $\lambda$ ($ms^{-1}$) | # Blks | Measured Mean (ms) | Measured $\sigma^2$ ($ms^2$) | Single Class Mean (ms) | Single Class $\sigma^2$ ($ms^2$) | Multiclass Mean (ms) | Multiclass $\sigma^2$ ($ms^2$) | Priority Mean (ms) | Priority $\sigma^2$ ($ms^2$) |
|---|---|---|---|---|---|---|---|---|---|
| 0.01 | 1 | 45.0 | 148.7 | 41.9 | 258.5 | 40.7 | 227.2 | 40.7 | 232.7 |
| | 2 | 44.3 | 135.2 | 42.5 | 179.0 | 43.8 | 208.6 | 43.6 | 202.4 |
| | 4 | 44.5 | 595.9 | 51.1 | 340.5 | 52.8 | 435.0 | 52.7 | 468.6 |
| | 5 | 41.8 | 494.3 | 47.8 | 271.7 | 52.7 | 403.1 | 52.1 | 387.4 |
| | 7 | 53.5 | 903.4 | 54.7 | 394.0 | 58.7 | 628.4 | 58.1 | 619.7 |
| | 8 | 57.2 | 1084.4 | 51.6 | 326.0 | 58.3 | 616.6 | 57.1 | 519.8 |
| | 10 | 65.8 | 1468.0 | 58.4 | 456.6 | 65.2 | 907.5 | 63.9 | 819.5 |
| | 11 | 64.9 | 1515.5 | 55.6 | 391.0 | 64.6 | 941.3 | 62.6 | 696.9 |
| | 13 | 64.16 | 1630.6 | 62.3 | 530.4 | 72.4 | 1295.4 | 70.1 | 1076.6 |
| | 14 | 77.0 | 1992.6 | 59.9 | 468.4 | 71.7 | 1414.5 | 68.4 | 926.1 |
| | 16 | 93.9 | 3327.9 | 66.4 | 616.1 | 80.3 | 1822.2 | 76.8 | 1400.7 |
| | 17 | 89.3 | 3216.2 | 64.4 | 559.9 | 79.6 | 2087.2 | 74.8 | 1216.2 |
| | 19 | 106.7 | 4710.2 | 70.7 | 715.5 | 89.0 | 2526.2 | 84.0 | 1803.2 |
| | 20 | 102.0 | 4331.6 | 69.2 | 667.7 | 88.6 | 3029.9 | 81.6 | 1578.2 |
| 0.02 | 1 | 51.9 | 278.4 | 48.4 | 466.8 | 47.1 | 429.1 | 46.9 | 475.6 |
| | 2 | 50.4 | 251.9 | 50.1 | 411.5 | 52.2 | 472.1 | 51.2 | 454.8 |
| | 4 | 71.7 | 2496.7 | 69.0 | 975.0 | 75.4 | 1430.4 | 75.2 | 1726.6 |
| | 5 | 65.3 | 2139.6 | 66.3 | 847.9 | 79.9 | 1731.1 | 75.4 | 1669.2 |
| | 7 | 98.4 | 5359.0 | 76.4 | 1235.3 | 90.3 | 2534.0 | 86.8 | 2509.8 |
| | 8 | 102.5 | 5863.0 | 74.8 | 1123.4 | 97.9 | 3588.7 | 86.6 | 2481.7 |
| | 10 | 137.7 | 10234.3 | 84.7 | 1573.2 | 110.1 | 4643.0 | 100.5 | 3686.9 |
| | 11 | 129.1 | 9171.6 | 84.6 | 1497.8 | 125.1 | 7967.5 | 100.0 | 3738.5 |
| | 13 | 164.5 | 18646.5 | 94.1 | 2014.7 | 137.7 | 8829.0 | 117.0 | 5471.1 |
| | 14 | 173.0 | 15746.0 | 96.1 | 2012.2 | 171.1 | 19712.4 | 116.3 | 5706.9 |

**Table 1. Mean and variance of request response time for the three RAID 5 write models.**

### 4.2. RAID 5

We validate our RAID 5 read model by comparing the measured and modelled mean response times in Fig. 3. Results are presented for two values of $\lambda$ (0.01 and 0.02 requests/ms) and for block sizes from 1 to 15. We generally see good agreement between model and measurement.

We now validate our three models for RAID 5 partial stripe write requests against device measurements. We refer to the model presented in [10] as the single class model and to the two models presented here in Section 3.1 as the multiclass and priority models respectively.

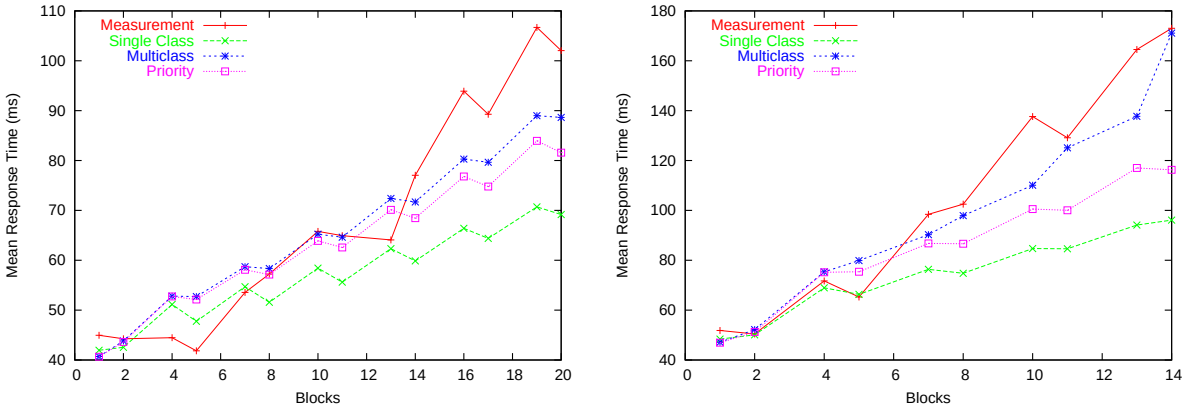
In Fig. 4 mean response times are presented for the three different models against device measurements for increasing block sizes and for arrival rates of 0.01 and 0.02 requests/ms. For small block sizes and loads, the single class model most often predicts means closest to the measured results. As block size increases, the means predicted by the multiclass and priority models are closer to the measured results. For large block sizes, the multiclass model clearly outperforms the other two models. However, the priority model means are reasonably close to the measured results for all block sizes. Table 1 contains means and variances for all cases.

Fig. 5 compares the pdfs and cdfs of the three models with measurements in the cases where each model had the closest mean and variance to the measurements. Fig. 5(a) is a 2-block write with an arrival rate of 0.02 requests/ms; the single class model gives the best mean and variance. Fig. 5(c) is a 14-block write with an arrival rate of 0.02 requests/ms; the multiclass model gives the best mean and variance. Fig. 5(b) is an 8-block write with an arrival rate of 0.01 requests/ms; the priority model gives the best mean and variance.

### 4.3. Mixed Reads and Writes

To validate both our RAID 01 and RAID 5 models for mixed reads and writes, we consider arrival streams of 25% reads/75% writes, 50% reads/50% writes, and 75% reads/25% writes. Each of these streams was generated for $\lambda = 0.01$ and 0.03 requests/ms and requests sizes between 1 and 5 blocks inclusive.

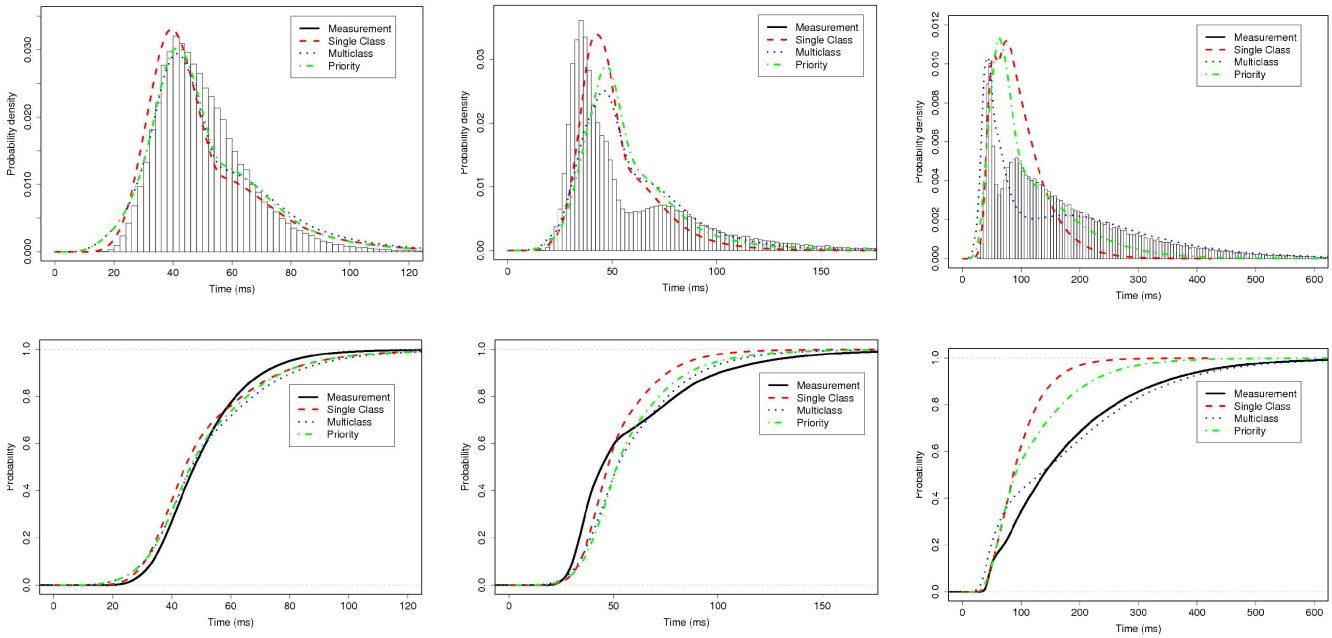
**4.3.1. RAID 01** Table 2 compares modelled and measured variances for this model. Fig. 6 presents the pdfs and corresponding cdfs for 2-block mixed read and write requests for the three read/write combinations at an arrival rate of $\lambda = 0.03$ requests/ms. We observe excellent agreement between measured and modelled means, variances, pdfs and cdfs.




(a) $\lambda = 0.01$ request/ms

(b) $\lambda = 0.02$ requests/ms

**Figure 4. Comparison of mean response time for all models against block size for RAID 5 partial stripe writes for different values of $\lambda$.**




(a) 2-block requests, $\lambda = 0.02$ requests/ms

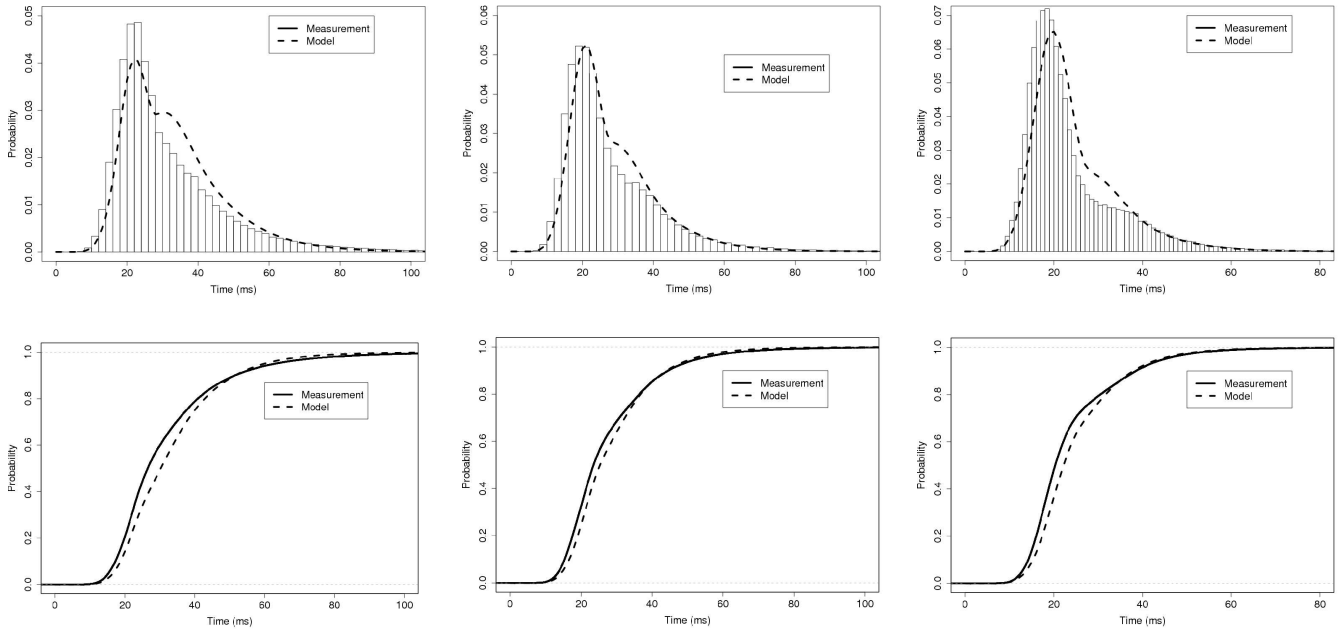
(b) 8-block requests, $\lambda = 0.01$ requests/ms

(c) 14-block requests, $\lambda = 0.02$ requests/ms

**Figure 5. Selected pdfs and cdfs of RAID 5 write request response times for the three models.**

| $\lambda$ ($ms^{-1}$) | # Blks | 25% Reads, 75% Writes | | | | 50% Reads, 50% Writes | | | | 75% Reads, 25% Writes | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Measured | | Modelled | | Measured | | Modelled | | Measured | | Modelled | |
| | | Mean (ms) | $\sigma^2$ ($ms^2$) | Mean (ms) | $\sigma^2$ ($ms^2$) | Mean (ms) | $\sigma^2$ ($ms^2$) | Mean (ms) | $\sigma^2$ ($ms^2$) | Mean (ms) | $\sigma^2$ ($ms^2$) | Mean (ms) | $\sigma^2$ ($ms^2$) |
| 0.01 | 1 | 21.0 | 41.8 | 18.8 | 27.7 | 19.4 | 39.7 | 17.8 | 27.2 | 17.7 | 32.7 | 16.8 | 25.6 |
| | 2 | 24.4 | 66.6 | 23.1 | 45.3 | 22.6 | 62.3 | 21.7 | 38.8 | 20.6 | 48.3 | 20.3 | 31.7 |
| | 3 | 27.3 | 90.1 | 25.0 | 54.6 | 25.1 | 81.8 | 23.8 | 48.4 | 22.6 | 63.4 | 22.6 | 41.0 |
| | 4 | 29.2 | 102.1 | 27.0 | 67.3 | 26.9 | 98.0 | 25.8 | 62.3 | 24.2 | 75.8 | 24.7 | 54.8 |
| | 5 | 32.5 | 137.5 | 28.5 | 78.9 | 29.7 | 131.9 | 27.1 | 72.6 | 26.4 | 98.0 | 25.8 | 62.5 |
| 0.03 | 1 | 22.9 | 82.6 | 21.1 | 60.4 | 21.0 | 72.8 | 19.5 | 51.3 | 18.8 | 54.5 | 18.0 | 42.0 |
| | 2 | 31.5 | 262.6 | 33.1 | 195.4 | 27.6 | 180.1 | 28.5 | 134.0 | 23.6 | 112.3 | 24.8 | 95.6 |
| | 3 | 37.3 | 404.8 | 38.7 | 279.4 | 32.4 | 283.8 | 34.6 | 220.5 | 27.3 | 176.2 | 31.0 | 166.9 |
| | 4 | 42.5 | 628.8 | 45.7 | 419.0 | 36.8 | 441.7 | 42.6 | 372.9 | 30.5 | 254.2 | 39.4 | 307.3 |
| | 5 | 50.4 | 946.6 | 50.7 | 550.5 | 42.4 | 596.1 | 46.6 | 485.0 | 34.3 | 347.7 | 42.5 | 385.9 |

**Table 2. Comparison of mean response times and variances for mixed read and write request streams for RAID 01.**




(a) 25% read requests, 75% write requests (b) 50% read requests, 50% write requests (c) 75% read requests, 25% write requests

**Figure 6. RAID 01 2-block request response time pdfs and cdfs for arrival streams of mixed reads and writes and an arrival rate $\lambda = 0.03$ requests/ms.**

| $\lambda$ ($ms^{-1}$) | # Blks | 25% Reads, 75% Writes | | | | 50% Reads, 50% Writes | | | | 75% Reads, 25% Writes | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Measured | | Modelled | | Measured | | Modelled | | Measured | | Modelled | |
| | | Mean (ms) | $\sigma^2$ ($ms^2$) | Mean (ms) | $\sigma^2$ ($ms^2$) | Mean (ms) | $\sigma^2$ ($ms^2$) | Mean (ms) | $\sigma^2$ ($ms^2$) | Mean (ms) | $\sigma^2$ ($ms^2$) | Mean (ms) | $\sigma^2$ ($ms^2$) |
| 0.01 | 1 | 42.5 | 544.1 | 34.9 | 292.6 | 34.0 | 408.8 | 28.2 | 258.6 | 25.8 | 257.8 | 21.9 | 166.1 |
| | 2 | 43.3 | 537.8 | 36.4 | 217.2 | 35.4 | 401.0 | 30.4 | 200.6 | 28.1 | 248.8 | 24.6 | 134.7 |
| | 3 | 27.6 | 126.9 | 23.8 | 46.5 | 25.8 | 113.8 | 23.0 | 42.2 | 23.9 | 89.3 | 22.2 | 37.6 |
| | 4 | 33.6 | 244.8 | 43.9 | 365.0 | 30.2 | 206.9 | 36.9 | 312.0 | 26.7 | 142.0 | 30.2 | 211.5 |
| | 5 | 33.7 | 234.5 | 41.6 | 268.7 | 30.7 | 198.4 | 35.7 | 229.1 | 27.3 | 144.4 | 30.0 | 155.2 |
| 0.03 | 1 | *sat.* | *sat.* | 44.3 | 670.5 | 105.8 | 9642.2 | 33.2 | 455.0 | 48.0 | 1722.9 | 24.2 | 241.3 |
| | 2 | *sat.* | *sat.* | 49.4 | 710.8 | 117.9 | 11921.4 | 38.7 | 496.4 | 57.5 | 2612.2 | 29.7 | 278.5 |
| | 3 | 53.1 | 1770.2 | 35.7 | 220.5 | 47.3 | 1386.0 | 32.8 | 182.4 | 40.1 | 865.4 | 30.2 | 148.8 |
| | 4 | 116.5 | 12192.3 | 85.5 | 2349.0 | 76.2 | 4596.2 | 65.3 | 1470.7 | 53.0 | 1880.7 | 49.3 | 786.1 |
| | 5 | 109.7 | 9268.7 | 88.4 | 1947.2 | 77.4 | 4607.6 | 67.4 | 1122.3 | 55.6 | 2096.4 | 51.2 | 615.0 |

**Table 3. Comparison of mean response times and variances for mixed read and write request streams for RAID 5.**

**4.3.2. RAID 5** In this section, we focus on the single class write model as it was, in the case of 100% write requests, the most accurate for small block sizes. Table 3 presents modelled and measured variances – note that the results for 1 and 2-block requests for the 25% read arrival stream at $\lambda = 0.03$ requests/ms display saturation on the RAID system. We note that agreement between measured and modelled results is not as good as for RAID 01. In particular, we observe that the measured mean response times for 1 and 2-block requests are higher than for 4 and 5 block requests, but that this does not occur in the modelled results. Furthermore, the measured mean response times for mixed reads and writes exceed the measurements for 100% writes (which are significantly larger than the measurements for 100% reads) for the same block size under each load presented. This can be seen most clearly under heavier loads and for higher percentages of write requests. We need to investigate further the performance of the RAID system under mixed arrival streams to understand why such behaviour occurs.

## 5. Conclusion

In this paper we have presented an improved performance model for RAID systems capable of calculating full request response time distributions. By employing multiclass queues, we gain the ability to analyse mixed arrival streams of reads and writes. This mixture of request type more accurately reflects the workloads experienced by real-life RAID systems. Adopting multiclass queues also enables us to more realistically model the way in which partial stripe writes are conducted for RAID 5. Our results are extensively validated against device measurements from a real RAID system. This exercise has revealed some interesting discrepancies between model and measurement for certain types of I/O requests (e.g. RAID 5 mixed read and writes), which we will investigate further.

There are a number features which we still need to model in order to have a comprehensive model capable of representing real I/O workloads. Firstly, caching is not yet supported in our model. Secondly, we would like to support sequential as well as random I/O, to better model the effects of locality. Thirdly, we currently constrain the alignment of RAID 5 write requests to start at the beginning of a stripe in all cases. In the future, we would like to allow for requests that start with a partial stripe, followed by further data. Fourthly, all our models assume fixed request sizes and we would like to extend them to incorporate distributions of block sizes. 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.


## Acknowledgements

We would like to thank our anonymous referees and our shepherd for their many constructive comments. We are also grateful to Peter Harrison and Soraya Zertal for helpful discussions. This work is supported by EPSRC research grant EP/F010192/1.

## References


[1] J. Abate and W. Whitt. The Fourier-series method for inverting transforms of probability distributions. *Queueing Systems Theory and Applications*, 10(1-2):5–88, 1992.

[2] E. Anderson. Simple table-based modeling of storage devices. Technical Report HPL-SSP-2001-4, HP Labs, 2001.

[3] P. M. Chen, E. K. Lee, G. A. Gibson, R. H. Katz, and D. A. Patterson. RAID: High-Performance, Reliable Secondary Storage. *ACM Computing Surveys*, 26(2):145–185, 1994.

[4] S. Chen and D. Towsley. A performance evaluation of RAID architectures. *IEEE Transactions on Computers*, 45(10):1116–1130, 1996.

[5] R. W. Conway, W. L. Maxwell, and L. W. Miller. *Theory of Scheduling*. Addison-Wesley, 1967.

[6] H. A. David. *Order Statistics*. John Wiley, 1981.

[7] P. G. Harrison. Teaching M/G/1 theory with extension to priority queues. *IEE Proc. Computers and Digital Techniques*, 147(1):23–26, January 2000.

[8] P. G. Harrison and N. M. Patel. *Performance Modelling of Communication Networks and Computer Architectures*. Addison-Wesley, 1993.

[9] P. G. Harrison and S. Zertal. Queueing models of RAID systems with maxima of waiting times. *Performance Evaluation*, 64(7-8):664–689, August 2007.

[10] A. S. Lebrecht, N. J. Dingle, and W. J. Knottenbelt. A response time distribution model for zoned RAID. In *15th International Conference on Analytical and Stochastic Modelling Techniques and Applications (ASMTA)*, June 2008.

[11] A. S. Lebrecht and W. J. Knottenbelt. Response time approximations in fork-join queues. In *23rd UK Performance Engineering Workshop (UKPEW)*, July 2007.

[12] E. K. Lee. *Performance Modeling and Analysis of Disk Arrays*. PhD thesis, UC Berkeley, 1993.

[13] M. P. Mesnier, M. Wachs, R. R. Sambasivan, A. X. Zheng, and G. R. Ganger. Modeling the relative fitness of storage. In *Proc. ACM SIGMETRICS '07*, pages 37–48, 2007.

[14] R. Nelson and A. N. Tantawi. Approximate analysis of fork/join synchronization in parallel queues. *IEEE Transactions on Computers*, 37(6):739–743, June 1988.

[15] D. A. Patterson, G. Gibson, and R. H. Katz. A case for Redundant Arrays of Inexpensive Disks (RAID). In *Proc. Int. Conf. on Management of Data (SIGMOD)*, 1988.

[16] Seagate. Barracuda ES Data Sheet, 2007. http://www.seagate.com/docs/pdf/datasheet/disc/ds_barracuda_es.pdf.

[17] E. Varki. Mean value technique for closed fork-join networks. In *Proc. ACM SIGMETRICS*, pages 103–112, 1999.

[18] E. Varki. Response time analysis of parallel computer and storage systems. *IEEE Transactions on Parallel and Distributed Systems*, 12(11):1146–1161, November 2001.

[19] E. Varki, A. Merchant, J. Xu, and X. Qiu. Issues and challenges in the performance analysis of real disk arrays. *IEEE Transactions on Parallel and Distributed Systems*, 15(6):559–574, June 2004.

[20] S. Zertal and P. G. Harrison. Multi-RAID queueing model with zoned disks. In *High Performance Computing and Simulation Conference (HPCS'07)*, June 2007.


## Appendix

We summarise the RAID 01 and RAID 5 models of [10]. The response time cdf for a RAID 01 $b$-block read is:

$$W_{\text{read}}(t) = \begin{cases} \left(W_d\left(t, \frac{\lambda b}{n}, \frac{1}{E[R]+E[S]+E[T_1]}\right)\right)^b & \text{if } b < n \\ \left(W_d\left(t, \lambda, \frac{1}{E[R]+E[S]+E[T_{\frac{b}{n}}]}\right)\right)^n & \text{otherwise} \end{cases} \tag{5}$$

The response time cdf for a RAID 01 $b$-block write is:

$$W_{\text{write}}(t) = \begin{cases} \left(W_d\left(t, \frac{2\lambda b}{n}, \frac{1}{E[R]+E[S]+E[T_1]}\right)\right)^{2b} & \text{if } 2b < n \\ \left(W_d\left(t, \lambda, \frac{1}{E[R]+E[S]+E[T_{\frac{2b}{n}}]}\right)\right)^n & \text{otherwise} \end{cases}$$

We model a RAID 5 read in the same way as a RAID 01 read (see Equation (5)).

The simplest RAID 5 write request is one which consists only of a number of complete stripes. Computation of parity does not require pre-reading of existing data in this case. The response time cdf is:

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

For each case of the single class partial stripe write model, the mean service rate and number of queues are averaged for each class. A small partial stripe write, has an overall response time cdf of:

$$\begin{aligned} W_{\text{write}}(t) &= \left(W_d\left(\tfrac{t}{2}, \frac{2\lambda(b+1)}{n}, \mu\right)\right)^{b+1} \\ \mu &= \frac{1}{\frac{(2b+1)(E[R]+E[S])+R_{max}}{2(b+1)} + E[T_1]} \end{aligned}$$

For a large partial stripe write, the response time cdf is:

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

If at least one full stripe write occurs followed by a small partial stripe write, the response time cdf is:

$$\begin{aligned} W_{\text{write}}(t) &= \left(W_d\left(\tfrac{t}{2}, \frac{\lambda(n+b_{mod}+1)}{n}, \mu\right)\right)^{\frac{n+b_{mod}+1}{2}} \\ \mu &= \frac{1}{\frac{(n+b_{mod})(E[R]+E[S])+R_{max}}{n+b_{mod}+1} + E[T_{\frac{k}{2}+\frac{b_{mod}+1}{n}}]} \end{aligned}$$

If at least one full stripe write occurs followed by a large partial stripe write, the response time cdf is:

$$\begin{aligned} W_{\text{write}}(t) &= \left(W_d\left(\tfrac{t}{2}, \frac{\lambda(n+b_{mod}+1)}{n}, \mu\right)\right)^{\frac{n+b_{mod}+1}{2}} \\ \mu &= \frac{1}{\frac{(n+b_{mod}+0.5)(E[R]+E[S])}{2n} + E[T_{\frac{k+1}{2}}]} \end{aligned}$$