SIMULATION AND MODELLING OF RAID 0 SYSTEM PERFORMANCE

F. Wan N.J. Dingle W.J. Knottenbelt A.S. Lebrecht
Department of Computing, Imperial College London, South Kensington Campus, London SW7 2AZ
email: {yffw05,njd200,wjk,asl102}@doc.ic.ac.uk

ABSTRACT

RAID systems are fundamental components of modern storage infrastructures. It is therefore important to model their performance effectively. This paper describes a simulation model which predicts the cumulative distribution function of I/O request response time in a RAID 0 system consisting of homogeneous zoned disk drives. The model is constructed in a bottom-up manner, starting by abstracting a single disk drive as an M/G/1 queue. This is then extended to model a RAID 0 system using a split-merge queueing network. Simulation results of I/O request response time for RAID 0 systems with various numbers of disks are computed and compared against device measurements.

KEYWORDS

Queueing models; Simulation models; Storage system performance

INTRODUCTION

The *Redundant Array of Inexpensive Disks* (RAID) [8] is a storage technology which can ease the widening performance gap between the processor and I/O subsystem. By spreading I/O operations over multiple disks, the bandwidths of several hard disks can be utilised and overall I/O throughput can be increased. Once designed for high-end servers and mainframes, products with RAID functionality are increasingly popular and today can even be found in chipsets for desktop computers, for example the Intel 915P chipset for Intel Pentium 4 processors [3]. Given the widespread adoption of RAID systems and the fact that the I/O subsystem is often a performance bottleneck, it is important to be able to predict RAID system performance.

In this paper we present a simulator that predicts the performance of a RAID level 0 (i.e. striping without redundancy) system. We work in a bottom-up manner, basing our single disk simulation on the analytical model for a zoned disk drive presented in [7]. Since this model abstracts a disk drive as an M/G/1 queue, we construct a validated M/G/1 queueing simulator to implement it. We further abstract a RAID system as a split-merge queueing network [4], and develop a corresponding simulator which outputs a cumulative distribution function (cdf) of the I/O response time for a specified arrival rate, size of request and request type (read or write). To validate our simulator, we compare cdfs of the simulations and measurements.

SINGLE DISK MODEL

Before we can model the performance of a RAID system, we must be able to model the performance of its constituent disk drives. We therefore begin with the construction of an effective single disk simulator, implementing an established analytical model for a zoned disk drive. Zoning arises on modern hard drives because there are more sectors on cylinders on the outside of the platter than those closer to the centre. A zone is a contiguous collection of cylinders which have the same number of sectors and thus the same storage capacity. Since disks rotate with constant angular velocity, data throughput is higher for outer zones than for inner ones [7].

We model a disk drive as an M/G/1 queue, in which the service time is defined as the sum of seek time, S, rotational latency, R, and k-block transfer time, T_k.

Seek Time

Seek time, S, is the time taken to move the actuator arm to the track where the destination sector lies. This is a function of D, the distance between the starting and target cylinders [2]:

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

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

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

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

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. We note that $minseek$ and $maxseek$ are potentially dependent on whether the I/O operation is a read or a write.

Assuming I/O accesses are uniformly randomly distributed across disk sectors, the seek time cdf, $F_S(t)$, can be defined in terms of the seek distance cdf, $F_D(t)$ [2]:

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

The probability density function of D, $f_D(x)$ is calculated in [10] as:

$$f_D(x) = A + Gx + Ex^3 \qquad (0 \le x \le C - 1)$$

The constants are defined as follows:

$$
\begin{aligned}
A &= \frac{V(Cyls-1)}{3\gamma^2} \\
G &= -\frac{V+\beta^2(Cyls-1)^2}{3\gamma^2} \\
E &= \frac{\beta^2}{3\gamma^2} \\
V &= 6\alpha^2 + 6\alpha\beta(Cyls-1) + 2\beta^2(Cyls-1)^2 \\
\alpha &= \frac{SEC[0]}{spb} \\
\beta &= \frac{SEC[Cyls-1]-SEC[0]}{(Cyls-1)\ spb} \\
\gamma &= \alpha(Cyls-1) + \frac{\beta}{2}(Cyls-1)^2
\end{aligned}
$$

$SEC[0]$ and $SEC[Cyls-1]$ are the number of sectors on the innermost and outermost tracks respectively, and spb is the number of sectors per block.

Rotational Latency

Rotational latency, R, is the time taken for the disk to rotate until the required sector is under the read/write head. It is uniformly distributed between 0 and the time for a disk to make a full rotation (R_{max}) [2].

Data Transfer Time

Data transfer time, T_k, is the time taken to transfer k data blocks to or from the read/write head. The function to calculate T_k for cylinder x of a zoned disk is [10]:

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

The cdf of the data transfer time, $F_{T_k}(t)$ can then be derived as [7]:

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

with

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

t_{min} and t_{max} are the times to transfer to a single sector on the outermost and innermost tracks respectively.

Table 1: Seagate ST3500630NS drive parameters

Capacity	500GB
Total number of cylinders	60 801
RPM	7 200
Sector size	512 bytes
Sectors per block	256
Time to write a single physical sector on the innermost track (t_{max})	0.012064 ms
Time to write a single physical sector on the outermost track (t_{min})	0.005976 ms
Track-to-track seek time (Read)	0.8 ms
Full-stroke seek time (Read)	17 ms
Track-to-track seek time (Write)	1 ms
Full-stroke seek time (Write)	18 ms

The Single Disk Simulator

Our single disk simulator is a simulation of an M/G/1 queue and is implemented using the JINQS Java-based queueing network library [5]. Interarrival times for I/O requests are sampled from an exponential distribution with rate parameter λ. The service time for each request is generated by summing samples from the random variables S, R and T_k (for some fixed k). The simulator processes a specified number of I/O requests (usually 5 000) and outputs the response time for each request. These are then used to generate the cumulative distribution function of I/O request response time, as well as other summary statistics such as the mean, sample standard deviation and median.

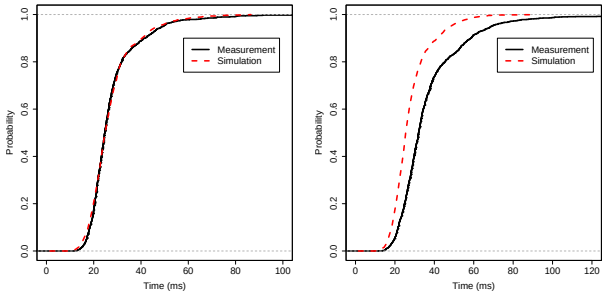
The operation of the M/G/1 queue simulator was validated for a simple Erlang service time distribution by comparing the cdf generated by our simulator with a known analytical result. In particular, the Laplace transform of the response time density of an M/G/1 queue can be derived using the Pollaczek-Khintchine transform equation [6], and then numerically inverted using Euler inversion [1] to yield the response time cdf. Agreement between simulated and numerical cdfs was found to be excellent, giving us confidence in the correct operation of our simulator.

To generate samples for disk service time, the simulator extracts samples from the analytical cdfs for $F_S(t)$ and $F_{T_k}(t)$. Specifically, we sample from a cdf $F(t)$ by computing the value of t for which $U = F(t)$, where U is a uniformly distributed random variable, $0 \le U \le 1$. Since rotational latency is uniformly distributed, a random sample is UR_{max}. The overall disk service time is the sum of the samples for seek time, rotational latency and transfer time.

Comparing with Measurements

The output of our single disk simulator was compared with the measurements from a Seagate ST3500630NS disk drive [9]. Table 1 gives the model parameters for this drive. Tests were conducted with an arrival rate of I/O requests (λ) of 0.01 and 0.02 requests/ms with workloads consisting of

$k = 1, 2, 5, 10, 20, 40$ 128KB blocks. For each value of k we ran two tests – one for reads and one for writes. Our single disk simulation matches device measurements very well, especially for workloads consisting of a lower arrival rate and smaller request sizes. We note, however, that the simulated write results do tend to slightly underestimate the measured results – see for example Fig. 1.



(a) 5-block read (b) 5-block write

Figure 1: Selected response time cdfs on a single disk for $\lambda = 0.01$ requests/ms.

When $\lambda = 0.02$ requests/ms and $k > 10$ the simulation results begin to underestimate the device measurements for both reads and writes. For $k \geq 20$ at both arrival rates, the model fails to give accurate predictions of response time because the requests are arriving faster than they are being served, and the queueing theory upon which our simulator is based requires a stable state to function properly.

RAID 0 MODEL

Split-Merge Queues

We now extend the single disk model to a RAID 0 model using a split-merge queueing network [4]. In a split-merge queue with N servers, 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. Therefore, for each request, the response time is defined as the maximum of the subtasks' response times.

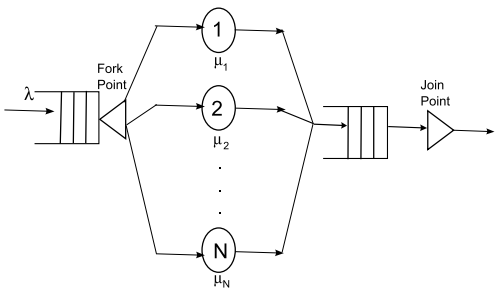


Figure 2: A split-merge queueing network

The split-merge queueing model fits well with the way in which a RAID system operates, since disk drives in the RAID system can be treated as servers in the queueing network. Users issue I/O requests to the RAID controller which holds arriving requests in a queue. The RAID controller repeatedly dequeues requests and splits them into several sub-requests, each to be serviced by one of the disk drives in parallel. After all sub-requests of a request have been serviced, they are merged and the I/O request completes.
We note that newer RAID 0 systems have buffering queues before and after the disks, and so disks can serve the sub-requests of new requests continuously without waiting for earlier requests to complete. Our model will slightly overestimate I/O request response time for such systems.

Modifying the Single Disk Simulator

We constructed a RAID 0 simulator by building a split-merge layer on top of our existing single disk simulator. When the RAID 0 simulator receives an I/O request from its queue, it splits the request into smaller sub-requests, and calls the single disk models to calculate the service times of those sub-requests. The RAID simulator selects the maximum service time returned by the single disk models to be the service time of that I/O request.

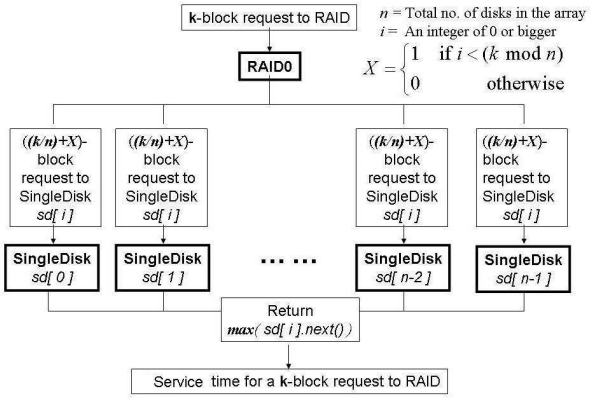


Figure 3: Calculating RAID 0 request service times

Upon receiving a k-block read or write request, the RAID simulator (`Raid0` hereafter) starts to allocate the blocks to different instances of single disk models (`SingleDisk` hereafter). The allocation scheme is illustrated in Fig. 3.
Each `SingleDisk` object, `sd[i]`, will read or write a minimum of $\lfloor \frac{k}{n} \rfloor$ blocks. Additional blocks, X, will be added as follows:

$$X = \begin{cases} 1 & \text{if } i < (k \bmod n) \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

For example, if a 7-block write request, `write(A,B,C,D,E,F,G)`, is issued to a 3-disk RAID 0 system, then the 7 blocks will be written to the disks in the following way:

sd[0] `write(A, D, G)` ($k = 3$)

sd[1] `write(B, E)` ($k = 2$)

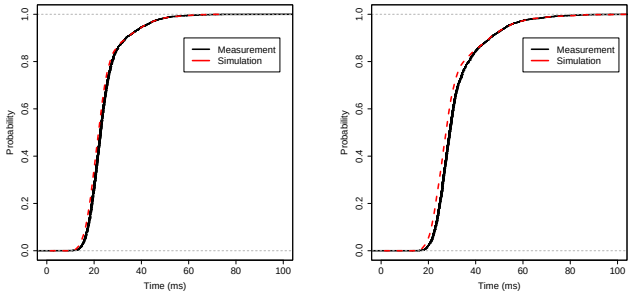
sd[2] `write(C, F)` ($k = 2$)

`Raid0` will then pick the largest service time generated from the array of `SingleDisk`s and return it as the service time of the I/O request. Adding time spent queueing yields the overall I/O request response time.

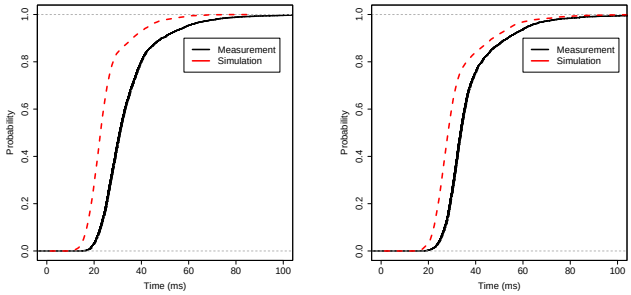
RESULTS AND ANALYSIS

We now validate our simulator by comparing cdfs of generated response time with actual measurements from RAID 0 systems with 2, 3 and 4 disk drives. In all cases, we used an Infortrend A16F-G2430 RAID system containing four Seagate ST3500630NS disks. We set the stripe width on the array to 128 KB and disabled the write caches of both the disk drives and the RAID system. For each test, we read or wrote k blocks ($1 \leq k \leq 20$) for arrival rate $\lambda = 0.01$ requests/ms. The sample size for constructing a cdf is 5 000 requests.

2-Disk Results



(a) 5-block read (b) 10-block read

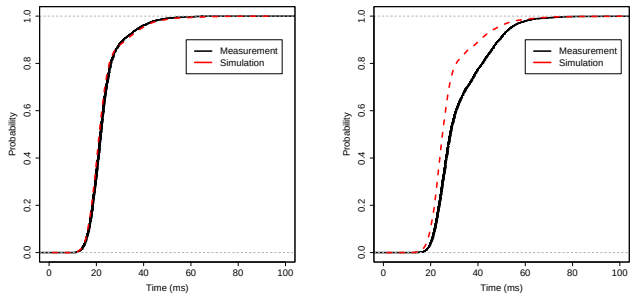


(c) 5-block write (d) 10-block write

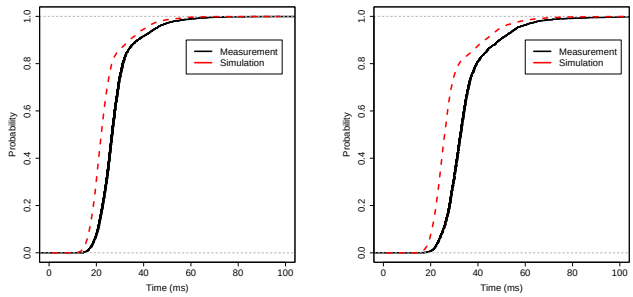
Figure 4: Selected response time cdfs on a 2-disk RAID 0 system for $\lambda = 0.01$ requests/ms.

Fig. 4 shows a selection of read and write results (presented in terms of the cdf of I/O request response time) for a number of request sizes on a 2-disk RAID 0 system with $\lambda = 0.01$ requests/ms. We observe close agreement between simulated and measured results for reads at all block sizes, while for writes we note that the simulation tends to slightly underestimate the measured results in all cases. This suggests that there is an additional overhead inherent in writes for which our model does not account. This discrepancy was also visible in the case of the single disk model, which suggests that the source is likely to be at the level of the disk rather than the RAID controller. Understanding the reason for this difference is a key area of our future work.

3-Disk Results



(a) 5-block read (b) 10-block read



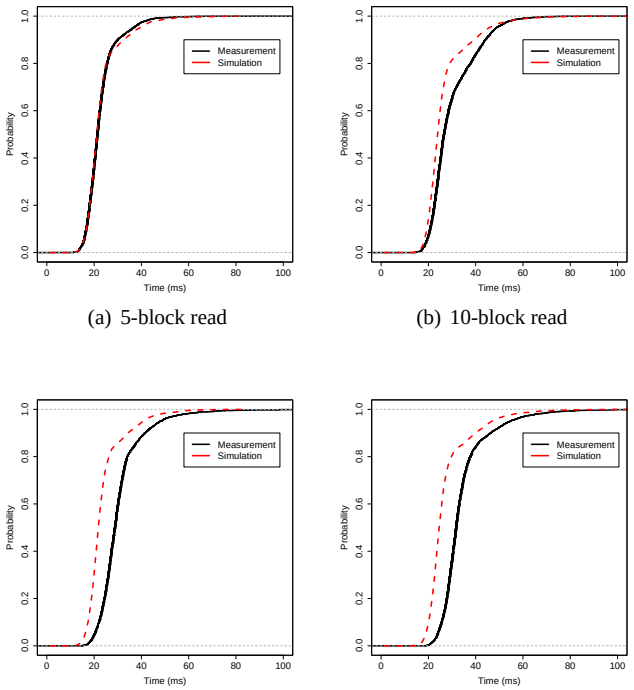
(c) 5-block write (d) 10-block write

Figure 5: Selected response time cdfs on a 3-disk RAID 0 system for $\lambda = 0.01$ requests/ms.

Fig. 5 shows a selection of read and write results for a number of request sizes on a 3-disk RAID 0 system with $\lambda = 0.01$ requests/ms. We observe close agreement between simulated and measured results for reads at all block sizes except $k = 10$, while for writes we note again that the simulation tends to slightly underestimate the measured results in all cases.

4-Disk Results

Fig. 6 shows a selection of read and write results for a number of request sizes on a 4-disk RAID 0 system with $\lambda = 0.01$ requests/ms. We observe similar trends in the results as for the 3-disk case.



(a) 5-block read (b) 10-block read

(c) 5-block write (d) 10-block write

Figure 6: Selected response time cdfs on a 4-disk RAID 0 system for $\lambda = 0.01$ requests/ms

CONCLUSION

In this paper we have presented a simulation model which predicts the cdf of I/O request response time in a RAID 0 system consisting of homogeneous zoned disk drives. We first constructed an M/G/1 simulation model of a single disk drive, which we validated against device measurement, and then used a split-merge queueing network to model the RAID system. We validated our resulting RAID 0 simulator against device measurements and demonstrated its accuracy.

There are a number of avenues for future work. Firstly, we need to account for the difference observed between our simulator and the measured results for write operations. Secondly, our simulator currently only models RAID 0 but there are several other commonly-deployed RAID configurations, particularly RAID 01 (mirrored stripes) and RAID 5 (distributed parity), which it could be extended to represent. Finally, our simulator is currently only capable of analysing request streams composed entirely of reads or entirely of writes. In the real world, streams of I/O requests will almost always be composed of a mixture of both, and we need to extend our simulator to be able to model this. Real I/O traffic can also consist of variably-sized requests and typically exhibits considerable burstiness, neither of which are supported by our simulator. We will therefore work to add support for these behaviours.

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

[3] Intel Corporation. Intel® 915P Express Chipset. http://www.intel.com/products/chipsets/915P/.

[4] A. Duda and T. Czachórski. Performance evaluation of fork and join synchronization primitives. *Acta Informatica*, 24(5):525–553, 1987.

[5] A. J. Field. *JINQS: An Extensible Library for Simulating Multiclass Queueing Networks V1.0 User Guide*, August 2006.

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

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

[8] D. A. Patterson, G. Gibson, and R. H. Katz. A case for redundant arrays of inexpensive disks (RAID). In *Proc. International Conference on Management of Data (SIGMOD)*, 1988.

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

[10] S. Zertal and P. G. Harrison. Multi-RAID Queueing Model with Zoned Disks. In *Proc. 21st European Conference on Modelling and Simulation*, June 2007.