Available online at www.sciencedirect.com
ScienceDirect
ELSEVIER
Electronic Notes in Theoretical Computer Science 232 (2009) 5–16
Electronic Notes in Theoretical Computer Science
www.elsevier.com/locate/entcs

Bus Modelling in Zoned Disks RAID Storage Systems

Peter Harrison[a] and Soraya Zertal[b]

[a] *Imperial College London, South Kensington Campus, London SW7 2AZ, UK*
Email: pgh@doc.ic.ac.uk

[b] *PRiSM, Université de Versailles, 45 Av. des Etats-Unis, 78000 Versailles, France*
Email: Zertal@prism.uvsq.fr

Abstract

A model of bus contention in a Multi-RAID storage architecture is presented. Based on an M/G/1 queue, the main issues are to determine the service time distribution that accurately represents the highly mixed input traffic of requests. This arises from the coexistence of different RAID organisations that generate several types of physical request (read/write for each RAID level) with different related sizes. The size distributions themselves are made more complex by the striping mechanism, with full/large/small stripes in RAID5. We show the impact of the bus traffic on the system's overall performance as predicted by the model and validated against a simulation of the hardware, using common workload assumptions.

Keywords: Multi-RAID, zoned disks, M/G/1 queues, IO and bus modelling, simulation

1 Introduction

Storage systems have evolved from small collections of interconnected disks to large disk-arrays, shared by multiple applications serving a very large user community. RAID (Redundant Arrays of Independent Disks) [2] systems were the first widely accepted, large scale storage architecture, for which many proposals for enhancement have been put forward. These include a sequence of RAID variants to improve speed of access and reliability (from RAID0 to RAID6) and the HPautoRAID [7] which supports two RAID organisations on the same storage system at separate space locations. In the Multi-RAID system of [5], different RAID configurations coexist on the same disk devices without physical space boundaries and jointly fulfill dynamically varying performance and space requirements. On this architecture, with multiple schemes implementing the most used RAID organisations (RAID0-1 and RAID5), requests of different type and size are executed in parallel on asynchronous disks connected via a bus to the RAID controller. The bus data transfers

doi:10.1016/j.entcs.2009.02.047

occur, with a wide range of lengths, at different steps during the requests' execution. The bus is a critical shared resource for these essentially independent, asynchronous parallel disks and is subject to congestion. The delay so introduced has a significant impact on the whole system response time, especially for the RAID5 partial stripe writes. Modelling such a storage architecture and related mechanisms is important in analysing the performance it can deliver and predicting its behaviour for different workloads. We have subsequently elaborated the disk-array model and carefully validated it, in particular the approximate solution for the specific fork-join problem that arises in assuming independent operation of the disks [5]. We extended this work to modern zoned disks by finding appropriate seek distance distributions, linear and non-linear in [14,15] respectively.

Other work in the area relate to single RAID levels and their performance in a specific working mode (normal or degraded) [1,8,10]. Some of these studies focused on the delivered throughput [9], which can be limited by the shared bus bandwidth.

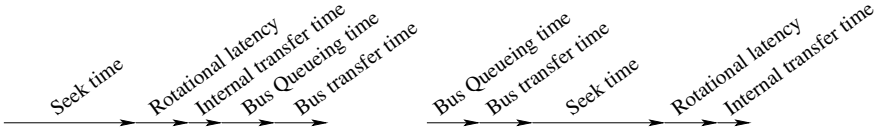
In this paper, we focus on the bus connecting devices in single RAID organisations as well as Multi-RAID systems. The very specific context characteristics which motivate this study are discussed in Section 2, the proposed model is described in Section 3 and its parameters are determined in Section 4, mainly as moments, using the detailed description of the principles of operation of the system. Validation against simulation is shown in Section 5 and the paper concludes in Section 6.

2 Context and motivation

We model the RAID-connecting bus using an M/G/1 queue – extracting its input rate from traffic generated in various contexts – before, during or/and after the disk's service period. We then evaluate its effect on the overall IO response time.

In RAID storage systems, an IO request's execution varies according to its type (read/write) and size (data blocks), as well as the RAID organisation type [1] [2]. The resulting different combinations may lead to the transfer of data on the bus to a native, mirror or parity disk; before, during or/and after an actual disk service, on one or more different disks. The transfers before and after service are well studied, in both the parallel computer architecture and, especially, the networking communities [13,4]; the former, of course, is closer to our interest in RAID storage systems. The presence of transfers at different steps in each overall disk access splits the process into phases separated by bus delays, each of which has a different impact on the whole response time. In fact, a delay can change the execution of the following phase and this impact is more important in a Multi-RAID system that has much more complex execution schemes, hence generating more complex bus traffic. The simplest case is a read or write request on a RAID0-1 (striped and mirrored RAID). The bus transfer is performed once: after the disk service for reads and before the disk service for writes, as shown in Figure 1. However, the number of data blocks (native or mirror) transfered to/from each disk is different in each case and depends on the request type and size as shown in Figures 2 and 3 for

[1] From RAID0 to RAID6, as defined in Berkeley's classification.



(a) RAID01 physical read diagram (b) RAID 01 physical write diagram

Fig. 1. RAID0-1 service time diagram

an 8-block request from $B0$ to $B7$. On RAID0-1, we can read either the (native) block on a native disk or its mirror on the mirror disk, according to the selected scheduling policy [2]. In the example shown in Figure 2 [3], native versions of blocks $\{B0, B1, B2, B6\}$ are read on native disks $\{disk0, disk1, disk2\}$ and mirror versions of the remaining blocks. This scheduling leads to a two-consecutive-blocks read request on $disk1$ (native) and $disk3$ (mirror) and a single block read on the other disks.

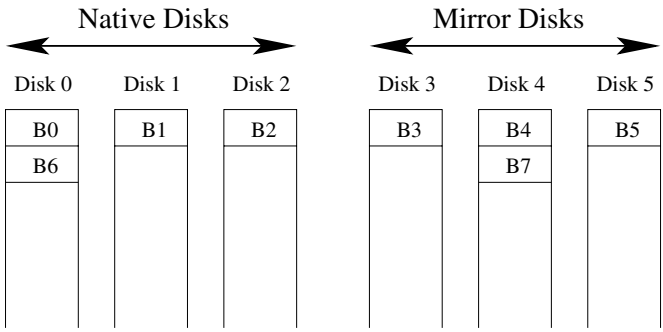


Fig. 2. RAID0-1 logical read

The RAID0-1 write requests need an access to both native and mirror versions on the associated disks. The 8-block logical request then generates a three-consecutive-blocks request to $\{disk0, disk1\}$ to write the native version of blocks $\{B0, B3, B6\}$, as well as their mirrors $\{disk3, disk4\}$ to write the mirror version of the same blocks. The native disk $\{disk2\}$ as well as its mirror $\{disk5\}$ is accessed by a two-consecutive-blocks request $\{B2, B5\}$.

RAID5 (distributed parity) is much more complex due to its data and redundancy patterns. On an N disks RAID5, a stripe is a collection of $(N-1)$ data units with an associated single parity unit controlling them, each of which is on a different disk. This is also called a full stripe because it covers all the disks. A partial stripe is a stripe with a parity block and less than $(N-1)$ data blocks. For every stripe, regardless of its width, the parity stripe unit (block) is calculated and associated with a disk in a round-robin manner. A read on a RAID 5 is similar to

[2] Random, shortest queue, shortest seek distance,...etc.

[3] Only accessed blocks are indicated on disks for all Figures.

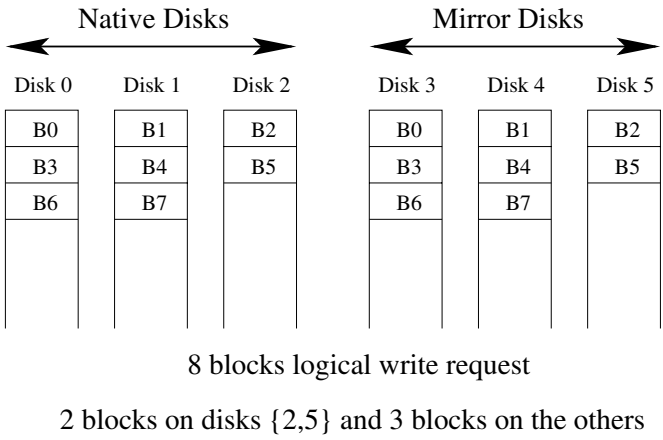


Fig. 3. RAID0-1 logical write

RAID0-1, with unique target disks rather than having the choice between a native and a mirror target. However, the write is completely different and much more complex. According to a logical request's size, the generated stripes may be full, large or small, respectively when a physical request's size covers all data units in the stripe, at least half of the data stripe units and less than half of the data stripe units. A RAID5 write might generate a combination of these three write types: full write(s) followed by a large or a small one. Every write request among these three and their possible combinations has an appropriate parity calculation and pre-read operations that involve different numbers of disks. These are respectively none of the disks for the full stripe writes, the non-target disks (those not concerned by the write request) and the parity disk only for large stripe writes, and the target disks with the parity disk only for small stripe writes, as shown in Figure 4. In the case of a combination (full followed by large or small stripe), the parity calculation is performed according to its two components. The transfer is performed once for reads (after the disk service) and according to the diagram in Figure 5 for writes. The Multi-RAID we considered is a combination of RAID0-1 and RAID5, which makes its generated data transfers quite complex combinations of the diagrams in Figures 1 and 5.

3 Bus response time model

We analyse the bus response time of a logical request by considering its type (read/write), its size (number of blocks) and the system's workload intensity (arrival rate in terms of the number of logical requests per second). We model the delay caused by contention at the bus by a conventional M/G/1 queue with a single workload class, which is the aggregate of all IO transfer types and sizes for the different coexisting RAID organisations. Because there are multiple, diverse input streams of IO requests arriving at the bus, which behave independently to a great extent, the Poisson assumptions are not unreasonable [12]. The challenge is to estimate the probability distribution of these sizes – or at least its moments – and to

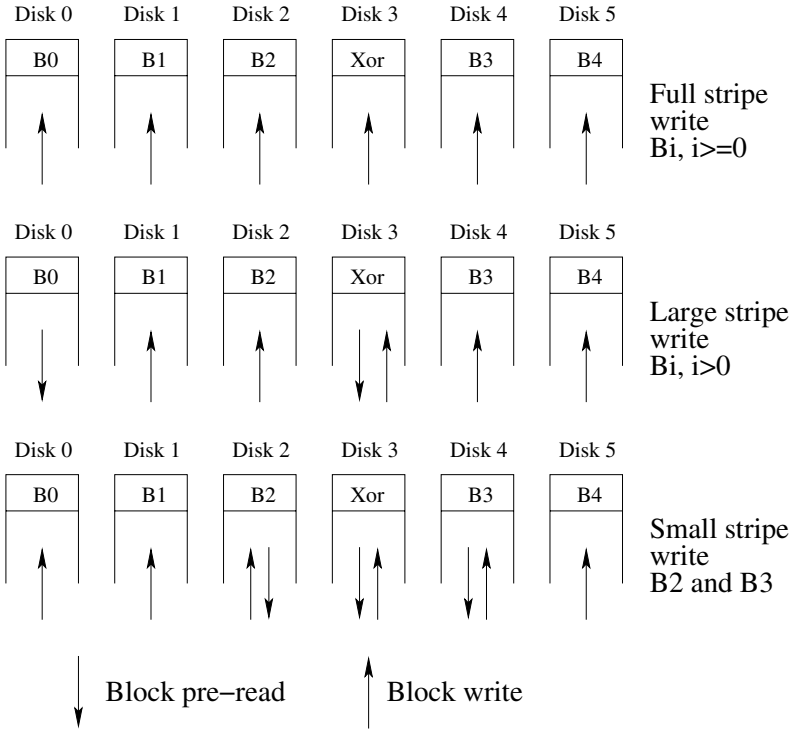


Fig. 4. RAID5 logical writes

calculate the aggregate arrival rate of such transfers, a rather easier task. The bus response time is then composed of the queueing time Q_{bus} and a service time S_{bus} that depends only on the size of the particular IO transfer in question :

$$T_{bus} = Q_{bus} + S_{bus} = Q_{bus} + K \times T$$

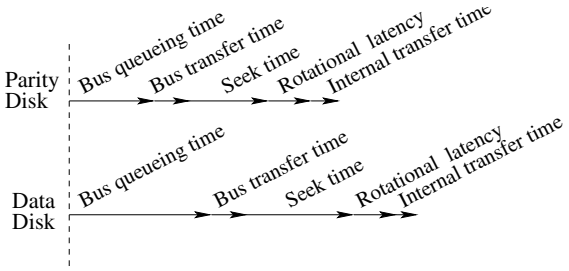
where K is the size of the IO transfer (in blocks), related to a logical request and estimated in Section 3.2, and T is the time taken to transmit one block, the reciprocal of the bus bandwidth.

3.1 Bus Queueing time (Q_{bus}) :

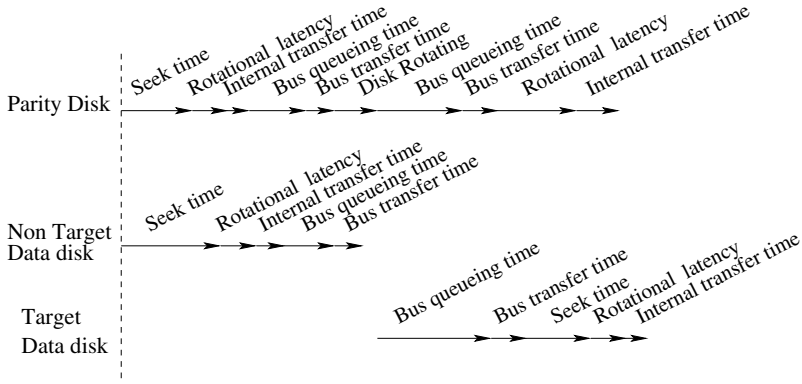
We use the Pollaczek-Khinchin formula [6], as used for the queueing time at the disk level in [5], with bus-related parameters:

$$E[Q_{bus}] = \frac{\lambda_{bus} S_{bus}}{2(1 - \rho_{bus})} \quad (1)$$

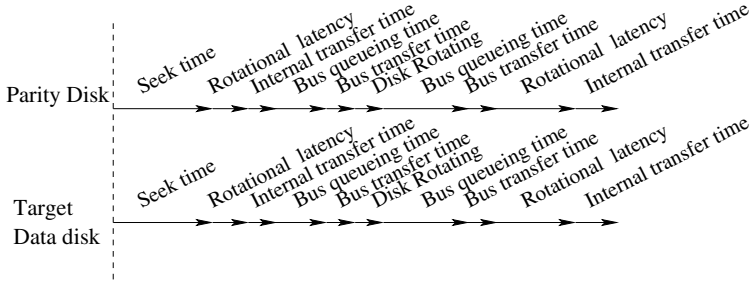
The traffic load is now defined by:
$\rho_{bus} = \lambda_{bus} \times S_{bus}$
where S_{bus} denotes the mean bus service time and, as we will use later, S_{bus} denotes its second moment. In fact, we use n overbars to denote the nth moment of a random variable in general.
As we are considering a Multi-RAID storage system, the bus traffic intensity λ_{bus}



(a) Full stripe write



(b) Large stripe write



(c) Small stripe write

Fig. 5. RAID5 service time diagram

derives from a mixture of RAID layout requests on top of the different types of IO transfer generated by the requests' pre-reads, native and mirror accesses, as shown in Figures 1 and 5. Considering the two most used RAID organisations, RAID0-1 (mirroring and striping) and RAID5 (distributed parity), the bus traffic can be detailed as below [4]:

$\lambda_{bus} = \lambda_{bus\_RAID0-1} + \lambda_{bus\_RAID5}$

where

$\lambda_{bus\_RAID0-1} = \lambda \times P_{raid0-1} \times B(2 - p_r)$, and

[4] "%" denotes the modulo function.

$\lambda_{bus\_RAID5} = \lambda \times P_{raid5}(p_r \times B + p_w((cdt_{full} \times N \times [\frac{B}{(N-1)}]) + (cdt_{lg}(N+1)) + (cdt_{sm} \times 2(B\%(N-1)+1))))$
B is the logical request's size (number of blocks), p_r and p_w are the probabilities of a request to be a read or a write respectively, $P_{raid0-1}$ and P_{raid5} are the probabilities of accessing the RAID0-1 area or the RAID5 area respectively, and finally λ is the arrival rate of the logical requests.

The boolean parameters cdt_{full}, cdt_{lg} and cdt_{sm} indicate respectively if the stripe is full, large or small [5] [5] . They are defined for RAID5 write requests and depend on the logical request size (B) and the RAID5 width (N disks):
$cdt_{full} = ([B/(N-1)] > 0)$,
$cdt_{lg} = (B\%(N-1) >= [N/2])$ and
$cdt_{sm} = (B\%(N-1) < [N/2])$

3.2 Bus service time (S_{bus}) :

A bus service consists of transfering K data blocks (native/mirror/redundancy) via the bus to/from the disk devices from/to the RAID controler. The number of blocks to transfer depends on the request type, request size and the associated RAID organisation as follows :
For the RAID0-1 read and write operations :

$$K_{R01} = \begin{cases} B & \text{w.p. } p_r \\ 2 \times B & \text{w.p. } p_w \end{cases}$$

For RAID5 operations, in addition to the read and write modes, we need to distinguish full stripes from partial (small or large) stripes. Hence we have:

$$K_{R5} = \begin{cases} B & \text{w.p. } p_r \\ K_{R5W} & \text{w.p. } p_w \end{cases}$$

[5] We interpret "false" as 0 and "true" as 1 to simplify later calculations.

Then the number of data (native and parity) blocks transfered via the bus for a RAID5 write (K_{R5W}) can be calculated for every possible case as:

$$K_{R5W} = \begin{cases} K_f & \text{if } (cdt_{full} \times (1 - cdt_{lg}) \times (1 - cdt_{sm})) \\ K_{lg} & \text{if } (cdt_{lg} \times (1 - cdt_{full})) \\ K_{sm} & \text{if } (cdt_{sm} \times (1 - cdt_{full})) \\ K_f + K_{lg} & \text{if } (cdt_{full} \times cdt_{lg}) \\ K_f + K_{sm} & \text{if } (cdt_{full} \times cdt_{sm}) \end{cases}$$

$K_f = [\,{}^{B}_{(N-1)}] \times N$
$K_{lg} = N + 1$
$K_{sm} = ((B\%(N-1)) + 1) \times 2$

4 Moments of bus delay

We wish to calculate the mean and variance of the bus delay and consequently require the first *three* moments of the:

(i) **Number of blocks to transfer (K) :**
The nth moment of K is simply $K^n = P_{raid01}K^n_{R01} + P_{raid5}K^n_{R5}$ where the moments of K_{R01} and K_{R5} are respectively

$$K^n_{R01} = (p_r + 2^n p_w)B^n$$
$$K^n_{R5} = p_r B^n + p_w\Big(cdt_{full}(1 - cdt_{lg})(1 - cdt_{sm})K^n_f + (1 - cdt_{full})cdt_{lg}K^n_{lg} + (1 - cdt_{full})cdt_{sm}K^n_{sm} + cdt_{full}cdt_{lg}(K_f + K_{lg})^n + cdt_{full}cdt_{sm}(K_f + K_{sm})^n\Big)$$

(ii) **Service time (S_{bus}) :**
The first three moments of the bus service time are simply:

$$S_{bus} = \overline{K} \times T, \qquad S_{bus} = T^2 \times \overline{\overline{K}} \qquad \text{and} \qquad S_{bus} = T^3 \times \overline{\overline{\overline{K}}}$$

(iii) **Bus queue (Q_{bus})**
The first moment is calculated using the Pollaczek-Khinchin formula as in Equation 1. The second moment can be used to assess the accuracy of our model and is obtained as (see, for example, [5]):

$$Q_{bus} = \frac{\lambda^2 S_{bus}^{\ 2}}{2(1 - \rho_{bus})^2} + \frac{\lambda S_{bus}}{3(1 - \rho_{bus})} \tag{2}$$

5 Bus model validation

In order to validate our model, we developed a hardware simulator representing all the architecture components and the functions they perform. The simulator is event-driven, written in C, and composed of 3 modules: the workload generator, the logical to physical address translator and the event (including IO requests) diary and execution engine. For scalability, portability and reliability, the execution engine uses hardware libraries for hardware characteristics such as the disks, connecting bus ...etc. The workload generator and the model share certain common assumptions: uniform distribution of requests' addresses over the storage space and Poisson arrival streams of requests. We focus on the impact of three parameters on the bus queueing time: the request type (read/write), by varying the probability of a logical request being a read (p_r); the logical request size, by varying B; and the architecture configuration (i.e. RAID organisation here), by considering an exclusive RAID0-1, an exclusive RAID5 or a mixture of the two, using the above model and the event-driven simulator for a storage system composed of 16 FujitsuMAN3367 disks connected to a 40MB/s Wide Ultra SCSI bus.

However, we first examine the relation between the request size B and the generated traffic (K), which shows the heavy traffic source of bus contention.

(i) **Request size vs. generated traffic**
For RAID5 writes, Figure 6 confirms the significant effect of the requests' sizes on the generated traffic and the variations in this traffic according to the stripe width (full, large or small). As we are considering a 16-disk storage array, organised as a RAID5, full stripe writes are obtained for logical request sizes $B \in \{j + 15i \ , \ i > 0, \ \forall j\}$, small stripe writes are obtained when $B \in \{j + 15i \ , \ i \geq 0 \ , \ j \in [1..7]\}$ and large stripes are obtained when $B \in \{j + 15i \ , \ i \geq 0 \ , \ j \in [8..14]\}$. The generated traffic increases with B during small stripe phases, decreases as B increases during large stripe phases (because the number of pre-read IOs decreases), and finally reaches its lowest values for full stripes. Figure 6 shows this effect for an exclusive write request stream ($p_r = 0$) and a mixed request stream with equal proportions of reads and writes ($p_r = 0.5$), contrasting with an exclusive read stream ($p_r = 1$) for which the generated traffic is linear in the logical request size.

For RAID0-1 writes, the generated traffic is proportional to the logical request size B with a factor of two. Considering a Multi-RAID storage system with an equal proportion of RAID0-1 and RAID5 on Figure 7, we can see the explosion of the generated traffic in an exclusive small write mode. In this case, the generated bus traffic is composed of the double RAID0-1 logical requests and quadruple RAID5 ones.

(ii) **Request type**
In order to analyse the effect of reads and writes on bus queueing behaviour, we vary the probability of a request being a read, p_r, from 0, for an exclusive read workload, to 1 for an exclusive write workload, including 0.5 for a balanced workload. We can see the variation for RAID0-1 in Figure 8, the model and

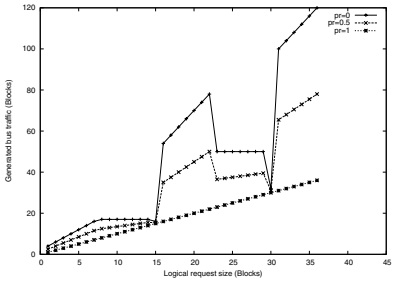


Fig. 6. Generated transfer blocks on RAID5

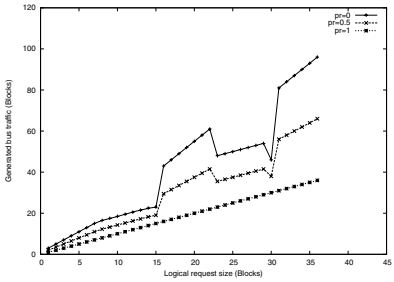


Fig. 7. Generated transfer blocks on Multi-RAID

the simulation results showing good agreement and indicating the increase in the queueing time with the proportion of writes, doubling the traffic for this category of RAID. This is confirmed for RAID5 as well, as shown in Figure 9 where the traffic is multiplied by four for small ($B = 1$) write requests.

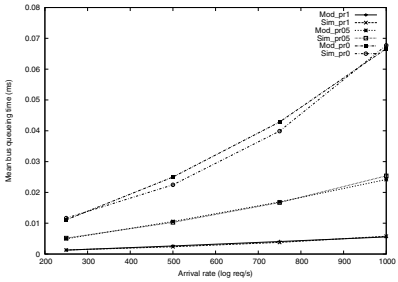


Fig. 8. Request type impact on RAID0-1 ($B = 1$)

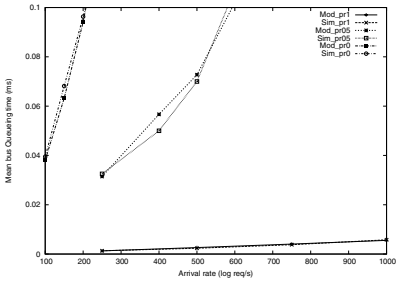


Fig. 9. Request type impact on RAID5 ($B = 1$)

(iii) **Request size**
We observed the behaviour of the bus at various request sizes, going from one block (4KB) to 4 blocks, then 8 blocks for RAID0-1. Figure 10 shows the impact of the logical request size on the generated bus traffic and hence on the bus queueing time, mutually validated by the analytical model and the simulation, which show good agreement. We choose the exclusive read mode to isolate the size impact from any write operation effect. We can see that for 8-block requests, the queueing time rises rapidly at relatively low arrival rates (reaches 9ms at 150 req/s). RAID5 reads are similar to RAID0-1; thus Figure 10 is representative of both organisations.

(iv) **Architecture configuration**
We notice different traffic intensities generated by the same logical request stream on an exclusive RAID0-1 storage system, on an exclusive RAID5 storage system and on a mixture of both in a Multi-RAID with equal proportions. Figure 11 shows clearly the effect of the architecture configuration on the generated traffic and the agreement of the model and the simulation results.

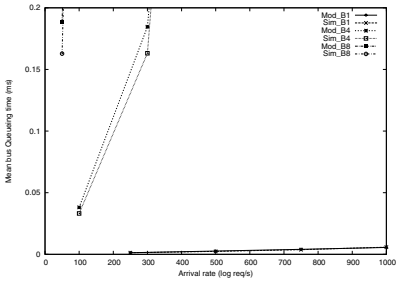


Fig. 10. Request size impact (RAID0-1)

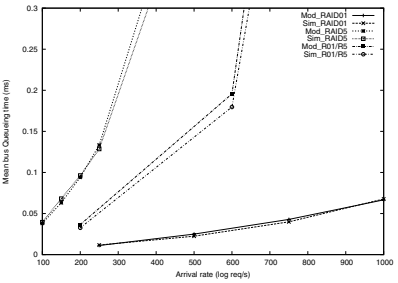


Fig. 11. Architecture config. impact on Q_{bus}

6 Conclusion

In this paper, we took an additional step in our hierarchical multi-disks storage system model: the disks' connecting component model and can be integrated with the disks array one. This is one of our Intelligent Performance Optimisation of virtualised Data Storage systems (IPODS) project's aims. The model showed good accuracy when compared with simulation in preliminary experiments. Further validation is required in which we consider request arrival streams with non-uniform addresses, and higher moments of response times and much larger file sizes. The model should then be validated against data monitored from an actual RAID system in a controlled environment. Similarly, at greater computational expense, response time distribution functions themselves could be obtained from the Pollaczek-Khinchin result for Laplace transforms, using a numerical inverter. This is likely to be much more sensitive to validation, especially in the tail. Extension of our storage architecture to a matrix of disks for very large systems can be achieved using this work, by adding new bus components and related connections in the simulator and extending our model to a multi-bus version using an M/G/n queue. Finally, it remains to model the controlling component of a RAID system, with its request scheduling schemes and data caching policies, which influence heavily the net storage system performance.

References

[1] S. Chen and D. Towsley. A performance evaluation of raid architectures. *IEEE Transactions on Computers*, 45(10), October 1996.

[2] G. Gibson D. A. Patterson and R. H. Katz. A case for redundant arrays of inexpensive disks (RAID). In *Proceedings of SIGMOD Conference*, June 1988.

[3] J. Xu E. Varki, A. Merchant and X. Qiu. An integrated performance model of disk arrays. In *proc. International Symposium onModelling, Analysis and Simulationof Computer and Telecommunications Systems (MASCOTS)*, 2003.

[4] Michael J. Flynn. *Computer architecture-pipelined and parallel processsor design.* Jones and Bartiett Publishers, 1995.

[5] P.G. Harrison and S. Zertal. Queueing models of RAID systems with maxima of waiting times. *Performance Evaluation*, 2007.

[6] Ng Chee Hock. *Queueing Modelling Fundamentals.* John Wiley Publisher, 1996.

[7] C Staelin J. Wilkes, R. Golding and T. Sullivan. The hp autoRAID hierarchical storage system. *ACM transactions on Computer systems*, 14, Feb 1996.

[8] E.K. Lee and R.H. Katz. An analytic performance model of disk arrays. In *Proc. ACM SIGMETRICS*, May 1993.

[9] G. A. Alvarez M. Uysal and A. Merchant. A Modular, Analytical Throughput Model for Modern Disk Arrays. In *Proceedings of the International Symposium onModelling, Analysis and Simulation of Computer and Telecommunications Systems (MASCOTS)*, August 2001.

[10] A. Merchant and P.S. Yu. An analytical model or reconstruction time in mirrored disks. *Performance evaluation*, 20, May 1994.

[11] Sangsoo Park and Heonshik Shin. Rigorous modeling of disk performance for real-time applications. In *proc. Real-Time and embedded Computing Systems and Applications (RTCSA)*, 2003.

[12] B.O. Shubert and H. J. Larson. *Probabilistic Models in Engineering Sciences*. John Wiley Publisher, 1979.

[13] A. van de liefvoort and N. Subramanian. A new approach for the performance analysis of a single-bus multiprocessor system with general service times. *IEEE Transactions on Computers*, 42, Mar 1993.

[14] S. Zertal and P.G. Harrison. Multi-RAID Queueing Model with Zoned Disks. In *High Performance Computing and Simulation*, 2007.

[15] S. Zertal and P.G. Harrison. Non-linear seek distance for optimal accuracy of zoned disks seek time in multi-raid storage systems. In *High Performance Computing and Simulation*, 2008.