ELSEVIER

Simulation Practice and Theory 7 (1999) 153–170

SIMULATION PRACTICE AND THEORY

Performance vs. cost of redundant arrays of inexpensive disks [1]

Francesco Quaglia *, Bruno Ciciani

Dipartimento di Informatica e Sistemistica, Università "La Sapienza" Via Salaria 113, 00198 Rome, Italy

Received 1 November 1997; received in revised form 8 December 1998

Abstract

Data redundancy has been widely used to increase data availability in critical applications and several methods have been proposed to organize redundant data across a disk array. Data redundancy consists of either total data replication or the spreading of the data across the disk array along with parity information which can be used to recover missing data in the event of disk failure. In this paper we present an extended comparative analysis, carried out by using discrete event simulation models, between two disk array architectures: the Redundant Arrays of Inexpensive Disks (RAID) level 1 architecture, based on data replication; and the RAID level 5 architecture, based on the use of parity information. The comparison takes both performance and cost aspects into account. We study the performance of these architectures simulating two application environments characterized by different sizes of the data accessed by I/O operations. In addition, several scheduling policies for I/O requests are considered and the impact of non-uniform access to data on performance is investigated.

Keywords: RAID; I/O subsystems; Simulation study; Data replication; Redundant information; Non-uniform access

1. Introduction

The spreading of data across a disk array usually increases the performance of the I/O subsystem (compared to what happens when using a single disk) because I/O operations are allowed us to execute concurrently [19,22]. On the other hand, if a large number of disks are used, the probability of some disk failing may increase to unac-

* Corresponding author. E-mail: quaglia@dis.uniroma1.it.

[1] This is a revised/extended version of a paper appeared in Proceedings of ISCS'96 Conference – Rome, Italy, 18–19 December, 1996.

0928-4869/99/$ – see front matter
PII: S0928-4869(98)00026-3

ceptable levels, thus leading to the need for mechanisms to recover data in case of disk failure. To this purpose, two different approaches have been proposed, both based on data redundancy, and the resulting disk array architectures are known collectively as Redundant Arrays of Inexpensive Disks (RAID) systems [4].

In the first approach, namely *data replication*, two copies of the same data are stored on different disks, so that, if one disk fails, the data is still available from the other disk [11]. If the two data copies are simultaneously active (i.e., they can both be used for read/write operations), the additional data copy improves not only availability, but, usually, also performance during both normal and failure modes. On the other hand, coherency requires the updating of both copies when write operations occur, which, in turn, introduces an overload on the I/O subsystem. As a consequence, performance of data replication is good mostly in the presence of workloads characterized by a high percentage of read operations.

In the second approach, data is spread across the disk array along with *parity information* that is used to recover missing data when a disk fails [10,17,21]. Usually this approach does not require a large amount of physical disk space to be reserved for redundancy, thus being attractive from an economic point of view. However, it is prone to drawbacks due to the updating of the parity information when a write operation is executed on data. Consequently, as in the case of data replication, the system performance is affected by the workload characteristics.

In the case of data replication, several techniques have been proposed to organize the two copies of data. In these techniques usually both copies are active for access during normal operation [1,11]. In the *mirrored declustering* technique, the disks are organized into a set of identical pairs (the resulting disk array architecture is known as RAID1). In *interleaved declustering*, the backup copy of the primary data in a disk is distributed over the other disks in the array. In *chained declustering*, the primary data copy in disk i has a backup copy in disk $i + 1$ (it has been shown that this solution leads to higher availability in the shared-nothing environment [11], i.e., an environment where distinct processes have distinct address spaces).

Data replication has been studied from the point of view of both performance and reliability [2,7,9,15]. The performance analysis has been carried out by using both simulation and analytical models, assuming, in most of cases, workload uniformly distributed over the disks. Concerning non-uniform access to data, an analysis is presented in Ref. [12] for the case of chained declustering, and in Ref. [20] a study is proposed to point out the trade-offs between replication techniques and striping strategies in the context of database environment, where the workload consists of both transactions and read-only queries. For such an environment, a dual striping strategy is presented in Ref. [20], which allows different stripe sizes for the two active copies of data.

In the case of data availability by means of parity information, the most commonly used disk array architecture is known as RAID5. In this architecture, the parity information is spread across all the disks in order to overcome the presence of hot spots due to parity information updating. This architecture has been the object of several performance and reliability studies. In Ref. [7] an analytical model of RAID5 systems has been introduced. In Ref. [3,8], the performance of RAID5 has

been studied through simulation, especially to investigate the effect of the striping unit size on the performance. The obtained results have shown that the best striping is not the same when considering read-intensive and write-intensive I/O operations. In Refs. [6,18] an analysis of strategies for spreading parity information on the disk array has been presented. However, as for the case of data replication, most of the attention is focused on workloads uniformly distributed over the disk array.

The cost of a RAID architecture is determined by the number of disks and the cost of a single disk; whereas, the storing capacity is the amount of memory space destined for data (i.e., it does not include space destined for redundant information). Given the different kinds of redundancy, RAID1 and RAID5 usually have different costs for the same amount of storing capacity (usually RAID1 is more expensive than RAID5 because of the need for physical disk space to store the secondary copy of data). On the other hand, RAID1 and RAID5 systems with the same cost (i.e., the same number of disks) may have very different storing capacities. Therefore, the cost of an architecture and its storing capacity are parameters which cannot be neglected while defining the goodness of the architecture itself.

In this paper we propose a comparative analysis of RAID1 and RAID5 architectures, focusing our interest on both performance (i.e., response time) and cost/storing-capacity aspects. To this purpose, we consider RAID1 and RAID5 architectures having either the same cost or the same storing capacity, and we study the response time vs. the I/O request arrival rate.

We have carried out the performance comparison by using discrete event simulation models, considering two different application environments: OLTP (on-line-transaction-processing), where each I/O operation accesses 1 block (4096 bytes), and LARGE (e.g., scientific computations, image processing), where each I/O operation accesses 256 blocks (1 MB). We have considered different sizes for the striping unit and, in the case of RAID1, three different scheduling policies (Random-Join, Shortest-Queue and Minimum-Seek) for dispatching read requests to one or the other copy of data [7] have been simulated. Furthermore, our analysis focuses on both uniform and non-uniform access to data, and takes both read-intensive and write-intensive workloads into account.

Beyond performance and cost/storing-capacity aspects, another basic feature of RAID architectures is the reliability. This feature is taken into account in the paper in a section devoted to a comparison of the architectures based also on other metrics. Reliability is here evaluated by using the analytical model presented in Ref. [21].

The remainder of the paper is organized as follows. In Section 2, RAID1 and RAID5 architectures are described, together with the scheduling policies. In Section 3, we briefly present the RAID systems simulator and characterize the workloads used during the experiments. In Section 4, the simulation results are presented and discussed. Section 5 is devoted to compare the two architectures by using other metrics (e.g., reliability etc.). Finally, some concluding remarks are given in Section 6.

2. Disk array architectures

In this section we first introduce basic concepts and assumptions on the disk array; then we describe in detail RAID1 and RAID5 architectures. While describing these architectures, we also report the parameter values selected for the simulation experiments.

2.1. Disk array model

We consider a disk array consisting of N identical disks with synchronized rotations and non-synchronized arms. Data is block interleaved across W disks, where W is called the stripe size ($W \leqslant N$). Each file is logically divided into blocks F_1, F_2, ..., F_n, which are stored on contiguous disks. Note that n may be larger than W, in this case the blocks of the file are distributed over more than one stripe (see Fig. 1). Each I/O request accessing an integer multiple of W blocks in a stripe (i.e., aligned with the stripe boundaries) is called *full stripe* I/O request, otherwise it is called *partial stripe* I/O request. Each disk has its own queue; data requests (jobs) in this queue are served in FCFS order. Disk parameters adopted in the simulation experiments are similar to those used in Ref. [7] and are reported in Table 1.

2.2. RAID1 architecture

In RAID1 architecture, data is spread across disks following the *mirrored declustering* policy. The N disks of the array are configured as $N/2$ pairs of mirrored disks, and the ith pair is called *mirror i*. Data is totally duplicated. The two copies are striped over the $N/2$ mirrors and each mirror contains both the primary and the secondary copy of the same data. Thus only the uniform striping is allowed given that both copies of the data are broken into partitions of the same size and dual striping

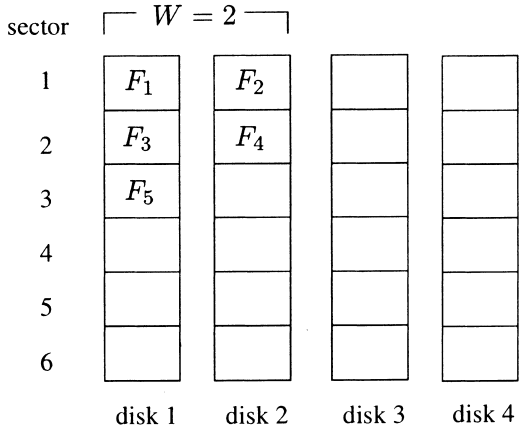

Fig. 1. An example of disk array with $N=4$ disks and stripe size $W=2$.

Table 1
Disk parameters

Parameter	Value
Number of cylinders (C)	1200
Transfer rate	3 MB/s
Time for a complete rotation	16,7 ms
Number of blocks in each trace	12
Acceleration time	3 ms
Searching factor	0,5
Block transfer time	1,3 ms
Block dimension	4096 bytes

strategies, such as those presented in Ref. [20], cannot be applied. We will consider different stripe sizes in our experiments, and point out the impact of the stripe size on performance.

In Fig. 2, a RAID1 architecture with 6 disks is shown. Blocks F_i and f_i are, respectively, the primary copy and the secondary copy of the ith block of file F. The two copies of data are not necessarily stored at the same location on the two mirrored disks; in our experiments we considered an offset of $C/2$ (i.e., 600 cylinders).

Having both the copies active, RAID1 permits the completion of a read operation by accessing one or the other copy of data, whereas a write request must be dispatched to both copies. Three main policies have been proposed for dispatching a read request to one or the other disk of the mirror [7]. A dispatcher executes the scheduling actions by selecting the disk a read request has to be sent to. On the other hand, when a write request arrives, two tasks are generated which enter both queues of the disks in a mirror. The policies for scheduling read requests are reported below.

Random Join (RJ) policy: A read request is randomly assigned to the queue associated to disk i with probability α_i (where $i = 1,2$ and $\alpha_1 + \alpha_2 = 1$). With regard to simulation parameters, we selected the following values $\alpha_1 = \alpha_2 = 0.5$.

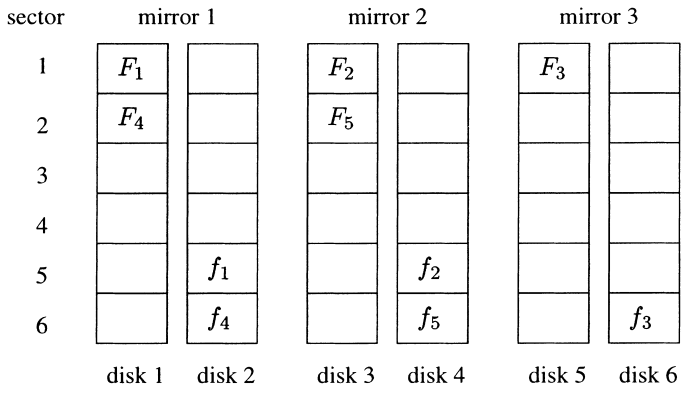

Fig. 2. An example of the RAID1 architecture with $N = 6$ disks.

Shortest Queue (SQ) policy: A read request is assigned to the disk with the shortest queue at the time of the request arrival (ties are broken in an arbitrary manner). If both disks are idle at the arrival time of the request, it is assigned to the disk whose arm is closest to the cylinder containing the block to be read.

Minimum Seek (MS) policy: A read request is assigned to the disk where the last job in the queue has the minimum seek distance from the new request.

2.3. RAID5 architecture

In the RAID5 architecture, data availability is achieved by using parity information which is distributed across the disk array following the *block-interleaved distributed parity* technique. In this architecture, only one block of redundant information is introduced for every $N-1$ blocks of data; furthermore, parity information is spread across all disks in order to prevent the parity information updating from becoming a bottleneck of the system.

Parity blocks are computed as the XOR of all the other blocks in the same stripe, thus, if one disk fails, the array is still operational since the missing data can be restored by executing an XOR operation of the data on all the other disks.

Performance of RAID5 may be poor in environments where write operations partially update a stripe. In this case, the new parity block for the modified stripe must be computed by executing an XOR operation between old data, new data and old parity, thus old data and old parity have to be read before the new parity is computed. This problem disappears in environments where write requests update a complete stripe (the new parity is computed by executing the XOR of all new data blocks).

An example of the RAID5 architecture with 5 disks is shown in Fig. 3, where P_i denotes the ith parity block (i.e., the parity block for the ith stripe).

Several methods for distributing parity information across the array have been proposed and analyzed in the literature (the reader can refer to Ref. [14] for a complete description). In our simulation study we consider the *left-asymmetric* distribution of parity blocks, which has been shown to ensure the best performance under

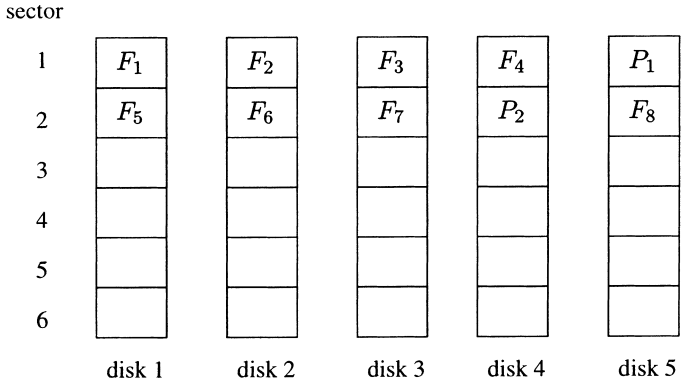

Fig. 3. An example of the RAID5 architecture with $N = 5$ disks.

several different workloads [14]. According to this distribution, the parity block of the first stripe is on disk N, the one of the second stripe is on disk $N-1$ and so on, and data blocks are stored according to increasing values of their index while moving from disk 1 to disk N (as an example, the architecture shown in Fig. 3 adopts the left-asymmetric distribution).

The most widely used scheduling policy for RAID5, namely *after read-out* (AR), has been proposed in Ref. [5] (we adopted this policy in the simulation study). Under AR, two queues are maintained for each disk: one (D queue) for incoming read/write requests, the other (P queue) for parity update requests. When a read/write request arrives, the dispatcher sends it to the D queue. When a write request is scheduled for service, it reads the old data and then it generates a parity update request which is sent to the P queue of the disk containing the corresponding parity block. The P queue has higher priority than the D queue in order to ensure that an outstanding parity request starts as soon as possible, since the corresponding data update has already started. The description of other scheduling policies is out of the aim of this paper. The interested reader can refer to Ref. [5] for a complete description.

3. The RAID simulator

We compared performance of RAID1 and RAID5 by using the S-RAID-UR (Simulator RAID of University of Rome) simulator of disk array systems, developed at the University of Rome "La Sapienza". This simulator is event driven and has the capability to model several disk array architectures including non-redundant arrays and redundant arrays (from level 1 to level 5). The simulator does not model CPUs, buses or other components out of the I/O subsystem, thus, the only resources explicitly simulated are disks. To this purpose, a special module for the generation of synthetic workloads is included.

Statistics have been collected by using the independent replication methods [13], and the confidence interval is based on the Jackknife estimator [16]. The simulation run length for a specific workload depends on the statistics gathered during the execution. In particular, the run stops when the mean response time is within 5% of the true mean at 95% confidence level.

Concerning RAID1, in Fig. 4 we show the basic structure of the simulation model for a single pair of mirrored disks. λ_{in} represents the arrival rate of I/O requests to that mirror; P_r represents the probability for a request to be a read one. Each write request is sent to the queues associated to both mirrored disks. Instead, a read operation passes through a dispatcher which selects the queue the request has to be sent to. The dispatcher implements one of the scheduling policies described in Section 2 (RJ, SQ or MS).

Concerning RAID5, the simulation model has the basic structure shown in Fig. 5. A dispatcher selects the disk a request has to be sent to, and forward it to the D queue of the selected disk. If the request is a write one, a parity information update request is produced and is sent to the P queue of the disk storing the corresponding

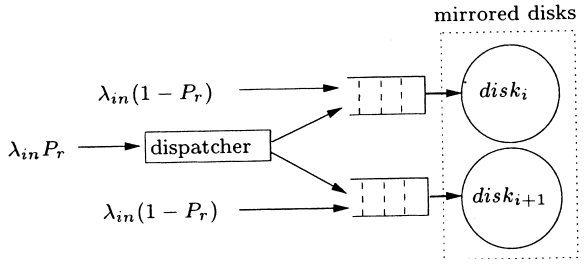

Fig. 4. A detail on the simulation model of RAID1.

parity block. As already outlined in Section 3, pending requests in the P queue of any disk have higher priority than those in the D queue.

Two different application environments have been simulated, characterized by different sizes for the amount of data accessed by a read/write request. In the OLTP (on-line-transaction-processing) environment, each request accesses a single data block (4096 bytes), thus this environment is representative of database applications. In the LARGE environment, requests access 256 data blocks (i.e., 1 MB); this environment nicely models such applications like image processing. Furthermore, for each environment we consider two different values of the probability P_r for an I/O request to be a read one. These values are $P_r = 0.75$ and $P_r = 0.25$, so both read-intensive and write-intensive workloads are considered.

We assume the interarrival time between I/O requests distributed according to the Poisson process. The cylinder accessed by each request is chosen according to a uniform distribution.

For the OLTP environment we study the effect of both uniform and non-uniform access to data on performance. For the LARGE environment, the investigation on non-uniform access is much less significant as I/O requests are full stripe ones, thus loading all the disks in a group. Therefore, for this environment only the uniform access is considered.

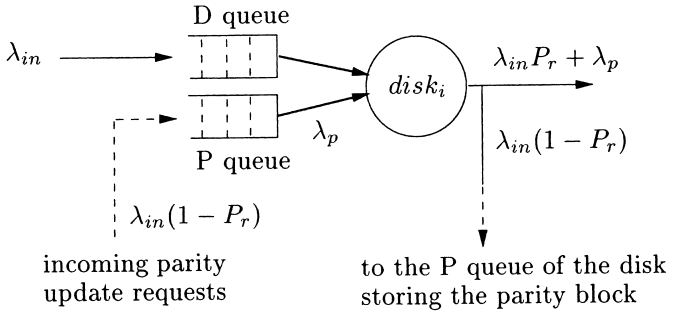

Fig. 5. A detail on the simulation model of RAID5.

4. A comparison between RAID1 and RAID5

4.1. Results in the OLTP environment

We simulated a RAID1 architecture with $N = 16$ disks organized in 8 mirrors, thus, the stripe dimension W is equal to 8 for both the primary and the secondary copy of data. Performance of such architecture is compared to that of two distinct RAID5 architectures, one having the same cost and one having the same storing capacity of the considered RAID1. The RAID5 architecture with the same cost has 16 disks, whereas the RAID5 architecture with the same storing capacity has 9 disks. In the case of RAID5 with 16 disks, the stripe dimension is fixed at $W = 16$ (15 data blocks and 1 parity block), whereas, in the case of RAID5 with 9 disks we set $W = 9$ (8 data blocks and 1 parity block).

In Fig. 6 (read-intensive environment) and in Fig. 7 (write-intensive environment) the average response time of previously described architectures is shown vs. the arrival rate of requests in the case of workload uniformly distributed across the array.

The results show that both RAID5 with 16 disks and RAID5 with 9 disks perform worse than RAID1, and that such performance difference is more evident in the case of write-intensive environments. This points out that the updating of the parity information due to the partial write of a stripe (which characterizes the OLTP environment) is a key factor adversely affecting performance of RAID5. Furthermore, RAID5 with 9 disks performs worse than RAID5 with 16 disks, outlining that the small number of disks actually allows low degree of concurrency for the service of I/O requests.

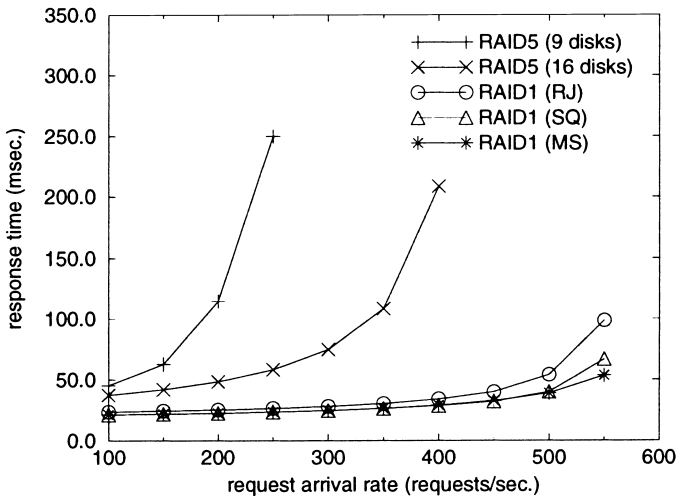

Fig. 6. Response time in the read-intensive OLTP environment ($P_r = 0.75$), with workload uniformly distributed across the array.

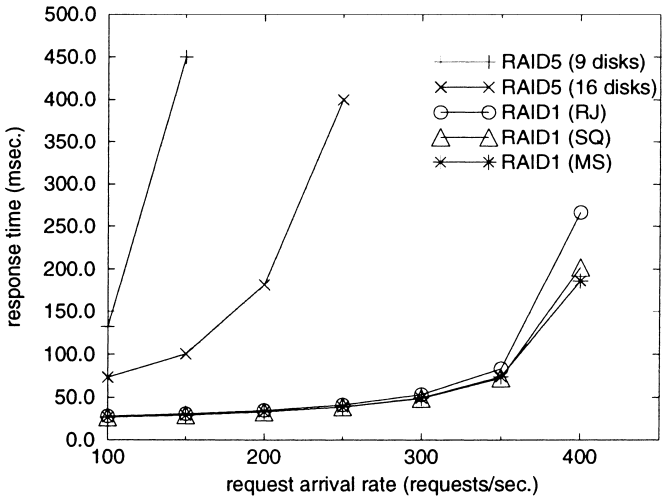

Fig. 7. Response time in the write-intensive OLTP environment ($P_r = 0.25$), with workload uniformly distributed across the array.

Let us now investigate how non-uniformly distributed workload affects performance of these architectures. In such an investigation we consider only RAID1 and RAID5 with 16 disks (thus from now on we do not explicitly recall the number of disks when referring to RAID5). We define *logic disk i* the disk corresponding to the primary copy of data stored into mirror *i* of the RAID1 architecture. Given that 8 mirrors constitute the architecture, 8 logic disks are identified. We consider two different probability distribution functions, namely D_1 and D_2 shown in Fig. 8, for a request to access data on logic disk *i*.

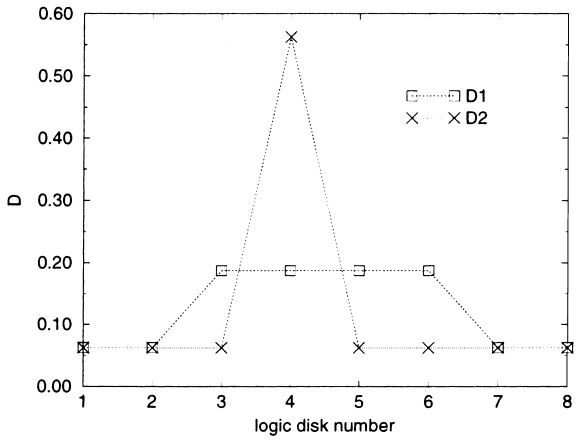

Fig. 8. Probability distribution functions for the access to logic disks.

These distributions define non-uniform access to logic disks. In particular, according to distribution D_1 four logic disks are accessed more frequently than the others. According to D_2 there is a hot spot logic disk, accessed more frequently than all the other ones. For the case of RAID5, data of a logic disk are divided into two equal parts which are stored on different physical disks. Given this mapping and the definition of logic disk stated above, the introduced distribution functions define the access to data as follows: in the case of RAID1 architecture, if a request must access to data on logic disk i, such a request is addressed to mirror i; in the case of RAID5 architecture, if a request must access to data on logic disk i, such a request is addressed to the physical disk on which the data is mapped onto. D_1 stipulates that 4 mirrors of RAID1 and 8 disks of RAID5 are overloaded; whereas, D_2 stipulates that one mirror of RAID1 and 2 disks of RAID5 are overloaded. We consider a fraction of $1/2$ of the workload equally distributed across the array and the remaining fraction distributed according to either D_1 or D_2.

In Fig. 9 (read-intensive environment) and in Fig. 10 (write-intensive environment) the response time is shown when considering distribution D_1 for half of the workload. In the case of read-intensive environment the results point out that RAID5 shows almost the same performance as RAID1, and a saturation point higher than that of RAID1 for both the RJ and SQ scheduling policies. In the case of write-intensive environment, the parity information updating becomes the major factor adversely affecting performance of RAID5. In this environment, RAID1 shows better performance independently of the adopted scheduling policy.

In Fig. 11 (read-intensive environment) and in Fig. 12 (write-intensive environment), the response time is shown when considering distribution D_2 for half of the

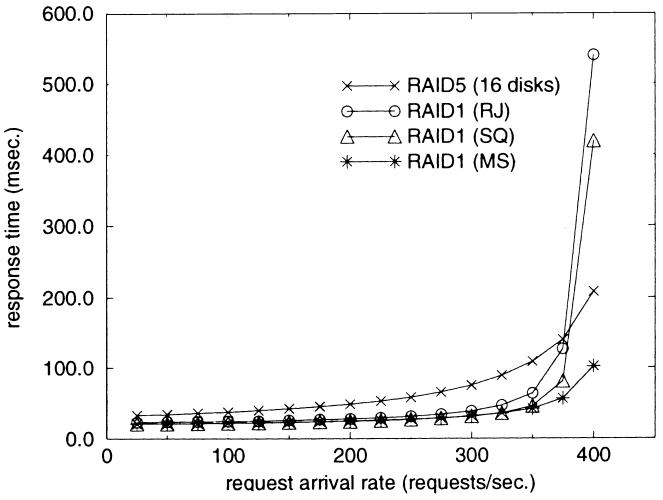

Fig. 9. Response time in the read-intensive OLTP environment ($P_r = 0.75$), with half of the workload distributed across the array according to D_1.

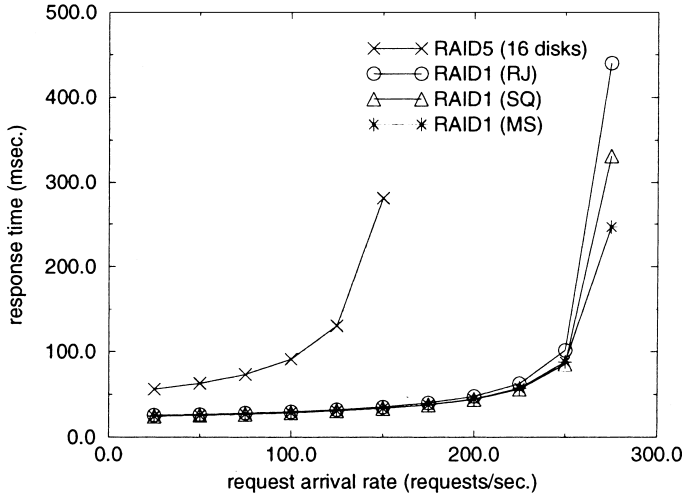

Fig. 10. Response time in the write-intensive OLTP environment ($P_r = 0.25$), with half of the workload distributed across the array according to D_1.

workload. In this case, RAID1 shows better performance in both the read-intensive and the write-intensive environments. However, in spite of the good gain achieved by RAID1 in the case of low request arrival rate, the saturation points of the architectures are not so different especially in the read-intensive environment.

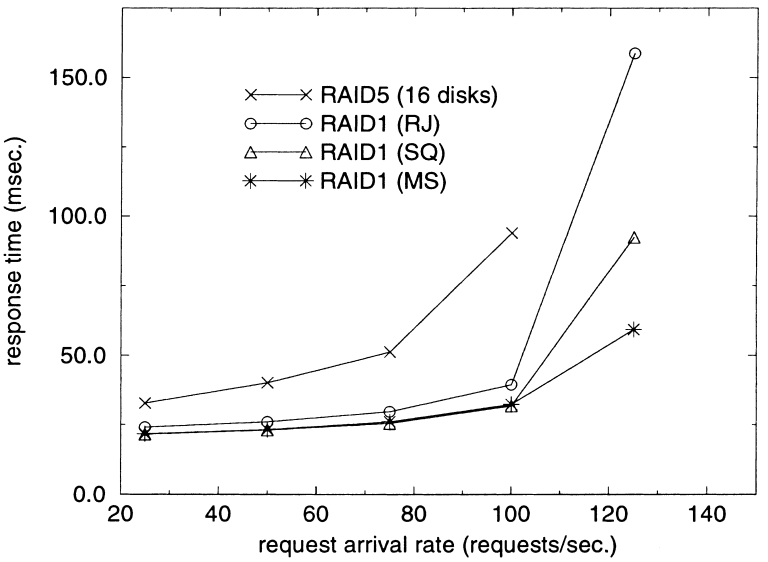

Fig. 11. Response time in a read-intensive OLTP environment ($P_r = 0.75$), with half of the workload distributed across the array according to D_2.

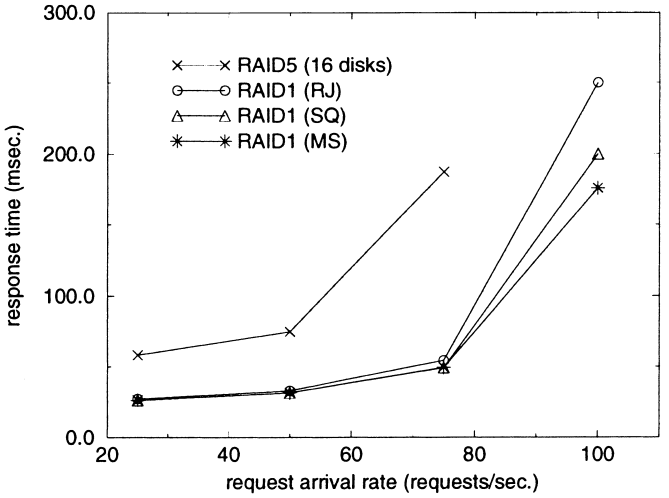

Fig. 12. Response time in a write-intensive OLTP environment ($P_r = 0.25$), with half of the workload distributed across the array according to D_2.

4.1.1. Comments on the results

Previous results show that, when the workload is uniformly distributed across the array, RAID1 shows better performance compared to RAID5. In particular, if the two architectures have the same storing capacity, two factors determine their performance difference:

1. RAID1 has the potential to split read requests on both of the data copies, thus taking advantage of high degree of concurrency between read operations;
2. RAID5 suffers from high cost of the parity information updating, which arises when a partial updating of the stripe (characteristic of the OLTP environment) is executed due to write requests.

Concerning RAID1 and RAID5 having the same cost (16 disks in each architecture), the performance difference decreases, especially in the read-intensive environment. In this environment the updating of the parity information in RAID5 is not the dominant factor affecting performance, whereas in RAID1 the write request traffic, being duplicated, adversely affects the average length of the queues of both disks in a mirror. Latter phenomenon, combined with non-uniform access to data, leads the spot disks of RAID1 to quickly saturate due to the fast increasing of the length of their queues.

In conclusion, under the same cost, RAID1 ensures good performance in the case of a workload uniformly distributed across the array, but has a much smaller storing capacity, whereas, RAID5 offers the advantages of a larger storing capacity and sometimes better tolerates non-uniform access to data.

4.2. Results in the LARGE environment

We simulated a RAID1 architecture with $N = 64$ disks organized according to two different configurations. In the first configuration the 64 disks are organized

in 32 mirrors, thus, the dimension W of the stripe is equal to 32; in the second, the 64 disks are organized into 2 groups of 32 disks, thus, each group consists of 16 mirrors and the dimension of the stripe is $W = 16$.

The RAID5 architecture with the same cost has 64 disks. We considered two different configurations for this architecture. In the first one, disks are organized in a single group with the stripe dimension $W = 64$ (63 data blocks and 1 parity block). In the second one, disks are organized into 8 groups with 8 disks in each group, thus the stripe dimension is $W = 8$ (7 data blocks and 1 parity block).

We simulated also two RAID5 architectures having the same storing capacity as RAID1. The former has 33 disks organized in a single group, with stripe dimension $W = 33$ (32 data blocks and 1 parity block). The latter has 36 disks organized into 4 groups, each of 9 disks, with $W = 9$ (8 data blocks and 1 parity block).

In Fig. 13 (read-intensive environment) and in Fig. 14 (write-intensive environment) the average response time of previously described architectures is shown vs. the request arrival rate in the case of workload uniformly distributed across the array.

Results point out that, while the request arrival rate grows, the architectures with large number of groups show better performance. This is because large number of groups (i.e., small dimension of the stripe) allows high degree of concurrency for the service of full stripe I/O requests. Furthermore, each I/O request executes a complete access to n adjacent stripes (with $n \geqslant 1$), thus reducing the average seek time.

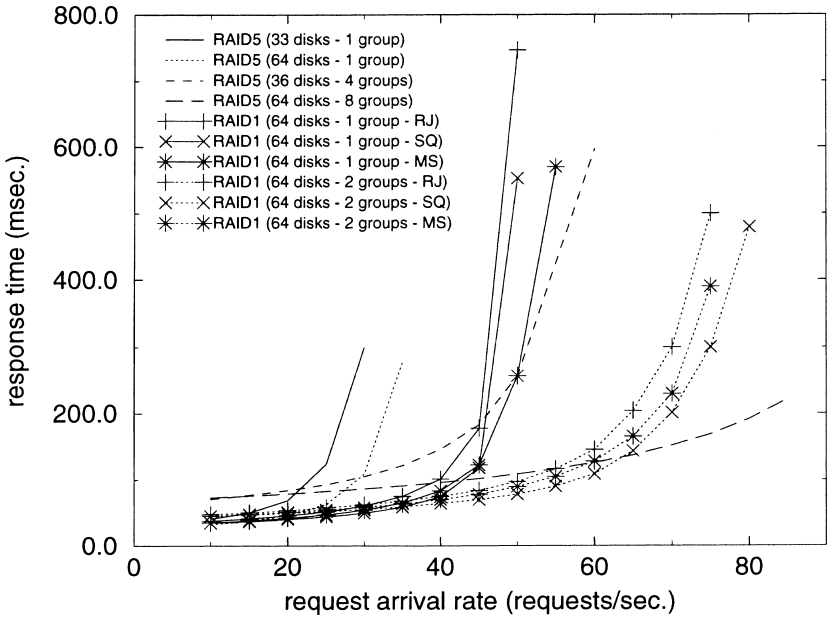

Fig. 13. Response time in a read-intensive LARGE environment ($P_r = 0.75$), with workload uniformly distributed across the array.

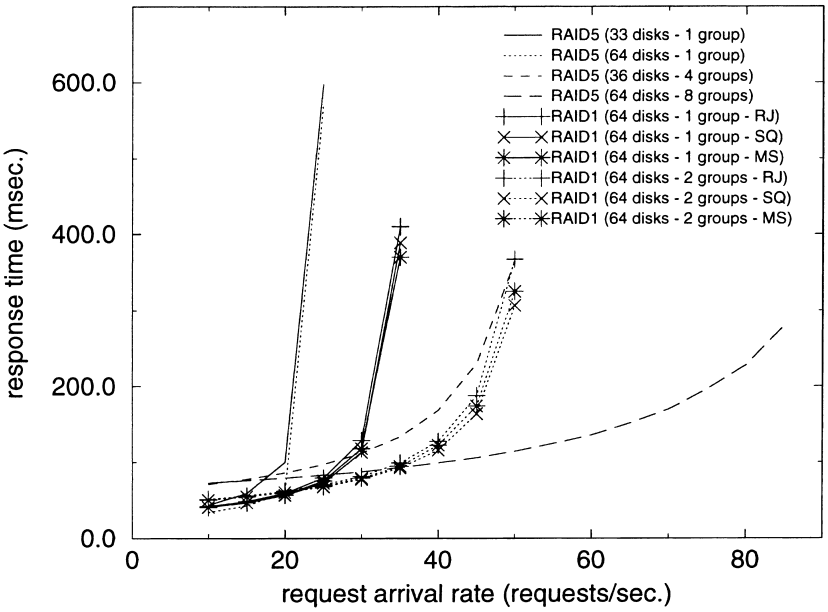

Fig. 14. Response time in a write-intensive LARGE environment ($P_r = 0.25$), with workload uniformly distributed across the array.

These results match those already proposed in Ref. [8]. Differently from what happens in the OLTP environment, in the LARGE environment the performance of RAID5 can be strongly improved by carefully adopting the policy for organizing disks into groups. Results presented here show that RAID5 (36 disks – 4 groups) is much less expensive than all the considered RAID1 architectures and shows very good performance especially in the read-intensive environment. In the case of high arrival rate, the best performer architecture (i.e., the one with the higher saturation point) is RAID5 (64 disks – 8 groups), which allows maximum degree of concurrency for the service of requests.

4.2.1. Comments on the results

Due to the full stripe access, characteristic of the LARGE environment, RAID5 takes advantage of the low cost of the parity information updating. Therefore, it tolerates write-intensive workloads better than RAID1. In the read-intensive environment the average response time of RAID1 (64 disks – 2 groups) is constantly around the same as the one of RAID5 (64 disks – 8 groups). In the write-intensive environment RAID1 quickly saturates as the length of the queue of each disk in a mirror is adversely affected by the traffic of write requests which is totally duplicated.

In addition, under the same cost RAID5 allows larger amount of storing capacity. As an example, RAID5 (64 disks – 8 groups) has a storing capacity 75% larger than the one of both RAID1 (64 disks – 1 group) and RAID1 (64 disks – 2 group).

Table 2
Other metrics for the comparison between RAID1 and RAID5

RAID system	RAID1	RAID1	RAID1	RAID5	RAID5	RAID5	RAID5	RAID5	RAID5
Number of disks	16	64	64	9	16	36	64	33	64
Groups	1	1	2	1	1	4	8	1	1
Storing efficiency	50%	50%	50%	88.89%	93.75%	88.89%	87.50%	6.97%	98.44%
Over-cost	100%	100%	100%	12.5%	6.67%	12.5%	14.29%	3.125%	1.587%
Reliability (yr)	2853	713	713	634	190	158	102	43	11

5. Other metrics for the comparison

In the previous section we have presented results for comparing RAID1 and RAID5 by using response time as performance metric and the number of disks in the architecture as cost metric. Below we show a table comparing the architectures by using also other metrics. In particular, we report data related to the following parameters:

- *Storing efficiency*: percentage of physical space destined for application data and not for redundancy (note that storing efficiency is not the same as storing capacity as latter parameter is not a percentage value);
- *Over-cost*: overprice spent for physical space destined for redundancy;
- *Reliability*: average time between failures (evaluated by using the formula proposed in Ref. [21] and considering the average time for a failure of a single disk equal to 200000 h).

The results in previous sections and those in Table 2 suggest that:

(a) RAID1 disk-mirroring shows small response time and good capability to tolerate heavy workloads especially in the case of I/O requests accessing small amount of data. Therefore, it is well suited for database environments where the workload consists of transactions and read-only queries. Furthermore, it offers high degree of reliability. It has the drawback that its overprice is constantly 100%, so it would be preferable to other architectures only when performance and reliability aspects largely dominate over the economic one;

(b) RAID5 shows acceptable performance in the case of a low rate of I/O requests and good performance in the case of I/O requests accessing full stripes. In addition, it shows an acceptable reliability achieved with usually very small overprice. It results well suited to environments characterized by large amount of data being accessed during I/O operations and to database environment where performance is not a critical factor.

6. Conclusion

In this paper we have presented a comparison between two disk array architectures. The first one (RAID1) adopts data replication, whereas the second one (RAID5) uses parity information in order to achieve data availability. The compar-

ison takes both performance and cost aspects into account. Performance of these architectures has been investigated through a simulation study considering a broad variety of scenarios. In particular, we considered both an *on-line-transaction-processing* environment, where a small amount of data (1 block – 4096 bytes) is accessed at each I/O request and a *large* environment in which I/O requests access 256 blocks (1 MB). In the first environment we also considered the impact of non-uniform access to data on the performance. In the second environment, we considered different sizes for the data stripe, discovering that, for both architectures, small stripes tend to have better performance compared to large ones. The main result of our comparison is that none of the architectures can be considered optimal from the point of view of both performance and cost, thus the choice towards one or the other architecture strongly depends on the applications and on the available capital to be spent for the I/O subsystem.

Future work will be focused on a comparative analysis when considering the event of disk failure, which introduces the data reconfiguration problem.

References

[1] D. Bitton, J. Gray, Disk shadowing, in: Proceedings of the 14th International Conference on Very Large Databases, Los Angeles, California, 1988, pp. 331–338.
[2] W. Burkhard, J. Menon, Disk array storage system reliability, in: Proceedings of the 23rd International Conference on fault Tolerant Computing, Toulouse, France, 1993, pp. 432–441.
[3] P.M. Chen, E.K. Lee, Striping in a RAID level 5 disk array, in: Proceedings of the Joint International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS'95/PERFORMANCE'95), Ottawa, Canada, 1995, pp. 136–145.
[4] P.M. Chen, E.K. Lee, G.A. Gibson, R.H. Katz, D.A. Patterson, RAID: high-performance reliable secondary storage, ACM Computer Surveys 26 (2) (1994).
[5] S. Chen, Design, Modeling and evaluation of high performance I/O subsystems, PhD thesis, University of Massachusetts at Amherst, September, 1992.
[6] S. Chen, D. Towsley, RAID5 vs. parity striping: Their design and evaluation, Journal of Parallel and Distributed Computing 17 (1992) 58–74.
[7] S. Chen, D. Towsley, A performance evaluation of RAID architectures, IEEE Transactions on Computers 45 (10) (1996) 1116–1130.
[8] P.M. Chen, D.A. Patterson, Maximizing performance in a striped disk array, in: Proceedings of the 1990 Symposium on Computer Architecture, 1990, pp. 322–331.
[9] R. Geist, K. Trivedi, An analytic treatment of the reliability and performance of mirrored disk subsystems, in: Proceedings of the 23rd International Conference on fault Tolerant Computing, Toulouse, France, 1993, pp. 442–450.
[10] J.Gray, B. Horst, M. Walker, Parity striping of disk arrays: low cost reliable storage with acceptable throughput, in: Proceedings of the 16th International Conference on Very Large Databases, Brisbane, Australia, 1990, pp. 148–161.
[11] H. Hsiao, D. DeWitt, Chained declustering: A new availability strategy for multiprocessor database machines, in: Proceedings of the Sixth International Conference on Data Engineering, Los Angeles, California, 1990, pp. 456–465.
[12] H. Hsiao, D. DeWitt, A performance study of three high availability data replication techniques, in: Proceedings of the First International Conference on Parallel and Distributed Information Systems, Miami Beach, Fle., 1991, pp. 18–28.
[13] S.S. Lavemberg, Computer Performance Modeling Handbook, Academic Press, New York, 1983.

[14] E.K. Lee, R.H. Katz, An analytic performance model of disk arrays and its application, Technical Report UCB/CSD 91/660, University of California at Berkeley, 1991.
[15] M. Malhotra, K. Trivedi, Reliability analysis of redundant arrays of inexpensive disks, Journal of Parallel and Distributed Computing 17 (1993) 129–139.
[16] R.J. Miller, The Jackknife – a review, Biometrica 61 (6) (1974) 1–15.
[17] M. Kim, Synchronized disk interleaving, IEEE Trans. on Comput. C-35 (11) (1986) 978–988.
[18] E.K. Lee, R.H. Katz, The performance of parity placements in disk arrays, IEEE Transactions on Computers 42 (6) (1993) 651–664.
[19] M. Livny, S. Khostafian, H. Boral, Multi-disk management algorithms, in: Proceedings of the International Conference on Measurement and Modeling of Computer Systems (ACM SIGMETRICS'87), 1987, pp. 69–77.
[20] A. Merchant, P. Yu, Analytic modeling and comparisons of striping strategies for replicated disk arrays, IEEE Transactions on Computers 44 (3) (1995) 419–433.
[21] D. Patterson, G. Gibson, R. Katz, A case for redundant arrays of inexpensive disks (RAID), in: Proceedings of the 1988 SIGMOD International Conference on the Management of Data, Chicago, 1988, pp. 109–116.
[22] K. Salem, H. Garcia-Molina, Disk striping, in: Proceedings of the Second International Conference on Data Engineering, 1986, pp. 336–342.