Hot mirroring : A method of hiding parity update penalty and degradation during rebuilds for RAID5

Kazuhiko Mogi
Institute of Industrial Science, The University of Tokyo
mogi@tkl.iis.u-tokyo.ac.jp

Masaru Kitsuregawa
Institute of Industrial Science, The University of Tokyo
kitsure@tkl.iis.u-tokyo.ac.jp

Abstract

This paper proposes a storage management scheme for disk arrays, named hot mirroring. In this scheme, storage space is partitioned into two regions. One is the mirrored region, which is characterized by high performance and low storage efficiency. The other is the RAID5 region, which is characterized by low performance and high storage efficiency. Hot data blocks are stored in the former area, while cold blocks are stored in the latter. In addition, mirrored pairs and RAID5 stripes are orthogonally laid out, through which the performance degradation during rebuilding is minimized. Hot block clustering in hot mirroring achieves higher performance than conventional RAID5 arrays. The potential of hot mirroring is examined through extensive simulation.

1 Introduction

Recently, due to the progress of semiconductor technologies, microprocessor performance has improved dramatically while that of secondary storage systems have not kept pace. For secondary storage, the main efforts have been devoted towards increasing capacity and reducing size, with only slight improvements in performance. This has caused the access gap between main memory and secondary storage systems to become even larger. As a means of decreasing this access gap, disk array systems have attracted strong attention as high performance secondary storage systems.

RAID[PGK88] utilizes a large number of commodity inexpensive drives in parallel to achieve higher performance as well as obtaining higher reliability by recording redundant information. In [PGK88], Patterson et. al classified RAID into five levels. Among these levels, level 1 (mirrored disk array) and level 5 (RAID5 disk array) are regarded as the most promising approaches for providing highly reliable secondary storage systems which support concurrent access to small blocks. Mirrored disk arrays maintain a copy of all disk blocks for redundancy. In contrast, RAID5 disk arrays employ parity encoding for redundancy, which leads to a much larger storage capacity than that of mirrored disk arrays when the same number of disks is used.

But there are two major problems in using RAID5 disk arrays. One is the overhead of recording the redundancy information. The new parity for a small write is derived as follows:

$$P_{new} = P_{old} \oplus D_{old} \oplus D_{new} \quad (1)$$

Thus a single block update requires 4 disk accesses: old block read (D_{old}), old parity read (P_{old}), new block write (D_{new}) and new parity write (P_{new}). This deteriorates the throughput of the write operations. The other problem is the overhead of reconstructing data when some disks fail. When disk k in disk group j fails, the lost data is rebuilt as follows ($D_{j,i}$ is the data location on disk i of disk group j where the parity stripe was made):

$$D_{j,k} = P_j \oplus D_{j,1} \oplus \cdots \oplus D_{j,k-1} \oplus D_{j,k+1} \oplus \cdots \oplus D_{j,n} \quad (2)$$

Thus the rebuild process requires disk accesses for all disks in the disk group of the failed disk. The impact of this operation on performance is quite large.

In mirrored disk arrays, a copy is stored for redundancy. In normal mode, only two write accesses are required for a block update. During rebuild mode, only a read access on the live disk and a write to the new disk are required for the replacement of a failed disk. Therefore the overhead on a block update and the load incurred by the rebuild process are much smaller than those of RAID5 disk arrays. However, mirrored disk arrays pay the penalty of much smaller data capacity than that of RAID5 disk arrays, because of the data replication.

In order to get not only higher performance but also larger usable capacity, we studied a method which combines a mirrored disk array which is characterized by higher access performance, but lower storage efficiency and a RAID5 disk array which is characterized by higher storage efficiency but has lower performance. Usually there are localities in the access pattern. In this paper we propose a new storage management scheme named hot mirroring[MK95b, MK95c], where hot data blocks are stored in the mirrored area and cold data blocks are stored in the RAID5 area. The idea of

SIGMOD '96 6/96 Montreal, Canada
© 1996 ACM 0-89791-794-4/96/0006...$3.50

increasing performance through the exploitation of the access locality was investigated in our previous paper[MK95a]. There we proposed the method named hot block clustering for RAID5 disk arrays. We allocate a contiguous region for hot blocks, which reduces the seek cost significantly and improves performance. We studied its effectiveness for RAID5 disk arrays employing an LFS[RO91] based approach and the virtual striping[MK94] approach. Based on these results, we assign a small contiguous area for mirroring in the hot mirrored disk array. The hot data in the mirrored area has low overhead for maintaining redundancy information when compared with cold blocks in the RAID5 area, which can further increase the performance.

Although hot block clustering in RAID5 can attain much higher performance than naive RAID5, it still has performance problems during rebuild operations on disk failure. In hot mirroring, as we will explain in detail, the mirrored pairs and RAID5 stripes are orthogonally laid out. In the mirrored area, the chained declustering scheme[HD90] is employed. Through this new method, we can achieve very efficient load balancing, which minimizes the degradation during rebuilds.

The HP AutoRAID[WGSS95] similarly provides a two-level (mirror and RAID5) storage hierarchy inside a single disk array controller.[1] However, there is no clear description of their data placement strategy in [WGSS95]. The hot block clustering and orthogonal layout employed in hot mirroring improves performance significantly, which should be not addressed in the AutoRAID paper. In addition, we carefully designed the system so that performance degradation during rebuilds is minimized, which is not addressed in the AutoRAID paper.

In section 2, the methods which have been proposed for improving performance of mirrored disk arrays and RAID5 disk arrays are surveyed. In section 3 the hierarchical structure of mirrored disk arrays and RAID5 disk arrays is described. In section 4 the details of the mapping scheme in the hot mirroring method are explained. Section 5 gives implementation issues for the hot mirroring method. Extensive simulation is done to identify the performance characteristics of hot mirroring. Section 6 describes the results of the analysis. Section 7 concludes this paper.

2 Related work

In this section, an overview of the methods which have been proposed to improve the performance of RAID5 disk arrays and mirrored disk arrays is presented.

2.1 The methods to improve the performance during normal mode

One of the major problems of RAID5 disk arrays is the overhead for small write accesses and the cost of rebuilding a failed disk. First, we introduce the methods which have been proposed for improving the performance of small write accesses. These methods can be roughly divided into three categories.

The first category contains methods which use a cache to take advantage of access localities in order to reduce the number of disk accesses, such as smart caching[MC93] and dynamic parity grouping[YWD93]. The smart caching method reduces the number of disk accesses and the cost of each access through proper cache controls. Dynamic parity grouping also uses a cache to decrease the number of disk accesses during parity updates. In this scheme, the blocks which have high access rates are grouped together to make new stripes. The parity of these stripes are maintained in the cache of the disk controller so that the number of accesses to the parity disk are reduced.

The second category contains methods which decrease the cost of disk accesses through bulk parity updating, such as parity logging[SG93] and LRAID[BD92]. Both methods delay parity updating by recording update information in a log, which is XORed value of the old and new data. Using this log, all the delayed parity updates are performed at one time. In parity logging, log areas and parity areas are distributed over the whole array, whereas LRAID physically separates the log and parity disks from the data disks. This type of parity updating method incurs extra disk accesses, but access efficiency is much higher than in traditional update methods. In all, these methods show higher performance than naive RAID5 disk arrays.

The last category floats the data to decrease the cost of small writes. In RAID5 disk arrays, traditional methods requires two pairs of read-modify-write accesses for each small write access as shown in equation (1). Therefore the floating parity/data method[MK92] alters the position of data or parity freely in the cylinder by using a mapping table. This method reduces latency delay so that the latency delay of a read-modify-write operation becomes nearly equal to one access delay.

The LFS[RO91] based method and the virtual striping method[MK94] change data positions freely in the disk array and make new stripes from the updated data and write them onto free areas. With these methods, there is no need to read the old parity or old data when small writes are performed. Although the action of freeing an area is required, garbage collection can be efficiently executed in a background fashion. Moreover hot block clustering[MK95a] helps to reduce the cost of garbage collection by taking advantage of access localities.

In mirrored disk arrays, distorted mirror[SO91] and doubly distorted mirror[OS93] also use the floating method. Distorted mirror introduced the mapping table named distorted map for slave drive, while master drive employs the static data layout. This can improve the performance for burst updates frequently occurred in transaction system, while static clustering can be maintained in the master drive. Improved traditional mirror[OWS93] and doubly distorted mirror use

[1] We proposed hot mirroring independently of AutoRAID. [MK95b, MK95c]

the write twice technique which floats the data in the short term. In these methods, write operations are efficiently scheduled.

2.2 Methods to minimize the degradation during rebuilds

Next, we examine the methods for decreasing the performance degradation during rebuilds. Proper caching improves performance during rebuild mode[MM91]. It can reduce the number of disk accesses to the blocks on a failed disk. The unit of rebuild processing is an important parameter because this value affects both the performance of the rebuild operation and the total time required for reconstruction[HMP93, HP93]. This is also true for mirrored disk arrays.

Clustered RAID[ML90] and parity declustering[HG92] use the same idea that the number of disks in the parity stripe should be set smaller than the number of physical disks. With this configuration, the accesses required to restore a damaged disk can be distributed over a larger number of disks, so that the load on each disk for reconstructing data becomes smaller.

Distributed sparing[MM92] and parity sparing[2][RCB93] use standby disks. Distributed sparing allocates a standby area on all disks. There is no need to reconstruct the data in the standby area, so less data needs to be restored during the rebuilding phase. The parity sparing method records parity in the standby area and reduces the number of data disks in a stripe by half. When a disk fails, parity merging and data reconstruction are performed. Since the number of disks in a stripe is small, the impact of rebuilding is smaller than that of naive RAID5 disk arrays.

In mirrored disk arrays, in order to reduce the impact of restore accesses, two copy placement policies, the chained declustering method[HD90] and the interleaved declustering method[Ter85] were proposed. They differ in how to perform load balancing on disk failure. The interleaved declustering method distributes a copy equally among all the disks in a predefined cluster, and reads the copy to restore the data when a disk fails. On the other hand, the chained declustering method stores the copy to the adjacent disk. During disk failure, this data allocation allows the access requests to be ideally distributed over the remaining drives.

3 Hierarchical structure combining mirror and RAID5

With respect to high storage efficiency with high reliability, RAID5 is the best amongst the five levels. But there are two big performance problems. One is the extra time required to record the redundancy information. The other is the large number of disk accesses required to reconstruct data when a disk fails. For both it is the parity encoding for redundancy that causes these problems. From the point of view of performance, mirrored disk arrays, which use block copying for redundancy, are better than RAID5 disk arrays. But mirrored disk arrays have significant space overhead. If it were possible to merge the characteristics of high storage efficiency but lower performance in RAID5 disk arrays with the higher performance but lower storage efficiency in mirrored disk arrays, it might be one of the best configurations of disk arrays.

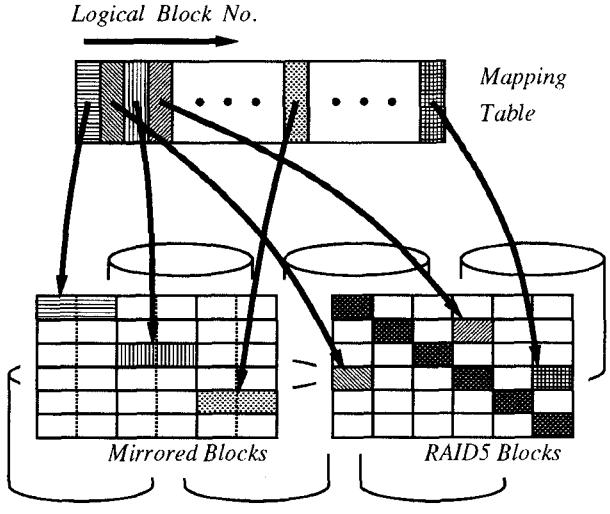


Figure 1: The mapping table for the hierarchical disk array

In general, there are access localities, which can be exploited to improve the performance of RAID5 disk arrays using methods such as caching, dynamic parity grouping, and hot block clustering. Based on access frequency, data blocks are separated into two storage classes, one contains blocks with high access rates (hot blocks) and the other with low access rate blocks (cold blocks). With this separation, the mirror scheme and the parity encoding scheme are combined to get higher performance and larger capacity.

It is desirable that the penalty of recording redundant information on hot blocks should be small. This requirement is well suited to the mirror scheme, thus hot blocks are mirrored. For the cold area, we use parity protection for redundancy to obtain higher storage efficiency. The penalty for maintaining redundant information on the cold blocks has little effect on performance because these blocks are infrequently updated. Thus the cold blocks are stored in RAID5 area.

In hierarchical disk arrays, hot blocks are mirrored and cold blocks are stored on the RAID5 class. That is, the storage class to which the data block belongs is altered according to its access ratio. The access frequency will change as time goes on. Data blocks might migrate from RAID5 to mirror or vice versa. The location should be transparent from the application program. Thus we assume a mapping table which translates the logical block address to the physical block address, which is essential for hierarchical disk arrays (illustrated in Figure 1).

[2]This term is used in [MM92] but not used in [RCB93].

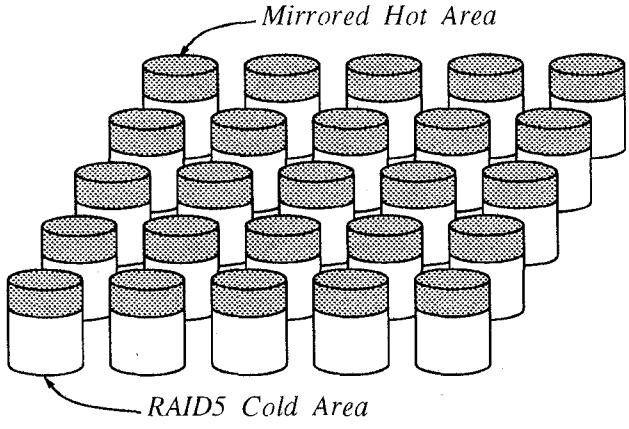


Figure 2: Hot mirroring

4 Hot mirroring [MK95b, MK95c]

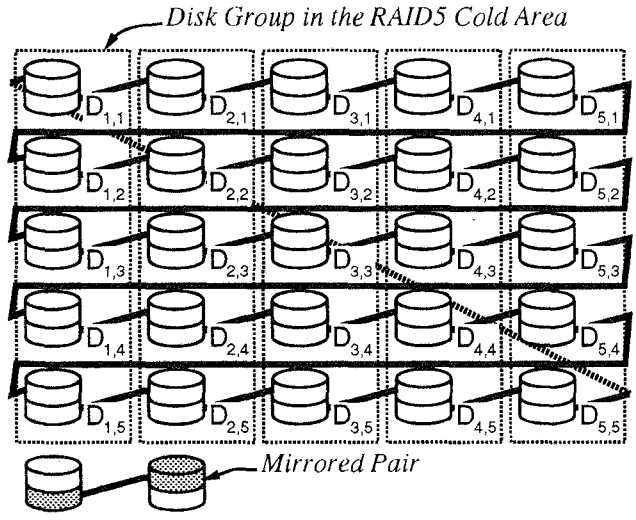
In this section, the layout scheme employed in hot mirroring is described, which is one of the key features in our method.

In hot mirroring, all disks are physically divided into two contiguous regions. One is the mirrored area and the other is the RAID5 area. Hot blocks are stored in the mirrored area, while the cold blocks are stored in the RAID5 area (Figure 2). The other possible approach is just to use two distinct arrays independently. Namely, instead of partitioning a single drive into two regions, we could use a mirrored array and a RAID5 disk array and put some control software which migrates the data blocks between two arrays. Hot mirroring does not adopt such an approach, but distributes the mirrored area evenly over all the disk drives in order to make full use of the disk heads of all the drives.

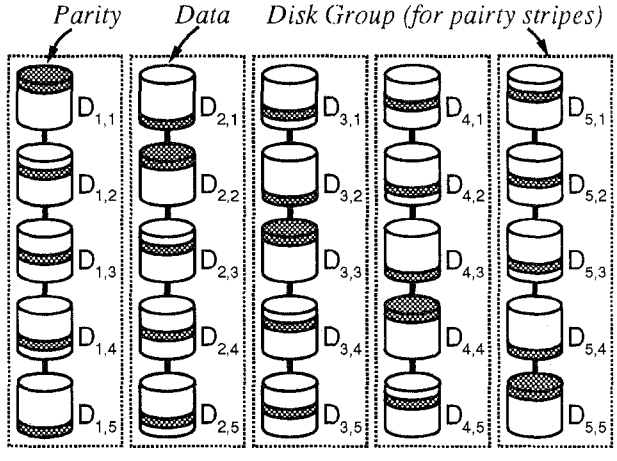
In addition, we allocate a contiguous region for the mirror. We proposed the method named hot block clustering for RAID5 disk arrays in [MK94]. We showed that instead of allocating the hot blocks randomly over all the disk space, we can increase performance by clustering the hot blocks into a specific contiguous region. The seek cost can be significantly reduced by the hot block clustering method. Based on these results, we reserve a contiguous region for hot blocks on each disk.

The hierarchical structure of hot mirroring improves the performance of normal operations. However, we also have to focus on the performance during the rebuild operation. At present, mission critical applications are rather reluctant to use RAID5 disk array, since the performance during rebuilds considerably decreases compared with normal operations. RAID5 survives even if one of the disk fails, but performance falls below acceptable levels. In order to address this problem, we employ the following orthogonal layout.

Figure 3 illustrates the data layout in hot mirroring. It is assumed that there are M disk groups which have N member disks. In the figure, the label $D_{j,i}$ ($i = 1, \cdots, N$ and $j = 1, \cdots, M$) means that this disk is the ith disk in the jth disk group. In the RAID5 area, parity stripes are made into a disk



(a) Mirrored hot area



(b) RAID5 cold area

Figure 3: Data placement policy for hot mirroring ($M = 5, N = 5$)

group, that is, a parity stripe consists of the blocks in the disks labeled $D_{j,1}$, $D_{j,2}$, $\cdots$, and $D_{j,N}$. This is shown vertically in Figure 3(b). In the mirrored area, the copy is allocated on different disk groups, employing the chained declustering method[HD90].[3] The mirrored pair consists of the disks labeled $D_{j,i}$ and $D_{j+1,i}$ ($j = 1, \cdots, M-1$) or the disks labeled $D_{M,i}$ and $D_{1,(i \bmod N)+1}$. This is shown horizontally in Figure 3(a). Through this pairing, there is only one mirrored pair chain which goes round disk groups in a round robin manner, which may lead to good load balancing, especially during disk failure.

The reason why we employ such orthogonal placement for parity stripes and mirrored pairs is as follows. The load

[3] We adopt the chained declustering method instead of the interleaved declustering method because chained declustering has a higher mean time to data loss than that of the interleaved declustering method.

of disks which belong to the disk group with a failed disk becomes high, since it has to serve both the normal access to the broken data and the accesses for reconstruction. The disks on the disk group including the failed disk should work on rebuilding as much as possible in order to minimize the rebuild time. Therefore, frequent accesses against the hot blocks stored over the mirrored area of the broken disk group should be directed towards other disk groups. In hot mirroring, the allocations of parity stripes and mirrored copies are orthogonal each other as shown in Figure 3. A pair of mirrored blocks are placed so as not to be on the same disk group, through which the mirrored pair of the block on the broken disk group is always found on an alive disk group. Accesses against the mirrored area can be absorbed by live drives without degradation using the chained declustering method. Thus, hot accesses can be served by surviving disk groups, while all the drives on the broken disk group can concentrate on rebuilding.

The HP AutoRAID[WGSS95] adopts the same hierarchical structure as hot mirroring, combining mirror and RAID5. However there is no precise description of its data layout. As described above, we employ the orthogonal layout for hot mirroring. This layout can not only improve the performance of ordinary accesses but also reduce the performance degradation during rebuilds.

5 Implementation issues for hot mirroring

In this section, we discuss three key implementation issues in hot mirroring. The first issue is the identification scheme for hot blocks. The second issue is on when to invoke migrations. The third issue is on how to balance the load among the disk drives. All of them are critical to performance.

5.1 Identification of hot blocks

How to determine the hot blocks is one of the important issues in hierarchically structured disk arrays. The hot mirroring method uses the same scheme as the hot block clustering method[MK95a] to identify the hot blocks. Usually almost all blocks written by write requests tend to be used again soon. Thus in hot mirroring, all blocks of normal write requests are assumed to be hot and all write accesses are issued on the mirrored area.

Even if the block is cold, our current strategy moves that cold block to the mirrored hot area on a write operation, which consumes the free space of the mirrored area. Cold blocks in the mirrored area need to be migrated back to the RAID5 area. The time at which each block in the hot area was last accessed is recorded. By checking the oldest access time, we can determine out the cold blocks occupying the mirrored area. In another words, a migration target is selected by the Least-Recently-Used policy. If the amount of free space in the hot area falls below a threshold value, this migration is invoked. The HP AutoRAID has a similar mechanism of changing storage class[WGSS95]. Data which resides in the RAID5 class are written into the mirrored class on their updates. To make free space in the mirrored class, data migrations are performed when the array is "idle". The target of migration is selected by the Least-Recently-Written algorithm.

Although cold block migrations from the mirrored class to the RAID5 class requires extra disk accesses, this is not a significant overhead as will be clarified in section 6.

Here and in the subsequent simulation, we assumed that the ratio of the mirrored hot class to the RAID5 cold class is determined in advance by the requirement of available data capacity. The dynamic decision of the ratio of mirrored hot area is left for future work.

5.2 Invocation of block migrations from mirror to RAID5

Block migrations are required to make free space on the mirrored area when we update the block on the RAID5 area. We have to migrate some blocks on the mirrored area to the RAID5 area. The victim block is read from one of the mirrored disk and written to a free block in the RAID5 area. The cost of block migrations affects performance. If a disk array has enough free space in the mirrored hot area, migration operations can be delayed until the disk array becomes idle in order to decrease the performance impact. However, if there remains insufficient free space in the mirrored area, migrations are forced to occur. Thus migration should be invoked based on the amount of remaining free space.

In addition, space balancing also has to be considered. If free space in the mirrored area exists only a part of mirrored disk pairs, write operations from the RAID5 class concentrate these mirrored disk pairs. Thus the unbalanced allocation of free space among mirrored disk pairs may cause performance degradation. Free space in each mirrored disk pair is independently managed to balance resource utilization.

Finally we determine the migration condition as follows.

1. The migration process is invoked when the number of free blocks remaining in the target mirrored disk pair falls below the first threshold value and the shorter queue length of the disk in the target mirrored disk pair is smaller than a certain number of accesses (3 in our simulation).
2. The migration process is always invoked if the number of free blocks in the target mirrored disk pair falls below the second threshold value. This eagerly makes free space for writing when there remains only a little free space in the mirrored disk pair.

We set the first threshold value to around half of the total free blocks divided by the number of disks and set the second threshold value to one third of it.[4] We found that migration

[4] In the simulation done in section 6 the first threshold is set to 60 blocks and the second threshold to 20 blocks.

can be done without large influence to the normal accesses under this condition.

5.3 Queue length based load balancing on read and write operations

Load balancing is another important issue in disk arrays to obtain high performance. Each disk has an associated access request queue. We adopted a queue length based load balancing algorithm.

In the mirrored area, the data is stored on two different disks. On reading the block in the mirror, the disk with shorter access queue is selected. On reading the block in the RAID5 area, there is no choice, since the RAID5 area does not have any copy.

On write and migration operations, more complex control could be considered. In hot mirroring, the physical location of the paired blocks in the mirrored area are statically fixed, while in distorted mirror[SO91], distorted map is introduced, which allows an arbitrary location relationship between master block and slave block. The parity stripes of the RAID5 area for hot mirroring are also fixed as in ordinary RAID5 arrays, while in virtual striping[MK94] the blocks in a stripe can occupy different locations stripe by stripe. Such a further sophistication would further increase performance. However, in this paper, we simplify the location assignment in order to focus on the effectiveness of the hierarchical combination of mirror and RAID5 and its orthogonal layout. Further performance tuning is left for future work.

Even under such static physical layouts, we can still balance the load over the drives on both write and migration operations.

There are three opportunities to assign new locations on write and migration operations. First, when we update the block in the RAID5 area, we migrate that block onto the mirrored area. We can choose the appropriate mirrored disk pair. Second, when we migrate the block from the mirrored area to the RAID5 area, we can choose the appropriate location in the RAID5 area. Third, when we update the block in the mirrored area, we could choose a different location from the original one.

The location of the block should be transparent from the application program. When migration occurs between the mirrored area and the RAID5 area, the mapping table keeps track of the relationship between logical address and physical address.

This change of the location could contribute to the load balancing of the drives and improve the performance of the system. Over a short time, if write and migration operations are performed on the disk pair which has the shorter access queue length, we can expect it will reduce the response time in the worst case. Over a long time, we can expect that the loads among the disks become uniform.

Basically the new position is determined based on the access queue length. However, as described in the previous section, balancing space utilization also has to be considered.

```
modified_queue_length( the disk pair )
{
  queue_length = longer queue length of the disk pair;
  if ( write operation is performed to the RAID5 area )
    rate = number of free blocks in the disk pair
           / average number of free blocks
             in a disk pair in the RAID5 area;
  else  /* write to the mirrored area */
    rate = number of free blocks in the disk pair
           / first threshold value to perform migrations;
  if ( rate ≤ 1 ) /* small free space */
    mod_queue_length = queue_length;
  else            /* large free space */
    mod_queue_length = (3 - rate)/2 * queue_length;
  return  mod_queue_length;
}
```

Figure 4: Calculation of modified queue length for a disk pair

In order to reflect space utilization effects on the access queue length, we introduce the metric, modified access queue length. Here we define a disk pair. A disk pair in the mirror is straight forward, namely, the pair of drives where pair of blocks are stored. A disk pair in RAID5 means that the two drives which contains the data and the parity. If the disk pair whose access queue is long has lots of free space, we shorten its access queue length by multiplying a certain coefficient. Figure 4 shows the way to calculate the modified access queue length.

The new location is determined using the following procedure.

1. If updated data is written to the same area as it was previously stored, the candidate disk pair into which the data is written is set to the previous disk pair. Otherwise it is set to "null".
2. Compare the access queue length of the candidate disk pair with that of all the other disk pairs containing free blocks. We select one with the shortest access queue length. This comparison is performed in a round robin manner. In addition, for every request, the disk pair from which the comparison starts shifts in a round robin fashion in order to attain an even distribution.
   As in figure 4, the queue length of the disk pair is defined as the longer queue length of the two. If two disk pairs have the same queue length, we do the following decision. If in the RAID5 area, we compare the queue length of the data disk rather than the parity disk, since reading data should have higher priority than parity. Otherwise, we select one of the two in a round robin manner. Thus we determine the disk pair into which the updated data will be written.
3. If that disk pair is not changed from the disk pair where it was previously stored, the data is written to the original

capacity	318MB
cylinders/disk	949
tracks/cylinder	14
sectors/track	6
sector size	4096 bytes
revolution time	13.9ms
seek time model	$\text{seek}(d) = 2.0 + 0.01 \cdot d + 0.46 \cdot \sqrt{d}$
track skew	1 sector

Table 1: Disk model parameters

	Effective data capacity
Naive RAID5	83.3 %
Floating D&P	79.4 %
Hot mirroring	76.5 %
Mirror	50.0 %

Table 2: Data capacity for each configuration (normalized by total disk volume)

location.[5] Otherwise, choose a new location from the free blocks in that disk pair.[6]

Load balancing plays a very important role for improving performance. There is no detailed description on this in AutoRAID.

6 Evaluation of hot mirroring

6.1 Simulation assumptions

Table 1 shows the disk model parameters, which are taken from [HG92]. The block size is 4KB. The striping unit is set to the block size. The position of the parity is incremented by one track when rotated among the disks of the RAID5 region.

To compare performance, four configurations are examined, hot mirroring, naive RAID5, RAID5 with floating data and parity[7] (abbreviated as floating), and ordinary mirroring. Mirrored disk arrays adopt the chained declustering method and the queue length based load balancing scheme for read operations.

Table 2 shows the effective data capacity of these configurations in the simulation. The AutoRAID employs a LFS based management scheme for the RAID5 region, which would require large amounts of free space. Since hot mirroring employs an ordinary RAID5 scheme instead of LFS for the cold region, basically we do not need a large free space. However, since write and migration processes lock both old and new locations simultaneously while these operations perform, we need only a small amount of free space. In this simulation, we assume that 0.16% of the whole disk capacity (3054 blocks) are assigned to free blocks in hot mirroring. In the floating method, 5% of the blocks in a cylinder (4 blocks in this simulation) are assigned as free for floating blocks.

Hot mirroring has 4 disk groups, which have 5 data disks and a parity disk in the RAID5 area. 4*(5D+P) denotes this parity configuration later. 20% of the total physical disk capacity is allocated to the mirrored hot region, which means that about 13%[8] of the total data blocks can be stored in this area. Naive RAID5 disk arrays and floating disk arrays also have a 4*(5D+P) configuration.

In this simulation, the effective capacity of hot mirroring, naive RAID5, and floating RAID5 are set equal. That is, 76.5% of the total physical disk blocks[9] have valid data for hot mirroring, naive RAID5 disk arrays, and floating disk arrays. In other words, naive RAID5 and floating RAID5 have a smaller number of cylinders than hot mirroring. Ordinary mirror has the same number of disks as hot mirroring for performance comparison. Therefore, the effective data capacity of mirror is smaller than that of the others.

To simplify the simulation, it is assumed that the disk array controller and the bus between the controller and disks is sufficiently fast so that the overhead of the controller and the communication overhead between the controller and disks is negligible. Based on this assumption, the controller can find free disks and dispatch accesses for rebuilding as soon as some disks become free. All the control tables are maintained by the controller.[10]

Disk accesses including migration accesses and rebuild accesses are performed on a first come first serve basis. The size of requests are fixed at 4KB. The interval of access request arrivals has a negative exponential distribution. The load is controlled by changing the mean time between access requests. Access locality is as follows: blocks are divided into two groups and $y\%$ of the blocks belong to the first group. In each group, the access probabilities are equal, but $x\%$ of the access requests are concentrated on the first group. Later we refer it as x-y access locality.

The performance of each array is simulated as follows. Initially, hot blocks and cold blocks are scattered randomly on both mirrored and RAID5 areas. Then two million write accesses are given in order to separate hot blocks from cold blocks. In the following simulations, we specify a certain read/write ratio. After two million access, another

[5] The reason why we just use the original position is due to simplicity. We could search more for a optimal location if there are several free locations. Since we employed static layout as described in section 4, we think optimizing the location would not have a large influence on performance.

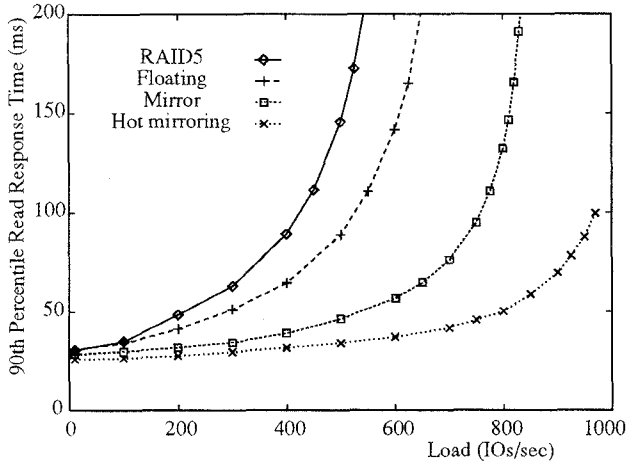
[6] Here we select the new position randomly for simplicity. The reason is same as in the previous footnote.

[7] This method is described in section 2.

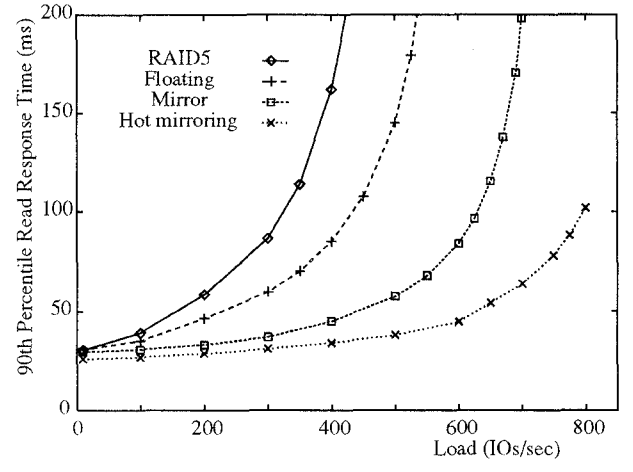
[8] $\frac{0.2/2}{0.8 \times 5/6 + 0.2/2} = 0.13$

[9] For naive RAID5 disk arrays and floating disk arrays, the total capacity is quantized by the capacity of a cylinder. In actuality, the ratio of valid data blocks is a little smaller than 76.5 % in these configurations.

[10] Currently many commercial RAID5 products employ dual controllers as discussed in [MC93]. Important tables are stored in the NV-RAM of the controllers and consistency between the two copies is maintained through appropriate implementation.



(a) Read : Write = 7 : 3



(b) Read : Write = 1 : 1

Figure 5: 90th percentile read response time in normal mode (90-10 access locality)

one million accesses with specified read/write ratios are given. These three million access have a low arrival rate. We perform another one million accesses with the specified read/write ratio with the specified arrival rate. Then we start to take the statistics.

6.2 Read response time analysis in normal mode

Figure 5 shows the 90th percentile read response time for 100,000 access requests for 90-10 access locality for two different write ratios ((a) Read:Write = 7:3, (b) Read:Write = 1:1). The horizontal axis shows the mean arrival rates for I/O requests, the vertical axis shows the read response time.[11]

Hot mirroring shows much better performance than the naive and the floating RAID5's. It has even better performance than the ordinary mirror. In hot mirroring, many of accesses are performed to the mirrored hot area, which are stored on a limited contiguous region. Therefore the average seek time is considerably reduced, which leads to a significant performance improvement.

In hot mirroring, we found that the ratio of write operations which cause block migrations is about 10% for each of two cases. This is the same as the ratio of cold access in 90-10 access locality. This means that only cold block writes cause the block migration. Thus by exploiting the access locality, hot mirroring can attain very high performance, where the migration cost is small enough.

6.3 Read response time analysis during rebuild mode

6.3.1 Details on Rebuilding

In hot mirroring, the rebuilding process is performed as follows. The method of dispatching the access requests for rebuilds has a lot of impact on the performance of the rebuild mode. As described in [HMP93], the unit of reconstructing data affects the performance of the rebuild operation and the rebuild period. In this simulation, we adopt a track as the unit of rebuilding, considering the efficiency of rebuilding process and the influence on normal requests. It is assumed that disks have an intelligent controller and begin track accesses on the sector which the disk head encounters first after the head becomes available. We regard the length of a track access for rebuilding as 4/3 of a normal access.[12]

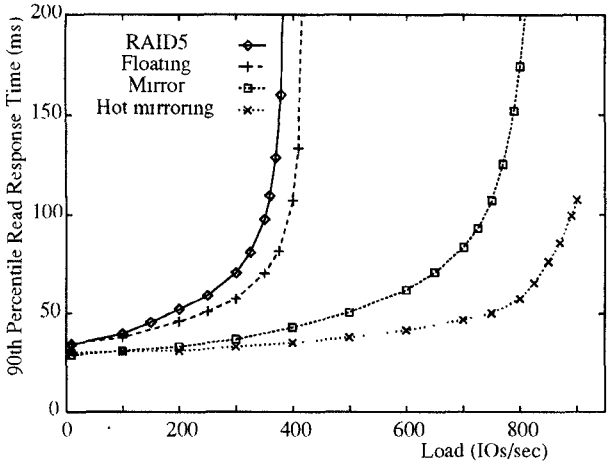
A baseline rebuild is employed in this simulation. For limiting the buffer usage on the array controller, rebuild read requests are issued when the access queue length of the replacement disk is not more than five accesses. In principle, rebuild read requests are queued separately from the normal requests and are served while disks are idle. In the RAID5 area, rebuild read requests are not issued until the previous rebuild read requests are completed. When all the alive data of the fault stripe have been read, a rebuild write request to the replacement disk is issued immediately. We put the limit on the maximum rebuild time. The check period is set to the maximum rebuild time divided by the number of tracks in the reconstructing area. Every check period, we check how much data has been reconstructed. The amount of reconstructed data should be proportional to the elapsed time. If reconstruction is insufficient, we issue more rebuild read requests and chain them to the queue for the normal requests.

Requests against the already rebuilt data are performed on the replacement disk. Requests against the failed block which are not yet rebuilt are served as follows. If it is in the mirrored area, the request is served by the live pair. If it is in the RAID5 area, requests are issued to all the remaining drives which belongs to that stripe.

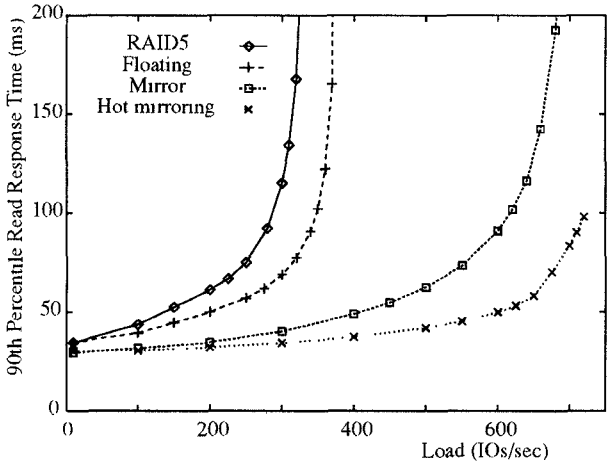
The rebuilding process on naive RAID5, floating RAID5, and mirror is performed in the same way as that of hot mirroring. The procedure is a bit different in floating RAID5.

[11] In the hot mirroring method, we terminated the simulation when there are no free blocks on the mirrored hot area. In such high traffic situation, requests consumes all the disk bandwidth, where there remains no room for migration. This results in all the free blocks being used up.

[12] $\frac{\text{(average seek time)+(half block rotation time)+(one rotation time)}}{\text{(average seek time)+(half rotation time)+(one block rotation time)}}$

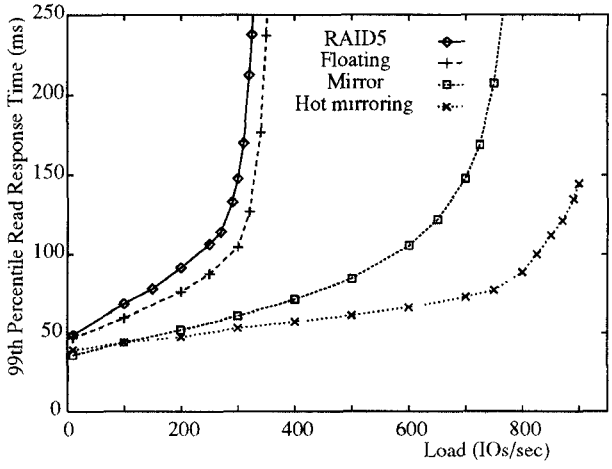


(a) Read : Write = 7 : 3

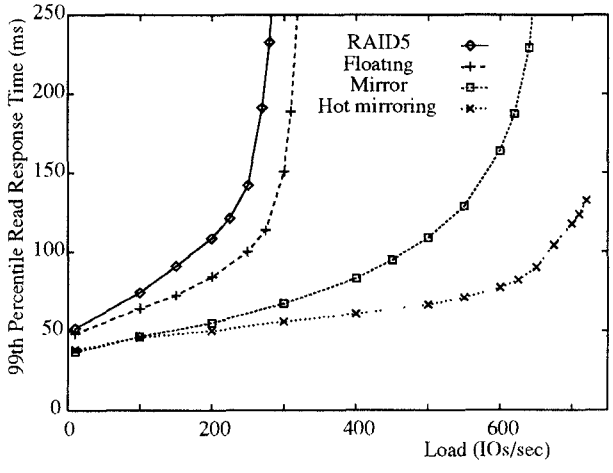


(b) Read : Write = 1 : 1

Figure 6: 90th percentile read response time during rebuild mode (90-10 access locality)



(a) Read : Write = 7 : 3



(b) Read : Write = 1 : 1

Figure 7: 99th percentile read response time during rebuild mode (90-10 access locality)

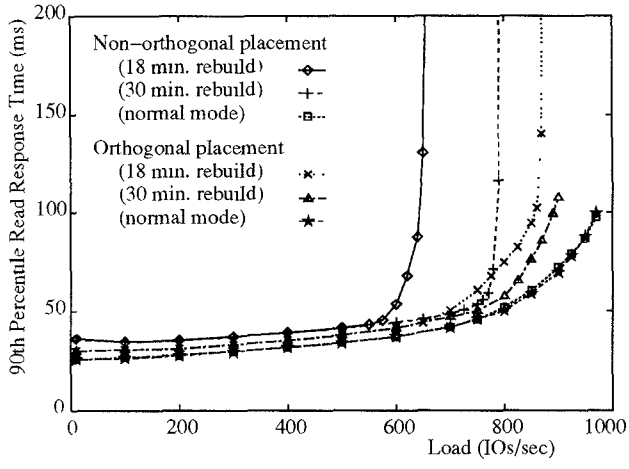
Since the location of the blocks are floating inside a cylinder, rebuilding is done not track by track but cylinder by cylinder. That is, after reading all the data blocks in the cylinder, we can initiate writing the reconstructed data. However, writing the reconstructed data into the replacement drive are not done cylinder by cylinder. If requests come to the already rebuilt data on replacement drive while writing the reconstructed data onto it, we suspend the reconstruction write track by track and serve the normal requests in order to improve the response time of normal accesses. While reading whole data blocks in a cylinder, reading proceeds track by track in the same way. Normal requests are served between track reads.

6.3.2 Response time analysis

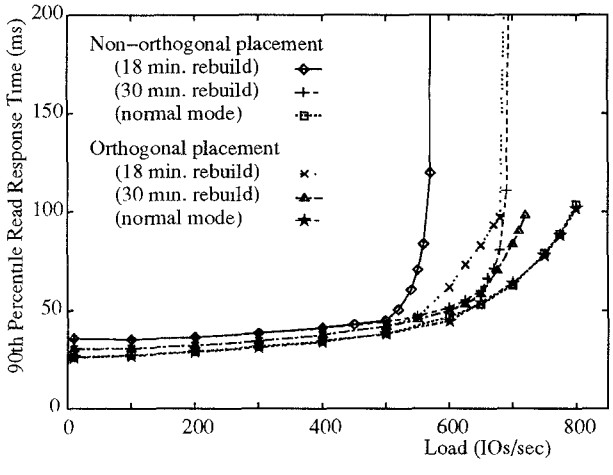
Performance during rebuild mode is as important as that in normal mode. We measured the 90th percentile read response time with 90-10 access locality during the rebuild process. In this simulation, the maximum rebuild time is set to 30 minutes, which is about 5 times longer than (the average time to perform a random seek and one track access) $\times$ (the number of tracks on a disk).

Figure 6 shows the results. The horizontal axis shows the mean arrival rate for I/O requests, the vertical axis shows the read response time. All methods show worse performance than that of normal mode. However, the performance degradation of hot mirroring is much smaller than RAID5 and floating RAID5. Orthogonal layout of hot mirroring minimizes the degradation. During rebuilding also, hot mirroring gives a higher performance than ordinary mirror.

In this simulation, naive RAID5, floating RAID5, and hot mirroring have four disk groups. Only the quarter of the total accesses are given to the broken disk group. The 90th percentile read response time might hide some long delay due to the conflicts with the reconstruction process. In order to clarify this effect, the 99th percentile read response time is examined (Figure 7).



(a) Read : Write = 7 : 3



(b) Read : Write = 1 : 1

Figure 8: Orthogonal placement v.s. non-orthogonal placement during rebuild mode (90-10 access locality)

By comparing figures 6 and 7, we can easily see that naive RAID5 and the floating RAID5 are affected more by disk failure than is hot mirroring. Since the rebuilding requests and the normal requests to the broken data are served by the same group of disk drives, some of the requests take a very long time. On the other hand, in hot mirroring, orthogonal layout and smart load balancing can remove this phenomena.

6.4 Effects of load balancing

6.4.1 Orthogonal placement v.s. non-orthogonal placement

In order to further clarify the effect of the orthogonal layout we compared the 90th percentile read response time between the orthogonal layout and the non-orthogonal layout. In the non-orthogonal layout, mirrored pairs are placed on the same disk group in which parity stripes are maintained. Figure 8 shows the result. The rebuild time limit is set to two different

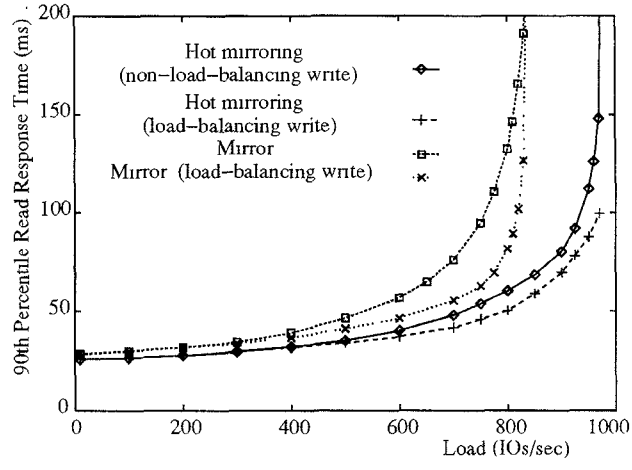


Figure 9: Effect of load balancing on write operations (Normal mode, 90-10 access locality, Read:Write = 7:3)

values, 30 minutes and 18 minutes. In the figure, we also plot the performance during normal mode. As you can see from the two figures, hot mirroring employing an orthogonal layout attains higher performance than that employing a non-orthogonal layout. The rebuild process on the RAID5 area requires a lot of disk bandwidth. If mirrored pairs are placed on the same disk group, the load on the disk group which has the broken disk becomes very high. The orthogonal placement leads to good load balancing on read operations and makes it possible that disks in alive disk groups will be almost fully utilized. This is the reason why the orthogonal placement shows better performance than the non-orthogonal layout, especially when the ratio of disk bandwidth used for the rebuilding process is high.

The degradation from the normal mode in the orthogonal layout is almost the same for the two write ratios. Now we would like to examine why the non-orthogonal layout works better for higher write ratio. The load balancing strategy described in section 5.3 also works in the non-orthogonal layout, which makes the write requests on the broken disk group be served by a different disk group by changing its location. This helps balance the load among the disks and it is effective in a short time. This load balancing is more effective with higher write ratios. Thus the relative performance of the non-orthogonal layout over the orthogonal placement with a 50% write ratio is better than that with a 30% write ratio. However, we cannot do this load balancing for a long time because there are limited free blocks on which blocks escaping from the broken disk group can be stored.

Thus although our smart load balancing works better for the non-orthogonal layout, the orthogonal layout achieves much higher performance than the non-orthogonal one.

6.4.2 Load balancing on write operations

In hot mirroring, the write locations are determined ac-

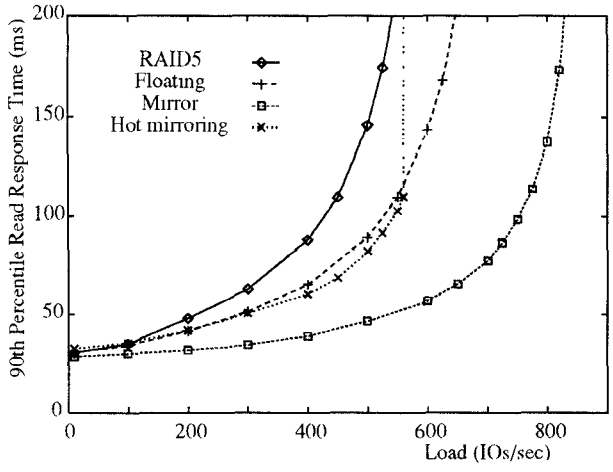


Figure 10: 90th percentile read response time in normal mode (80-20 access locality, Read:Write = 7:3)

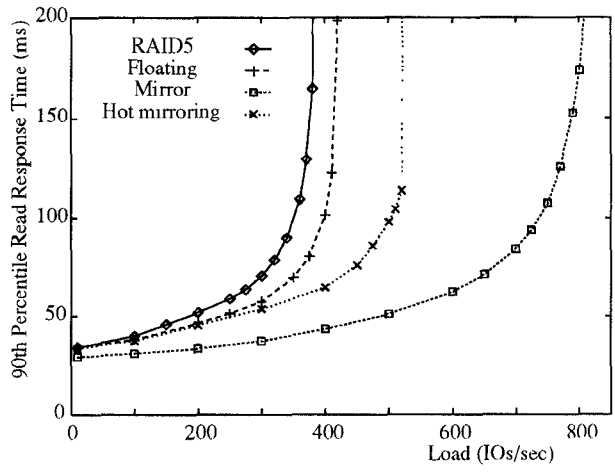
cording to the disk loads as described in section 5.3. When migration occurs between the mirrored area and the RAID5 area, we have to assign a new location. However, if the data on the mirrored area is updated, we have two alternatives. The method we call non-load-balancing write does not change the location. Namely the updated data in the mirrored area is written to its original position. The method we call load-balancing-write changes the location. The new location is determined based on the length of the access queue of each drive. We examined the difference between the two methods by comparing the 90th percentile read response time. The performance is measured in normal mode, where requests have 90-10 access locality with a 30% write ratio.

In the figure, we also plot the performance of the ordinary mirror and the ordinary mirror with load-balancing-write. The difference between with and without load-balancing write in hot mirroring is smaller than in the ordinary mirror. In hot mirroring, the load balancing is already working on migration between mirror and RAID5. This is the reason why the effect of load-balancing-write on the mirrored area in hot mirroring is less than that in the ordinary mirror.

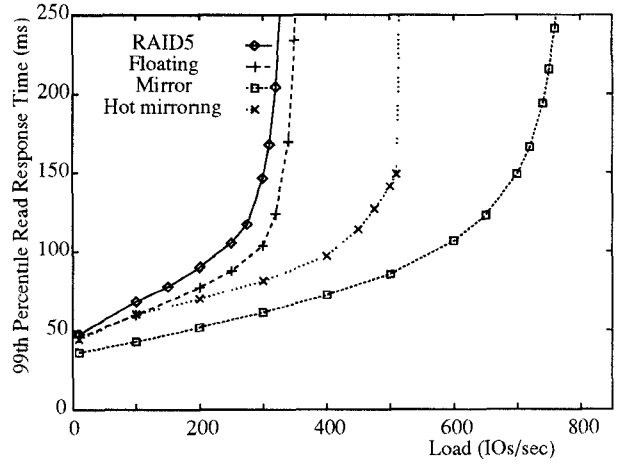
6.5 Effects of access locality

Hot mirroring makes use of access locality to improve performance. The degree of access locality may affect performance. Figure 10 shows the 90th percentile read response time for 100,000 access requests with 80-20 access locality with 30% write ratio in normal mode. The horizontal axis shows the mean arrival rates of requests, the vertical axis shows the read response time.

For 80-20 access locality hot mirroring shows much worse performance than for 90-10 access locality. Comparing between hot mirroring and RAID5 based disk arrays, hot mirroring shows better performance than that of the naive RAID5 but cannot bear as high a load as the floating RAID5 can.



(a) 90th percentile read response time



(b) 99th percentile read response time

Figure 11: Read response time during rebuild mode (80-20 access locality, Read:Write = 7:3, 30 min. rebuild)

In hot mirroring, the hot area occupies about 13% of the data capacity. For 90-10 access locality, all hot blocks can fit in the mirrored hot area. But for 80-20 access locality, all of the hot blocks cannot be stored in the mirrored hot area, which causes frequent block migrations. In this simulation, about 56% of the write operations cause migration. Thus the effectiveness of hot mirroring decreases when access locality is not high enough.

Figure 11 shows the performance for 80-20 access locality during rebuild mode. In the same way as in 90-10 access locality, the performance of the naive RAID5 disk array and the floating disk array are noticeably affected by disk failure and rebuild processing. On the other hand, hot mirroring can still balance the load amongst all the disks although the overhead for write accesses is large. Therefore the performance of hot mirroring becomes better than that of floating RAID5 during rebuild mode even if the locality is low.

7 Conclusion

This paper presents the storage management scheme named hot mirroring. The feasibility of hot mirroring was examined through simulation. For high access localities, hot mirrored disk arrays show much higher performance than naive RAID5, floating RAID5, and ordinary mirror in both normal mode and during rebuild mode. Even for low access locality, hot mirroring does not perform worse than naive RAID5 and performs better than floating RAID5 during rebuild mode. Clustering of hot blocks improves performance significantly. Orthogonal layout and smart load balancing plays an important role during the rebuild process.

There remain several issues which require further investigation, such as dynamic adaptation of the ratio between hot and cold areas. The cost of the mapping table could be reduced by increasing the size of the unit of mapping. Its effects on performance have to be examined carefully.

References

[BD92] Anupam Bhide and Daniel Dias. RAID Architectures for OLTP. Research Report RC 17879 (78489), IBM Research Division, Mar. 1992.

[HD90] H. Hsiao and D. DeWitt. Chained Declustering: A New Availability Strategy for Multiprocessor Database Machines. In *Proc. of 6th Int. Conf. on Data Engineering*, pp. 456–465, Feb. 1990.

[HG92] Mark Holland and Garth A. Gibson. Parity Declustering for Continuous Operation in Redundant Disk Arrays. In *Proc. of 5th Int. Conf. on Architectural Support for Programming Languages and Operating Systems*, pp. 23–34, Oct. 1992.

[HMP93] Robert Y. Hou, Jai Menon, and Yale N. Patt. Balancing I/O Response Time and Disk Rebuild Time in a RAID5 Disk Array. In *Proc. of Hawaii Int. Conf. on System Sciences*, pp. 70–79, 1993.

[HP93] Robert Y. Hou and Yale N. Patt. Comparing Rebuild Algorithms of Mirrored and RAID5 Disk Arrays. In *Proc. of ACM SIGMOD Conf.*, pp. 317–326, May 1993.

[MC93] Jai Menon and Jim Cortney. The Architecture of a Fault-Tolerant Cached RAID Controller. In *Proc. of 20th Annual Int. Symp. on Computer Architecture*, pp. 76–86, May 1993.

[MK92] Jai Menon and Jim Kasson. Methods for Improved Update Performance of Disk Arrays. In *Proc. of 25th Hawaii Int. Conf. on System Sciences*, vol. I, pp. 74–83, Jan. 1992.

[MK94] Kazuhiko Mogi and Masaru Kitsuregawa. Dynamic Parity Stripe Reorganizations for RAID5 Disk Arrays. In *Proc. of 3rd Int. Conf. on Parallel and Distributed Information Systems*, pp. 17–26, Sep. 1994.

[MK95a] Kazuhiko Mogi and Masaru Kitsuregawa. Hot Block Clustering for Disk Arrays with Dynamic Striping — exploitation of access locality and its performance analysis . In *Proc. of 21st VLDB Conf.*, pp. 90–99, Sep. 1995.

[MK95b] Kazuhiko Mogi and Masaru Kitsuregawa. Preliminary Evaluation of Hot Mirroring. IPSJ SIG Notes 95–DBS–104–7, Information Processing Society of Japan, Jul. 1995.

[MK95c] Kazuhiko Mogi and Masaru Kitsuregawa. Performance Evaluation of Hot Mirrored Disk Arrays on Disk Failure. IEICE Tech. Report CPSY95–82, Institute of Electronics, Information and Communication Engineering, Dec. 1995.

[ML90] Richard R. Muntz and John C.S. Lui. Performance Analysis of Disk Arrays under Failure. In *Proc. of 16th VLDB Conf.*, pp. 162–173, Aug. 1990.

[MM91] Jai Menon and Dick Mattson. Performance of Disk Arrays in Transaction Processing Environments. Research Report RJ 8230, IBM Research Division, 1991

[MM92] Jai Menon and Dick Mattson. Comparison of Sparing Alternatives for Disk Arrays. In *Proc. of 19th Annual Int. Symp. on Computer Architecture*, pp. 318–329, May 1992.

[OS93] Cyril U. Orji and Jon A. Solworth. Doubly Distorted Mirrors. In *Proc. of ACM SIGMOD Conf.*, pp. 307–316, May 1993.

[OWS93] Cyril U. Orji, Mark A. Wiess, and Jon A. Solworth. Improved Traditional Mirror. In *Proc. of 4th Int. Conf., FODO '93*, pp. 329–344, Oct. 1993.

[PGK88] David A. Patterson, Garth Gibson, and Randy H. Katz. A Case for Redundant Arrays of Inexpensive Disks (RAID). In *Proc. of ACM SIGMOD Conf.*, pp. 109–116, Jun. 1988.

[RCB93] A. L. Narasimha Reddy, John Chandy, and Prithviraj Banerjee. Design and Evaluation of Gracefully Degradable Disk Arrays. *Journal of Parallel and Distributed Computing*, 17:28–40, 1993.

[RO91] Mendel Rosenblum and John Ousterhout. The Design and Implementation of a Log-Structured File System. In *Proc. of 13th ACM Symp. on Operating Systems Principles*, pp. 1–15, Oct. 1991.

[SG93] Daniel Stodolsky and Garth A. Gibson. Parity Logging: Overcoming the Small Write Problem in Redundant Disk Arrays. In *Proc. of 20th Annual Int. Symp. on Computer Architecture*, pp. 64–75, May 1993.

[SO91] Jon A. Solworh and Cyril U. Orji. Distorted Mirrors. In *Proc. of 1st Int. Conf. on Parallel and Distributed Information Systems*, pp. 10–17, Dec. 1991.

[Ter85] Teradata Corporation. *DBC/1012 Database Computer System Manual Release 2.0*, Nov. 1985. Doc. No. C10-0001-02.

[WGSS95] John Wilkes, Richard Golding, Carl Staelin, and Tim Sullivan. The HP AutoRAID Hierarchical Storage System. In *Proc. of 15th ACM Symp. on Operating Systems Principles*, Dec. 1995.

[YWD93] Philip S. Yu, Kun-Lung Wu, and Asit Dan. Dynamic Parity Grouping for Efficient Parity Buffering to Improve Write Performance of RAID-5 Disk Arrays. Research Report RC 19041 (83137), IBM Research Division, Jul. 1993.