Stripped mirroring RAID architecture

Hai Jin \textsuperscript{a,b,*}, Kai Hwang \textsuperscript{b}

\textsuperscript{a} The University of Hong Kong, Pokfulam Road, Hong Kong
\textsuperscript{b} Department of Electrical Engineering, University of Southern California, Los Angeles, CA 90089, USA

Received 10 September 1998; received in revised form 17 February 1999; accepted 1 July 1999

Abstract

Redundant arrays of independent disks (RAID) provide an efficient stable storage system for parallel access and fault tolerance. The most common fault tolerant RAID architecture is RAID-1 or RAID-5. The disadvantage of RAID-1 lies in excessive redundancy, while the write performance of RAID-5 is only 1/4 of that of RAID-0. In this paper, we propose a high performance and highly reliable disk array architecture, called stripped mirroring disk array (SMDA). It is a new solution to the small-write problem for disk array. SMDA stores the original data in two ways, one on a single disk and the other on a plurality of disks in RAID-0 by stripping. The reliability of the system is as good as RAID-1, but with a high throughput approaching that of RAID-0. Because SMDA omits the parity generation procedure when writing new data, it avoids the write performance loss often experienced in RAID-5. © 2000 Elsevier Science B.V. All rights reserved.

Keywords: Disk array architecture; Mirroring; Parallel I/O; Fault tolerant; Performance evaluation

1. Introduction

Redundant arrays of independent disks (RAID) \cite{2,5} systems deliver higher throughput, capacity and availability than can be achieved by a single large disk by hooking together arrays of small disks. RAID technology is an efficient way to solve the bottleneck problem between CPU processing ability and I/O processing speed \cite{4}. The tremendous growth of RAID technology has been driven by three factors. First, the growth in processor speed has outstripped the growth in disk data rate. The imbalance transforms traditionally computer-bound applications to I/O-bound applications. Therefore, I/O system throughput must be increased by increasing the number of disks. Second, arrays of small-diameter disks often have substantial cost, power and performance advantages over larger drives. Third, such systems can be made highly reliable by storing a small amount of redundant information in the array. Without this redundancy, large disk arrays have unacceptable low data reliability because of their large number of component disks.

Fig. 1 presents an overview of the RAID systems considered in this paper. This figure only shows the first few units on each disk in different RAID levels. “D” represents a block, or unit, of user data (of unspecified size, but some multiple of one sector) and “Px – y” a parity unit computed over user data units x through y. The numbers on the left indicate the offset into the raw disk, expressed in data units. Shaded blocks represent

* Corresponding author.

E-mail addresses: hjin@ceng.usc.edu, hjin@eee.hku.hk (H. Jin), kaihwang@usc.edu (K. Hwang).
redundant information, and nonshaded blocks represent user data.

RAID-0 is nonredundant and does not tolerate faults. RAID-1 is simple mirroring, in which two copies of each data unit are maintained. RAID-5 exploits the fact that failed disks are self-identifying, achieving fault tolerance using a simple parity (exclusive-or) code, lowering the capacity overhead to only one disk out of six in this example. In RAID-5, the parity blocks rotate through the array rather than being concentrated on a single disk, avoiding parity access bottleneck [12]. RAID-10 [11,19] combines RAID-0 and RAID-1 in a single array. It provides data reliability through RAID-1 and enhanced I/O performance through disk stripping.

While RAID-5 disk arrays offer performance and reliability advantages for a wide variety of applications, they possess at least one critical limitation: their throughput is penalized by a factor of four to RAID-0 for workloads of small writes. This penalty arises because a small-write request may require that the old value of user's targeted data be read (pre-read), overwriting this with new user data, pre-reading the old value of the corresponding parity, then overwriting this second disk block with the updated parity. In contrast, systems based on mirrored disks simply write the user's data on two separate disks and, therefore, are only penalized by a factor of two. This disparity, four accesses per small write instead of two, has been termed the small-write problem [21].

Small-write performance is important. The performance of on-line transaction processing (OLTP) system is largely determined by small-write performance. A single read-modify-write of an account record will require five disk accesses for RAID-5, while the same operation would require three accesses on mirrored disks, and only two on RAID-0. Because of this limitation, many OLTP systems continue to employ the much more expensive option of mirrored disks.

In this paper, we propose a new RAID architecture with high reliability and high performance, called stripped mirroring disk array (SMDA). It is a new solution to the small-write problem for disk array. SMDA stores the original data in two ways, one in a single disk and the other in a plurality of disks in the way of RAID-0. Section 2 reviews related works. Section 3 discusses the stripped mirroring disk array mechanism. Section 4 analyzes the read/write performance of different RAID architectures. Section 5 closes with the conclusions and future work.

2. Related works

Many studies have been previously proposed to deal with the RAID write penalty. In this section, we will give a brief review of the related works.

Parity stripping [3] stripes the parity across the disks, but does not stripe the data. It is based on the fact that the mirrored disk has higher availability and higher throughput. Small random request can achieve higher transfer throughput without stripping large data into small pieces. The disk utilization and throughput of parity stripping are similar to that of the mirrored disk, and have cost/GB comparable to RAID-5. Parity stripping has preferable fault containment and operations features compared with RAID-5.

Floating data/parity structure [18] uses the method of shortening "read-modify-write" time when modifying data or parity information so as to solve the small-write problem. All the data blocks and parity blocks correspond to the different cylinders separately. In each cylinder, one track is
preserved to keep the modification result of the data block and parity block. This method can reduce response time of small write by the floating address physical space on the disk. But it is not so convenient to the large size request. After many instances of small write, many logically continuous blocks are physically separated. Thus it enlarges the rotation delay of accessing logically continuous data.

Parity logging \[21,22\] sets a large capacity of cache or buffer to combine some small writes to a large writes to improve the data transfer rate and reduce the time of modifying parity information. It stores all the modification of parity information as a log in the logging cache. When the logging cache is fulfilled, it writes parity information in large blocks to the parity log disk in a serial way. When the parity log disk is fulfilled, it reads all the information in the parity log disk and the information of the parity disk or data disk to reconstruct parity information. One of the disadvantages of parity logging is that when the parity log disk is fulfilled, the I/O request should be blocked so as to reconstruct parity information, and these operations should be carried on in a front end way. Besides, it can only reduce the parity block accessing time and has no influence on the data block.

Disk caching disk (DCD) \[7\] uses a small log disk, referred to as cache-disk, as secondary disk cache to optimize write performance. While the cache disk and the normal data disk have the same physical properties, the access speed of the former differs dramatically from the latter because of different data units and different ways in which data are accessed. DCD exploits this speed difference by using the log disk as a cache to build a reliable and smooth disk hierarchy. A small RAM buffer is used to collect small-write requests to form a log that is transferred onto the cache disk whenever the cache disk is idle. Because of the temporal locality, the DCD system shows write performance close to the same size RAM for the cost of a disk.

The floating-location technique improves the efficiency of writes by eliminating static association of logical disk blocks and fixed locations in the disk array. When a disk block is written, a new location is chosen in a manner that minimizes the disk-arm time devoted to the write, and a new physical-to-logical mapping is established. An example of this approach is the log-structure file system (LFS) \[13,14\]. It writes all modifications to the disk sequentially in a log-like structure, thereby speeding up both file writing and crash recovery. The log is the only structure on the disk; it contains indexing information so that the file can be read back from the log efficiently. In order to maintain large free areas on disk for fast writing, LFS divides the log into segments and uses a segment cleaner to compress the live information from heavily fragmented segments. However, because logically nearby blocks may not be physically nearby, the performance of LFS in read-intensive workloads may be degraded if the read and write access patterns differ widely.

The distorted-mirror approach \[20\] uses the 100% storage overhead of mirroring to avoid this problem: one copy of each block is stored in fixed location, while the other copy is maintained in floating storage, achieving higher write throughput while maintaining data sequentially. However, all floating-location techniques require substantial host or controller storage for mapping information and buffered data.

Write buffering \[16,24\] delays users’ write requests in a large disk or file cache to achieve deep queue, which can then be scheduled to substantially reduce seek and rotational positioning overheads. Data loss on a single failure is possible in these systems unless fault-tolerant caches are used \[17\].

In the traditional mirrored system, all disks storing a mirrored collection are functional, but each may offer a different throughput over time to any individual reader. In order to avoid the performance consistency, the graduated declustering approach \[1\] will fetch data from all available data mirrors instead of picking a single disk to read a partition from. In the case where data are replicated on two disks, disk 0 and disk 1, the client will alternatively send a request for block 0 to disk 0, then block 1 to disk 1; as each disk responds, another request will be sent to it, for the next desired block.
3. Stripped mirroring disk array architecture

This section discusses the architecture of striped mirroring disk array (SMDA). Our approach is motivated by the fact that RAID-0 has the highest data transfer rate and the maximum I/O rate for both read and write, while RAID-1 has the highest reliability among all the RAID levels. SMDA stores the original data in two ways, with original data stored in one disk drive and duplicated data distributed and stored in different disk drives with the method of RAID-0.

Fig. 2 shows a typical data layout of SMDA that composes of five disk drives. This figure only shows the first few units (where 16 units) on each disk. The numbers on the left indicate the offset into the raw disk, expressed in data units. Non-shaded blocks represent the original information, and shaded blocks represent the duplicated data that are distributed among all the other disks in the array.

SMDA comprises a plurality of disk drives, while the number of disk drives in the array is at least 3. Suppose the number of disks in the array is \( N \), then \( N \geq 3 \). The disk array controller controls the writing of data to each of the disks in the array and reading of data from each of the disks in the array.

A disk drive unit connected to the disk control unit of the disk array has a logical group formed of a plurality of disk drives. In our example, there are two logical groups. The length of each logical group in each disk drive is \( 2(N - 1) \) blocks. In each logical group, both original data and duplicated data of the original data are stored. Each logical group is divided into two sub-groups, while the original data are stored in the first sub-group of each disk drive and the duplicated data are stored in the second sub-group distributed among all the other disk drives. The original data blocks D0, D1, D2 and D3 are stored in the first sub-group of disk 0 in our example. The duplicated data blocks D0, D1, D2 and D3 are stored in the second sub-group of the first logical group in disks 1, 2, 3 and 4, respectively.

We call each of the locations stored in the original data in the first sub-group and the locations stored in the duplicated data distributed in the second sub-group among all the other disk drives in the array an area pair. In the example, the locations of first sub-group of disk 0 and the locations of the fifth data block in disks 1, 2, 3 and 4 belong to one area pair. The locations of the first sub-group of disk 1 and the locations of the fifth data block in disk 0 and the sixth data block in disks 2, 3 and 4 belong to another area pair. The area for original data and an area for duplicated data belonging to a same area pair are located in different respective disk drives. A set of \( n \) area pairs (where \( 2 \leq n \leq N - 1 \)) which have their areas for original data on a common disk drive, the corresponding areas for duplicated data are distributed in one-to-one correspondence across each of \( n \) other disk drives. In this way, the original data and the duplicated data are stored in different drives.

When a plurality of data stored in the disk array are to be read, the original data in one area or the duplicated data distributed in the same area pair among all the other disks are read in parallel from the different drives. In the above example, if data blocks D0, D1, D2 and D3 are to be read from the array, they can be read from disks 1, 2, 3 and 4 in parallel. If only one data block, say D0, is to be read from the array, it can be read both from disk 0 and 1 in parallel. As the duplicated data are stored in the area among all the other disk drives.
in the array in the fashion of RAID-0, SMDA illustrates the higher I/O performance by reading the duplicated data out in parallel from the disk drives.

When a plurality of data are written to the disk array, they are written to the area for original data and another area for duplicated data belonging to the same area pair in parallel with the different disk drives in the array. In the above example, if data block D0 is to be written to the array, it can be written to the disk 0 and 1 in parallel. As it only writes the original data and duplicated data to the disk drives in the array, without keeping the parity information of the data information, it avoids the write performance loss when using SMDA architecture. Because it omits the parity generation procedure when writing the new data, the overall performance of SMDA is the same as that of RAID-0.

The fault tolerance of SMDA architecture is realized by using the original data and the duplicated data among all the disks in the array. In case of one disk drive crashes in SMDA, the locations to store the original data can be read out from all the other disk drives in the disk array. The locations to store the part of duplicated data can be read out from the disk drive storing the original data in the same area pair of disk array. In the above example, suppose disk 2 is in failure, let us consider the data blocks in the first logical group. The original data blocks D8, D9, D10 and D11 can be read from disks 0, 1, 3 and 4, respectively. The duplicated data blocks D1, D5, D14 and D18 can be read from disks 0, 1, 3 and 4, respectively.

4. Modeling and performance evaluation

In this section we present a utilization-based analytical model to compare the I/O access performance of RAID-0, RAID-1, RAID-5, RAID-10 and SMDA. RAID-0 here is just for comparison only, we will not use it in any disk array system because of lack of fault tolerance. This model predicts saturated array performance in terms of achieved disk utilization.

The variables used in this model are defined as follows:

- \( B \): Amount of data to be accessed
- \( N \): Numbers of disks in array
- \( D \): Data units per track
- \( T \): Tracks per cylinder
- \( S \): Average seek time
- \( M \): Single track seek time
- \( R \): Average rotational delay (1/2 disk rotation time)
- \( H \): Head switch time

We define the unit read as the read access from only one data block in the array, while unit write as the write access to only one data block in the array. Unit read time \( r \) and unit write time \( w \) do not include the start-up mechanical delay time, which may include seek time, head switch time and rotational delay.

For the read access, the situation is quite simple. All the different RAID architectures have the same unit read time, that is \( 2R/D \).

For the write access, the situation for RAID-5 is quite different from other RAID architectures. For RAID-5, small writes require four I/Os: data pre-read, data write, parity read, parity write. These can be combined into two read–rotate–write accesses. Each read–rotate–write access can be done in an I/O that reads the data, waits for the disk to spin around once, then updates the data. Each unit write time for RAID-5 is \( 2R/D + (2R - 2R/D) + 2R/D \). For each small write, there are two unit writes.

For RAID-0, RAID-1, RAID-10 and SMDA, no pre-read is required. The unit write time is \( 2R/D \).

Next, we discuss the start-up mechanical delay for different RAID architectures. There are three different types of mechanical start-up delays for each I/O accesses, they are seek time, head switch time and rotational delay. Seek operation happens when the head of the disk seeks user data among different cylinders. Head switch happens when the head of the disk changes in the same cylinder. Rotational delay happens when the head of the disk waits for the data to rotate under the head. Because the data layout in each RAID architecture is different, the head switch times \( m_1 \) and cylinder
switch time \((m_2)\) are different. All these values are listed in Tables 1 and 2.

Tables 1 and 2 compare the read and write access time for different RAID architectures among RAID-0, RAID-1, RAID-5, RAID-10 and SMDA.

From the discussion, we can see that using the architecture of SMDA can greatly improve the I/O throughput of disk array. Because of the small-write problem, RAID-5 has the lowest I/O throughput among five RAID architectures. RAID-1 has the limited throughput because only one pair of disks can be accessed in parallel. RAID-10 has half the peak throughput as only one pair of disks can be accessed in parallel. For SMDA, as the maximum disks that can be accessed in parallel is \(N - 1\), the total throughput can achieve as high as \(N - 1/N\) peak throughput. Here we assume RAID-0 can achieve peak throughput.

SMDA has the higher throughput even in the degraded mode and the rebuilt mode. In the case of degraded mode, there is no need to modify the extra data to keep the data consistent as in the RAID-5. It only writes one copy, whether original or duplicated depends on the locations where the data blocks are written. Thus, even in the degraded mode, SMDA performs the highest I/O performance just as in the normal mode.

In the time of rebuild, it is much more easy to recovery the failure data to the newly replaced disk drive just using the copy operation avoiding the operation of reading all the other data as well as the parity information to perform exclusive OR operation in RAID-5. Therefore, it greatly reduces the rebuild time and MTTR.

The reliability of SMDA is the same as RAID-1 and RAID-10, which has the highest reliability among all the RAID architectures.

### Table 1

Comparisons of read access time for different RAID architectures

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Head switch times ((m_1))</th>
<th>Cylinder switch times ((m_2))</th>
<th>Unit read time ((r))</th>
<th>Total read time</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAID-0</td>
<td>(B_i/ND)</td>
<td>(B_i/NDT)</td>
<td>(2R/D)</td>
<td>((S + R) + r + (m_2 - 1) \times (M + R) + r + (m_1 - m_2 - 1) \times (H + R) + r)</td>
</tr>
<tr>
<td>RAID-1</td>
<td>(B_i/2D)</td>
<td>(B_i/2DT)</td>
<td>(2R/D)</td>
<td>((S + R) + r + (m_2 - 1) \times (M + R) + r + (m_1 - m_2 - 1) \times (H + R) + r)</td>
</tr>
<tr>
<td>RAID-5</td>
<td>(B_i/(N - 1)D)</td>
<td>(B_i/(N - 1)DT)</td>
<td>(2R/D)</td>
<td>((S + R) + r + (m_2 - 1) \times (M + R) + r + (m_1 - m_2 - 1) \times (H + R) + r)</td>
</tr>
<tr>
<td>RAID-10</td>
<td>(B_i/2D)</td>
<td>(B_i/2DT)</td>
<td>(2R/D)</td>
<td>((S + R) + r + (m_2 - 1) \times (M + R) + r + (m_1 - m_2 - 1) \times (H + R) + r)</td>
</tr>
<tr>
<td>SMDA</td>
<td>(B_i/(N - 1)D)</td>
<td>(B_i/(N - 1)DT)</td>
<td>(2R/D)</td>
<td>((S + R) + r + (m_2 - 1) \times (M + R) + r + (m_1 - m_2 - 1) \times (H + R) + r)</td>
</tr>
</tbody>
</table>

### Table 2

Comparisons of write access time for different RAID architectures

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Head switch times ((m_1))</th>
<th>Cylinder switch times ((m_2))</th>
<th>Unit write time ((w))</th>
<th>Total write time</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAID-0</td>
<td>(B_i/ND)</td>
<td>(B_i/NDT)</td>
<td>(2R/D)</td>
<td>((S + R) + w + (m_2 - 1) \times (M + R) + w + (m_1 - m_2 - 1) \times (H + R) + w)</td>
</tr>
<tr>
<td>RAID-1</td>
<td>(B_i/2D)</td>
<td>(B_i/2DT)</td>
<td>(2R/D)</td>
<td>((S + R) + w + (m_2 - 1) \times (M + R) + w + (m_1 - m_2 - 1) \times (H + R) + w)</td>
</tr>
<tr>
<td>RAID-5</td>
<td>(B_i/(N - 1)D)</td>
<td>(B_i/(N - 1)DT)</td>
<td>(2R/D)</td>
<td>((S + R) + w + (m_2 - 1) \times (M + R) + w + (m_1 - m_2 - 1) \times (H + R) + w)</td>
</tr>
<tr>
<td>RAID-10</td>
<td>(B_i/2D)</td>
<td>(B_i/2DT)</td>
<td>(2R/D)</td>
<td>((S + R) + w + (m_2 - 1) \times (M + R) + w + (m_1 - m_2 - 1) \times (H + R) + w)</td>
</tr>
<tr>
<td>SMDA</td>
<td>(B_i/(N - 1)D)</td>
<td>(B_i/(N - 1)DT)</td>
<td>(2R/D)</td>
<td>((S + R) + w + (m_2 - 1) \times (M + R) + w + (m_1 - m_2 - 1) \times (H + R) + w)</td>
</tr>
</tbody>
</table>
5. Conclusions and future work

This paper presents a new solution to the small-write problem and high I/O load applications in disk array. We store the original copy on one disk drive while distributing the duplicated copies to other drives in the array. The proposed technique achieves substantially higher performance than conventional RAID-5 arrays. The data should not be read in advance. There is no need to keep the parity information as it does not use the method of parity encoded fault tolerant algorithm.

Compared with other RAID architectures, stripped mirroring RAID architecture (SMDA) can achieve nearly the peak throughput \((N - 1/N)\). Although the reliability of SMDA is the same as RAID-1 and RAID-10, the SMDA may lead to higher throughput than RAID-1 and RAID-10.

One application of SMDA architecture is in the design of I/O systems for cluster of computers. Clusters of workstations [9] are often used in I/O intensive applications, especially in the business world. High availability in cluster operations demands both high bandwidth and fault tolerance in the distributed disk arrays. Different distributed RAID architectures were proposed to enhance the reliability of clusters [6,15,23]. We proposed a hierarchical checkpointing scheme using mirroring architecture to build high availability cluster of workstations [8,10]. In order to improve the throughput of mirroring architecture, we use SMDA architecture to store mirrored checkpoints. We hope SMDA architecture can be adopted by RAB [19] and as many manufacturers as possible to be an extension of standard RAID levels.

References


Kai Hwang received the Ph.D. degree in electrical engineering and computer science from the University of California at Berkeley in 1972. He is a Professor of Computer Engineering at the University of Southern California. Prior to joining USC, he has taught at Purdue University for many years. An IEEE Fellow, he specializes in computer architecture, digital arithmetic, parallel processing, and distributed computing. He has published over 150 scientific papers and six books in computer science and engineering.

He has served as a distinguished visitor of the IEEE Computer Society, on the ACM SigArch Board of Directors, and the founding Editor-in-Chief of the Journal of Parallel and Distributed Computing. He has chaired the International Conferences, ARITH-7 in 1985, ICPP '86, IPPS '96, and HPCA-4 in 1998. His current interests focus on fault tolerance and single system image in multicore clusters and integrated information technology for multi-agent, Java, Internet, and multimedia applications.

Hai Jin is the Professor of computer science at Huazhong University of Science and Technology, Wuhan, China. He obtained B.S., M.S. degree of computer science from Huazhong University of Science and Technology in 1988 and 1991, respectively. He obtained his Ph.D. degree in electrical and electronic engineering in 1994 from Huazhong University of Science and Technology. He is the associate dean of School of Computer Science and Technology at Huazhong University of Science and Technology. In 1996, he got the scholarship awarded by German Academic Exchange Service (DAAD) for academic research at Technical University of Chemnitz-Zwickau in Chemnitz, Germany. Now he is a post-doctoral research fellow in Department of Electrical and Electronic Engineering at the University of Hong Kong, where he participated in the HKU Pearl Cluster project. Presently, he worked as a visiting scholar at the Internet and Cluster Computing Laboratory at the University of Southern California, where he engages in the USC Trojan Cluster project.

He served as program committee member of PDPTA’99, IWCC’99. He has co-authored three books and published nearly 30 papers in international journals and conferences. His research interests cover computer architecture, parallel I/O, RAID architecture design, high performance storage system, cluster computing, benchmark and performance evaluation, and fault tolerant.