A Technique for Managing Mirrored Disks

Darrell D. E. Long†
Computer Science Department
Jack Baskin School of Engineering
University of California, Santa Cruz

Abstract

Disk mirroring is a highly effective technique for improving the fault tolerance and performance of storage systems. Mirroring requires that those disks holding the current data must be determined when recovering from total system failure. In a two disk system, this is only a minor problem since the operator can indicate which disk is current. The complexity increases when more disks are introduced, and requiring human intervention is impractical when the target environment is the consumer market. In these situations, automatic recovery must be used when possible.

We describe an efficient algorithm for managing the consistency of mirrored storage. Our algorithm requires only $n + \log n$ *bits of state per disk and does not require logging or quorum collection.*

1 Introduction

Mirrored storage provides several advantages, including better fault tolerance and increased read performance. The use of this technology has normally been confined to high-end systems, since it was considered to be expensive both in terms of the extra storage required, and in terms of the processing necessary to implement it. Trends in technology have mitigated both of these concerns; in particular the cost of storage has dropped dramatically, and it is now practical to provide mirroring in small systems as well as large.

When disks fail in a mirrored system, writes are allowed to continue on the surviving disks. This is one of the fundamental reasons for employing mirroring. Temporary failures of the system are common (such as the loss of power), and these events make it necessary to determine which disks are current and which are out of date in a reliable and automatic manner.

Consider the scenario in which a disk suffers a temporary failure, such as a loose connection. The system will continue operation, but will eventually be shut down so the connection can be repaired. When the system recovers, all disks will be again on-line, but the system must determine which disks are current.

†The author is a Visiting Scientist at IBM Almaden Research Center. He is grateful for the input of his colleagues, in particular R. Fagin, N. Hanami, S. Edelman, A. Lam, J.-F. Pâris and J. Wyllie.

While this may seem to be an uncomplicated task, it is easy to show that simple schemes, such as using version numbers, are insufficient, since there are sequences of failures that make an out of date disk indistinguishable from the current disk. For example, suppose there are two disks, both of which have failed. When one of the disks recovers, and has version number k, is it the current disk? There is no way to tell until the second disk has been repaired and the version numbers can be compared. The problem is equivalent to determining the last process to fail [8].

We have developed a technique for disk mirroring that has very small metadata requirements. It provides better reliability than most other methods that are employed in practice. It can be implemented efficiently for as few as two disks and scales to an arbitrary number of disks.

Maintaining correct metadata is critical to the operation of the system. It should also be noted that failures are rare events. The only time metadata is modified using our scheme is when a failure is noticed, so most of the time the system will simply apply each write to every disk. As a result, the amortized cost of maintaining the metadata is nearly zero.

In the following sections, we describe our method, and using standard Markov analysis show that it provides excellent availability. In particular, it is shown to be significantly better than quorum consensus.

2 Related Research

The amount of published research on how to implement mirrored storage is surprisingly small. In contrast to logging, which has been carefully described [5, 2], most references discuss mirroring in a general context, and leave the details of implementation to be resolved later.

There are several methods in use for managing mirrored storage. Perhaps the simplest technique is to designate one disk as the *primary*. Its advantage is in its simplicity; its primary disadvantage is that it limits availability and necessitates manual intervention if the primary copy fails to recover following a total failure.

A possible solution is to use *quorum consensus* to determine which copies of the data are current. This has the advantage of making automatic recovery possible, but has the disadvantage of requiring at least three disks to improve availability.

0-7803-7001-5/01/$10.00 © 2001 IEEE

A third common technique for determining which disk is current is to use a log. This has the advantage of providing excellent availability, but has the disadvantage of the added complexity of maintaining the log. In particular, the issue of providing a reliable log becomes an interesting one. Hardening of the log is done through redundancy, such as mirroring. While the semantics of logs allow them to be implemented reliably, we believe that a technique that does not require this added complexity has many benefits.

Another advantage of mirrored storage is its performance benefits. Mirroring has been shown to have better read performance than RAID [10], and that it can improve read performance over that of a single disk. While in practice, a read operation is usually issued on only one disk, if the read were issued in parallel on all disks then its expected performance would improve. The analysis is simple: since there are several read/write heads available, and if we assume that the disks are not synchronized, then the read request can return as soon as the first disk has completed the read operation.

Conversely, a mirrored storage system requires that all disks be written before the write operation can complete (this may be relaxed with the addition of a stable cache, for example using non-volatile memory). It can be shown that the seek time will tend toward the maximum seek time for a single disk as the number of disks increases. In practice, though, a small number of disks will be used. In the case of three disks, the expected divergence from the mean seek time is only 50% [9].

Our method is related to research into efficient methods for implementing the *available copy protocol* [1]. Our earlier research [7] led to an efficient method for finding the last site to fail using a data structure called *was-available sets*. The contents of these sets change only when a failure or a recovery operation occurs, and then only at those sites that participated in the operation. As a result, the computation of a closure operation on those sets was required. Version numbers were then used to determine which sites in the closure were current.

Our new method replaces this closure operation with a simple test for equality, which results in a significantly simplified algorithm and a slight increase in the availability provided. The need for version numbers has also been removed, with the result that the metadata storage requirement is reduced.

3 Mirroring Algorithm

The central issue for managing mirrored storage is determining the current state of the system following a failure. Since partitions are not a concern in disk mirroring, we were able to exploit this fact to develop an algorithm that can tolerate $k-1$ failures among k mirrored disks.

Our algorithm works by tracking the current version of the data using a simple data structure we call *cohort sets*. These sets, a copy of which is stored on each disk, indicate which disks participated in a given write operation. Using only this information, we can determine which disks hold the current version of the data.

3.1 Cohort Sets

The goal of the recovery algorithm is to determine which disks are current. A current disk is, by definition, the one that participated in the most recent write operation.

A *cohort set* for a given disk is the set of all mirrored disks which last participated with that disk in a write operation. Cohort sets require n bits per disk, where n is the number of disks. If there is no explicit numbering on the disks, then an additional $\log n$ bits per disk are required to provide that numbering.

Cohort sets are similar to the *was-available sets* used for the available copy protocol, except that the condition for finding the last copy to fail is much simpler. In fact, our method has been extended to the available copies protocol, resulting in a significant simplification in the recovery procedure [4].

Our approach is to record membership information in these cohort sets. For example, if there are three disks, A, B and C, and C_A, C_B and C_C are the corresponding cohort sets, Table 1 can be used to illustrate what happens when disks fail and recover. Suppose that the system starts with a full complement of disks. At some time in the future, disk C fails and a write operation occurs. The state of the system is reflected in the second row of the table, where $C_A = C_B = \{A, B\}$, which indicates that disks A and B were the only participants in the last write operation. At some point further on, disk B also fails, followed by a write operation. This is reflected in the third row, where only disk A is current. Suppose now that the other two disks recover, then the state of the system is reflected in the fourth row. Which disks are current? Only disk A, since $C_A = \{A\}$.

Table 1: An Example of Failure and Recovery

C_A	C_B	C_C
{A, B, C}	{A, B, C}	{A, B, C}
{A, B}	{A, B}	—
{A}	—	—
{A}	{A, B}	{A, B, C}

We observe that the set of current disks must have cohort sets that are both *equal* and *complete*. By equal, we mean that the cohort sets of all disks in this set must agree.

By complete we mean that all disks listed in the cohort sets must be accessible.

For example, suppose that we have three disks, and let $C_A = \{A, B\}$, $C_B = \{A, B\}$ and $C_C = \{B, C\}$. In this case C_A and C_B are both equal and complete, and so the disks are current, while C_C indicates a disk that is out of date.

Definition 3.1 *The* current set *is the set of all disks* G *such that for each disk* $i \in G$, *its cohort set* $C_i \subseteq G$ *meets the condition*:

$$\forall j \in C_i, C_j = C_i.$$

That is, for any disk in the current set, its cohort set must contain exactly those disks in the current set. Since the cohort sets contain exactly the current set, it is necessary and sufficient to verify that these cohorts sets are equal.

Our algorithm operates by recording changes in the system configuration as they are detected, through write operations. The write operation records in the cohort sets of the participating disks the identity of all disks involved in that operation.

When a disk has been repaired, it must perform a recovery procedure to ensure that it holds the most current data. There are two ways for a disk to become current: it may find that it is already a member of the current set, or it may find that it has been excluded from the current set. In the latter case it must copy the current state of the data from one of the members of the current set and then join the current set.

The algorithm is correct – the data is guaranteed to be consistent if there is always exactly one current set. Its correctness is established by the following theorem.

Theorem 3.1 *There is at any time exactly one current set.*

Proof. Initially there is only one current set consisting all disks. There are only two ways for the current set to be modified. The first is for a failure to occur and to be noticed by a failure detection mechanism such as a write operation. The second is for a disk to be repaired and go through the recovery procedure.

Case 1. Suppose some disk d which is a member of the current set fails. The algorithm will remove d from the current set. The removal of d cannot introduce a second current set since d would had to have been the only shared element among several intersecting sets. Since all cohort sets in the current set are equal, this is impossible. Disk d does not introduce a new current set since it unchanged by the failure, and lists all other members of the current set of which it was a member.

Case 2. Suppose that some disk d is repaired. The recovery procedure designates d as current if it is a member of the current set, or it copies the state from the current set and then joins it. If d is a member of the current set, then there is no change in the single current set. If d joins the existing current set, it cannot introduce a new current set since joining the existing current set only modifies that set. *Q.E.D.*

3.2 Reading and Writing

Cohort sets are the critical metadata items for mirrored operation and must be correct for the recovery algorithm to operate correctly. As discussed above, it was assumed that cohort sets were completely written to stable storage following the detection of a failure. While extremely rare, it is possible that a second failure could occur while the cohort sets were being written.

In order to mitigate the effects of this unlikely failure scenario, two phases are used to write the cohort sets to stable storage. In the first phase, the so-called tentative cohort sets are written. If this fails, the system can fall back to the original committed cohort sets. In the second phase, the tentative cohort sets are cleared and the committed cohort sets are written. Should this fail, the tentative cohort sets that remain can be used in conjunction with the newly committed cohort sets.

Cohort sets are modified when a write operation occurs following a failure. As long as cohort sets are written every time a failure is detected, the data is assured to be correct.

Frequent writes have the effect of speeding recovery. If writes are too infrequent to provide sufficiently fine grained failure detection, then cohort sets can be modified when read operations occur. If an asynchronous failure notification mechanism is available, then it can be used to modify cohort sets as well.

3.3 Recovery

The recovery from total failure is the most intricate operation in the system. Since failures may occur at any time, cohort sets must be carefully maintained. We write cohort sets using the two-phase write described in the previous section. In this case, the following recovery procedure can be applied.

3.3.1 Recovery Procedure

In order to declare a set of disks current, the cohort sets for each disk are checked for equality and completeness in the following order:

1. All cohort sets are tentative. This will succeed if there was a failure after the tentative cohort sets were written, but before the committed cohort sets were written.

2. There is a mixture of tentative and committed cohort sets that form a current set. This will succeed if there was a failure while the committed cohort sets were being written.

3. There are only the committed cohort sets. This is the most common case and will succeed if the two phase write completed successfully.

There is one further case – when a failure occurs during the initial writing of the tentative cohort sets. In this case, the tentative cohort sets can be safely ignored since the committed cohort sets represent a consistent view of the system.

The system considers each on-line disk in turn as it becomes active. Each disk will have its cohort set compared to the cohort set of all on-line disks. When the system is able to contact all disks in a cohort set, and the cohort sets of each of these disks agree (the equal and complete property), then this disk and all disks in its cohort set can be declared to be current.

Disks which are unable to complete this procedure are out-of-date and must be repaired from one of the current disks.

3.4 Single-copy semantics

There is a possibility that a write operation may fail, or be interrupted, before all of the disks can be written. As a result, only a subset of the disks hold the latest version of the data. Subsequent read operations may get different results, depending on which disk satisfies the request.

Instead, it is desirable to have *single-copy semantics*, where the mirrored storage behaves like a single disk. In order to do this, a *dirty bit*[1] is used. This bit is replicated on each disk and is set before a write operation first occurs to some portion of the disk. This bit is reset (*cleaned*) when the disk is quiescent. To speed recovery, a dirty bit is assigned to each region of the disk that is mirrored. In this way, recovery is more rapid since only a subset of the disk sectors are likely to be accessed before the dirty bits are again cleaned.

When the system recovers, the dirty bit is used to provide single-copy semantics in this way: if the dirty bit is set, then a valid (usually the first) disk is copied to the others that make up the mirrored set. A valid disk is any disk that satisfies the currency requirements of the recovery procedure. Since a write may have failed before it could complete on all disks, a small number of writes may be lost. While it may be desirable to never lose a write, the performance cost is prohibitive. This method provides semantics equivalent to those of a single disk and at a very low cost.

The amount of storage represented by each dirty bit is an engineering decision. Finer granularity yields more rapid recovery, but increases the cost of writing since the dirty bit much be set more often. The costs can be significant, since the dirty bit must be forced to disk before the write can proceed.

[1] Based on research by J. Menon on RAID controllers.

4 Dependability Analysis

Availability is the most common measure of dependability for reparable systems that are expected to remain operational over an extended period. It is traditionally defined as the fraction of time that the system is expected to be available. In the case of mirrored storage, it is the fraction of time that the system will allow read and write operations to occur.

The system model consists of a set of storage devices (disks and the associated controller) with independent failure modes. A non-independent failure, such as the loss of power, can be modeled separately as a simple multiplicative factor.

When a device fails, a repair process is immediately initiated. Should several devices fail, the repair process will be performed in parallel on the devices. Device failures rates are assumed to be exponentially distributed with mean λ, and repair rates are assumed to be exponentially distributed with mean μ. The rate at which all write requests occur, and which serve as the failure detection mechanism, are assumed to be characterized by a Poisson process with mean κ.

The analysis of the mirrored disk system is similar to the analysis of the available copy protocol for distributed data base systems [3]. Here we present a model for two disks, larger systems are similar but have a much larger state space.

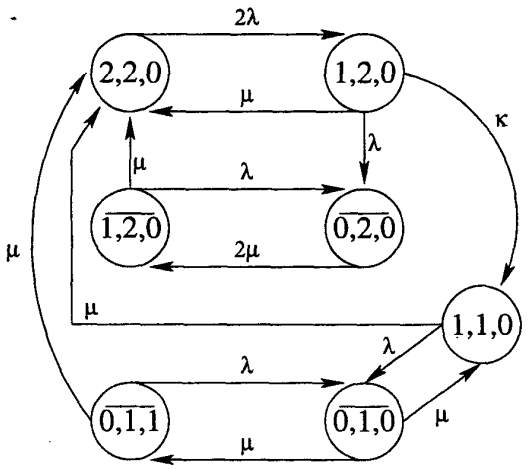

Figure 1: State Transition Diagram

As shown in Figure 1, the states of the Markov model are labeled by the ordered triple $\langle i, j, k \rangle$, where i represents the number of current (or up-to-date) disks, j represents the cardinality of the current cohort set and k represents the number of disks that are out-of-date. When a state is

marked with a bar, for example $\langle 1, 2, \bar{0} \rangle$, this indicates that the system is unavailable.

A system of equations can be derived from this state transition diagram, and solved either algebraically or using numerical methods. If we let $\rho = \lambda/\mu$ and $\phi = \kappa/\mu$, then the equations are significantly simplified.

The availability of the system using algorithm X with k disks is denoted $A_X(k)$. $A_M(k)$ indicates the availability of mirroring with k disks, and $A_V(k)$ indicates the availability using voting (quorum consensus).

The availability of the mirrored disk system with two disks is the sum of the probabilities of being in an available state and is given by the expression:

$$A_M(2) = \frac{\phi\rho^2 + 3\rho^2 + 3\rho\phi + 4\rho + \phi + 1}{(\rho+1)^3(\rho+\phi+1)}.$$

If the writes occur with sufficient frequency that the cohort sets can be assumed to be up-to-date, then the availability is given by the expression:

$$\lim_{\phi\to\infty} A_M(2) = \frac{\rho^2 + 3\rho + 1}{(\rho+1)^3}.$$

It should be noted that for conservative estimates of λ and μ, the difference between these two expressions is less than 0.000002.

The availability of the mirrored disk system with three disks can be derived in a similar manner. In this case, the state diagram has sixteen states. The resulting expression is very large and has been omitted for the sake of brevity. If we again make the assumption of frequent writes, then the availability of a system with three disks is given by the expression:

$$\lim_{\phi\to\infty} A_M(3) = \frac{2\rho^4 + 11\rho^3 + 17\rho^2 + 9\rho + 2}{(\rho+1)^3(2\rho^2 + 3\rho + 2)}.$$

This analysis can be done for any number of disks, although the equations quickly become unmanageable. If the frequent write assumption is made – writes occur much more frequently than failures – then a closed form solution has been derived [3].

It is instructive to compare the availability afforded by our method with an ideal, but unrealizable, system where reads and writes can occur as long as there is at least one disk accessible. This scheme provides no consistency, but its availability is an upper-bound on the performance of method.

The availability of a single disk is given by

$$A = \frac{\mu}{\lambda+\mu} = \frac{1}{1+\rho}.$$

From this, it is easy to see that the probability of finding exactly k disks available is:

$$p_k = \binom{n}{k} A^k (1-A)^{n-k} = \frac{\binom{n}{k}\rho^{n-k}}{(1+\rho)^n}.$$

This means that an upper bound on the availability of a mirrored system with n disks is given by [7]:

$$A(n) \le 1 - p_0 = 1 - \frac{\rho^n}{(1+\rho)^n}.$$

It is also instructive to compare the availability of our method with that of quorum consensus. In quorum consensus, the current disks of the data must be found among a majority of accessible disks.

For an odd number of disks, the availability of quorum consensus was derived by Pâris [6] (it has been shown that an even number of disks k yields the same availability as an odd number of disks $k-1$):

$$A_V(n) = \sum_{i=\lceil \frac{n}{2} \rceil}^{n} \frac{\binom{n}{n-i}\rho^{n-i}}{(1+\rho)^n}.$$

In the following graphs, we will compare our algorithm with a system that does not enforce consistency and with quorum consensus. The ordinate axis is ρ, which ranges from 0.0 to 0.01. These are conservative estimates based on a maximum MTTF of approximately five years (a common warranty period) and a MTTR of two weeks. The coordinate axis indicates the availability provides by each scheme.

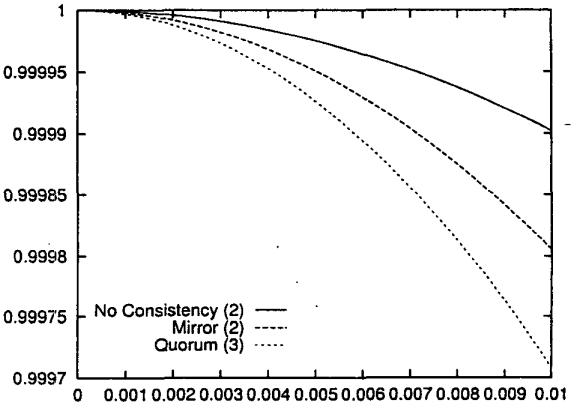

Figure 2: Compared Availability, $0 \le \rho \le 0.01$

We now compare our algorithm using two disks, with both a system that maintains no consistency, and one that uses quorum consensus with three disks to determine the current disk. We see from Figure 2, that while as is to be

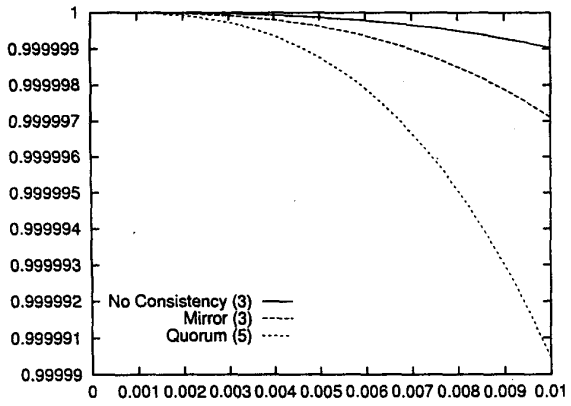

Figure 3: Compared Availability, $0 \leq \rho \leq 0.01$

expected, our algorithm provides lower availability than a system with no consistency, it provides better availability than quorum consensus with an added disk.

Next, in Figure 3, we compare our algorithm using three disks with a system that maintains no consistency and a system using quorum consensus with five disks. Our algorithm again provides availability that is superior to quorum consensus with nearly twice as many disks.

We believe that this analysis shows that our algorithm provides superior availability at a reduced cost in terms of both metadata and complexity.

5 Discussion

While the reliability of disks has increased to the point that a single disk can be expected not to fail for several years, the integrity of data is still an important issue. In the case of small systems, experience indicates that back-ups are seldom performed. Losing data even once every few years is a potential disaster. In the case of large installations, even though each disk is expected to last several years, the expected time to have one of the disks fail is measured in weeks and in some cases days.

By using mirroring, the likelihood of data loss is greatly reduced. In the case of large installations, it will essentially eliminate data loss provided that failed disks are replaced in a timely fashion. Given the cost trends of magnetic storage, mirroring may remove the need to perform routine back-ups in most small installations.

6 Conclusions

We have developed a simple technique for efficiently managing mirrored storage. Our technique requires a only a small amount of metadata, and does not require logging or quorum collection. It allows automatic recovery from failure with an arbitrary number of disks, and its simplicity and robustness makes it viable even in the consumer market.

A significant advantage of this scheme is that its cost is low, both in terms of storage required and in terms of overhead. For n disks of the data, it requires n bits per disk to represent the set, and $\log n$ bits to store the ordinal number of each disk. While these bits are hardened metadata, they need only be modified when a failure is detected (typically, when a write fails to complete).

We have shown, using standard Markov analysis, that our algorithm provides excellent availability. In particular, it is very close to the upper-bound where no consistency is enforced and is clearly superior to the availability provided by quorum consensus. Its use can effectively eliminate the risk of data loss.

References

[1] P. A. Bernstein, V. Hadzilacos, and N. Goodman, *Concurrency control and recovery in database systems*. Reading, Massachusetts: Addison-Wesley, 1987.

[2] J. Gray and A. Reuter, *Transaction processing: concepts and techniques*. Morgan Kaufmann, 1993.

[3] D. D. E. Long, *The Management of Replication in a Distributed System*. Ph.D. dissertation, University of California at San Diego, 1988.

[4] D. D. E. Long and J.-F. Pâris, "A leaner, more efficient, available copy protocol," in *Proceedings of the Symposium on Parallel and Distributed Processing*, (New Orleans), IEEE, October 1996.

[5] C. Mohan, D. Haderle, B. Lindsay, H. Pirahesh, and P. Schwarz, "ARIES: A transaction recovery method support fine-granularity locking and partial rollbacks using write-ahead logging," *ACM Transaction on Database Systems*, vol. 17, March 1992.

[6] J.-F. Pâris, "Voting with a variable number of copies," in *Proceedings Sixteenth Fault-Tolerant Computing Symposium*, (Vienna), pp. 50–55, IEEE, 1986.

[7] J.-F. Pâris and D. D. E. Long, "The performance of available copy protocols for the management of replicated data," *Performance Evaluation*, vol. 11, pp. 9–30, 1990.

[8] D. Skeen, "Determining the last process to fail," *ACM Transaction on Computer Systems*, vol. 3, pp. 15–30, 1985.

[9] K. S. Trivedi, *Probability & Statistics with Reliability, Queuing and Computer Science Applications*. Englewood Cliffs, New Jersey: Prentice-Hall, 1982.

[10] J. Wilkes, R. Golding, C. Staelin, and T. Sullivan, "The HP AutoRAID hierarchical storage system," in *Proceedings of the Fifteenth Symposium on Operating Systems Principles*, (Copper Mountain), ACM, 1995.