RAID0.5: Active Data Replication for Low Cost Disk Array Data Protection

John A. Chandy
Department of Electrical and Computer Engineering
University of Connecticut
Storrs, CT 06269-2157
john.chandy@uconn.edu

***Abstract*— *RAID has long been established as an effective way to provide highly reliable as well as high-performance disk subsystems. However, reliability in RAID systems comes at the cost of extra disks. In this paper, we describe a mechanism that we have termed RAID0.5 that enables striped disks with very high data reliability but low disk cost. We take advantage of the fact that most disk systems use backup systems for disaster recovery. With the use of these backup systems, the disk system needs to only replicate data since the last backup, thus drastically reducing the storage space requirement. Though RAID0.5 has the same data loss characteristics of traditional mirroring, the lower storage space comes at the cost of lower availability. Thus, RAID0.5 is a tradeoff between lower disk cost and lower availability while still preserving very high data reliability. We present analytical reliability models and experimental results that demonstrate the enhanced reliability and performance of the proposed RAID0.5 system.*

I. Introduction

Disk arrays have long been used to improve the performance of storage systems [1], [2]. The parallelism inherent in multi-disk systems can significantly boost both the throughput and response times as compared to a single disk system. However, the increased performance comes at the cost of lower reliability. As a result, disk arrays need some form of redundancy to improve reliability. The most common and cost effective solution to improve the reliability of disk systems is the use of Redundant Array of Inexpensive (or Independent) Disks (RAID) [3]. The various levels of RAID specify different methods of redundancy to provide reliability. The most commonly used forms of RAID are RAID1 or mirroring, which entails replication of the data on multiple disks, and RAID5 or parity striping, which involves spreading data along with parity across multiple disks. Choosing which RAID level to use is typically determined by cost and application requirements. At the disk array level, the redundancy choice is usually RAID5 as it provides excellent availability, moderate storage overhead, and adequate performance.

All RAID levels require extra disks to provide redundancy. In the case of RAID5, the redundancy overhead is $1/D$ where D is the number of data disks in a redundancy group. With RAID1, the overhead is 100%. This overhead makes RAID cost prohibitive in many environments, particularly single user desktop computers and laptops. RAID5 requires the installation of either SCSI controllers or multiple IDE controllers and enclosures to contain the multiple disks. In a two-disk scenario that is feasible for most single PCs, the 100% overhead of RAID1 mirroring becomes costly.

It is in this light that we present RAID0.5, a midway point between RAID0 and RAID1 in that it provides equivalent data loss guarantees to RAID1 with just slightly more overhead than RAID0. However, as we will show, the tradeoff between RAID0.5 and RAID1 is the lower availability of a RAID0.5 system. The remainder of the paper describes the overall architecture and analysis of RAID0.5, experimental results future directions, and related work.

II. RAID0.5

The goal of the RAID0.5 architecture is to provide a disk subsystem that can achieve high reliability with low cost. Our particular aim is to make the method feasible for systems with relatively few disks. A typical RAID5 system will have 5-7 disks in the array, leading to a redundancy overhead of 14-20%. As you use fewer disks and approach RAID1 mirroring, the overhead quickly approaches 100%. With a RAID0.5 system, we would like to maintain a redundancy overhead of less than 20% for even 2 disk systems. In a two-disk system, in order to prevent data loss, it would seem that we would need to replicate all the data on a disk thus leading to 100% overhead. Compressing the replicated data may allow for smaller overheads, but this is not guaranteed and the compression process can be compute intensive.

The key to achieving low redundancy overheads is to replicate only a portion of the data on a disk. If we assume that the system is being periodically backed up, then we need only to replicate data that has been changed since the last backup. We define this data as the *active* data. In such a scenario, if a disk fails, active data can be recovered from the remaining disk(s) and inactive data can be recovered from the backup media.

The backup window determines how much data needs to be replicated. For example, a weekly backup will create a larger active data set size than a daily backup. HP's study of working set sizes showed that, on average, only 2% of the storage space is written to over a 24 hour period [4]. The largest observed 24-hour write set size was just over 10%. Thus, assuming daily backups, we need to only replicate 10% of the data in the array. Assuming similar access patterns over a week, the weekly working set size should not change much, so a 15% replication overhead may be sufficient for weekly backups.

The data distribution of a RAID0.5 system is shown for a 4 disk system in Figure 1. Each row in the disk represents one of n stripes as in a RAID0 disk array, and each cell represents a data block or chunk of a stripe on a disk. On each disk, we dedicate a region for replication, the size of which is determined by the desired redundancy overhead which is determined by the backup frequency as described above. In Figure 1a, the data distribution is shown for a system where the active data set is empty, i.e. immediately after a backup. After writes to *D10*, *D21*, and *D03*, we can see in Figure 1b that the writes have been replicated to replication regions on adjacent disks. By replicating on the next disk (mod number of disks), we can ensure that the active data is preserved on at least one disk in the event of a single disk failure. If a disk replication region fills up, we can replicate the data to another disk or grow the replication region. If this is not possible, the file system or disk driver will respond as if the disk is full.

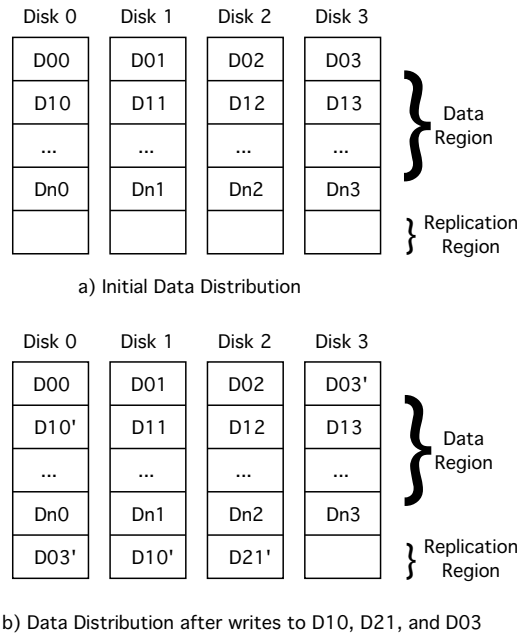

Fig. 1. RAID0.5 data distribution.

III. Reliability Analysis

When a disk fails, any requests to active data can be delivered from the replication region. However, requests to inactive data can not be serviced because the data is not available online and must be retrieved from backup media. Thus, in a RAID0.5 system, as soon as failure is detected, the system must block all future accesses to disk until the failed disk can be reconstructed from backup. To prevent the chance of a second failure, the system should power down all disks in the system. Note that this implies that a RAID0.5 system is not available under failure even though there has been no data loss. This behavior is different from a RAID1 or RAID5 system where data is still available even under a single disk failure. Thus, for a RAID0.5 system there is a distinction between availability and data loss protection, whereas no such distinction exists for other RAID levels. A RAID0.5 system trades off system availability for low redundancy costs. Therefore, using RAID0.5 is not appropriate for mission critical applications that require 24/7 operation, but it is appropriate for desktop applications that may not require nonstop operation but do require absolute data protection. Because of this high-reliability low-availability characteristic, we placed RAID0.5 between the low-reliability, low-availability RAID0 and high-reliability, high-availability RAID1.

In light of this distinction between system availability and data loss protection, when doing reliability analysis we are more interested in the mean time to data loss (MTTDL) rather than the more conventional mean time to failure (MTTF). The analysis is similar but MTTDL implies nothing about system availability as MTTF does.

A. RAID0.5 with replication region

We can develop a model for the MTTDL and availability of a RAID0.5 system using a analysis methodology similar to that outlined in [3]. Each disk is assumed to have independent and exponential failure rates. We assume d disks in the array and a replication ratio of r, i.e. the ratio of the replication region size to data size. Each disk is assumed to have a mean time to failure of $MTTF_{Disk}$. The mean time to failure of one disk in the array is $\frac{MTTF_{Disk}}{d}$ and data loss occurs when a second disk fails while the first failed disk is being repaired.

$$MTTDL_{RAID0.5} = \frac{MTTF_{Disk}}{d} * \frac{1}{Pr[\text{data loss failure during repair}]} \quad (1)$$

If we define the mean time to repair the disk as

$MTTR_{Disk}$, then assuming exponential failure rates, the probability of a failure during the repair time is:

$$1 - (e^{-MTTR_{Disk}/MTTF_{Disk}})^{(d-1)} \quad (2)$$

Since $MTTR_{Disk}$ is almost always much less than $MTTF_{Disk}$, the above expression can be approximated as:

$$\frac{MTTR_{Disk}(d-1)}{MTTF_{Disk}} \quad (3)$$

Substituting into Equation 1, we get

$$MTTDL_{RAID0.5} = \frac{(MTTF_{Disk})^2}{d(d-1)MTTR_{Disk}} \quad (4)$$

The MTTDL equation for a RAID0.5 system is equivalent to the MTTDL for a RAID5 system except for the derivation of the disk MTTR. In a RAID1 or RAID5 system, the MTTR is comprised of the time to replace the failed disk as well as the time to reconstruct the data onto the new disk. In systems with hot spares the time to replace the failed disk can be zero, but for the small scale systems that we are considering, hot sparing is not likely to be present. In a RAID0.5 system, since the system is powered off after a failure, there is almost no chance of a second failure during the time it takes to replace the failed disk[1]. Therefore, when determining the MTTR of the disk, we need to only include the time to reconstruct the data to the failed disk. The reconstruction time of a RAID0.5 system is determined primarily by the speed of the backup media - tape can be slow but a D2D (disk to disk) backup system can be relatively fast. Assuming the backup media is fast enough to keep up with the reconstruction, reconstruction on a RAID0.5 will be much faster than RAID5 reconstruction because there is no need to read all disks in the parity group to regenerate the lost data. RAID0.5 reconstruction will be slightly slower than RAID1 reconstruction because of the need to read the replication region and copy over the active data changes since the last backup. Note that the replication region on the new disk will be invalid, so the data must be copied over from the original disk.

B. RAID0.5 with replication disk

Though the rationale for RAID0.5 was established for disk arrays with few disks, RAID0.5 can be advantageous in larger arrays and can actually show equivalent MTTDLs to mirroring if we modify the data distribution slightly. Instead of putting the replication region on each disk, we dedicate a separate disk to serve as the replication disk as shown in Figure 2. With $d = 5$ and $r = 0.2$, the two data distributions from Figures 1 and 2 are equivalent in terms of overhead. We call this modified data distribution RAID0.5 with disk-based replication. Note that the advantage of this distribution is that the array can tolerate more combinations of multiple failures. Figure 3 illustrates how the system can tolerate more than one failure and still recover the most recent data. Even though two disks have failed, the active data blocks, *D11'* and *D32'*, are still available from the replication disk. In fact if all the disks except for the replication disk fail, disk-based replication allows for the recovery of all active data.

[1]The probability of failure while the disks are turned off is not absolutely zero, but it is much lower than the chance of failure while the disks are powered up, so for the most part we can ignore it.

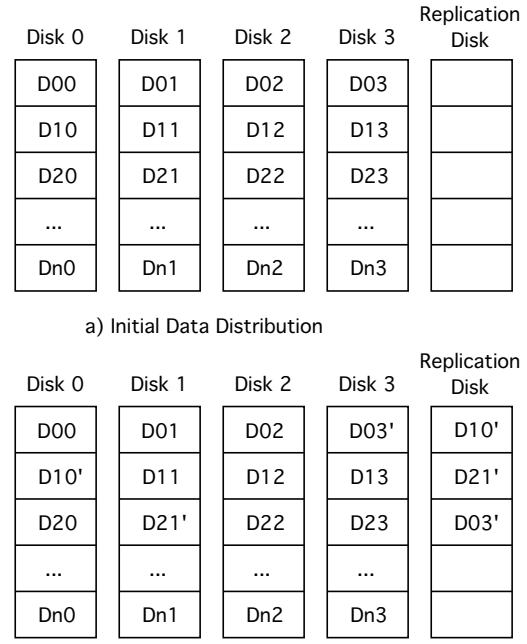

Fig. 2. RAID0.5 disk-based replication.

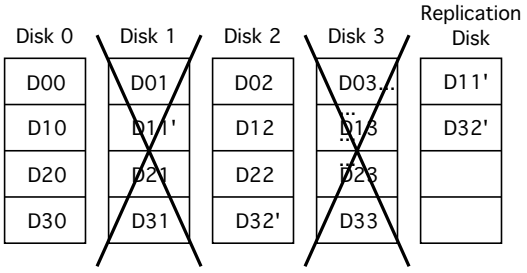

Fig. 3. RAID0.5 disk-based replication failure.

With disk-based replication, the overall MTTDL is as described in Equation 1, but the derivation of the probability of data loss during failure is different. Data loss during the repair time of the failed disk can happen in two cases: 1) the first failed disk was the replication disk and any other disk fails, and 2) the first failed disk was not the replication disk and the replication disk fails. Thus, the probability of data loss causing failure during the repair time is as follows:

$$\begin{aligned} &Pr[\text{data loss failure during repair}] = \\ &\quad Pr[\text{first failed disk was the replication disk}]p_f + \\ &(1 - Pr[\text{first failed disk was the replication disk}])p_{rf} \end{aligned} \tag{5}$$

where p_f is the probability that any one of the remaining $d-1$ disks fails during the repair time and p_{rf} is the probability that the replication disk fails during the repair time. The derivation of p_f is the same as the derivation of Equation 3. Thus,

$$p_f = \frac{MTTR_{Disk}(d-1)}{MTTF_{Disk}} \tag{6}$$

Similarly, p_{rf} is equal to $\frac{MTTR_{Disk}}{MTTF_{Disk}}$. Substituting into Equation 5, we arrive at:

$$\begin{aligned} &Pr[\text{data loss failure during repair}] \\ &= \frac{1}{d}\frac{MTTR_{Disk}(d-1)}{MTTF_{Disk}} + (1-\frac{1}{d})\frac{MTTR_{Disk}}{MTTF_{Disk}} \\ &= (\frac{2}{d})\frac{MTTR_{Disk}(d-1)}{MTTF_{Disk}} \end{aligned} \tag{7}$$

Substituting into Equation 1, we arrive at:

$$MTTDL_{RAID0.5} = \frac{(MTTF_{Disk})^2}{2(d-1)MTTR_{Disk}} \tag{8}$$

If we define D as the total number of data disks in the system, i.e. $d-1$, we can rewrite $MTTDL$ as $\frac{(MTTF_{Disk})^2}{2D\ MTTR_{Disk}}$. The redundancy overhead to support replication is one disk. By comparison, a mirrored system with D data disks has a similar MTTDL of $\frac{MTTF^2_{Disk}}{2D\ MTTR_{Disk}}$ but with an overhead of D disks, and a RAID5 system has an MTTDL of $\frac{MTTF^2_{Disk}}{D(D+1)MTTR_{Disk}}$ and an overhead of 1 disk. The RAID0.5 system actually has similar MTTDL times to a mirrored system with significantly less redundancy overhead.

The drawback to disk based replication region is that the disk with the replication region can become a hot spot. Rotating the replication region as is done with rotated parity in RAID5 reduces the system in reliability terms to the replication region method described earlier. Thus, the choice of disk-based replication should only used in environments where there is not very high loads such that a single disk could become a bottleneck.

C. Availability analysis

We have presented RAID0.5 as a compromise choosing high reliability at the cost of lower availability. In this section, we will present a model for comparing the availability of the various systems. Availability is defined as the percentage of time that the system is not available. In reliability analysis terms, this is simply the ratio of the MTTF of the system to the sum of the MTTF and MTTR of the system. For a RAID or mirrored system the MTTF of the system is equivalent to the MTTDL that was calculated in the previous section. The MTTR of the system is equivalent to the MTTR of a disk since the system will be repaired when the disk is repaired. In practice, though, this is not true since when there is a system failure, the time to recover will probably be longer because of the time required to reinstall software, recover data from backups, etc. For the purposes of this discussion, however, we will assume, in the absence of hot spares, that the majority of the repair time is the time to install the new disk. The availability of the RAID5 and mirrored systems can be expressed as follows:

$$\begin{aligned} A_{RAID5} &= \frac{MTTDL_{RAID5}}{MTTDL_{RAID5} + MTTR_{Disk}} \\ &= \frac{MTTF^2_{Disk}}{MTTF^2_{Disk} + D(D+1)MTTR^2_{Disk}} \end{aligned} \tag{9}$$

$$\begin{aligned} A_{Mirror} &= \frac{MTTDL_{Mirror}}{MTTDL_{Mirror} + MTTR_{Disk}} \\ &= \frac{MTTF^2_{Disk}}{MTTF^2_{Disk} + 2D\ MTTR^2_{Disk}} \end{aligned} \tag{10}$$

For a RAID0.5 system, the system is unavailable whenever a disk goes down even though data may not have been lost. Thus, we can derive the availability of a RAID0.5 system as follows:

$$\begin{aligned} &A_{RAID0.5} = \\ &\frac{MTTF_{Disk}}{MTTF_{Disk} + \text{Time to replace and restore disk}} \end{aligned} \tag{11}$$

Comparing Eqs. 9 and 10 with Eq. 11, we see that the availability of a RAID0.5 system is reduced by a factor equal to $MTTF_{Disk}$. In fact, RAID0.5 availability

Configuration	MTTDL (years)	Availability		Accessible storage
		A	Nines ($-log_{10}(1-A)$)	
RAID0	1.9	0.9989212	2.97	6 disks
RAID5	1584	0.9999983	5.76	5 disks
Mirror	7922	0.9999996	6.46	3 disks
RAID0.5 (r = 0.2)	3169	0.9989212	2.97	4.8 disks
RAID0.5 (disk-based replication)	9506	0.9989212	2.97	5 disks

TABLE I
MTTDL AND OVERHEAD FOR A 6 DISK ARRAY. ($MTTF_{Disk}$ = 100000 HOURS AND $MTTR_{Disk}$ = 24 HOURS)

Configuration	MTTDL (years)	Availability in nines	Accessible storage
RAID0	5.7	3.32	2 disks
RAID5	23766	6.94	1 disk
Mirror	23766	6.94	1 disk
RAID0.5 (r = 0.2)	47532	3.32	1.6 disks
RAID0.5 (disk-based replication)	47532	3.32	1 disk

TABLE II
MTTDL AND OVERHEAD FOR A 2 DISK ARRAY. ($MTTF_{Disk}$ = 100000 HOURS AND $MTTR_{Disk}$ = 24 HOURS)

is no better than a non-redundant disk system. This is borne out in Tables I and II which show MTTDL, availability, and accessible storage for six and two disk arrays. We assume that the time to reconstruct the disk is 6 hours for mirrored and RAID5 systems. For RAID0.5, the reconstruction time is assumed to 12 hours because of slow backup tape and the need to reinitialize the replication region on the new disk. Since we are assuming non mission-critical small scale systems, we do not presume any hot sparing is present, and thus the time to replace the disk is assumed to be 18 hours. This gives a $MTTR_{Disk}$ of 24 hours for the RAID5 and mirrored systems, and 12 hours for RAID0.5.

The tables show that RAID0.5 can offer equivalent or better MTTDL to RAID1 with significantly less storage overhead. For large arrays, the disk-based replication approach is best and for small arrays, the replication region approach is best. The cost, however, is the availability of the RAID0.5 system. While the availability of RAID0.5 is good (99.9%), this is due entirely to the reliability of the disk and not due to the structure of the array. Enterprise systems typically demand six nines of reliability which can be delivered with the RAID5 and mirrored systems, but is not possible with RAID0.5. Thus, the use of RAID0.5 becomes a choice between high availability with low storage overhead and high availability with high storage overhead. As mentioned before, the ideal environment for such a system is for desktop or single-user systems that contain high value data but do not require 24/7 availability.

IV. EXPERIMENTAL RESULTS

We implemented the RAID0.5 as part of the md (multidisk) subsystem in Linux (2.6.11). The test platform is a 500MHz Pentium III with 2 IDE disks as masters on separate IDE channels. Each disk is managed with the LVM2 logical volume manager and the RAID arrays are formed from two 160MB logical volumes - one each from disk. The array was reformatted with the ext3 filesystem after each run. We used the IOZone [5] filesystem benchmark to measure I/O performance. With the *-a* option, IOZone uses a broad mix of access patterns to read/write a test file. As the results in Figures 4 and 5 show, RAID0.5 performs favorably to RAID1, as would be expected because for each write they each require two writes to separate disks. RAID0.5 does exhibit slightly slightly better performance on writes in part due to the write logging that eliminates the access time for writes because the head does not have to move.

V. FUTURE EXTENSIONS

A. Snapshots

Because RAID0.5 replicates active data, it makes it easy to provide storage snapshotting features. If we were to maintain the replication log as a strict log rather than update blocks in place, we would have a log of all updates since the last backup. Adding a timestamp to the BDS allows us to retrieve a block from any period any in time. In essence, rolling back the log to any point in time means that we can provide a continuous snapshotting capability. This does not require any separate copy-on-write capabilities as is common with snapshot systems, since we always maintain the older versions of blocks.

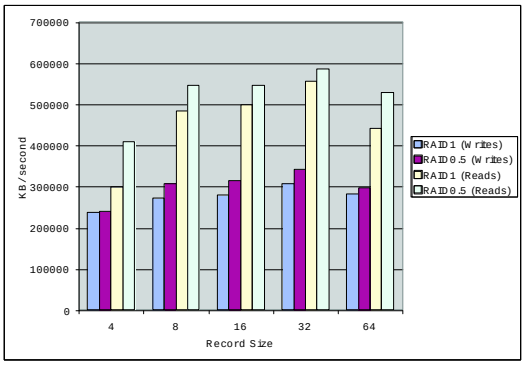

Fig. 4. RAID1 and RAID0.5 results for 64K file reads and writes.

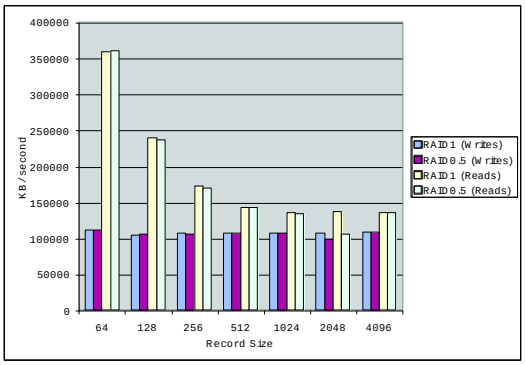

Fig. 5. RAID1 and RAID0.5 results for 32M file reads and writes.

B. Backups

RAID0.5 is inherently dependent on a judicious backup procedure at somewhat frequent intervals. However, in desktop environments where the cost of backup can dwarf the cost of a RAID system, backups may be difficult. In an office environment, though, automated backup systems can run in the background and perform backup to an offline system. Since the restore process is block based, the backup process must also be block based which may not be compatible with existing backup mechanisms. With the use of a snapshotting feature as described above, however, it is not necessary to quiesce the system since arbitrary checkpoints can easily be chosen within the snapshot. Moreover, the process of determining what blocks need to be backed up in an incremental backup is trivial since the replication region contains all data since the last backup.

C. File System Implementation

The RAID0.5 system that we have described is block based since it is at the device driver level. However, it is possible to do the same at the file system level. The replication copy information can be placed in the inode, and recovery procedures need not depend on a replication log and can instead use the filesystem's standard fsck procedure. However, to ensure reliability, the file system needs to be aware of physical disks in order to place replication blocks on a different physical disk than the actual data block. In today's layered I/O systems, physical disk information is often not visible to the file system which tends to see the block device as simply a continuous segment of blocks. This disconnect is further amplified when using volume managers that completely obscure the physical layout of the storage.

Another approach to a file system implementation of RAID0.5 is to do replication on a file by file basis. In such a system, we can replicate file writes to a separate replication volume by intercepting file `write()` calls. The advantage of such an approach is that replication decisions can be made on a file-by-file basis rather than for an entire volume. This allows the user to limit the space required for replication by selecting only important files for replication. Thus, files that are easily replaced from installation CDs such as applications and operating system files can be marked as not needing replication. File based RAID0.5 is also more compatible with existing backup systems.

D. RAID5.5

A variation of RAID0.5 is shown in Figure 2 where we use a replication disk as as a cache for writes in a RAID5 system. In such a scenario, instead of writing to the data disk and the parity disk as in a normal RAID5 system, we write to the data disk and the replication disk. This allows writes to be quickly mirrored and thus avoid the parity generation problem in RAID5. The parity can be generated later when the system is at low load. Using this replication disk can improve RAID5 reliability as well as improve RAID5 performance.

VI. Related Work

The RAID0.5 system is probably most closely related to the various works in disk logging. Of particular interest are virtual logs [6], parity logging [7], data logging [8], and hot mirroring [9], [10].

In general, logging optimizes writes by writing data to empty locations near the current disk head location. The assumption is that if the disk is doing only writes, the disk head needs to seek only slightly, if at all, to write the new data, thus eliminating the seek time. The idea of write logging is not new and previous work has shown the effectiveness of the method in certain scenarios [6], [11], [12], [13], [14], [15]. RAID0.5 borrows from the logging idea in creating a replication log. However, since there is normal disk activity to the data region of the disk,

we can not depend on the disk head remaining in the log, so no advantages in terms of disk access are seen. However, the use of a log does simplify block allocation as described earlier.

The parity logging technique eliminates the need for parity disk accesses by caching the partial parity formed from the old and new data in non-volatile memory at the controller. The partial parity can then be periodically flushed to a log disk, which can then be cleaned out at a later time to generate the actual parity disk data. This process reduces the number of disk accesses from 4 to 2 and clearly, this reduction in accesses will greatly speed up the performance of writes in a RAID system. The process is similar to the RAID5.5 optimization discussed in Section V-D except that RAID5.5 uses a disk as a cache instead of non-volatile memory. Similar approaches to RAID5 caching have also been presented in [13], [14], [16], [17]. Data logging is similar to RAID5.5 except that it performs an old data read and stores that to the data log as well. This requires an extra disk access and the maintenance of log maps requires non-volatile memory at the clients as well.

Hot mirroring [9] and AutoRAID [10] are similar techniques that attempt to move actively used data to mirrored regions of the array and less frequently used data to parity logged regions (hot mirroring) or parity striped regions (AutoRAID). These systems require a background process that evaluates the "hotness" of data and then moves them to or from mirrored regions as required. If a data block is in the parity striped region, it will remain there until a background process has tagged it as hot even if it is experiencing high activity. The process is similar to RAID0.5 in that active data is mirrored. RAID0.5 systems, however, dynamically adjust to the activity of the data since the latest updated data is always pushed to the replication region or disk.

VII. Conclusions

In this paper, we have described a variation of disk array striping called RAID0.5. The proposed architecture has very high data loss protection characteristics compared to RAID0 with very little overhead. By taking advantage of the fact that offline backup will protect most data, RAID0.5 saves replication space by only duplicating data that has changed since the last backup. Therefore, RAID0.5 allows smaller disk arrays to have data protection without resorting to mirroring. The major drawback to RAID0.5 is that the storage system is not available after a failure and thus is not suitable in high availability 24/7 environments. However, RAID0.5 does provide a way to achieve low cost data protection in small desktop disk arrays. RAID0.5 can also be extended to provide snapshotting and RAID5 parity caching.

References

[1] M. Y. Kim, "Synchronized disk interleaving," *IEEE Trans. Computers*, vol. C-35, pp. 978–988, Nov. 1986.

[2] K. Salem and H. Garcia-Molina, "Disk striping," in *International Conference on Data Engineering*, pp. 336–342, 1986.

[3] D. A. Patterson, G. A. Gibson, and R. H. Katz, "A case for redundant arrays of inexpensive disks (RAID)," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 109–116, June 1988.

[4] C. Ruemmler and J. Wilkes, "A trace-driven analysis of working set sizes," Technical Report HPL-OSR-93-23, Hewlett-Packard, Palo Alto, CA, Apr. 1993.

[5] IOZone. http://www.iozone.org, Feb. 2005. v. 3.242.

[6] R. Y. Wang, T. E. Anderson, and D. A. Patterson, "Virtual log based file systems for a programmable disk," in *Proceedings of Symposium on Operating Systems Design and Implementation*, pp. 29–43, Feb. 1999.

[7] D. Stodolsky, G. Gibson, and M. Holland, "Parity logging: Overcoming the small write problem in redundant disk arrays," in *Proceedings of the International Symposium on Computer Architecture*, 1993.

[8] E. Gabber and H. F. Korth, "Data logging: A method for efficient data updates in constantly active RAIDs," in *Proceedings of the International Conference on Data Engineering*, pp. 144–153, 1998.

[9] K. Mogi and M. Kitsuregawa, "Hot mirroring: A method of hiding parity update penalty and degradation during rebuilds for RAID5," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 183–194, June 1996.

[10] J. Wilkes, R. Golding, C. Staelin, and T. Sullivan, "The HP AutoRAID hierarchical storage system," *ACM Transactions on Computer Systems*, vol. 14, pp. 108–136, Feb. 1996.

[11] R. M. English and A. A. Stepanov, "Loge: A self-organizing storage device," in *Proceedings of the Winter USENIX Symposium*, pp. 237–252, Jan. 1992.

[12] C. Chao, R. English, D. Jacobson, A. Stepanov, and J. Wilkes, "Mime: A high performance parallel storage device with strong recovery guarantees," Technical Report HPL-CSP-92-9 rev 1, Hewlett-Packard, Palo Alto, CA, Mar. 1992.

[13] T.-C. Chiueh, "Trail: A track-based logging disk architecture for zero-overhead writes," in *Proceedings of International Conference on Computer Design*, pp. 339–343, Oct. 1993.

[14] T.-C. Chiueh and L. Huang, "Track-based disk logging," in *Proceedings of International Conference on Dependable Systems and Networks*, pp. 429–438, June 2002.

[15] J. Menon, J. Roche, and J. Kasson, "Floating parity and data disk arrays," *Journal for Parallel and Distributed Computing*, vol. 17, pp. 129–139, Jan. 1993.

[16] M. Zhang, X. He, and Q. Yang, "Implementation and performance evaluation of RAPID-Cache under Linux," in *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications*, June 2002.

[17] Y. Hu and Q. Yang, "DCD - Disk caching disk: A new approach for boosting I/O performance," in *Proceedings of the International Symposium on Computer Architecture*, 1995.