The Case for Massive Arrays of Idle Disks (MAID)

Dennis Colarelli, Dirk Grunwald and Michael Neufeld
Dept. of Computer Science
Univ. of Colorado, Boulder

January 7, 2002

Abstract

The declining costs of commodity disk drives is rapidly changing the economics of deploying large amounts of on-line storage. Conventional mass storage systems typically use high performance RAID clusters as a disk cache, often with a file system interface. The disk cache is backed by tape libraries which serve as the final repository for data. In mass storage systems where performance is an issue tape may serve only as a deep archive for disaster recovery purposes. In this case all data is stored on the disk farm. If a high availability system is required, the data is often duplicated on a separate system, with a fail-over mechanism controlling access.

This work explores an alternative design using *massive arrays of idle disks*, or MAID. We argue that this storage organization provides storage densities matching or exceeding those of tape libraries with performance similar to disk arrays. Moreover, we show that through a combination of effective power management of individual drives and the use of cache or migration, this performance can be achieved using a very small power envelope.

We examine the issues critical to the performance, energy consumption and practicality of several classes of MAID systems. The potential of MAID to save energy costs with a relatively small performance penalty is demonstrated in a comparison with a conventional RAID 0 storage array.

1 Introduction

Robotic tape systems are designed to reduce the wait time for tape loading, and to increase storage density so as to provide large amounts of storage for a given footprint. Historically, tape libraries are preferred over disk arrays for large (100+ TB) mass storage environments, in large part due to the cost differential between tape and disk. This gap has been closing and is expected to continue to decrease. Additionally, there is the economics of powering and cooling large disk arrays. Files on tape not being accessed consume no power and generate no heat. Files on disk not being accessed do both.

Large tape libraries can accommodate many tape drives; for example, the the StorageTek 9310 tape libraries can support up to 80 T9940 tape drives [1]. Each cartridge for the T9940 drives can record 60GB of uncompressed data, and each 9310 library can support up to 6000 tapes, providing a total of 360TB of storage. The T9940 tape reader takes 18 seconds to load a tape, 90 seconds to rewind the tape and has an average search time of 41 seconds. Each tape can be subjected to a minimum of 10,000 loads/unloads before the media begins to fail. Migrating from one generation of media to another is problematic simply because of the volume of media and the (relatively) limited number of drives.

Consider the economics of using existing RAID arrays rather than tape libraries for such storage needs. Current disk drive capacities are approximately the same as tape; for point of comparison, we'll assume that 60GB drives are used. A single StorageTek 9310 tape library consumes 1.1Kw/h of electricity. To store 1,000TB of information one would require three 9310 libraries (3.3Kw/h). The T9940 tape reader consume 85 watts of power. If we assume half of the full compliment of readers is used (120), a further 11.5Kw/h of electricity is consumed, assuming the drives are constantly in use.

A consumer grade 60GB drive consumes about 8 watts in steady-state. Ignoring the cost of control logic, networking and the like, it would take 144Kw/h to power such a disk array.

Electric utility rates have held fairly constant over the last 10 years at 7.25 cents/Kw/h for commercial customers [2]. Assuming a 24x7 data center operation, it would cost $9,400 to power the tape library system *vs.* $91,500 to power the the disks in the disk array. This estimate discounts the additional electronics needed to actually build a disk array, and is thus an underestimate. Furthermore, additional power would be needed for cooling since the disk array dissipates more heat than the tape libraries.

Our hypothetical tape library would have an aggregate bandwidth of 1200 MB/s, while the disk array could provide a peak bandwidth of 2,880,000 MB/s. However, existing RAID systems provide more capability than needed by such large storage environments. Analysis of tape libraries at supercomputer centers has shown that 50% of the data is written and never accessed [3] and a further 25% of the data is accessed once. Also, tape libraries store data on a single tape with the limited reliability that implies. Such systems have little need of either the high performance or increased reliability of conventional RAID systems. Contrast this with [4] which allow services to "bid" for resources as a function of delivered performance.

We propose to build large storage arrays using *massive arrays of idle disks*, or MAIDs. The design goals for our system are: reducing the energy consumed by a large storage array while maintaining acceptable performance, increasing storage density and maintaining performance similar to conventional disk arrays or tape libraries.

In this paper, we use trace-driven simulation to compare the performance of a simple MAID cluster to a fully active drive array. Our simulator combines both performance and power estimates using a disk power model derived from measurements of sample drives. Our analysis demonstrates that MAID offers performance comparable to a constantly-on drive array for workloads representative of archival

storage systems.

2 The Design of MAID-Simple

The first decision is whether we should use data migration or duplication (caching). If migration is used, the intent would be to move storage to a cluster of "more active" drives, or to distribute the data based on the likelyhood of access; that data would not be duplicated on the remaining drives. Alternatively, we can dedicate a small number of drives as "cache drives" that would cache read data and act as a write-log for write data. Migration provides more usable storage, since no duplication is necessary, and is appropriate when distributing storage across a small number of drives. It may also provide better performance because varying usage patterns of the data will automatically aggregate the information on a few drives.

However, migration requires a map or directory mechanism that maps the storage across *all* drives. By comparison, caching requires maps or directories proportional to the size of the cache disks. If a MAID system is to hold 6,000 drives and total 1,000GB of storage, it is difficult to see how to build an efficient map or directory for such all drives in such a large system; even maintaining a map for the 600 drives that make up the cache may be difficult.

Access patterns will also govern the caching policy in a MAID system – should writes be cached? All reads? Only small reads (which may be likely to be meta-data)? The answer to these questions depend on the usage patterns of MAID systems, and we expect those patterns to differ from RAID access patterns.

The second decision is whether to provide a filesystem or block interface. File-level access would provide many benefits since accesses in large tertiary systems typically involves reading large files (*e.g.* climate model data or movies). Using file system information to copy or cache full files would probably provide a performance benefit. Block-level access would not require file system semantics and would work with contemporary systems.

2.1 Design choices used in this study

For our initial study, we choose to examine a non-migratory, block-level design. Figure 1 shows a schematic of the system design. The system is divided into zero or more "active drives" that remain constantly spinning; the remaining "passive drives" are allowed to spin-down following a varying period of inactivity. Requests from one or more initiators is directed to the set of virtual targets.

We examined two configurations: MAID-cache and MAID-no-cache. In MAID-cache, the active drives act as a cache for read and write traffic. The disk is partitioned into 512-sector "chunks" and a cache directory maintains an LRU ordering and information about the location of the contained data. The cache is examined for all read requests; any matching request is sourced from the cache (even if the corresponding passive drive is actually powered up). Write requests first probe the cache; if there is a cache entry corresponding to the write address, the data is written to the cache. All entries are placed in the write-log, which is used to eventually commit the writes to the passive drives. Writes that are the size of a full cache block (512 sectors) are written to the cache even if an existing block has not been allocated. To maintain consistency, all reads examine the entries in the write log prior to accessing data from the cache or the passive drives.

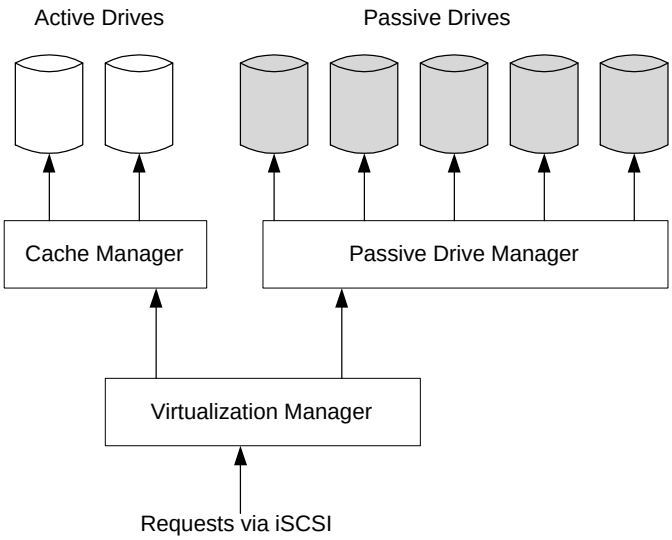

Figure 1: MAID configuration with caching

Passive drives remain in standby until either a read request misses in the cache or the write log for a specific drive grows too large. Once the drive is powered up, the queue of read and write requests is serviced. Following this, the drive is remains idle until the spin-down inactivity time limit is reached; varying the inactivity time limit is the primary way to influence energy efficiency and performance.

The MAID-no-cache design is essentially similar, but there are no cache disks; all requests are directed to the passive drives, and those drives will remain active until their inactivity time limit is reached.

In all, we compare the same number of *passive drives*, or drives that hold data. In the MAID-cache organization, some fraction of the drives act as "overhead", and this limits the potential energy savings. For example, assume the we have an array of ten drives with an additional active cache drive. The cache drive remains spinning at all times. This implies that this MAID-cache configuration can save no more than 90% of the energy of a system that is constantly active with 10 data drives.

3 Preliminary Results

We decided to compare the performance of the MAID system to that of a similarly configured RAID system rather than a tape library. In part, we felt that comparison against a tape library would be uninteresting because the tape load, seek and rewind times are so long. Furthermore, since each tape drive consumes as much energy as 10 disk drives, the tape system may have worse performance *and* energy usage than a simple MAID system which only powered up drives when requests were pending.

We used two sets of traces to drive a performance and power simulator for MAID and RAID systems. The sets are derived from server and interactive system performance, and thus have very different characteristics than accesses we would expect for a large tape library.

The interactive workload traces were acquired from StorageTek Inc., and were taken from a large, active database management system. The trace file represents 19 hours of transactions. The ratio of read requests to write requests was about 1.2:1. The server workload came from a programming development environment, and represents one weeks worth of requests. The ratio of reads to writes is

approximately 1:2.4.

As expected, the response time of MAID with a cache outperformed MAID with no cache. In particular, write response time were significantly better, as writes that do not hit in the cache are placed in the write buffer until they are transferred to the passive drives. This was particularly important for the server workload which had a larger percentage of write request than read requests.

Read performance was significantly effected by the spindown delay, particularly at small values. With small spindown delays, successive reads to drives that had been spun up were unable to take advantage of the drive being in a ready state when they arrived. Both read and write request performance suffered with very small spindown delay times due to the congestion caused by write log traffic that was required to wait while target drives spun up. Longer and longer spindown delay times did not contribute significantly to performance and increased energy usage.

Most problematic with small spindown delay times are the resulting large number of spinups required to service requests. Frequently spinning up the drives impacts reliability and the overall lifetime of the drive.

The interactive workload consumed less energy on the MAID with cache than the MAID with no cache. 82% of the reads were satisfied by the cache, resulting is fewer requests to spinup the passive drives and keep them spinning during throughout the delay interval. Writes hit in the cache only 12% of the requests, due to the dominance of the read requests which then required writes to go to the write buffer. Suprisingly the server workload consumed less energy on the MAID without a cache. The server workload was dominated by writes with 38% of the writes being satisfied by the cache. This dominance forced reads to the the passive disks, requiring more energy for spinup. In the MAID-no-cache configuration locality of read and write requests result if fewer spin-ups and less energy consumption.

The least amount of energy is consumed with a spindown delay around four seconds. Read and write performance improve with longer spindown delays, but only marginally after about a 60 second delay.

4 Conclusions and Future Work

The MAID concept seems to offer a good trade off on performance and energy efficiency. Further work is needed to compare caching policies, migration schemes and energy-conserving redundancy techniques. More representative traces are needed to determine the efficacy of these approaches with a diverse workload. We're currently deploying a large MAID system for further analysis.

References

[1] StorageTek Corp. 9310 tape silo information. `http://www.storagetek.com/products/tape/9310` 2001.

[2] Department of Energy. Historical electricity rate tables. http://www.eia.doe.gov/cneaf/electricity/page/at_a_glance/sales_tabs.html, 2001.

[3] Ethan Miller and Randy Katz. An analysis of file migration in a unix supercomputing environment,. In *USENIX Winter Technical Conf. Proceedings*, pages 421–433, 1993.

[4] Jeffrey S. Chase, Darrell C. Anderson, Parchi N. Thakar, Amin M. Vahdat, and Ronald P. Doyle. Managing energy and server resources in hosting centers. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles*, pages 103–116, 2001.