

Massive Arrays of Idle Disks For Storage Archives

Dennis Colarelli, Dirk Grunwald
{dennisc,grunwald}@cs.colorado.edu

Dept. of Computer Science
Campus Box 430
University of Colorado,
Boulder 80309-0430

July 26, 2002

Abstract

The declining costs of commodity disk drives is rapidly changing the economics of deploying large amounts of online or near-line storage. Conventional mass storage systems use either high performance RAID clusters, automated tape libraries or a combination of tape and disk. In this paper, we analyze 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 with effective power management of individual drives, this performance can be achieved using a very small power budget. In particular, we show that our power management strategy can result in the performance comparable to an always-on RAID system while using $1/15th$ the power of such a RAID system.

1 Introduction

Disk arrays are typically designed to improve performance or reliability. Existing RAID systems provide performance by striping data across multiple drives, allowing the use of multiple heads during reading and writing. Reliability is provided by duplication or parity blocks. RAID systems rely on multiple drives for

both performance and reliability. Robotic tape systems are designed to reduce the wait time of manual tape loading and increase storage density to provide large amounts of storage. Historically, tape libraries are preferred over drive arrays used for large (100+ TB) storage environments. In part, this is due to the costs of purchasing and powering large disk arrays.

Consider a scientific computing center that stores data on a variety of magnetic tape media; such a system may have 20,000+ tape cartridges and store more than 500TB of data. The growth of storage at such centers can be large, exceeding $\approx 5TB/month$, and the amount of storage transferred each day can exceed 680GB. Both the increase in storage and the rate at which storage is accessed are increasing because faster computers allow more accurate models that can use and generate more data. For example, the European Center for Medium-Range Weather Forecasts (ECMWF) expects their storage demands to grow to 1,500TB of storage by 2004 [1]. The Stanford Linear Accelerator Center's BaBar experiment has a database holding over 500TB and growing at more than 500GB daily [2].

Large tape libraries can accommodate many tape drives; for example, the StorageTek 9310 tape libraries can support up to 80 T9940 tape drives [3]. 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 uncompressed storage. Tape libraries are designed to work with different generations of media, although the tape drives usually have limited capability. The T9940 tape drive takes 18 seconds to load a tape, 90 seconds to

⁰0-7695-1524-X/02 \$17.00 (c) 2002 IEEE

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. Migrating from one generation of media to another is problematic simply because of the volume of media and the (relatively) limited number of drives. Many supercomputer sites that transition from one media format to another require over a year to complete the transition.

Disk systems provide an alternative to large tape libraries, but raise problems not encountered with tape systems [4]. Current disk drive capacities are approximately the same as tape – as of May, 2002, 160GB drives are plentiful and inexpensive (about US\$180). However, conventional disk-based RAID systems consume significantly more power than comparable tape systems.

For point of comparison, we'll assume that 60GB drives are used. A single StorageTek 9310 tape library consumes 1.1Kw/h of electricity. Storing 1,000TB of information would require three STK9310 libraries (at a total of 3.3Kw/h) and some number of tape drives. The T9940 tape drive consume 85 watts of power. If we assume half of the full compliment of drives is used (120), a further 11.5Kw/h of electricity is consumed, assuming the drives are constantly in use. A consumer grade 60GB IDE disk drive consumes $\approx 8 - 16$ watts. Ignoring the energy cost of controllers, backplanes and additional components needed by high performance RAID systems, it would take 144Kw/h to power a disk array similar to the tape system previously described. Assuming a 24x7 data center operation, it would cost $\approx \$9,400$ to power the tape library system vs. $\$91,500$ to power the the disks in the disk array.¹

This estimate ignores the additional electronics needed to actually build a disk array, such as RAID controllers, high performance servers, Fibre Channel switches and so on; thus, it is an underestimate. Additional power would be needed for cooling since the disk array dissipates significantly more heat than a tape library.

There are advantages to using a disk system for data archiving – the data would be available much quicker, allowing better utilization of the supercomputers, and a single homogeneous disk array may be easier to man-

¹This estimate is based on an electricity rate of 7.25 cents/Kw/h rate for commercial customers; electric utility rates have held fairly constant over the last 10 years at ≈ 7.25 cents/Kw/h for commercial customers [5].

age. Our hypothetical tape library would have an aggregate bandwidth of ≈ 1200 MB/s, while the disk array could provide a peak bandwidth of $\approx 2,880,000$ MB/s. However, existing RAID systems provide more capability than needed by such large storage environments. Analysis of mass storage systems in a supercomputer environment has shown that 50% of the data is written and never accessed [6] and a further 25% of the data is accessed only once. In our more recent analysis of a supercomputer storage system, we found that of the 10,489,550 file operations we examined, 60% of the 5,511,019 unique files were used by a single operation (*e.g.* read or write) and 84% were used by two or fewer operations. This file access pattern may be common in supercomputing centers, because it is more cost effective to store data than to recompute it and some centers must maintain data for longitudinal studies that span decades. Although such systems need a large amount of reliable storage, they have little need of either the high performance or increased reliability of conventional RAID systems, at least at this scale.

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 to large storage arrays while maintaining acceptable performance, increasing storage density and maintaining performance similar to conventional disk arrays or tape libraries. Unlike existing RAID systems, MAID systems are designed to emphasize performance *and* energy efficiency. Since performance, while important, is secondary, MAIDs can be constructed using inexpensive commodity disk drives and inexpensive servers. Like traditional RAID systems, MAID systems may emphasize caching and workload prediction for good performance, but the performance implications of these designs differs; if a MAID system were constructed following the design decisions of a conventional RAID system, it would have good performance but waste power.

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. Since supercomputer archival storage is one of the most immediate applications for large MAID arrays, we have used file access traces from a supercomputing center to drive our simulations. Our analysis demon-

strates that MAID offers performance comparable to a constantly-on drive array for workloads representative of archival storage systems while using about 15% of the power of such an array.

In §2, we measure and characterize the performance and power of commodity IDE disk drives. We then describe the workload we used to evaluate our initial design and variants of the MAID systems we have explored. In §5 we compare the performance and energy efficiency of MAID systems to conventional RAID drive arrays.

2 Disk Performance And Power Modeling

The design of MAID arrays requires accurate models for both drive performance *and* for drive power. We use a combination of existing performance models and measurements from sample IDE drives to develop a unified performance and power model. Our simulator model is simply an extension to a analytic model designed for SCSI drives [7]; however, we verified that the important aspects of the model are also appropriate for IDE drives. To make our model and analysis concrete, we measured various characteristics of several samples of a 20GB IDE drive; specifically, we used the IBM 60GXP Model IC35L020AVER07 because there is copious information available and we had several of the drives. The drive supports the ATA-5 standard and UltraDMA Mode 5 (100MB/sec) and uses a load/unload mechanism to avoid head-to-disk contact on startup.

Shriver *et al* [7] model the seek time within a single head using a bipartite equation (combining a linear portion and a sub-linear portion). We wrote a program to seek from cylinder to cylinder using the native IDE command interface. Using this program, we collected the data shown that appears to match the prior models sufficiently well. We then computed a least-squares fit for the more linear portion of the data and used that to estimate the value where the non-linear model takes over. We then used a non-linear regression model to fit the remaining data. The resulting parameters for the seek time in the Shriver *et al* [7] model are shown in Figure 1 where 610000 and *dis* are in units of sectors using a linear block address.

Most drive performance models focus on estimating

the effect of seeks, rotational delay and drive cache policies [8, 7]; we are also concerned with the time needed to spin-up disk drives. Although some papers have modeled spin-up performance, the models were either for laptop drives (which have different characteristics) or relied on vendor specifications of typical behavior [9, 10]. We found that disks had wildly varying times to transition from power-off to idle; most list 7-15 seconds, but we found values of 3-8 seconds until first byte read to be more typical. Spinning down the sample disk for even a short period requires a relatively long spin-up period (≈ 1.6 seconds). The spin-up period increases almost linearly with time once the disk begins to spin down. Once the platters have stopped rotating, spin-up time is relatively constant. The initial delay is caused by head loading – once the drive is placed in standby or sleep mode, the head is moved off the disk. Moving the head back requires servo calibration to accurately track the head as it moves over the drive.

We approximate the spin-up delay using a two-piece linear model; this slightly over-estimates the delay in spinning up the drive but means that our performance estimates for spinning up the drive are conservative.

As with spin-up performance, few papers have provided drive energy models for large IDE drives; the models were either for laptop drives (which have different characteristics) or relied on vendor specifications of typical behavior [9, 10].

Disk drives have two components that contribute to the overall power demands of the drive. The first is the 12V spindle motor used to spin the platters and drive the head motors. The other is the 5V supply used to power the analog-to-digital converters, servo-control DSP's and interface logic. Figure 2 show the power used on the 12V and 5V supplies, respectively, as the drive transitions from idle→standby→idle. During the idle period, we conduct numerous seeks. We believe the large peak (from $\approx 7200ms \rightarrow 9200ms$) is due to drive spin-up and the short peak (at $\approx 10,000ms$) is due to unparking the head. We couldn't observe significant power spikes due to the seeks – although a seek takes considerable power, the actuator is actively driven for a very short period of time, meaning the total energy from seeks is fairly small. We observed that the average power values are appropriate approximations for the idle (6.6 Watts) and standby (1.03 Watts) periods. We use similar measured data from many drives

$$SeekTime[dis] = \begin{cases} 0 & dis = 0 \\ 0.00169 + 1.740 \times 10^{-6} \sqrt{dis} & 0 < dis \leq 6100000 \\ 0.00456 + 2.364 \times 10^{-10} * dis & dis > 6100000 \end{cases}$$

Figure 1: Seek Time Model For Sample Disks

to calibrate our simulator for the disk spin-up power and drive power; the values described for this particular drive are representative of many of the drives we measured.

3 Workload

A massive disk array is clearly targeted for very large workloads. It is difficult to assess the efficacy of such a system without some model or trace of activity from a similar system; if a system is configured with many drives and there are few requests, the power savings can be dramatic.

Rather than use synthetic workloads or extrapolate behavior from small traces, we have evaluated the systems using a combination of small workloads representative of common workstation file access, a trace of database access and another trace constructed from a workload using logs of file-level access to a large storage system in use at a supercomputing facility. Due to space constraints, we will only report on the results for the supercomputer center. We first describe the traces used to evaluate the system and the performance metrics we used.

3.1 Traces For Evaluation

The first version of MAID we are evaluating presents a block-based interface. To evaluate this version, we used traces of file system activity that was reduced to a sequence of block requests (read, write) to a disk array. For each request, we recorded an arrival time for that request; requests are issued no sooner than the original arrival time. We also recorded the block number and operation type. In general, we must assume that each request must be completed before any further requests can be issued; for example, a read request for a directory may need to complete so the O/S can determine the blocks needed for future read. Without explicit dependence information, we assumed that each operation depended on all prior operations. In certain

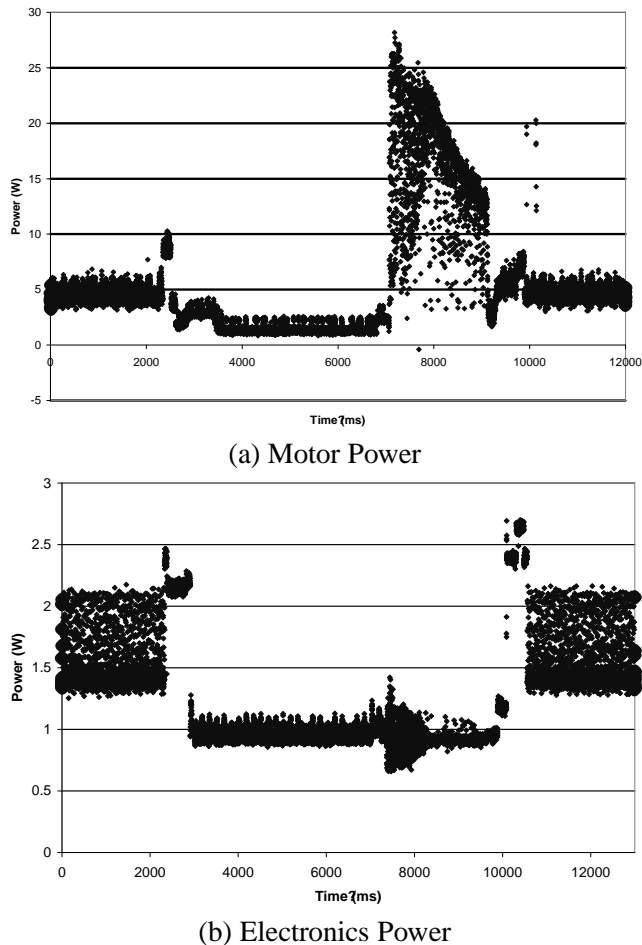


Figure 2: Power from the 12V and 5V Supply for the Spindle and Actuator

Attribute	Trace B	
Prior Requests In Trace	9,000,000	
Prior Files In Trace	4,615,188	
Prior Reference Volume In Trace	600,304,523 MB	
Requests In Trace	200,000	
Days In Trace	≈ 5	
Unique Files In Trace	174,178	
Reads	31,050	3,967,016 MB
Write	14,252	725,285 MB
Create	143,533	5,211,031 MB
Purge	11,165	2,140,427 MB

Table 1: Characteristics of Workload Used For Evaluation

circumstances, we were able to synthesize dependence information or determine that dependences did not exist. We determined the traces from the supercomputer center were largely non-dependent.

The file-level traces are the “cello96” traces provided by HP labs; these are a more recent set of trace similar to the “Cello92” traces [11]. The database traces were block level accesses from a 32GB RAID disk that was split into two partitions; the first was a swap partition and the second was used for an oracle database. Since this workload easily fits into a contemporary single drive, we scaled the drive sizes to use 10 3GB drives; although these are artificially small drives, they allow us to examine power and performance for a small workload. For space reasons we report only on the supercomputer center traces.

The supercomputer center trace was synthesized from a trace of a storage system that has been in use for many years and uses a combination of disk, robotic tape and hand-mounted tape systems. The storage system actions can either create a file, write to an existing file, read from a file or purge a file. The trace we used contained a total of 10,489,550 transactions covering 5,511,019 unique files. The read, write and create operations result in an average total of 1,871GB of data transfers per day; this is approximately 22MB/s of sustained activity throughout each day, although peak bandwidth demands are higher.

The supercomputer file-level traces provide a rich source of information for approximating a workload that might use very large disk arrays system, but they arise from a mixed-media storage system. We selected two sub-traces from the workload and used these to characterize MAID performance; we only report on

one here due to space constraints. Salient characteristics of the trace is shown in Table 1. The trace was from late in the year. We used the references prior to the start of the trace to populate a file system in our simulated disk array. For the trace, we allocated space for 4,615,188 files taking a total of 600TB of storage. Each ‘read’, ‘write’ or ‘create’ would cause a file to be created. Each ‘purge’ would cause a file to be deleted, making the space available for other files. Following this initial population of the disk array, we would then simulate the access of 200,000 file operations; only read, write and create operations actually generated disk transactions. The simulated disk array contained 1120TB of storage organized as 7,000 disks holding 160GB each.

Both of our evaluations traces are compromises. The database trace is small, but uses application level block operations. The supercomputer center configuration presents a more realistic number of drives, but synthesizes the location of block operations.

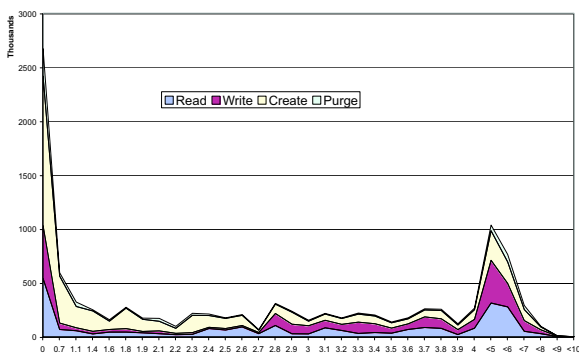
3.2 Metrics for Evaluation

There are two performance metrics we use to compare system configurations: energy and request response time. Energy is significant because the amount of energy required to spin-up a drive is considerably more than that required to keep the drive spinning. Hence a power model that would spin-up drives too frequently would consume more energy than simply leaving the drives spinning. Additionally, frequently spinning up drives reduces their lifetime. Request response time provides an accurate basis for performance comparison, particularly when using trace data in a simulation.

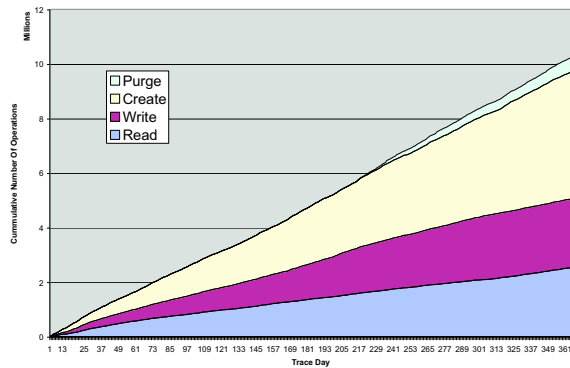
4 The Initial MAID Design

At first blush, the design of the MAID system is simple compared to the performance-oriented design of RAID systems; unlike RAID systems, there are no parity drives and thus no optimization of parity declustering or the like. However, maintaining the balance between performance and energy efficiency means there are several design decisions to consider.

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



(a) Logarithm of File Size, Broken Down By Operation Type



(b) Cumulative Size of File Operations Over Time, Broken Down By Operation Type

Figure 3: Characteristics of the supercomputing center workload showing file size distribution and file access rates. The file system is shown on a logarithmic scale on the horizontal axis; there are a large number of “small” files ($< 10MB$) and also many “large” files (from $10^4 \dots 10^7 MB$).

“more active” drives, or to distribute the data based on the likelihood 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 recently used data. Cache drives would always remain active to improve performance and reduce the need to spin-up idle data drives. 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 drives that make up the cache may be difficult.

There is also the question whether caching should be used at all. We could construct the MAID system as a simple RAID-0 and power down individual drives based on timers or access pattern heuristics. Caching decisions vary depending on the temporal and spatial locality in the file accesses. For large systems, we expect that file system metadata will be frequently ac-

cessed as people navigate a large MAID; this metadata would probably benefit from being cached. It is less clear if the actual data contents should be cached. Traces of file accesses on supercomputer mass-storage systems indicate that most data is written and never accessed. This same pattern will probably hold for data archiving in large companies, although data mining, inventory and other database operations may access the data more frequently. If large arrays are used for media servers or as storage containers for large proxy caches, different access patterns are likely – we would expect some degree of reference locality in a media server as users start, stop and replay music or videos.

These access patterns will also govern the caching policy in a MAID system – should writes be cached? All reads? Only small requests (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 typical RAID access patterns.

The second decision is whether to provide a file system 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 entire files would provide a performance benefit. For our initial study, we chose to use a block interface as this would let us deploy a MAID system across a broader array of systems; it also lets us

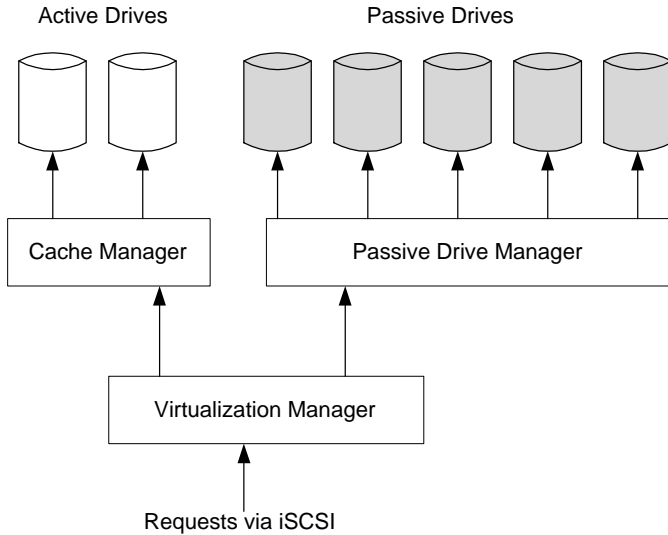


Figure 4: MAID configuration with caching

use block-level traces to quantify the performance of a MAID system. We may be able to capture the benefits of file system knowledge by prefetching blocks; this will be examined later when we can collect larger and more representative traces from a deployed MAID system.

4.1 Design choices used in this study

For our initial study, we choose to examine a non-migratory, block-level design. Figure 4 shows a schematic of the system design. The system is divided into zero or more “cache drives” that remain constantly spinning; the remaining “data drives” are allowed to spin-down following a varying period of inactivity. We assume that a large MAID system will be used by a number of clients using some form of storage virtualization. For our prototype we plan on providing an iSCSI interface since this provides an inexpensive block-level interface. Requests from one or more initiators is directed to the set of virtual targets.

Three parameters were used in the simulation studies: power management policy, data layout and cache use. Power management is controlled by a simple non-activity policy, where drives that have not seen any requests for a specified period are spun down, or an adaptive spin-down policy. The adaptive power management policy either lengthens or shortens the spin-down period, based on a recent request history to each drive.

Data layout is either linear, with successive blocks

being placed on the same drive, or striped across multiple drives. Blocks are grouped into “chunks” of 1MB to facilitate cache management and striping. The width of a stripe is limited to ten drives, which we assume to be reasonable to address both performance and power.

The cache parameter indicates the number of drives of the array which will be used for cache.

Simulation components consist of a fixed number of drives, some of which may be used as cache, a cache manager which implements the cache protocol, a power manager which enforces the power management policies, and a virtualization manager which controls data layout.

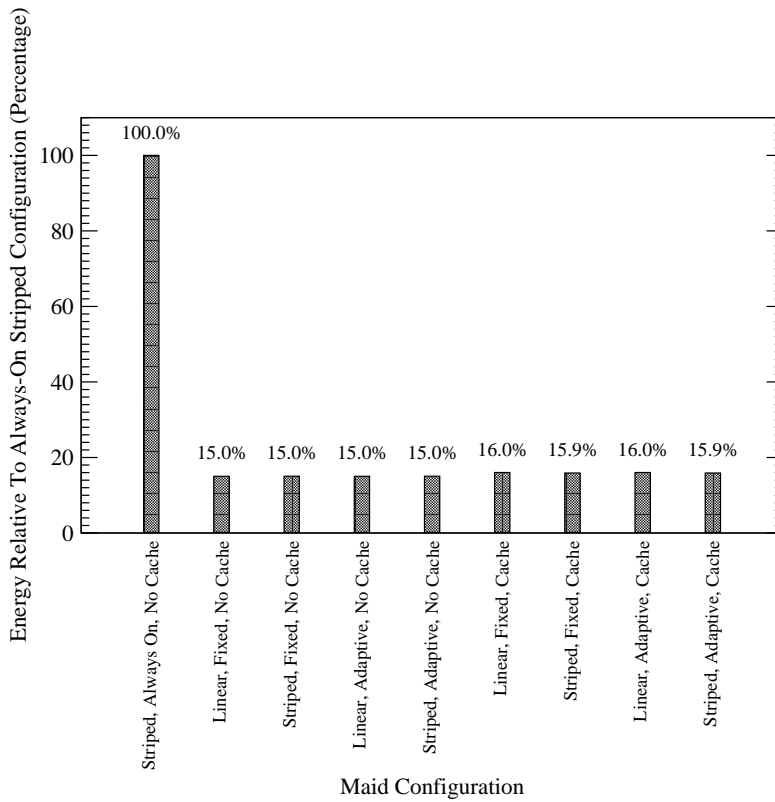
When a request arrives at the virtualization manager it is split in to some number of “physical” requests, predicated on the data layout policy. If no caching is enabled, requests are passed directly to the data drives. With caching, requests are checked in the cache directory. Requests that hit in the cache are sent to the cache drives and returned to the virtualization manager. Reads that miss in the cache are passed on to the data drives. Writes that miss in the cache are written to the cache, and written to the data drives when the target data drives transition from standby to active, typically from a read miss.

The write policy was designed to avoid spinning up idle drives. We assume that there will be limited reuse of data, and thus there is a low probability that any item will be read from cache and actually delay a disk spin-up; however, each write must be confirmed to stable storage, and storing the writes on the active drives should defer many drive spin-ups.

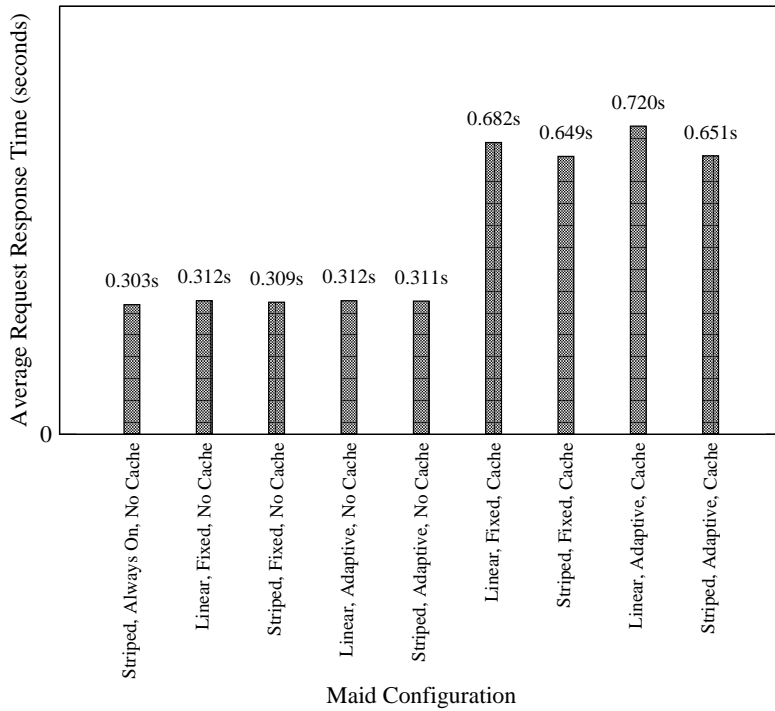
The power manager monitors request activity for all of the data drives. When requests arrive they are sent to the appropriate drive. Upon completion the request is returned and the spin-down interval timing is initiated.

Finally the virtualization manager coalesces all related physical requests and completes the original request.

These design choices were predicated on our target workload. We are currently investigating other options for different workloads and methods that adapt policies for the workload presented to the system.



(a) Relative energy for different MAID configurations



(b) Performance for different MAID configurations

Figure 5: Energy And Performance Comparison for Differing MAID Configurations

5 Results

Figure 5 shows the energy and per-request response time performance for seven different MAID configurations; we have examined a number of other traces and configurations, but these illustrate the primary point of our research: *intelligent power control of large very RAID clusters can yield performance comparable to an always-on disk array with significantly reduced power.*

The figure shows seven configurations in four groups. A “striped” configuration uses a RAID-0 organization where blocks are striped across multiple drives² We used a stripe group of ten drives, since that saturates the I/O bandwidth of the host system. A “linear” organization allocates data within a drive; this would result in fewer drives being spun up for a particular I/O than in the striped configuration, but can suffer from greater contention. These figures show different power management policies. The “full” power management policy always leaves each drive spun up. The “fixed” power management policy spins down any given drive following a specific period of no activity (512 seconds in this case). The “adaptive” policy uses a moving-average estimator for the idle period prior to spinning down a drive.

Lastly, cache configurations use 1% of the total drive budget.

From left to right, we show the energy and performance for a striped, full power system with no cache. Energy is given as a percentage of the always on, striped, no cache configuration. Note that we only measure energy from drives. No estimate was made for the overhead of virtualization, cache management, etc. For this study we were primarily interested in how power management techniques and cache behavior effect energy consumption in comparison to a similar system without these capabilities.

Performance is the sum of the response time for each I/O request. The full system time was the same for all configurations (*i.e.* the request stream could be satisfied with no delay by each configuration). The remaining groups show both fixed and adaptive policies for striped with no cache; striped with a cache; linear with no cache; and, linear with a cache.

Clearly, the full-power configuration (*i.e.* a con-

²In our initial study, we’re primarily concerned with non-redundant configurations.

ventional RAID-0 system) has the best performance; it also consumes the most energy because the drives are always on. There were some surprises in the remaining results. We had originally expected that the cache would benefit the “small” files seen in the supercomputer center traces; in studies using file system and database workloads, caches were essential to maintaining high performance. However, for this workload, the cache drives actually reduced performance. Analysis indicated that this occurs because of the lack of locality in the workload. Successive requests that hit in the cache drive compete with read misses that are written to the cache and dirty cache elements written to backing store for the I/O bandwidth on the few number of cache drives.

We had also expected that a linear data layout would be more energy efficient than a striped configuration, since fewer drives would be spun up for each request. However, the supercomputer center workload has many large transfers; this causes longer I/O queues for each drive, increasing both the response time and the time that those drives are spun-up. The greater power usage by the cache over the other non-cached power management configurations is due to the lack of block reuse in the cache. The 1% difference is attributed primarily to the cache comprising 1% of all drives.

Other workloads which exhibited greater locality saw a larger benefit from the use of cache drives.

6 Prior Work

Most work on disk drive power management to date has been concerned with mobile computers. Low power advances in microprocessors have increased the percentage of total system power consumed by drives to 20-30% and more [12]. While the characteristics of drives used for mobile computing differ from those used in large storage arrays, the research in this area is valuable for our studies.

Douglis et al. [12] compare fixed-threshold strategies against optimal offline algorithms. The authors also hint at predictive spin-down strategies. In [13] adaptive spin-down policies change the spin-down threshold based on perceived inconvenience to the user. Greenawald [10] discusses spin-down techniques with the assumption that arrivals of I/O requests are a Pois-

son process; the traces we examined have very bursty behavior. The author also studies how the reliability of drives is impacted by repeatedly spinning drives up and down.

Li et al. [9] look at several issues including fixed spin-down policies, performance impact as perceived by the user, adding memory as a disk cache, delayed writes and name-attribute caches. They conclude that a fixed spin-down policy with a small delay time can reduce almost all the energy consumed by the disk drive. This motivated our design - although the workloads are very different, their observations also appear

A machine learning approach is used by [14]. In their multiplicative-weight algorithm, input is taken from a set of "experts" which are other algorithms which make predictions. Predictions are combined with the goal of minimizing total error. They use Herbert and Warmuth's variable-share algorithm which takes some weight of each expert's, and shares it with the other experts. The algorithm gives the most credence to those experts performing well in the recent past.

7 Summary and Future Work

This work demonstrates that intelligent power management of large disk arrays can yield significant energy savings with only a modest performance penalty.

The MAID configurations examined in this study indicate that the success of a particular configuration is somewhat workload dependent.

With the large requests in this workload, striping improved performance over a linear data placement (which we expected) at no significant energy costs (which we didn't expect).

Our workload exhibited little locality. Consequently the cache configurations impeded performance and required more power than non-cache configurations. The lack of performance and energy gains with cache drives has motivated us to look at cache protocols that can intelligently handle low-locality workloads. Additionally we are looking at distributed caches to benefit both performance and energy savings.

As an alternative to using caches, we are investigating a migration scheme where blocks which are frequently referenced are moved to drives which remain spun up longer and infrequently referenced blocks are

moved to drives which may be spun-up less frequently.

The power management schemes require spinning up drives many more times than would be expected in an always on array. Based on the frequency of spinning the drives up, we are concerned about expected drive lifetime. As such, we are investigating redundancy methods to insure data integrity. We are particularly interested in redundancy techniques which require little power.

We are in the process of building a 2TB MAID prototype and have received funding to build a 20TB prototype.

8 Acknowledgements

We would like to thank Bernie O'Lear, Gene Harano and Bill Anderson for their help in defining supercomputer workloads. Michael Neufeld provided us with experimental results of power used in disk state transitions and seek time performance. We would also like to thank the anonymous reviewers for their many helpful suggestions. This was funded in part with support from the National Science Foundation through grant CCR-9988548 as well as support through the Colorado Center for Information Storage.

References

- [1] Jan 2002. http://searchstorage.techtarget.com/originalContent/0,289142,sid5_gci788800,00.html.
- [2] Jan 2002. <http://slac.stanford.edu/BFROOT/www/Public/Computing/Databases/Indes.shtml>.
- [3] StorageTek Corp. 9310 tape silo information. http://www.storagetek.com/products/tape/9310/9310_sp.htm—, 2001.
- [4] D.A. Sanders, L. M. Cremaldi, V. Eschenburg, D. N. Lawrence, C. Riel, D. J. Summers, and D. L. Petravick. Redundant arrays of ide drives. In *IEEE Transactions on Nuclear Science*, 2001.
- [5] Department of Energy. Historical electricity rate tables. http://www.eia.doe.gov/cneaf/electricity/page/at_a_glance/sales_tabs.html, 2001.

- [6] 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.
- [7] Elizabeth A. M. Shriver, Arif Merchant, and John Wilkes. An analytic behavior model for disk drives with readahead caches and request reordering. In *Measurement and Modeling of Computer Systems*, pages 182–191, 1998.
- [8] C. Ruemmler and J. Wilkes. An introduction to disk drive modeling. *IEEE Computer*, 27(3):17–28, 1994.
- [9] Kester Li, Roger Kumpf, Paul Horton, and Thowas Anderson. A quantitative analysis of disk drive power management in portable computers. In *USENIX Winter Technical Conf. Proceedings*, pages 279–291, 1994.
- [10] P. Greenawalt. Modeling power management for hard disks, 1994.
- [11] C. Ruemmler and J. Wilkes. UNIX disk access patterns. In *Proc. of the Winter 1993 USENIX*, pages 405–420. USENIX, Jan 1993.
- [12] Fred Douglis, P. Krishnan, and Brian Marsh. Thwarting the power-hungry disk. In *Proceedings of the 1994 Winter USENIX Conference*, 1994.
- [13] Fred Douglis and P. Krishnan. Adaptive disk spin-down policies for mobile computers. *Computing Systems*, 8(4):381–413, 1995.
- [14] David P. Helmbold, Darrell D. E. Long, and Bruce Sherrod. A dynamic disk spin-down technique for mobile computing. In *Mobile Computing and Networking*, pages 130–142, 1996.