Carnegie Mellon University

Information Networking Institute

THESIS

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
Master of Science in Information Networking

Title

RAID for Mobile Computers

Presented by Rachad Youssef

Accepted by the Information Networking Institute.

Thesis Advisor____Date__________

Garth A. Gibson, Assistant Professor

School of Computer Science and Dept. of Electrical and Computer Engineering

Reader____Date__________

Daniel P. Siewiorek, Professor

School of Computer Science and Dept. of Electrical and Computer Engineering

INI Dept. Chairman_____________________________________Date__________

Marvin A. Sirbu, Professor

Professor of Engineering and Public Policy, and Industrial Administration

Carnegie Mellon University

RAID for Mobile Computers

A Thesis Submitted to the
Information Networking Institute
In Partial Fulfillment of the Requirements
for the degree
MASTER OF SCIENCE
in
INFORMATION NETWORKING
by

Rachad Youssef
Pittsburgh, Pennsylvania

August, 1995

Table of Contents

Figures

Tables

Abstract

The requirement for high-performance, highly available storage for file servers and supercomputing systems led to the development of Redundant Arrays of Inexpensive Disks (RAID) and log-structured file systems (LFS). For mobile computers, however, performance is often a secondary requirement to long battery life. This study examines the design issues of low-power, highly available disk arrays for mobile computers. Specifically, by dynamically remapping the location of newly written data using a log-based allocation strategy and by deferring parity updates in an NV-RAM cache, the rate of drive spin-ups can be reduced by a factor of 2.

Key words: Data storage, RAID, mobile computer systems, low-power, log-structured, disk drives.

1.0 Overview

Processor power, software sophistication, and communications capabilities for portable systems are growing rapidly. While allowing portable systems to capture a larger fraction of a user's computing needs, these advances in turn, are creating demand for storage. This demand for storage translates into a need for large onboard storage since globally available, high-bandwidth communications is not a reality. As systems are increasingly targeted at the consumer electronics market, the need for solutions that reduce the frequency of backups and other preventive maintenance is also increasing.

Mobile computers are different from desktop machines because they require light weight, low-power, small-volume, high-shock-tolerant, low-connectivity components while still providing good interactive performance. Today's mobile computers are used predominantly as desktop machines which the user can take away and run the same applications. Current mobile systems only provide the user with several hours of autonomy. This makes battery life one of the most important product-differentiating factors in this market. Since battery technology alone will not provide the desired autonomy, there is a rich opportunity for design alternatives that reduce the power needed to perform a user's mobile work.

This thesis explores RAID-based secondary storage in mobile computers to provide adequate availability and capacity. RAID's single-failure tolerance allows the user to defer system maintenance, while capacity is provided by the scalable nature of RAID systems. Deferrable maintenance is a desired feature in mobile systems since they rarely have provisions for backups or other preventive maintenance. While RAID's availability and capacity features are desirable, its power characteristics are not. Replacing a single-drive system with a RAID level 5 architecture greatly increases the power consumption of the system due to the effects of striping and redundancy updating.

In this research study we examine the power requirements of alternative modifications to RAID architectures and algorithms. In particular, we highlight the importance of caching and scheduling to defer accesses until their execution minimally increases power consumption. Moreover, we propose and evaluate a *log-based* dynamic remapping architecture that allocates recently written data according to minimal additional power consumption.

This study is organized as follows. The following section looks at power consumption in a mobile system. Section 3 describes our RAID management scheme which emphasizes power. Section 4 describes our evaluation methodology, simulation environment and traces. We then describe the results of our simulations in Section 5, Section 6 discuses related work and Section 7 presents conclusions.

2.0 Design Considerations in Mobile Computers

Mobile systems depend on high-capacity batteries and low-power components for their autonomy. The most important limitation of a mobile system is considered to be short battery life. Mobile computers on the market today employ several techniques in order to maximize the amount of autonomy they can offer the user without compromising their performance.

All components are responsible for consuming power. Figure 1 shows a listing of major system components and their corresponding power usage for a particular system [Kester94]. In general, displays are the major power consumer in mobile computer. Though beyond the scope of this study, we heartily encourage new strategies for reducing their power consumption.

TABLE 1. Power Distribution of a Sample Mobile System

Component	Manufacturer & model	Power (watts)	Percent of Total
Display	Compaq Monochrome Lite25c	3.5	68%
Disk Drive (105Mbytes)	Maxtor MXL-105 III	1.0	20%
Memory (16 Mbytes)	Micron MT4C4MA1/B1	0.024	0.5%
CPU	3.3V Intel486	0.6	12%

2.1 Battery Technology

Battery technology and power management have become an important consideration in the design of portable computer systems. Portable computers can only operate for several hours before the battery power runs out. One of the design challenges for these systems has become how to maximize run time for a given battery weight.

Over the last 30 years the improvement in battery technology has been approximately a factor of two increase in the energy density (Watt-hours/lb) over three decades of nickel cadmium cells to their present value of around 22 Wh/lb. There is no reason known to us that will cause this rate of improvement to increase dramatically.The only likely new technology that will take over from NiCd will be nickel-metal hydride, since it alone provides a combination of enhanced performance (30-35 Wh/LB) and environmental safety. However, even for this new technology the projections are at best 40% improvement over the next five years [Srivastava94].

2.2 Non-Volatile RAM Variants

Magnetic disks are at present the technology of choice for secondary storage. However, disk power, size and weight constraints have made some designers try to escape to non-magnetic options. Currently the market offers flash memory and battery-backed DRAM storage systems as the predominant alternatives to disk-based storage at a higher cost.

Flash memory is a modified EEPROM that can be written by the host system. Flash's main advantage is that it requires no power to maintain state. Flash memory provides read-access times close to DRAM (45ns), but slower write-access times (about 4μs). One of the drawbacks of this technology, which is not expected to disappear in the coming years, is that flash memory requires an erasure step before any data can be written to it. This erasure step is a destructive operation; memory cells will eventually fail after a large number of cycles (on the order of 100,000 for the current technology). The use of Flash devices as the secondary storage unit for general mobile systems has been studied [Douglis93] and for special-purpose devices by [Forest94]. In both of these studies the use of Flash was established to be beneficial in special-purpose mobile applications by a factor of as much as 16%.

Flash memory consumes little power and has low latency and high throughput for read accesses. Flash memory comes in two forms: flash memory cards and disk emulators. Memory cards can be accessed in the same way the CPU accesses main memory. Flash-disk emulators are accessed through a disk-block interface. Flash disks have a device controller in the card that manages the flash-memory array, translating block I/O requests into the necessary memory operations. Flash disks provide the user with higher write performance by pre-erasing segments.

DRAM is currently available in high densities and is very fast to access. It is more economical than SRAM ($25MB versus $100/Mbyte) and is often used for main memory. It does require a small amount of power to maintain its state. DRAM is still considered a high-cost option. There are also serious reliability implications of having to power it with the host's batteries. Both of these reasons make its use in secondary storage unit not cost effective.

TABLE 2. Mobile Storage Comparison

Device	Latency	Cost
SRAM	70ns	$100Mbyte
DRAM	100ns	$30/Mbyte
Disk	9ms	$1/Mbyte
Flash	100ns(r) 4us(w)	$25/Mbyte

As can be seen from Table 2, the current alternatives to disk-drive technology remain too expensive; we believe that these technologies will not penetrate the market at this stage. As the processing capability makes it possible for a user to rely on a mobile computer for all computing needs, the cost of a desk-top system becomes prohibitive. In the case of a mobile-only computer user, the traditional desktop requirements for 500MB to 2GB must be met by the mobile unit. For this reason we believe that storage demands of mobile computer systems cannot be satisfied by NV-DRAM or Flash in a cost-effective way. However, industry analysts are convinced that their power consumption makes them a viable option for low-capacity specialized devices in the future [Wood93].

Due to a small NV-RAM write buffer, a single disk can perform comparable to Flash, even with an aggressive spin-down policy, but its power consumption is still an order of magnitude higher than Flash [Douglis93]. Our study makes use of this fact to reduce the power consumed in a multiple-drive architecture.

2.3 Disk Drive Technology for Mobile Computers

The main advantage disk drives have to offer the mobile-computer market is the ability to provide maximum on-line capacity at the lowest cost; they also provide sufficient throughput for large transfers.

Disk drives are a mature technology that offer high density and low cost, and their management is well understood. Because current popular form factors consume a lot of energy[1] and the recent explosion in demand for portable computers, disk-drive manufacturers have been enticed to develop a special breed of drives dedicated to serving this emerging market. These new drives enjoy special emphasis in several areas:

- They are able to tolerate shocks up to 300G [Integral92].
- They have a reduced physical volume and they weigh less.
- They consume less energy.

In particular, they have a new mode of operation called *Sleep* mode along with the standard *Standby*, *Idle* and *Active* modes as defined by the ATA-2 standard [ATA-2]; these modes are described in Table 3. The new Sleep mode allows a user to save up most of the energy that would otherwise be consumed by the drive. This energy-saving feature does not come without a cost to the user. An access to the disk while the disk is spun-down will incur a delay measured in seconds as opposed to the tens of milliseconds of delay expected from a spinning drive[2].

Major power savings can be achieved by applying an aggressive disk-spindown policy; a simple short time out drive-spindown policy can achieve close to optimal performance [Kester94][Klostermeyer95]. These results have been independently verified in various trace driven studies [Douglis94]. Current disk products use an idle time of two seconds as the appropriate spindown threshold. Figure 1 shows the standard transition diagram for a power optimized disk drive.

1. Newly developed small-form factor drives fail to deliver the appropriate capacity.

2. These new disks are also capable of withstanding multiple spin-up/spin-down transitions due to a new technology that handles parking unparking of the disk head which is known as *dynamic head-loading technology.*

TABLE 3. ATA-2 Description of Power Modes Used in Disk Drives

State of Drive	Description
OFF mode	The disk consumes no energy and is incapable of performing any functions except power-up.
SLEEP mode	The disk is powered up but the physical platters aren’t spinning.
IDLE mode	Disk is spinning but no disk activity is taking place.
ACTIVE mode	The disk platters are spinning and the arm is seeking or the disk head is actively reading or writing a sector. This mode consumes the most power, but occurs for short periods of time in a typical single user environment.

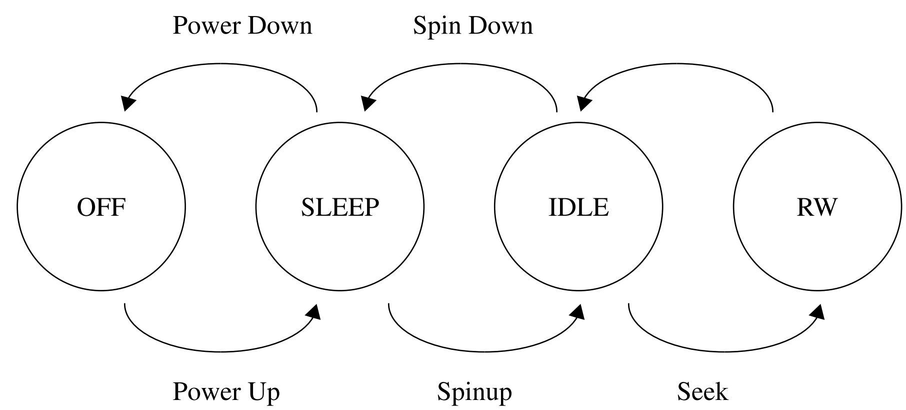

FIGURE 1. Description of Power Transition for Low-Power Disk Drives

To understand disk-power issues more fully, we can look at the sources of power dissipation in a disk-drive unit. The most important sources of power dissipation are:

1. Air shear power of the rotating disks (energy consumed preserving the angular momentum of the physical platters).
2. The actuator motion during seeking.
3. Arm-electronics power during read/write operations; power dissipated by the controller.
4. Other minor air-shear and frictional losses.

Although each of the enumerated factors contribute to disk-drive power dissipation, the first is the most significant. A much smaller fraction is spent powering the electrical components of the drive, controller and read/write channel. For example, the average actuator power is 25% of the spindle power [Grochowski93].

One empirical study of large-diameter disks (14 in.) derived the following expression for the power consumed maintaining the angular momentum [IIST90]:

$$\text{Air Shear Power } \alpha \text{ (diameter)}^{4.6} \text{ x(rotation rate)}^{2.8} \text{ x(\# of platers)} \qquad \text{(EQ 1)}$$

It was this expression that stimulated our initial interest in arrays for mobile computers. We observe that the small-diameter disks such as Hewlett Packard's 1.3 in. Kittyhawk, contained insufficient storage to satisfy a mobile computer user, but, according to this expression, might use as little as a factor of $(2.5/1.3)^{4.6}$ less power in the spindle motor than a 2.5 in. drive. This suggests replacing a single 2.5 in. drive with five 1.3 in. disks to match the capacity and reduce the power even with all five drives spinning. This story is certainly simplistic because:

1. The cost per Mbyte is not constant across the different form-factor drives.
2. Other sources of power consumption in the drives (see above list) are not governed by Equation 1.
3. Equation 1 may not hold for small-form-factor drives. It has been suggested that the surface friction observed on a large diameter drive is not experienced in a small diameter drive because the platters are also spaced much closer together. Therefore the shear surface becomes the cylinder formed by the outer edge plater rather than the surface of each platter [Brady94].

Regardless of the outcome of this comparison our interest in arrays for mobile computers stems from the belief that arrays can provide additional robustness and availability for mobile computers. In order to maintain the disk-drive volume acceptable to the size of portable machines while exploiting the robustness of RAID, we will need to move to smaller-form-factor drives. Fortunately, since the volume of a small array of seven 1.3 in. disks is the same as a single 2.5 in. drive, a RAID system can meet the volume constraints of current mobile systems. Our focus, then, is to minimize the number of spinups and spinning time of an array of small-diameter drives.

3.0 Power-Optimized RAID

We propose an innovative use of the basic RAID architecture in order to minimize power. Our design modifies the data layout of RAID level 4 and adds power-optimized caching policies.

RAID systems have traditionally been designed for performance and reliability. RAID systems primarily address the need for narrowing the I/O bandwidth gap and drastically increasing the time to data loss of secondary storage units. Disk drives have been increasing in performance at a much slower (20% per year) rate than CPU speeds (40% -100% per year) [Wood93] [Gibson95b]. Arrays use parallelism in the storage subsystem in order to meet the increasing demand for I/O bandwidth as well as the capacity requirements. As the number of drives goes up, the mean time to failure of an individual component decreases, thus the need for redundancy. More important is the fact that redundancy in disk arrays raises the mean-time-to-data-loss of the system well beyond that of a single drive [Gibson92, Patterson88].

Popular RAID architectures are optimized to use all drives concurrently in parallel to provide the user with the aggregate bandwidth of all the drives and minimize the overhead

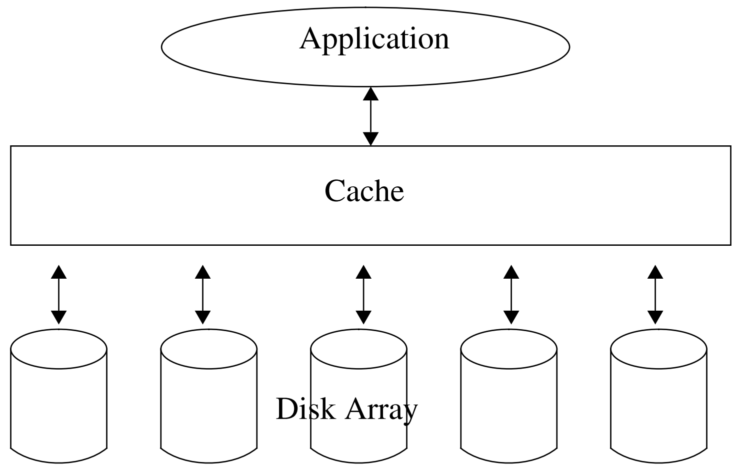

FIGURE 2. Basic System Architecture

time spent maintaining redundant information for failure recovery. In contrast the architecture proposed in this study attempts to use the drives one at a time. Caching policies are designed to cluster access into one disk in order to maximize the use of the disk it has spunup and to minimize the number of spinups.While these two ideas have the side effect that the aggregate bandwidth of all the drives is no longer used, this is as much bandwidth as is available now and disk bandwidth is not often put forward as the bottleneck of mobile systems.

The basic system architecture is based on a cache containing at least some NV-RAM and an array of disks. We assume each disk follows a spindown policy such as was discussed in section 2.3, that is, a disk is spundown after two seconds without new accesses. Once an aggressive spindown policy like this is used in each drive, the dominant power factor becomes the number of spinups that the system incurs. We have studied the effect of three complementary strategies for minimizing the number of spinups:

1. The addition of a read cache to reduce the number of read requests that result in drive activity and thus reduce the power used by the system.
2. The addition of a write cache to defer write activity and allow for more energy efficient scheduling of write operations.

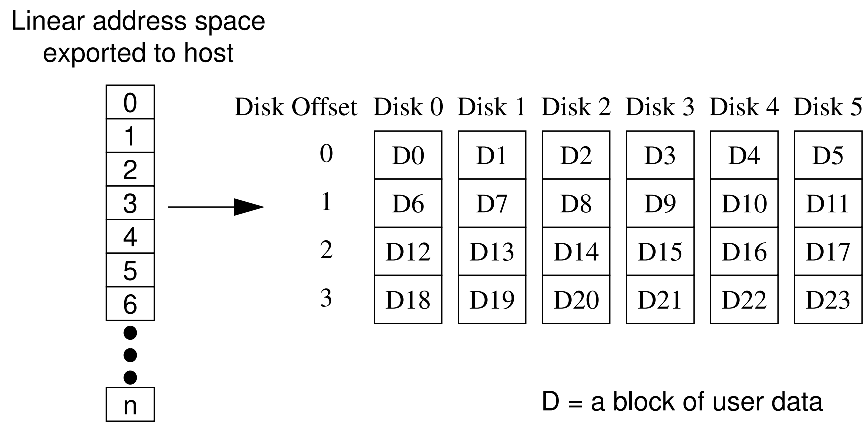

FIGURE 3. RAID level 0 Sample Layout

3. The appropriate data layout to maximize the number of user requests that can be satisfied for each spinup that is incurred.

In our evaluation, we compare Log Structured Storage to stand-alone drives, to standard RAID levels 0, 4 and 5 architectures, and to simple power optimizations of standard RAID architectures.

3.1 Standard RAID Architectures and Simple Optimizations

RAID level 0 refers to a non-redundant array of disks. Data is laid out by interleaving consecutive blocks of user data across N consecutive disks. Figure 3 shows one layout for a 5-disk, RAID level 0 device. It is important to note that RAID level 0 arrays have no protection against data loss when a disk fails; we included this architecture in our analysis to be able to evaluate the cost of maintaining parity information.

In RAID level 4, a set of N drives is divided into two sets: a set of N-1 drives is used to stripe user data; a single drive is dedicated to holding the parity units; each parity unit protects a set of N-1 data units. A possible RAID 4 layout is depicted in figure Figure 4.

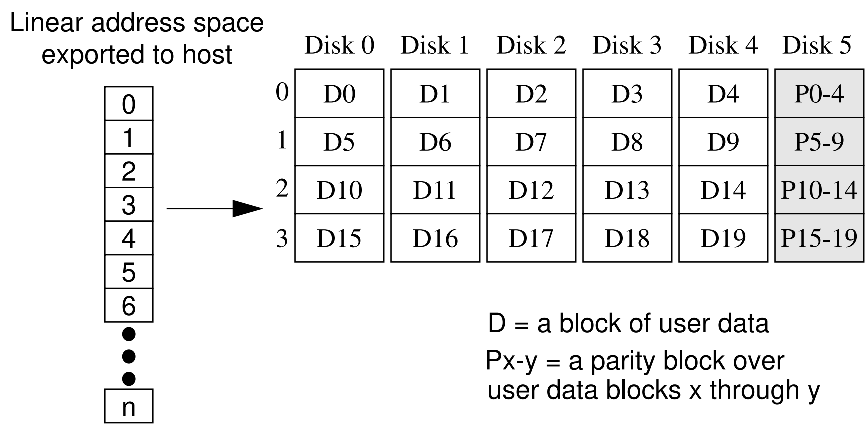

FIGURE 4. RAID Level 4 Sample Layout

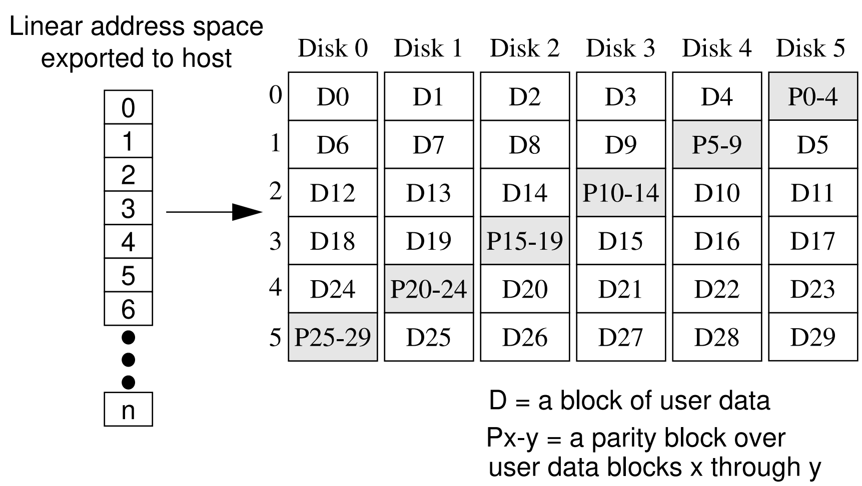

FIGURE 5. RAID Level 5 Sample Layout

In RAID level 5, N drives are used to stripe data and maintain redundancy information. This is achieved in the following way: user data is block-striped across (N-1) drives; a block of parity protects a set of (N-1) data units; parity blocks are distributed in the drives in a round-robin fashion. Figure 5 shows a possible data layout for a RAID level 5 system.

The following observations can be made about the expected power consumption of RAID levels 4 and 5:

1. The stripe unit is expected to play an important role in the power consumption of the system. Larger stripe units are expected to allow the user to satisfy single read requests

in a single drive, and even multiple requests close to each other in the exported address space without having to spin up multiple drives.

2. The location of the parity is expected to also play a role in the power performance of the system. Having all the parity blocks concentrated in a dedicated drive is expected to prove advantageous since it will allow all write operations to satisfy parity updates in the same drive, thus minimizing the probability that a parity update will induce a spinup.

Taking these two factors into account, we expect systems with large stripe units to require less power and the power performance of RAID level 4 to be better that RAID level 5 for the same stripe unit size. Hence we expect a RAID level 4 architecture configured with a stripe unit of infinite size (spanning the whole address space of a drive) to be the ideal standard RAID data layout for minimizing power consumption.

A cache can significantly reduce the amount of requests that translate into any drive activity for all storage organizations including RAID. Even without deferring writes, a read cache will reduce significantly the number of read requests that translate to drive activity, reducing the number of spinups incurred. Read caches are widely used in files systems and embedded disk controllers because their reduction of disk activity does not affect data integrity. A write-back cache, in contrast, will accumulate write activity in memory in order to schedule its execution in some more beneficial way at the cost of exposing the data to longer periods before it is safely on a disk. By deferring writes, these caches can cancel writes to data that is multiply written before update blocks are flushed to disk, thus reducing the amount of drive activity further. To increase the integrity of deferred write data, these caches typically protect against power failures by employing a non-volatile memory technology, in turn adding cost to the system.

The principal idea in our study is to defer writes until an appropriate disk is spinning for some other reason, to perform all possible write activity. In this approach we try to exploit spinups caused by read cache misses for all read and write activity.

The following caching policies are evaluated in our analysis:

1. A Write-Through (WT) caching policy will not retain any dirty blocks in the cache longer than necessary to update the disks.

2. A Write-Back (WB) policy keeps dirty blocks in the cache, flushing them to the drives as the cache becomes half full and retaining the data as clean blocks that are discarded in LRU fashion.

3. A Read-Miss Piggy-Back (PB) policy defers writes in the same way as the WD policy but it also takes advantage of read misses to issue pending writes to the drive that is now known to be spinning.

3.2 Log-Structured Storage Architecture

Log-structured Storage Architecture is inspired by the Log-Structured File System (LFS) [Rosenbum92], which treats underlying storage as a segmented, append-only log. The use of a disk storage manager similar to LFS in RAID systems has been previously studied as a possible solution to the small-write problem in [Mogi94] and implemented in-high end commercial products like StorageTek's Iceberg [StorageTek94]. In these previous implementations the basic idea was to accumulate writes to minimize the number of disk operations needed for parity maintenance and to maximize the write bandwidth. Neither study considered power consumption as a design factor.

Our use of the ideas behind LFS is to defer writes until a read miss induces a drive to spin up, then to dynamically remap pending writes to the closest free location on the now spinning drive as depicted in Figure 6. As with other LFS-inspired systems, performance of the system will depend on the workload; in systems whose read patterns closely mimic the write patterns for the same data most requests will be satisfied in the same drive, thus driving the performance of the system closer to the power consumption of a single drive-system.

Dynamically remapping written data may seem inappropriate for storage servers at first, but it can be done transparently so that a user or file system need not take note. As Figure 8 shows, the address space that is exported to the user is a linear address space that expands the aggregate capacity of the drives in the array minus the space designated to parity and the space reserved for the storing mapping information. The careful management of this "mapping information" (referred to as "functional Track Directory" in Iceberg terminology) is necessary. It must survive disk and power failures. As this constraint is familiar to many researchers [Menon93, Solworth91], we borrow their solutions as well —NV-RAM holds tables that are periodically written to disk with self-identifying information incorporated. We call this part of the system the Dynamic Remapping.

The function of the cache is to accumulate writes until there is enough data to fill in a segment worth of disk space. User-pending writes are then mapped to a free segment in the RAID address space and then written to the RAID system. The drives are configured as a RAID level 4 system with an infinitively large stripe unit. The Dynamic Remapping module translates addresses from the exported user-address space to the RAID level 4 system as shown in Figure 7.

3.2.1 Log-Structured Storage Algorithm

A user's read requests will be either satisfied in the cache or invoke disk accesses whose results are cached as clean data in the cache. Write requests are satisfied in the cache and

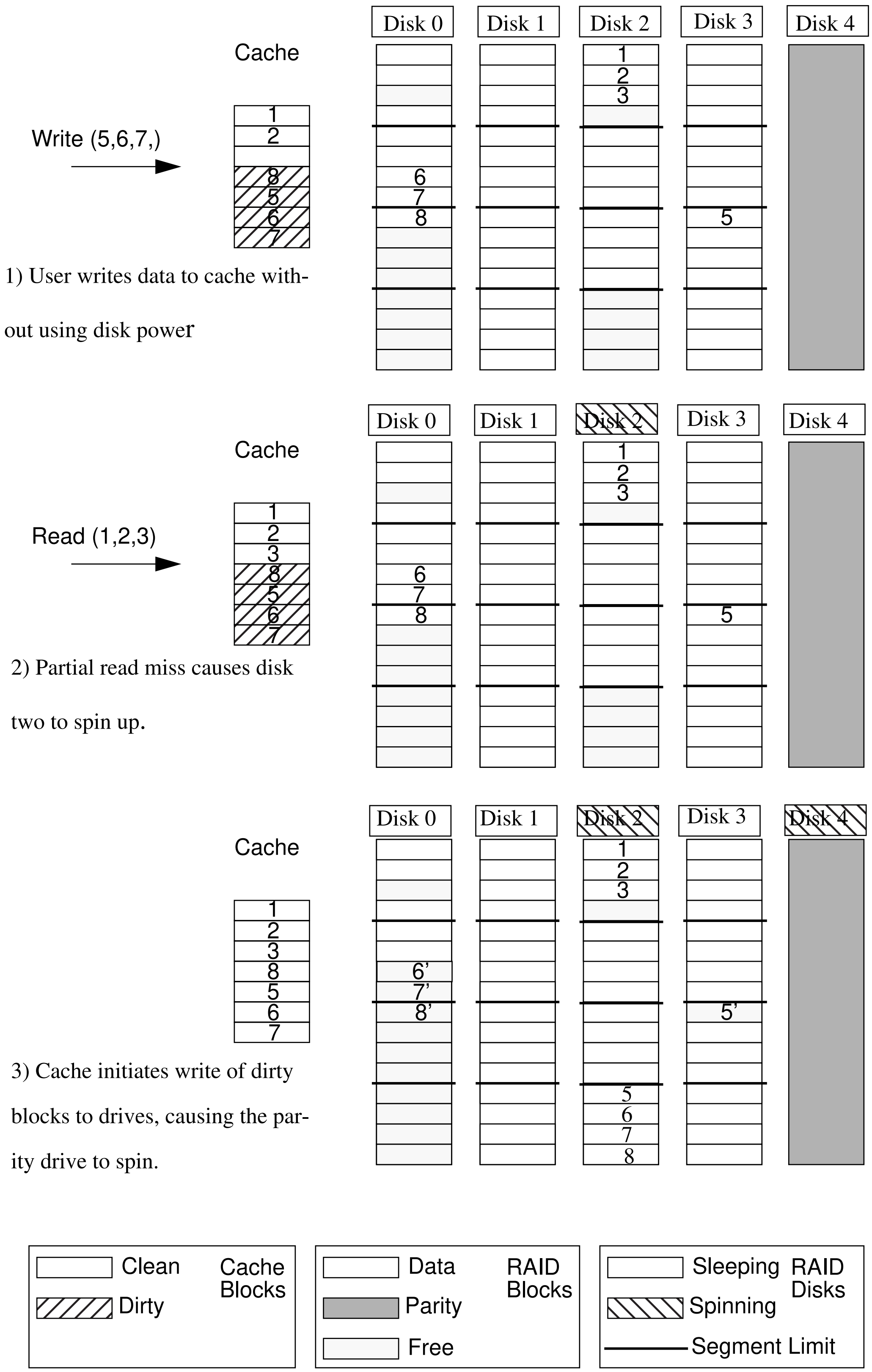

FIGURE 6. A LSS R/W Operation

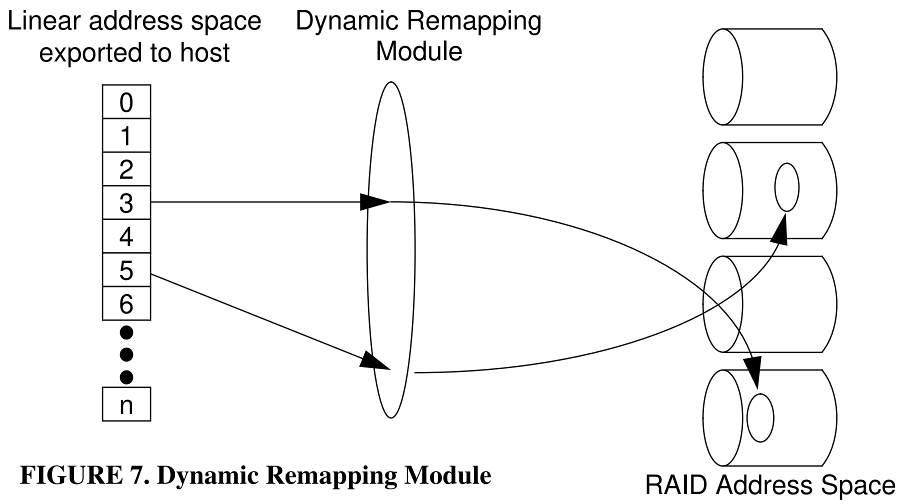

FIGURE 7. Dynamic Remapping Module

the disk's work deferred. The cache will accumulate writes until either: a read miss occurs and enough blocks are in the cache; the cache becomes low in clean blocks; or a user initiates a sync request. Once the cache triggers a write, accesses, grouped in segment sizes, are assigned and written to a clean segment in the last spinning or currently spinning drive.

This algorithm assumes there are available segments on all the drives at any given time. This means a certain amount of space is not visible to the user. This is needed in our system since the storage subsystem is typically not aware that files have been deleted until the storage locations are rewritten by a new data. Instead, the hidden space is constantly recycled. As new data blocks are allocated to free segments the old locations of those addresses are marked as free. The system keeps a pessimistic view of the utilization of the segments so it is able make decisions when it is time to clean up segments to provide the user with the space.

Segment cleanup involves selecting underutilized segments (live segments that have more than a threshold number of blocks not in use) and writing them into the cache, marking them as free, and allowing the newly cached data to follow the ordinary write path. Segment cleaning can automatically trigger when the system is recharging the power source, so that it has no impact on the life of the battery. Although of no impact to the results in this study, it is important to note that the following trade-offs are associated with the segment size: Large segment sizes greatly reduce the amount of mapping information that the system needs to keep track; large segment sizes lead to lower utilization of the storage space.

3.2.2 Mapping Structures

A set of maps constitute a “fuzzy image” of the dynamic map of the system; these mapping tables are stored both in memory and in predefined locations in the array. This means that, although not all structures are maintained at all times, there is always enough information in stable storage to recreate all the dynamic-map information of the system. This idea allows us to implement a fast, cheap and reliable way to maintain the information regarding the new current physical location of the user-address space presented to the user. It is fast because only a small portion needs to be updated on write access and a fast lookup can be performed on read access. It is cheap because only a minor part of the map needs to be resident in memory. It is reliable because the system can recover from failures at any point.

The main mapping structures are the mapping tables that keep track of the dynamic maps and maps that keep track of the segment allocation and utilization.

The Dynamic Mapping structures are made up of a global block map, a set of disk block maps and segment block maps. The system needs to permanently keep in memory an image of the user’s block-address space mapped to each drive; this constitutes the global block map. In our implementation the global block map stores four bits of information for each data block in the device for a maximum of 16 drives in the array. The global block map can be reconstructed in the event of a failure from the other maps, so it does not require non-volatile storage.

Each drive has a drive block map associated with it that contains a mapping of user location to drive offset. These maps are kept in fixed locations on the corresponding drive and are loaded into memory when a block needs to be looked up. They are updated in memory and written back to the drive when a set of new segments is written to the drive.

A segment block map is included at the beginning of each segment to aid the cleaner ramp segments back to user addresses. It is written when the segment is allocated and read back during segment clean-up operations.

A dedicated map keeps track of the utilization of each segment in the array, called the global segment map. This structure contains a data-block count for each segment in the array and is used to allocate space for new segment writes and in order to select victim segments for cleaning purposes.

The amount of space consumed by these map structures is summarized in Table 5.

4.0 Evaluation Methodology

The following section describes the techniques used to perform the analyses in this thesis. Trace-driven simulation was used to evaluate the performance and power consumption of LSS. The simulator used to produce this work relies on an accurate disk-simulation module managed by a real RAID driver, RAIDframe [Courtright95]. The following subsections describe the components of this environment

TABLE 4. LSS Mapping Summery

	Function	Location	Size	Fraction of array size
Global Block Map	User block to disk	Memory	A/(B*2)	1/(B*2)
Disk Block Map	User block to disk offset	Drive	A*4/B	4/B
		NV-RAM	A*4/(B*N)	4/(B*N)
Global Segment Map	Block count per segment	Memory	A/(S*B)	1/(S*B)
Segment Block Map	Segment offset to user block	Drive	A*4 /B	4/B
		NV-RAM	A*4/(B*S)	4/(B*S)

Size of Array=A; Bock Size=B; Number of Drives = N; Size of Segment=S

4.1 Disk-Simulation Module

The disk-simulation module used originates in a mature disk-array simulator called Raid-Sim [Chen90, Lee91]. It accurately models significant aspects of each access (seek, rotate and transfer) according to a table-driven cylinder layout and drive-performance parameters[3]. The geometry model was expanded to account for spin-up time and instrumented to capture power data. A trace of the power transitions of each drive is captured for postprocessing analysis. Because of the small-form-factor requirements, we chose the Hewlett Packard Kittyhawk ™ HP C3014A [HP94] as the drive to use in our simulations. Performance parameters where faithfully simulated.

3. Although the disk-simulation module does not simulate the drive cache buffer or any overhead associated with bus arbitration, we believe this to not be a disadvantage. Because the emphasis was on maximizing the amount of time the disks where in sleep mode and minimizing the amount of times a spinup occurred, the lack of those parameters is not considered important

TABLE 5. Disk Parameters

	C3013A
Data Surfaces per Drive	3
Data Bytes per Sector	512
Data sectors per Drive	10252
Data Cylinders per Drive	949
Total Bytes per Drive	20MB
Spin-up Time	1.5 sec
Disk Rotating Speed	5310 rpm
Cylinder Seek Time	5e-3sec
Max Stroke Seek Time	20e-3sec
Average Seek Time	6.1e-3sec
Spinup	3.5 W
Sleep	0.015 W
Idle	0.625 W
Active	1.5 W

4.2 RAIDframe driver

RAIDframe is as an extensible framework for rapid prototyping of parallel storage architectures being developed here at CMU. RAIDframe is built around an engine that executes RAID access without understanding the architecture at work. The engine executes directed acyclic graphs (DAGs) that express and control the concurrent execution of accesses on drives. RAIDframe's advantage is its configuration flexibility, allowing an architect to customize layout mapping, to easily tune critical sequencing of disk accesses and to build in architecture-aware caching.

The system can be used as a simulator, as a user-level software array controller that accesses physical disks using the UNIX raw device interface, or as a device driver in the kernel[4] [Courtright95]. The fact that RAIDframe can be used to control real disks is very powerful. It allows us to verify the correctness of the new algorithm on a configuration available in our lab [Gibson95a] before simulating, which validated the accuracy of our implementation. RAIDframe uses directed acyclic graphs to express and control the concurrent execution of drives. RAIDframe's advantage is the ability to separately configure a layer of mapping indirection, carefully tuned sequencing of disk accesses and architecture-aware caching.

4.3 Implementing LSS in RAIDframe

RAIDframe's flexible architecture allowed us to implement an extra layer of mapping, the Dynamic Remapping module. In all RAID architectures a user access is mapped to separate disks and offsets. We introduced an extra level of indirection between the cache module and the standard mapping code. This layer had access to all the maps needed to manage the segments and map user data to RAID addresses.

4.4 Traces

The traces used in this study where obtained from Panasonic Technologies Inc. These traces where used in a previous study also evaluating storage alternatives for mobile computers [Douglis94]. These traces where taken from two instrumented Apple Macintosh PowerBook Duo 230s. They record all file-system activity: each record specifies the type of operation, read or write, the file blocks that where affected, and the approximate amount of time elapsed since the previous operation.The major characteristics of these traces are reported in Tables 6,7 and 8.

A serious limitation of these traces for our research goals was the lack of information regarding the location of the files on the drive. Because this limitation forces us to approximate the layout of the files on the disk, we explored several allocation strategies for their

4. Currently RAIDFrame can be used in Digital AXP workstation running OSF /1 v2.0 and v3.2 as a device driver.

TABLE 6. Machines Simulated

	Machine A	Machine B
Number of Traces	15	22
Address Span	42MB	75MB
Number of Files	522	952
Average File Size	81KB	82KB
Max FIle Size	12MB	13MB

impact on our study. Because the allocation alternatives turned out to have less impact than expected we present just two:

1. Scatters files over the whole address space of the simulated storage device at random.
2. Lays out the files sequentially in the simulated storage address space according to the order they are first touched in the traces.

A random mapping is intended to be pessimistic; good file systems try to cluster related data in an organized fashion [McKusik83]. In contrast a temporally allocated layout is optimistic for log-based systems. For both schemes file lengths were approximated to the largest location accessed in each file.

While the complete set of traces were used to approximate a reasonable initial file allocation on disk, a particular set of four traces were used in the results reported in Section 5.

.

TABLE 7. Statistics for Traces Used with Machine A

	Machine A Trace 1			Machine A Trace 2		
Duration			1 Hours 0 Min.			7 hours 55 min
Number of Operations			54592			153551
Number Distinct locations touched			5.5MB			14.0MB
Fraction of Reads			17%			99%
Fraction of Writes			83%			1%
Percent of Bytes Read			26%			12%
Percent of Bytes Write			74%			88%
Async accesses			1%			1%
	Avg	Max	SDev	Avg	Max	SDev
Inter-arrival Time (msec)	69.13	39400	469.7	189.6	48200	922.9
Operation Size (Kb)	0.26	60.71	1.66	0.21	61.13	1.58
Read Size (Kb)	0.43	60.71	2.28	0.13	61.13	0.70
Write Size (Kb)	0.23	24.8	1.50	15.20	24.87	11.99

TABLE 8. Statistics for Traces Used with Machine B

	Machine B Trace 1			Machine B Trace 2		
Duration			3 Hours 30 Min.			3 hours 29 min
Number of Operations			169743			61539
Number Distinct locations touched			20.0MB			11.3MB
Fraction of Reads			54%			86%
Fraction of Writes			45%			13%
Percent of Bytes Read			55%			54%
Percent of Bytes Written			44%			46%
Async Accesses			4%			1%
	Avg	Max	SDev	Avg	Max	SDev
Inter-arrival Time (msec)	75.0	90783.	0.562	206.8	55683	974.7
Operation Size (Kb)	0.34	394.27	3.38	0.45	394.27	4.55
Read Size (Kb)	0.35	394.27	4.17	0.40	394.27	4.62
Write Size (Kb)	0.32	65.54	2.08	0.80	65.54	4.05

5.0 Evaluation

RAIDFrame was used as the simulation vehicle to explore the trade-offs of the different architectures. We focused on the following issues: the power performance and the response time of the system.

In order to evaluate the power consumption and performance of the different architectures the traces described in Section 4.2 were run through the RAIDframe simulator under a number of different configurations. The parameters varied were:

1. Architecture: RAID Level 0, RAID Level 4, RAID Level 5,LSS.
2. Cache Size: 0KB, 32KB, 256KB, 512KB.
3. Caching Policy: WT, WB, PB.
4. Initial FIle allocation: Scattered or consecutive.

Each of the two machines were configured in the following way:

Machine A: storage capacity 60MB (3 drives RAID level 0, 4 drives RAID levels 4 and 5 and LSS), the space utilization was constant at 68%.

Machine B: storage capacity 100MB (5 drives RAID level 0, 6 drives RAID levels 4 and 5 and LSS), the space utilization 73%.

5.1 File-Layout Sensitiveness

Because we lacked all the information necessary to recreate the layout of the file system, measurements were taken using the two different simulated file layouts explained earlier: scattered layout and consecutive layout. The results of the two different scenarios where

TABLE 9. Power Performance for Single Drive vs. RAID Levels 0, 4, and 5

Arch./ Stripe	A Trace 1	A trace 2	B Trace 1	B Trace 2
Single Disk/NA	4342.4J(100%)	30124.0J(100%)	13356.9J(100%)	14830.7J(100%)
RAID 0 /32KB	10895.5J(250%)	40698.8J(135%)	77684.6J(581%)	78661.7J(530%)
RAID 0 /inf.	15734.8J(362%)	40160.3J(133%)	51988.6J(389%)	53501.2J(360%)
RAID 5 /32KB	19741.4J(434%)	44868.6J(148%)	142014.0J(1063%)	105235.0J(709%)
RAID 5/1MB.	18879.9J(434%)	43861.3J(154%)	129144.0J(966%)	85178.3J(574%)
RAID 4 /32KB	19741.4(454%)	44781.0J(148%)	101143J(757%)	89522.9J(603%)
RAID 4 /inf.	16010.4J(368%)	44566.8J(147%)	73720.3J(551%)	63580.4J(428%)

compared in order to evaluate the relevance of the initial layout of the files in our results. The difference between both scenarios is minimal, on the average 1% for both machines with a maximum discrepancy was of 6% for Machine A and 8% for Machine B. For the rest of the paper we will report only scattered file layouts since we believe it to be more realistic.

5.2 Basic Comparisons

To provide a baseline for understanding the cost of RAID robustness, we include results where only one disk is simulated. Table 12 compares the performance of RAID level 0, RAID level 4 and RAID level 5 to a single drive for two machines and two different workloads each. These results show that without power-specific modifications RAID level 4 and RAID level 5 architectures use much more power than RAID 0 and that RAID 0 itself is expensive in power compared to a single disk with as much capacity as the array.

The difference between the RAID 0 32k stripe and RAID 5 32KB stripe numbers can be interpreted as the price for maintaining redundant data in a small stripe RAID 5 configuration. For the workloads shown here the average cost of maintaining parity is between 13% and 204% for Machine A (3/4 drives) and between 482% and 179% for Machine B (5/6 drives). The cost is lower for larger stripes and RAID 4 small stripe and lowest for RAID with an infinite stripe.

5.3 Architecture and Caching Policy Comparisons

With a fixed cache size we compared the energy consumption and performance of the different configurations. Each of the machines was simulated and two traces where run against it. With each architecture the three caching policies were evaluated (WT, WB, PB), with the exception of LSS that expects writes to be grouped in segment sizes and thus needs a write-back cache.

Figure 8 shows the energy needed to satisfy each trace for each of the architectures evaluated. We observe that, although LSS consistently consumes less power, the amount of improvement is dependent on the workload applied. It performs better as the access read/write ratio decreases but does not appear to be as sensitive to the amount of data that is written. Once more, larger stripe units yield less power consumption as we saw in the non-cached case, and RAID 4 tends to outperform RAID 5 with the exception of Machine A Trace 2 workload.

The performance of the different caching policies is not as consistent across the different workloads with the exception that deferring writes has a clear advantage to write through, especially for the write-hungry traces. The effect of PG is undetermined for RAID 4 and RAID 5; this is due to the fact that although spinups are saved in the data disks, spilling pending writes too often leads to an increased number of spinups to update parity.

The performance of each architecture and caching policy was evaluated in terms of the average response time seen by the user due to drive latency and throughput. The results are summarized in Figure 9. Once more RAID 4 out performs RAID 5 since the probabil-

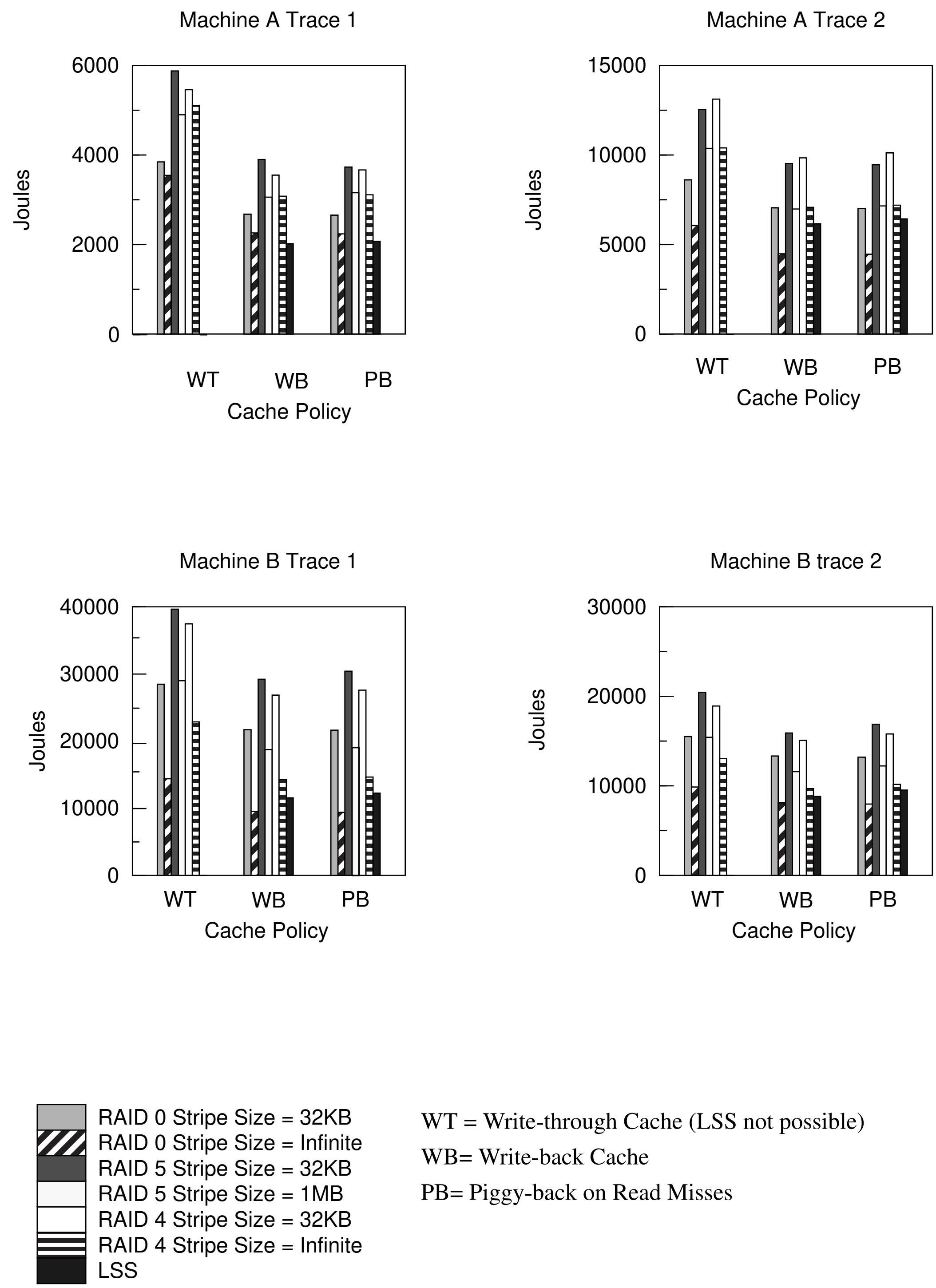

FIGURE 8. Energy Evaluation for Fixed Cache Size of 0.5 MB

ity of having to wait for a spinup in a RAID 5 array architecture is higher. LSS performance is not consistent, machine B trace 2 does not deliver better performance than RAID 4.

5.4 Impact of the Cache Size

Because of the access patterns of the applied workloads, even a small cache has a great impact on the performance of the system. A 32KB cache reduces the power consumed by the system by a factor between 100% and 400%.

We compared the energy consumed by the architectures under different cache sizes and concluded that no one architecture is favored more than the others by a larger cache size and that not much performance is gained by moving from a 256KB cache to a 512KB cache. Figure 10 shows the power consumed by all layouts tested using PB caching policy and cache sizes of 32KB, 256KB and 512KB.

Average response time scales inversely to the cache size across all architectures.

6.0 Related work

In addition to the work on Flash and NV-RAM discussed in Section 2.2, other studies have been done for optimizing power consumption in portable computers. Wu [WU94] discusses how to implement and manage a large non-volatile storage combining NV-RAM and Flash memory. Srivastava, Chandrakasan and Brodersen [Srivastava94] describe architectural technics for energy-efficient implementations of general computer systems.

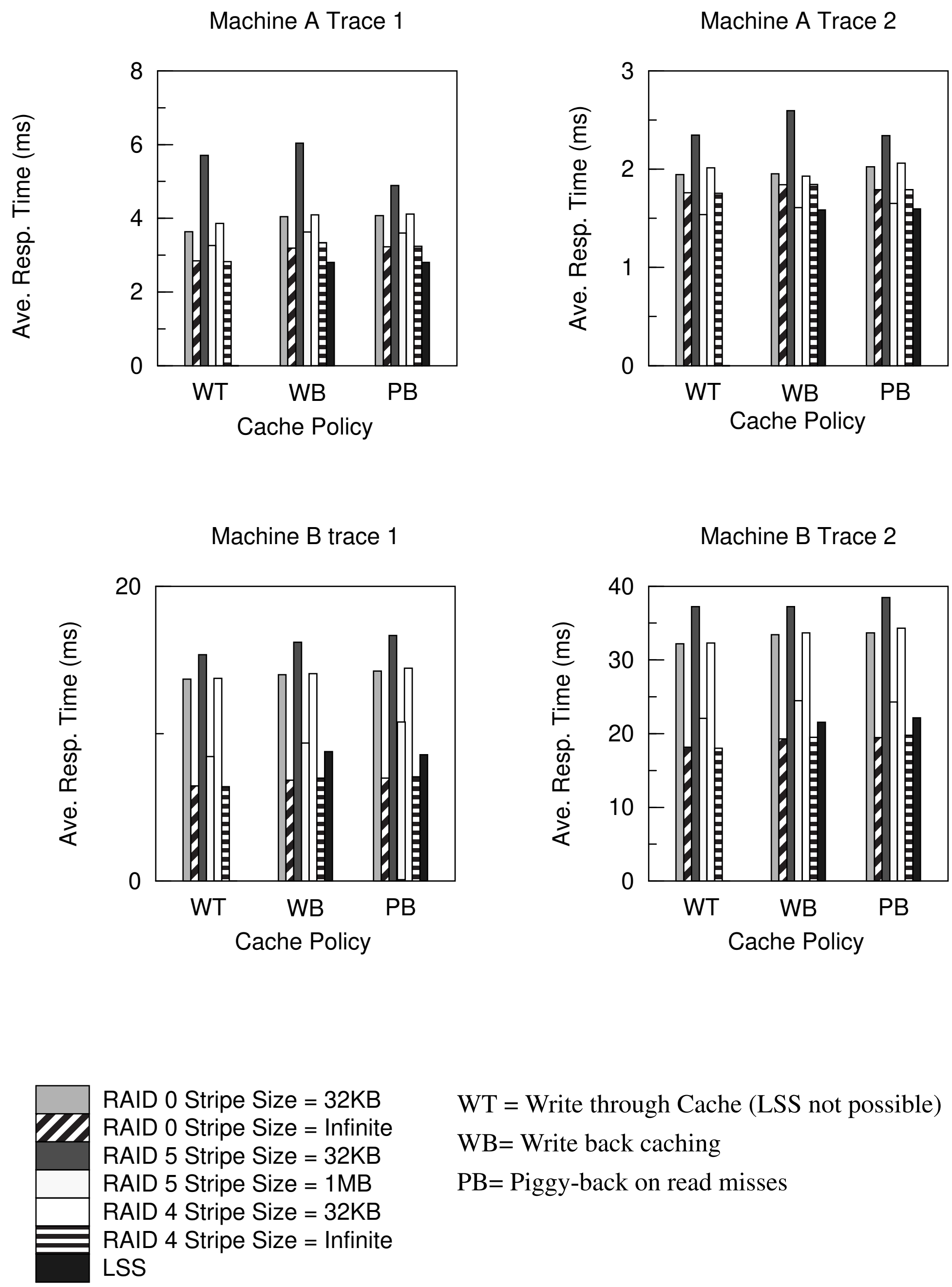

FIGURE 9. Average Response for Cache Size fixed to 0.5 MB

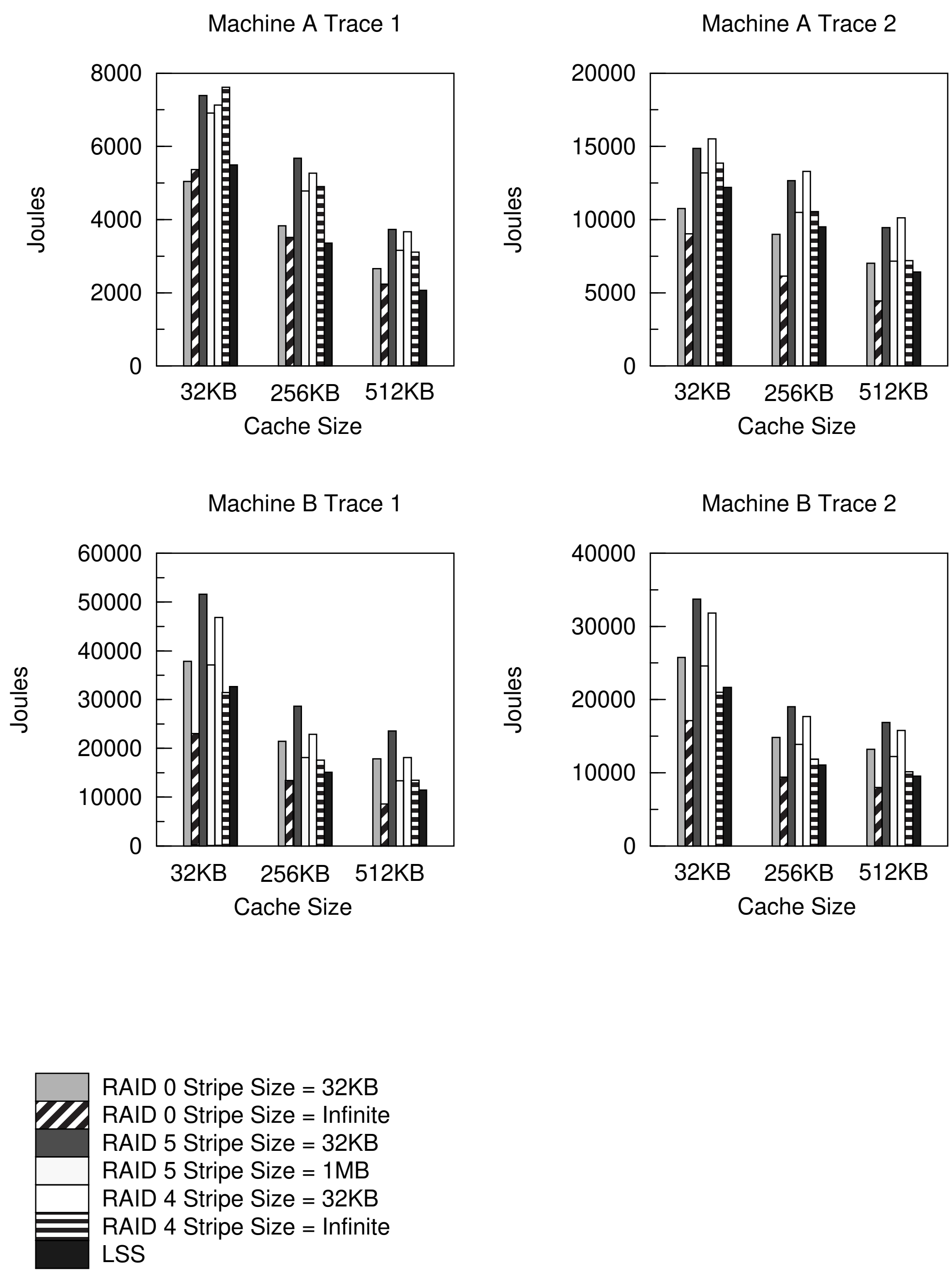

FIGURE 10. Energy Consumption Evaluation for Cache Policy Fixed to PB

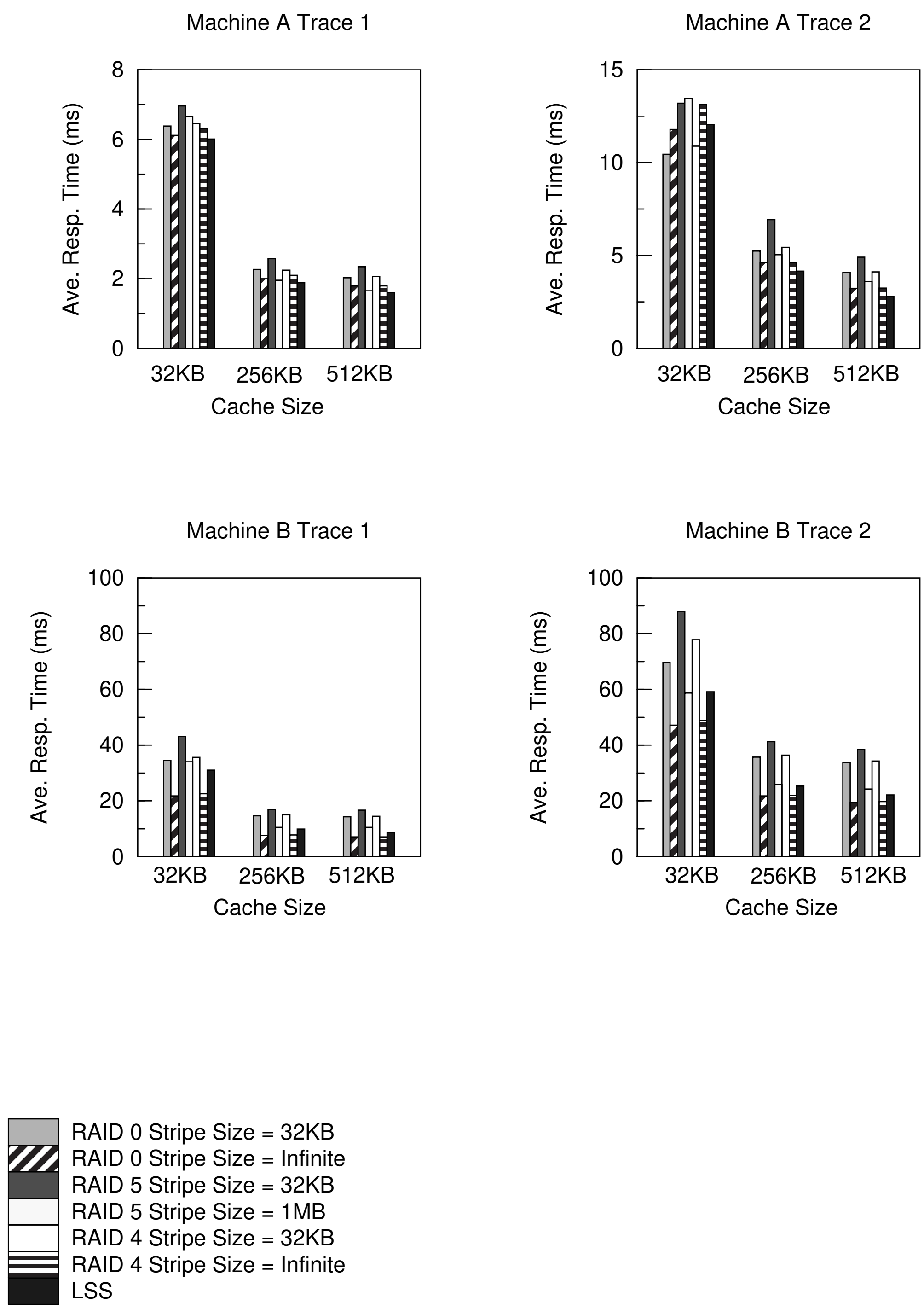

FIGURE 11. Average Response Time for Cache Policy Fixed to PB

7.0 Conclusions

In this paper we have examined the power consumption of RAID systems and developed simple improvements to standard RAID architectures to decrease the power consumption. We have compared this to a specialized architecture LSS. Although LSS out performs even the optimized standard architectures across all cache sizes and workload types we tested, it does not pose a clear win to justify the complexity and cost associated with it.

8.0 Acknowledgments

We are grateful to the Matsushita Information Technology Laboratory of Panasonic Technologies for sharing their traces, especially Fred Douglis and Hank Karth for answering all of our questions. We are grateful to Ralph Simmons and Dave McIntyre of Hewlett-Packard for the support of this project. We also thank Daniel Stodolsky for the concept idea and insightful comments. We owe much of the results of this paper to the RAIDframe team effort: Mark Holland, William Courtright, Claudson Ferreira Bornstein and Robert Findler. We also thank LeAnn M. Neal for comments on previous drafts.

9.0 Bibliography

[Brady94] J. Brady, Personal Communication, IBM 1994.

[Chen90] P. Chen and D. Pattterson, “Maximizing Performance is a Striped Disk Array,” Proceedings of International Symposium on Computer Architecture, 1990, pp. 322-331.

[Courtright95] W.V. Courtright II and G. Gibson, “RAIDframe: an Extensible Development Framework for Redundant Disk Arrays,” Work in progress 1995.

[Douglis93] F. Douglis, R. Caceres, B. Marsh, “Storage Alternatives for Mobile Computers,” Technical report MITL-TR-93-93.

[Douglis94] F. Douglis, P. Krishnan, and B. Marsh, “Thwarting the Power Hungry Disk,” Proceedings of 1994 Winter USENIX Conference, pp 293-306, San Francisco, CA, January 1994.

[Grochowski93] E. G. Grochowski and R. F. Hoyt, “Magnetic Hard Disk Form Factor Evolution”, IEEE International Magnetics Conference (INTERMAG ‘93); Stockholm, Sweden, pp 3-16 April 1993.

[Gibson92]G. Gibson, “Redundant Disk Arrays: Reliable, Parallel Secondary Storage”, MIT Press, 1992.

[Gibson95a] G. A. Gibson, D. Stodolsky, F. W. Chang, W. V. Courtright II, C. G. Demetriou, E. Ginting, M. Holland, Q. Ma, L. Neal, R. H. Patterson, J. Su, R. Youssef, J. Zelenka, “The Scotch Parallel Storage Systems,” IEEE Compcon Conference, May 5-8, 1995.

[Gibson95b] G. Gibson, “Reliable, Parallel Storage Architectures; RAID and Beyond”, Tutorial at the 1995 International Symposium on Computer Architecture, Santa Margherita, Ligure, Italy 1995.

[HP94] Hewlett Packard, ”Kittyhawk HP C3013A/Co14A Personal Storage Modules,“ Technical Reference Manual, Edition 5.1 January 1994

[IIST90] IIST Short Course, “Hard Disk Drive Design for Storage Efficiency,” Santa Clara, CA Dec. 1990.

[Kester94] L. Kester, R. Kumpf, P. Horton, and T. Anderson, “A Quantitative Analysis of Disk Drive Power Management in Portable Computers.”, Proceedings of the 1994 Winter USENIX, pp 279-291, 1994

[Klostermeyer95] W. F. Klostermeyer and K. Srinivas, “Reducing Disk Power Consumption in a Portable Computer”, ACM, Open System Review, Vol 29, No2 Apps 1995, pp. 27-32

[Lee91] E. Lee and R Katz, “Performance Consequences of Parity Placement in Disk Arrays”, Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems, 1991, pp. 190-199.

[McKusik83] M. K. McKusick, W. N. Joy, S. J. Leffler and R. S. Fabry,”A Fast File System for UNIX”, (Revised July 27. 1983), UCB:CSD-83-147.

[Menon93] J. Menon, J. Roche and J. Kasson, ”Floating Parity and Data Disk Arrays”, Journal of Parallel and Distributed Computing, 1993, pp “129-139”

[Mogi94] K. Mogi and M. Kitsuregawa, ”Dynamic Parity stripe Reorganizations for RAID5 Disk Arrays”, Proceedings of the third international conference on Parallel and Distributed Information systems, Sept. 28-30 1994.

[Patterson88] D. Pattterson, G. Gibson, and R. Katz, “A case for redundant arrays of Inexpensive Disks (RAID)”, Proceedings of the ACM Conference on Management of Data, 1988, pp. 109-116

[Rosenblum92] M. Rosenblum and J. K. Ousterhout, “The Design and Implementation of a Log-Structured File System”, Proceedings of the 13th Symposium on Operating System Principles, Asilomar, CA, October 1991, 1-15. Published as Operating Systems Review 25, 5 (October 1991). also available as Transactions on Computer Systems 10, 1 (February 1992),26-52.

[Solworth91] J.A. Solworth,C.U. Orji, ”Distorted mirrors”, Proceedings of the First International Conference on Parallel and Distributed Information Systems (Cat. No 91TH0393-4); Miami Beach, FL, USA; 4-6 Dec. 1991.

[Srivastava94] A. P. Chandrakasan, M. B. Srivastava. R. W. Brodersen.”Energy Efficient Programmable Computation,” Proceedings of the 7th International Conference on VLSI Design (IEEE), pp. 261-264, Calcutta, January 1994.

[StorageTek94] Storage Technology Corporation, “Iceberg 9200 Storage system Introduction,” Storage Technology Corporation Corporate Technical Publications, First Edition, March 1994.

[Wood93] Chris Wood and Paul Hodges, “DASD Trends: Cost Performance, and Form factor”, Proceedings of the IEEE, Vol. 81, No. 4, April 1993, pp. 573-585.

[Wu94] M. Wu, ”The Architecture of eNVy, A Non-Volatile, Main Memory Storage System”, Technical Report, Rice COMP TR94-229, April 1994.