Towards A Low Power File System

Kester Li

Computer Science Division
University of California
Berkeley, CA 94720

May 19, 1994

Abstract

Without power management, disks on portable computers consume 20-30% of the system's power. Previous papers showed that spinning down a 1 watt drive after 3-5 minutes of inactivity reduces the average power consumption to 0.5 watts with a slight performance penalty. We explored the addition of an infinite non-prefetch cache to filter disk traffic and showed it to be ineffective at reducing disk power consumption or improving performance. Modifying the cache to perform whole-file prefetching gave the next step of improvement by reducing the disk power consumption to 0.1 watts while retaining current performance levels. The traditional file system performance model focuses on cache hit rate as the performance determinant. This paper introduces a new performance model for systems that spindown disks and shows that performance is no longer a function of hit rates, but rather of read-interrequest times and disk spindown delays. Drive durability is an issue for low power file systems because of the friction induced wear during disk spinup. Periodic writes, which result in periodic disk spinups, will very likely lead to premature drive failure. We show that approximately 1 MB of flash memory used as a write buffer eliminates the need to spinup the disk for writes.

1 Introduction

The power limited environment of portable computers has fueled much work in the area of constructing low power components. Low power displays, processors, memories, and disks have been developed to extend battery life. Of these components, the disk is the only device whose transition from low power mode to high power mode occurs at mechanical speeds. Such large latencies cannot be hidden at the hardware level and limit the usefulness of low power drives. To optimally leverage the low power mode of the disk, operating systems must be designed to balance between the application's need for low latency file access and the system's need to reduce disk power consumption. Several previous papers have examined various aspects of disk spindown and the use of flash memory as power reduction techniques. This paper is another step towards the design of a reasonable cost, high performance, low power file system for portable computers.

Previous work has made some progress in designing low power file systems. [Li94] and [Douglis94] examined ways to reduce disk power consumption by spinning down the disk at various granularities. Coarse-grained spindown, spindown after 3-5 minutes of inactivity, saved half of the potential energy savings in exchange for a small performance penalty. Fine-grained spindown, spindown after 10-20 seconds of inactivity, approached the minimum energy consumption but resulted in large performance sacrifices. [Marsh94] examined the use of a nonvolatile, low power memory store as a transparent layer above the disk which caches recently referenced disk blocks. In that study, they assume a large warm cache, a stable stochastic working set, and prescient disk spindown and conclude that a disk can be virtually power free while providing good file system performance. This paper combines, refines, and builds upon the previous work. It reexamines the assumptions in [Marsh94] and evaluates the power-performance tradeoff using representative DOS traces. It shows that the three necessary ingredients for a low power file system are:

- Fine-grained disk spindown
- Whole-file prefetching cache
- 8-16 MB of low power, low read latency memory

For systems that spindown disks, the measure of performance is the absolute number of disk spinups per time period. This quantity is a function of the read-interrequest time distribution and the disk spindown delay. Using the new performance measure, non-prefetching caches, even of infinite size, do not significantly improve performance or reduce disk power consumption. Installing a whole-file prefetch cache approximating the trace working set size gives the next step in improvement beyond coarse-grained spindown. Preserving current performance levels, disk drive power consumption can be reduced from 0.5 watts to 0.1 watts. Drive durability is an issue for low power systems because of the friction induced wear during disk spinup. Periodic writes, which result in periodic spinups, could easily lead to premature drive failure. Buffering writes in nonvolatile flash memory eliminates the need to spinup the disk for writes.

The rest of this paper is organized as follows: Section 2 describes the hardware changes that enable the construction of low power file systems. Section 3 describes the methodology of this paper. Section 4 gives a statistical characterization of the traces. Section 5 introduces a new performance model for low power file systems. Section 6 reviews the results of [Li94] and [Douglis94] using the traces of this study. Section 7 examines the need for large, whole-file prefetching caches to eliminate the power-performance tradeoff. And finally, Section 8 studies the feasibility of caching writes in nonvolatile flash memory to reduce the disk durability impact of spinning up for periodic writes.

2 Hardware

Hardware technologies have evolved and emerged to satisfy the low power market consisting of portable computers and embedded systems. This section describes two such evolutions that aid in the construction of a low power file system.

2.1 Sleeping Disks

The recent explosion in the portable computer market has motivated disk drive manufacturers to develop a new breed of drives especially for the portable environment. In addition to high shock tolerances, reduced physical volume, and smaller weights, these drives provide a method for operating systems to spindown the physical disk platter. This capability is significant in a power limited environment because most of the energy consumed by a disk is expended in rotating the physical platter.

Figure 1 describes the operational states of a typical low power drive and its state transitions. "Off" mode is when the disk consumes no energy and is incapable of performing any functions except powerup. "Sleep" mode is when the electrical components are powered but the physical disk platter is not spinning. "Spin" mode is characterized by a spinning disk, but the absence of disk activity. A disk drive in a desktop computer spends most of its time in this mode. "Active" mode is when the disk platter is spinning and the disk head is actively reading or writing the disk.

Hardware characteristics determine the tradeoff between sleep mode and spin mode. Table 1 gives the power consumption and transition times for the Maxtor MXL-105-III drive, a typical low power drive. The difference in power consumption between spin mode and sleep mode is quite sizable. Spin mode consumes 1 watt while sleep mode consumes just 25 milliwatts. This difference is strong motivation to spindown the disk. Unfortunately, the difference in latency into active mode between spin mode and sleep mode is also quite sizable. The seek latency is on the order of 10 milliseconds while the spinup latency is 2 seconds. The hardware characteristics indicate that the primary tradeoff for spinning down disks is between lower power and higher latency. The central question for operating system designers to examine is when to take the spindown transition (the spindown policy) to provide a good balance between power and performance. A very conservative policy may undo the work of the hardware designers by not spinning down the disk. At the other extreme, a very aggressive spindown policy may highlight the two orders of magnitude latency difference and result in huge performance sacrifices.

Drive durability is another area of concern for systems that spindown disks. The number of spinups a contact start/stop drive such as the Maxtor MXL-105-III can tolerate before head wear becomes a problem is 40,000. Low power file systems that produce large numbers of disk spinups for fetching data or committing writes could easily result in drive failure. Limiting the total number of spinups is an important goal.

2.2 Flash Memory

Flash memory, an in-system programmable variant of an EEPROM, has several characteristics which make it useful for constructing low power file systems. Among these are low cost, low power, low read latencies, and non-

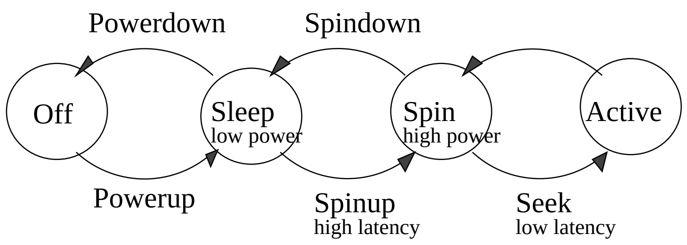

Figure 1: **Disk Drive State Diagram.** Low power drives have four modes of operation. The low power sleep mode is the recent addition.

Mode	Power (watts)	Transition	Latency (seconds)	Power (watts)
Off	0.0	Powerup	0.5	0.025
Sleep	0.025	Spinup	2.0	3.0
Spin	1.0	Seek	0.009	1.95
Active	1.95	Spindown	1.0	0.025
		Powerdown	0.5	N/A

Table 1: **Maxtor MXL-105 III.** This table gives the power consumption and average transition times of the major disk modes and transitions for a typical low power drive. Noteworthy points are the large power differences between sleep mode and spin mode and also the large latency differences between seek and spinup.

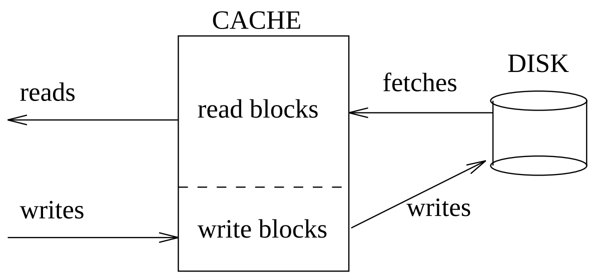

Figure 2: **File system cache model.** The simulated cache uses 4 KB block sizes and LRU replacement. Cache blocks were tagged "read" if they were fetched from disk and "write" if they were written by applications.

volatility. Although flash memory is presently quite expensive, the predicted growth in demand will promote larger economies of scale, quicken the pace of technology maturation, and result in sharp price drops. According to [Dipert93], flash memory is estimated to sell for $7.20 per megabyte by the mid-1990's. This allows for the possibility of adding 8-16 MB of flash memory to portable computers at a moderate cost. Flash memory is also a low power memory store. The power consumption while reading flash memory is 325 milliwatts. When flash memory is inactive, it can be placed in either standby mode or sleep mode which consumes 315 microwatts and 15 microwatts respectively [Intel]. Large amounts of flash memory can be added to a system without increasing power consumption. The read latency of flash memory is within a factor of three of DRAM speeds which makes it fast enough to be used as an executable read-only store. This study only requires that flash read latencies be comparable or smaller than read latencies for spinning disks, which it far exceeds. And lastly, flash memory is nonvolatile which implies it can be used to store writes without sacrificing data reliability.

Two limits of flash memory should be mentioned as well. First, flash memory is limited in the number of erasures each cell can tolerate. Current flash memory technologies can tolerate only 100,000 erasures which makes flash durability an issue. Second, the erasure-program step to write flash memory requires 9-300 microseconds per byte depending on implementation technology [Dipert93]. The slow write speeds may force buffering data in DRAM for short periods of time.

3 Methodology

To study the feasibility of low power file systems in the real world, we instrumented and collected system call traces on DOS machines. The traces were run through a discrete event simulator that simulated the Maxtor disk drive described in Section 2, various disk caches, and also portions of the DOS state. The cache model used in this study is shown in Figure 2. The unit of caching is 4 kilobyte blocks and LRU replacement occurs on cache overflow. Cache blocks were tagged either "read" or "write" depending on the block's origin. Read blocks refer to cache blocks whose data was fetched from disk while write blocks refer to cache blocks whose data was written by applications. The cache is assumed to be backed by nonvolatile flash memory which suggests a warm cache model. However a cold cache model was simulated. The reasoning behind this decision was to ensure high performance across shifts in working set or when large files flush the cache. During these times, the cache behaves like a cold cache.

The simulations in this paper assume that no disk spin-ups occur to satisfy name and attribute requests. We assumed the presence of a name-attribute cache in the file system that stores all recently accessed meta-data for active directories which are typically few in number and exhibit tremendous locality. This assumption was made because the traces do not contain complete information on meta-data accesses. We speculate that this assumption is not far from the truth as these are small personal file systems whose entire meta-data can reasonablely be stored in a small amount of flash memory.

4 Traces

This section describes the traces used in this study. It presents a description of the trace source, the motivation for eliminating paging traffic, and a statistical characterization of the traces.

4.1 Trace Source

The traces in this study were collected on two personal computers with 8 and 16 MB of DRAM while running Windows 3.1. These machines were traced while being used as the primary work platform for two Berkeley computer science graduate students running office applications. An attempt was made to make this trace set representative of current portable computer usage. All traces in this study were collected while running Windows hosted applications. The main applications were FrameMaker, PowerPoint, Word, Excel, Procomm, Terminal, Notepad, and various Central Point utilities. It should be noted that the traces of this study are different from the traces used in [Li94] which were collected at a student computing laboratory primarily running DOS hosted applications. Although those traces were more user diverse, we believe that they were not representative of expert users or current portable computer usage. The trace set of this study attempts to correct both of these shortcomings.

Fourteen traces were collected and analyzed for this study. Due to differences in trace lengths and the need to simulate entire traces to collect accurate statistics, direct comparisons of large numbers of traces was not attempted. Instead, the traces were separated by their characteristics into light and heavy workloads and a representative was chosen among them for detailed analysis.

4.2 Paging

A potentially controversial decision was made to eliminate all swap file activity from the trace analysis. The reasoning behind this decision comes from the observation that (in the absence of a permanent swap file) Windows 3.1 blindly constructs a several megabyte swap file on startup, irregardless of whether such a swap file is necessary. Empirical analysis of the traces show that even under heavy duress with three large applications running concurrently, not a single byte was ever read out of the swap file in any of the traces. However, many megabytes were written to the swap files during the startup process which were promptly deleted on shutdown. Only one trace exhibited further swap file activity beyond the initial construction salvo. This particular trace wrote many additional megabytes to the swap file, but never read a single byte back into memory. This behavior could be indicative of a large memory leak in one of the applications or just poor system design.

The controversial part of the decision to eliminate swap file activity stems from the large DRAM store present on the traced machines. Portable computers today typically come equipped with only 4 megabytes of DRAM which may be insufficient memory to run under heavy workload. In this case, virtual memory must be enabled and the swap file activity taken into consideration. However, since this paper considers adding flash memory as a write buffer, this would be tantamount to paging into flash. Only upon exhaustion of both DRAM and flash in a low power system will paging to disk occur and we speculate that users will more likely reduce resource demands than suffer the performance and energy cost of paging to disk.

4.3 Read Characteristics

The read characteristics of the traces can be separated into light and heavy workloads. The light workload is characterized by short bursts of disk activity, mostly during application startup, followed by long periods of inactivity (see Figure 3. These traces usually correspond to self-contained applications such as PowerPoint, Word, or Notepad which only generate read requests when they need access to large data sets such as the dictionary file or running embedded applications such as Equation Editor. Relating the traces in this study with those of [Li94], all of the traces used in [Li94] would be classified under light workload in the present study. The heavy workload is characterized by uniformly distributed read accesses, with bursts of activity during application startup. These traces were mostly collected while running FrameMaker.

The separation between light and heavy read workloads is primarily a function of application design. The application's structural organization determines the character-

Description	**Light Workload**	**Heavy Workload**
Application	PowerPoint	FrameMaker
Trace length	1.6 hours	3.5 hours
Bytes read	5.53 MB	10.76 MB
Bytes written	275.87 KB	14.39 MB
Bytes written and overwritten	0.82 KB	0.23 MB
Bytes written and deleted	58.79 KB	13.57 MB
Bytes deleted	58.80 KB	13.83 MB
Working set	4.77 MB	9.71 MB
Read working set	4.52 MB	8.87 MB
Write working set	0.25 MB	1.15 MB
Prefetch set	5.86 MB	13.57 MB

Table 2: **Trace statistics.** This table gives the read, write, deletion, and working set statistics for the light and heavy workloads.

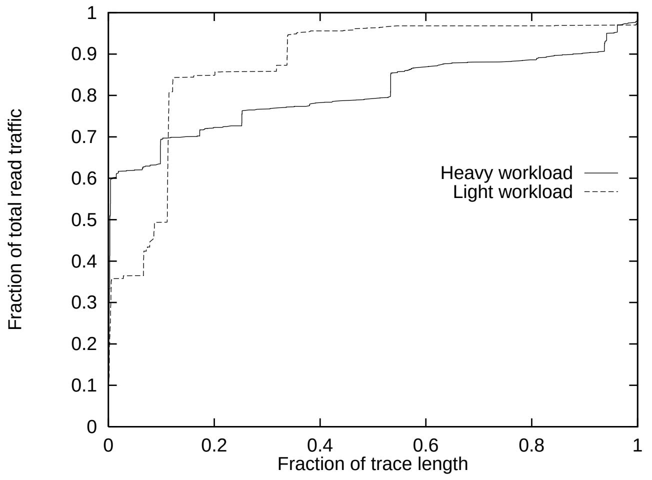

Figure 3: **Read traffic.** The light workload trace has bursts of read activity followed by periods of inactivity, while the heavy workload trace has a more steady stream of read activity.

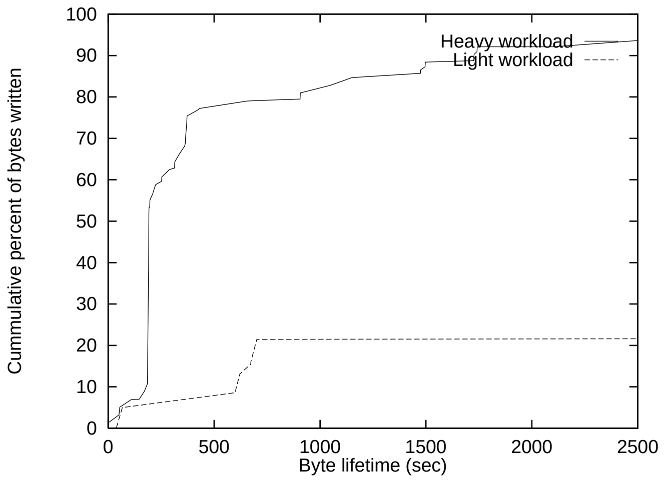

Figure 4: **Byte lifetimes.** This figure shows that most bytes that die within a session die within ten minutes. Spinning up the disk to commit these condemned bytes unnecessarily sacrifices drive durability.

istics of the resulting traces. Design decisions such as using load-on-demand or discardable segments or extensive use of dynamic link libraries have a significant impact on trace contents. Also, applications designed as a collection of independent executables using temporary files for inter-module communication leave distinguishing marks on the traces.

4.4 Write Characteristics

The write characteristics of the traces were mostly periodic and uniform. This is the expected behavior as users go through the edit-save cycle, sometimes enabling the auto-save feature of applications to perform periodic writes to nonvolatile storage. The density of write events again varies with the application and user. The representative traces were selected to also expose light and heavy write workloads. The light workload wrote 276 KB of data and the heavy workload, running FrameMaker with a 3 minute auto-save, generated 14.4 MB of data.

4.5 Deletion Characteristics

Table 2 shows the deletion characteristics of the two sample traces. Two conclusions can be drawn from this data. First, most bytes are killed through the delete system call with very few bytes getting overwritten. Second, most bytes that are deleted in a session were written in that session. The light workload and heavy workload traces differed in the fraction of bytes that were written in the session and subsequently deleted. A large fraction of the bytes in the light workload survived past the end of the session. This is attributable to its large write-delete period and the short trace length which combine to over-represent the last data save. The heavy workload trace had the opposite behavior with most bytes dying within the session. This is attributable to its small write-delete period. Figure 4 shows the byte lifetimes of writes in each trace. Bytes not deleted within the session are assumed to live forever. Most bytes that die within a session live for less than 10 minutes. Byte lifetimes under DOS are greater than byte lifetimes under the Sprite operating system as measured in [Baker91] (corrected in [Hartman93]).

4.6 Working Set

The working set size is the number of unique disk blocks referenced by the trace. The light workload trace had a working set size of 4.8 MB and the heavy workload trace had a working set size of 9.7 MB. The cache model further decomposed the working set into read and write working sets according to the block's origin. The statistics show that most bytes in the working set are fetched from disk as opposed to written by applications. The prefetch set size is the number of unique disk blocks referenced by a whole-file prefetching cache. The 20-40% difference in sizes between the working set and the prefetch set indicates that 20-40% of prefetched bytes are never referenced.

5 Performance Model

This section develops a new performance model for a low power file system. The performance in normal file systems with a disk cache and a spinning disk is measured by the average read access time given by the formula:

```
performance = average read access time
            = hit ratio * cache access time
            + (1-hit ratio) * disk access time
```

This performance model works well when the disk access latency is sufficiently small that users do not notice individual disk accesses. This assumption, however, is not true in a low power system where the disk is put to sleep and disk access latency increases to 2 seconds. While the disk is sleeping, the user will notice every cache miss. Performance with a sleeping disk is no longer determined by the fraction of read events filtered by the cache, but rather by the absolute number of disk spinups. Assuming only read events cause disk spinups and the system employs a fixed delay spindown policy, performance in a low power system should be stated as:

```
performance = #(disk spinups)
            = #(read-interreq > spindown delay)
```

Using this model, performance in a low power system is a function of the read-interrequest times and the spindown delay, not the cache hit ratio. Performance can be improved by increasing the spindown delay so that fewer read-interrequest times exceed it. This is the reasoning behind choosing a coarse-grained spindown policy. Alternatively, performance can be improved by modifying the

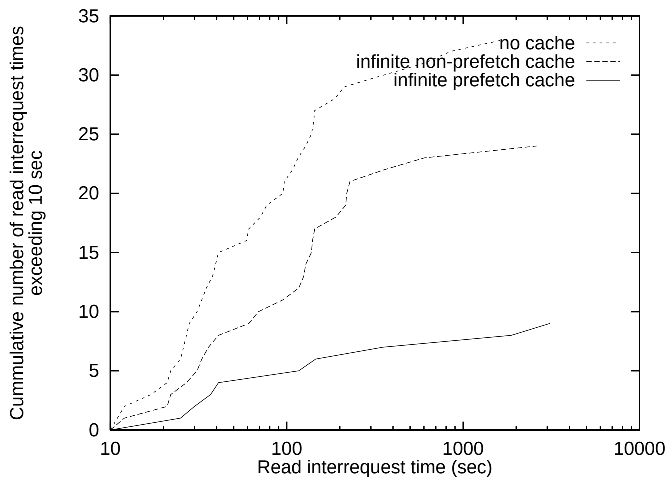

Figure 5: **(Light workload) Read interrequest times.** The non-prefetch cache does not significantly reduce the number of large read-interrequest times which explains its ineffectiveness at improving performance. The prefetch cache does reduce the number of large read-interrequest times.

read distribution so that fewer read-interrequest times exceed the spindown delay. [Li94] attempted to modify the read distribution by using non-prefetching caches to filter read traffic and found they were ineffective at improving performance. Figure 5 explains this result. The non-prefetch cache does not significantly reduce the number of large read-interrequest times over no caching. Performance improvements can only come from a cache which reduces the number of large read-interrequest times, such as a whole-file prefetching cache.

6 Spindown Policies

[Li94] and [Douglis94] studied the tradeoffs associated with spinning down a disk. Manufacturer's recommendations varied widely as to when disks should be spun down even though their drives have similar characteristics. Some recommend fine-grained spindown of the disk after 10-20 seconds of inactivity while others recommend coarse-grained spindown of the disk after 3-5 minutes of inactivity. These large differences are a cause of concern, possibly reflecting a lack of quantitative data or methods to evaluate the tradeoff.

6.1 Power-Performance Tradeoff

This subsection repeats the analysis of [Li94] and [Douglis94] on the traces of this study, with one exception. Instead of using a 1 MB non-prefetch cache, this analysis will present the best case non-prefetch cache by making it infinite. Figures 6 and 7 present all of the results of this study. Focusing on the infinite non-prefetch cache curves, we can rederive the conclusions of [Li94]. For the light workload trace, the energy consumption per hour for no spindown is 3,600 joules, for coarse-grained spindown is 1,800 joules and for fine-grained spindown is 400 joules. For the heavy workload trace, the energy consumptions are 3,600 joules, 1,400 joules, and 300 joules respectively. Half of the potential energy savings comes from spinning down the disk after 3-5 minutes of inactivity and the other half comes from spinning down after 10 seconds of inactivity. These numbers show slightly larger energy savings for coarse-grained spindown due to the presence of an infinite cache.

The middle graphs consider the other half of the tradeoff by plotting the number of synchronous spinups of the disk against the spindown delay. Synchronous spinups are those spinups of the disk which result in the user having to pause for the disk. For the light workload trace, the number of spinup pauses per hour for coarse-grained spindown is 2 and for fine-grained spindown is 15. For the heavy workload trace, the number of spinup pauses per hour are 3 and 9. Decreasing the spindown delay results in a dramatic increase in spinup pauses as the disk is spun down between many small cracks of inactivity. The performance penalty for coarse-grained spindown is likely to be acceptable, however the performance impact of going to a fine-grained spindown policy may be unacceptable for many users.

The bottom graphs make the power-performance tradeoff explicit. Energy consumption and disk spinup pauses can be freely exchanged.

6.2 Prescient Spindown

[Douglis94] originated the notion of a prescient spindown policy. The idea is that remaining in spin mode for a short period of time can actually consume less energy than spinning down, sleeping and spinning up the disk again. This is true because the energy cost of spinning up a disk is greater than the energy cost of just spinning the disk. Thus sleeping for very short periods of time can actually increase the disk energy consumption. The minimum period of time a disk must sleep before sleeping becomes beneficial varies with drives, but typically is on the order of a few seconds. The Maxtor MXL-105 III drive simulated in [Li94] has a minimum period of 6 seconds and the Hewlett-Packard Kittyhawk C3014A drive simulated in [Douglis94] has a minimum period of 5 seconds.

The prescient spindown looks into the future to optimally decide when to spindown the disk. If the next disk event occurs greater than 5 or 6 seconds into the future, then sleeping minimizes energy consumption. If, on the other hand, the next disk event is less than 5 or 6 seconds in the future, then keeping the disk in spin mode in anticipation of that request minimizes energy consumption. Running the prescient policy against a given trace yields the theoretical lower bound on the amount of energy required to satisfy that trace while varying the spindown policy. This quantity is useful to evaluate the remaining

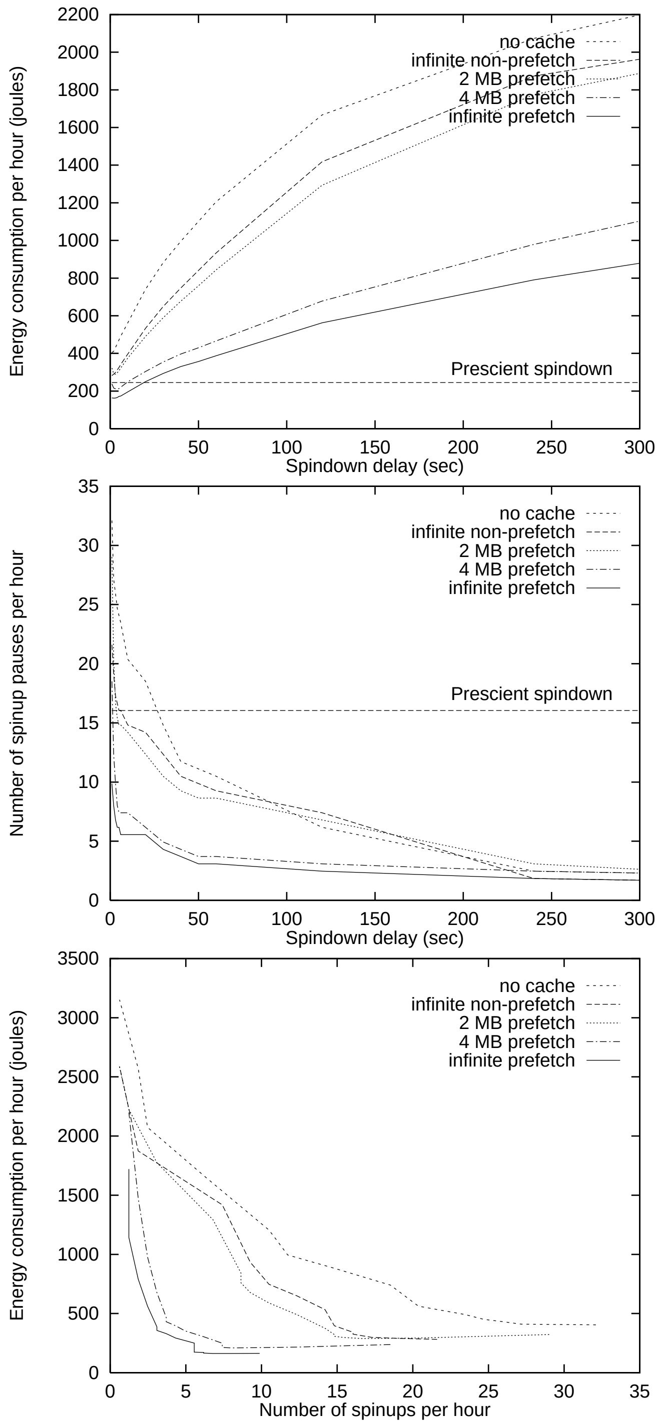

Figure 6: **(Light workload)** Power and performance characteristics under various disk spindown and cache policies.

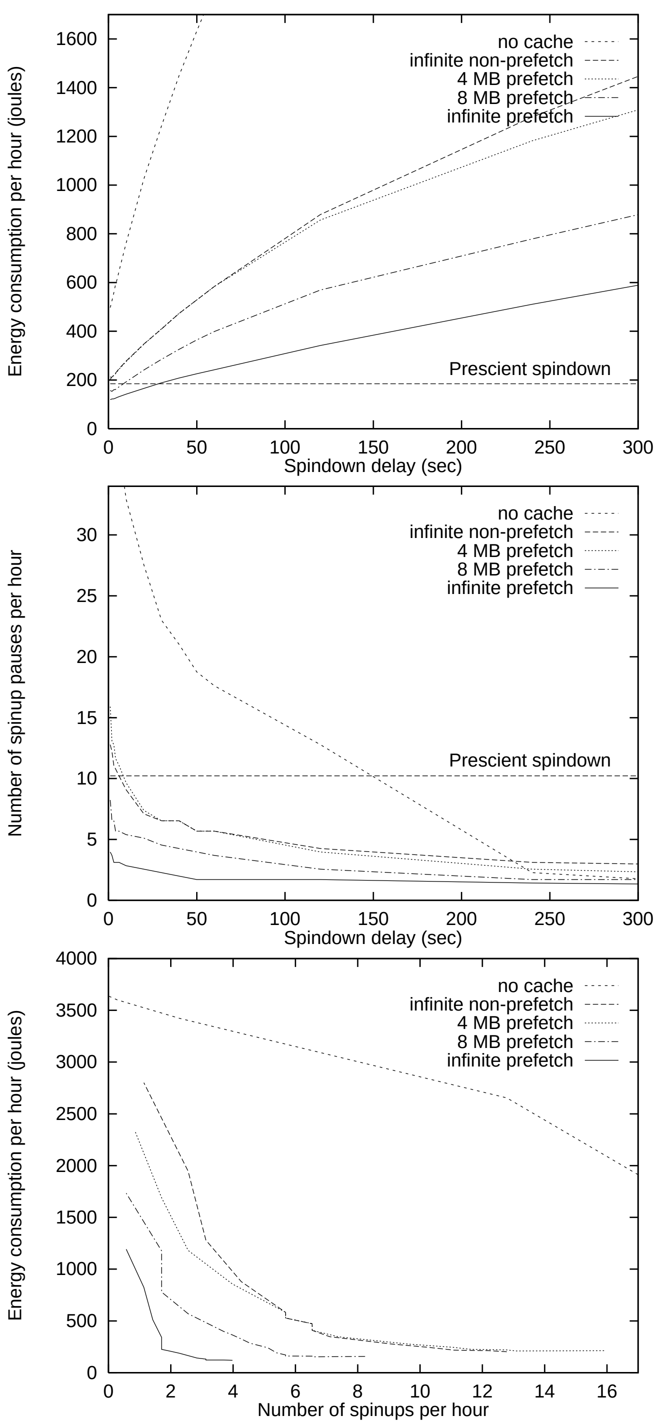

Figure 7: **(Heavy workload)** Power and performance characteristics under various disk spindown and cache policies.

improvement possible in choosing a different spindown policy. The results of [Li94] and [Douglis94] were similar and both indicate that a fine-grained spindown policy can get close to the minimum energy consumption. Figures 6 and 7 echo this result. The horizontal lines give the power consumption and performance for the prescient spindown policy against the infinite non-prefetch cache load. As the spindown delay approaches several seconds, the energy consumption approaches that of the prescient spindown policy. However, the performance graphs show that prescient spindown suffers from the same performance degradation as fine-grained spindown.

7 Whole-File Prefetch Cache

Previous sections gave hints of the ineffectiveness of non-prefetching caches. This section examines whole-file prefetch caches as a method of eliminating the power-performance tradeoff. Two styles of whole-file prefetch caches are possible. The first style is to prefetch all blocks of a file on every open. This policy gives preference to the most recently opened files, possibly flushing blocks from older files. The second style is to prefetch the file only if none of the file's blocks are present in the cache. This policy in conjunction with LRU replacement attempts to cache only the working set of each file. As was shown in Section 4, 20-40% of prefetched bytes are never referenced. By prefetching on the first open and not prefetching thereafter, this policy attempts to eliminate the prefetch overhead. Figures 6 and 7 show the results of the second style of prefetching.

The goal of this paper is to design a high performance, low power file system. This goal can be restated in the bottom graphs as "the closer we get to the origin, the better." Under this criterion, the infinite prefetch cache is the clear winner. As an example of the improvement of prefetching caches, we will make the assumption that current performance levels with 2 or 3 spinups per hour is acceptable and examine the resulting energy difference. Filtering the light workload trace with an infinite non-prefetch cache, the disk consumes 1900 joules/hour (0.53 watts) and spins down after 4 minutes of inactivity. Filtering with an infinite prefetch cache, the disk consumes just 350 joules/hour (0.10 watts) and spins down after 50 seconds of inactivity. The prefetch cache eliminates the number of large read inter-request times which allows us to spin down the disk at finer granuarities to achieve a fixed performance level. The smaller spindown delays along with further reductions in the number of disk requests result in a savings of 1550 joules/hour. The same is true for the heavy workload trace. Filtering the heavy workload trace with a non-prefetch cache, the disk consumes 1,300 joules/hour (0.36 watts) and spins down after 5 minutes of inactivity. Filtering with a prefetch cache, the disk consumes just 150 joules/hour (0.04 watts) and spins down after 10 seconds of inactivity which results in a difference of 1,150 joules/hour.

Infinite caches are impractical, so we must examine finite cache sizes. Prefetch cache sizes are shown at approximately the working set size of the trace and at half of the working set size. The prefetch cache performs best when its size approximates the working set size. As the cache size is reduced below the working set size, the prefetch cache approaches the power and performance characteristics of the non-prefetch cache. In order for prefetching to be useful, the cache size must approximate the working set size of the trace which is typically on the order of 8-16 MB. This brings up the issue of how to implement such a large cache. Powering 8 MB of DRAM for a prefetch cache defeats its purpose as 8 MB of DRAM consumes as much power as a spinning disk. Prefetching is saved by low power flash memory whose size is only limited by cost. This establishes both the benefit and feasibility of whole-file prefetch caches.

8 Write Buffer

The short byte lifetimes shown in Section 4 suggest that spinning up the disk to commit condemned bytes is a wasteful activity. The presence of periodic writes such as the 3 minute auto-save in the heavy workload trace along with drive wear during disk spinups can easily lead to premature drive failure if the disk is spun up to commit every write. [Li94] attempted to exploit the short byte lifetimes by delaying writes in the cache for 30 seconds. However, the byte lifetime analysis shows that bytes typically live longer than 30 seconds so a significant number of disk spinups still occur to commit writes. Buffering writes in flash memory is the obvious solution to eliminating these spinups. Table 2 shows that the entire write working set of both sample traces can easily be accommodated in a small amout of flash memory. The write working set size for the light workload trace was 250 KB and the heavy workload trace was 1.15 MB. With a little more than one megabyte of flash memory, neither trace would require any special disk spinups to commit writes to nonvolatile store.

However, all is not well as current flash memories also have durability limits and the short byte lifetimes may create durability issues for flash. The problem is not nearly as severe as for the disk, though. Each flash cell can tolerate 100,000 erasures and by using a write policy which uniformly degrades the cells in the flash store, flash durability is really the product of both the number of tolerable erasures per cell and the size of the flash memory. Unlike the disk, flash durability can be increased by increasing the amount of flash memory. As an example, we can compute a very pessimistic approximation of flash memory life for the heavy workload trace with 8 MB of flash memory. The heavy workload trace generated around 4 MB of write traffic per hour and assuming a 2,000 hour work year, this trace would generate 8 GB

	DRAM	Flash	Sleeping Disk
Size	Small	Small	Large
Persistence	Volatile	Nonvolatile	Nonvolatile
Power	High	Low	Low
Read latency	Low	Low	High
Write latency	Low	High	High
Durability	None	Slight	Severe

Table 3: **Memory characteristics.** Future portable computers will contain DRAM, flash memory and a sleeping disk. The memory characteristics plus the moderate working set sizes suggest the proper use for each memory type in a low power file system.

of write traffic per work year. 8 MB of flash memory with each cell able to tolerate 100,000 erasures permits 800 GB of writes before flash wearout. Dividing 800 GB by 8 GB/work year approximates the flash memory life at 100 work years. This estimate could be off by one order of magnitude due to small writes and other factors, however it does indicate that flash durability is unlikely to become an impediment in using flash as a write buffer.

Flash memory also has the problem of slow write speeds. As suggested in [Marsh94], this issue can be solved by doing background erasure to keep a clean block pool and buffering writes in a small DRAM cache. In the light workload trace, no 10 second region generated more than 72 KB of write traffic and in the heavy workload trace, no 10 second region generated more than 330 KB of write traffic. Flash programming rates may not be able to keep up with the peak write thruput of the heavy workload trace, but DRAM buffering can be used to rate match the two.

9 Big Picture

The results of this paper can be put into perspective by examining the characteristics of the three memory types in future portable computers (see Table 3). All three memories are necessary to construct a low power file system. A small amount of the high power, low write latency DRAM is required to buffer writes into flash. The remainder of the low power file system is composed of flash memory and a sleeping disk, which differ in two primary respects. First, the cost per megabyte of disk is at least a factor of ten smaller than for flash memory. If flash were cost competitive with disk, then we could easily achieve a zero power, zero spinup file system by replacing the disk with flash. However, the price difference dictates a small flash store and a large disk store. Second, flash memory has low read latencies whereas a sleeping disk has very high read latencies. This suggests the proper use of flash memory is as a file cache. Trace analysis shows that typical working set sizes for personal computers are around 8-16 MB. By installing 8-16 MB of flash memory and using file open events as hints of future read accesses, we can move the active files into the small, fast flash memory. Under the new performance model, this technique eliminates many compulsory misses which would otherwise have caused synchronous disk spinups and thereby hide the latency of a sleeping disk. Durability is a third difference between flash memory and a sleeping disk. Periodic writes to flash memory may create a slight durability problem which can be mitigated by increasing the flash store. Periodic writes to a sleeping disk will definitely lead to premature drive failure. This problem may be eliminated by the development of non-contact start/stop drives such as dynamic head loading drives [Parrish92] which can increase drive start/stop ratings to one million. However, the current trend is towards higher bit densities and not higher start/stop ratings, so the durability problem is likely to persist. At least for current contact start/stop drives, flash memory can be used to enhance drive durability by buffering writes.

10 Conclusion

The previously published papers, [Li94], [Douglis94] and [Marsh94], looked into techniques for reducing file system power consumption in portable computers. Each paper develops one aspect of a low power file system such as disk spindown or large flash caches. However, the sum of these papers do not add up to a high performance, low power file system under realistic conditions. This paper combines the previous work and adds the missing ingredient, a whole-file prefetching cache. The necessary parts for a low power file system are:

- Fine-grained disk spindown
- Whole-file prefetching cache
- 8-16 MB of low power, low read latency memory

Any system missing one of these three will trade-off power and performance. Without fine-grained disk spindown, file system power consumption will be high with good performance. Without a whole-file prefetching cache, the choice is between coarse-grained spindown with high power consumption and good performance or fine-grained spindown with low power consumption and poor performance. Without a memory store approximating the working set size, the prefetching cache exhibits characteristics of a non-prefetching cache. Without a low-power memory, the cache may consume as much power as a spinning disk. Without a low read latency for the cache, performance will suffer. The sum of the three form a low power file system.

11 Postscript

Upon further reflection and a gentle nudge from John Ousterhout, the true benefit of whole-file prefetching is likely to come from undoing Window's demand segmentation model. Because Windows operates under a scarce memory assumption, Windows hosted applications are designed to load segments and resources only at the time of first reference. This likely results in temporally scattered compulsory misses which whole-file prefetching eliminates. Prefetching just the segments and resources of opened executables and dynamic link libraries into flash memory seems a likely refinement to whole-file prefetching that will yield similar benefits. This would be a good starting point for others who wish to continue this investigation.

12 Acknowledgements

I would like to thank Tom Anderson for his advice, Roger Kumpf and Paul Horton for working on [Li94], Lok Tin Liu for volunteering to trace his computer and, lastly, Paul Hilfinger and John Ousterhout for the inspiration.

References

[Baker91] M. Baker, J. Hartman, M. Kupfer, K. Shirriff and J. Ousterhout, "Measurements of a Distributed File System" *Proceedings of the 13th Symposium on Operating System Principles*, Oct. 1991, 198-212

[Baker92] M. Baker, S. Asami, E. Deprit, J. Ousterhout and M. Seltzer, "Non-volatile memory for fast, reliable file systems" *Proceedings of the Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, Oct. 1992, 10-22

[Compaq] Compaq User's Manual, 1993

[Dipert93] Brian Dipert and Lou Herbert, "Flash memory goes mainstream" *IEEE Spectrum*, October 1993

[Douglis94] Fred Douglis, P. Krishnan and Brian Marsh, "Thwarting the Power-Hungry Disk" *Proceedings of the Winter 1994 USENIX Conference*, January 1994

[Eldridge93] James Eldridge, "Filing in a flash" *IEEE Spectrum*, October 1993

[Hartman93] John Hartman and John Ousterhout, "Letter of Correction" *Operating Systems Review*, January 1993

[Intel] Intel Flash Memory Products.

[Li94] Kester Li, Roger Kumpf, Paul Horton, and Thomas Anderson, "A Quantitative Analysis of Disk Drive Power Management in Portable Computers" *Proceedings of the Winter 1994 USENIX Conference*, January 1994.

[Marsh94] Brian Marsh, Fred Douglis and P. Krishnan, "Flash Memory File Caching for Mobile Computers", *Proceedings of the 27th Hawaii Conference on Systems Sciences*, 1994

[Maxtor] Maxtor Corporation, "The MXL-105-III" 1993

[Parrish92] T. Parrish and G. Kelsic, "Dynamic Head Loading Technology Increases Drive Ruggedness" *Computer Technology Review*, Winter 1992

[Shirriff92] K. Shirriff and J. Ousterhout, "A Trace-Driven Analysis of Name and Attribute Caching in a Distributed System", *Proceedings of the Winter 1992 USENIX Conference*, January 1992