A Quantitative Analysis of Disk Drive Power Management in Portable Computers

Kester Li, Roger Kumpf, Paul Horton, and Thomas Anderson

Computer Science Division
University of California
Berkeley, CA 94720

Abstract

With the advent and subsequent popularity of portable computers, power management of system components has become an important issue. Current portable computers implement a number of power reduction techniques to achieve a longer battery life. Included among these is spinning down a disk during long periods of inactivity. In this paper, we perform a quantitative analysis of the potential costs and benefits of spinning down the disk drive as a power reduction technique. Our conclusion is that almost all the energy consumed by a disk drive can be eliminated with little loss in performance. Although on current hardware, reliability can be impacted by our policies, the next generation of disk drives will use technology (such as dynamic head loading) which is virtually unaffected by repeated spinups. We found that the optimal spindown delay time, the amount of time the disk idles before it is spun down, is 2 seconds. This differs significantly from the 3-5 minutes in current practice by industry. We will show in this paper the effect of varying the spindown delay on power consumption; one conclusion is that a 3-5 minute delay results in only half of the potential benefit of spinning down a disk.

1 Introduction

Power management has become an important consideration in the design of new hardware and software. Portable computers today can only function for several hours before draining their battery source. Industry has taken the approach of course-grained shutdown of system components as the major power management technique. This approach works well when there are clear periods of system inactivity, but fails under more typical scattered activity patterns. We believe that opportunities exist for fine-grained power management in the portable computer environment.

Tackling the question of power management begins with an analysis of where the energy is being consumed. Table 1 gives a listing of the major system components and their power consumption in a typical portable computer. At 68%, the display clearly dominates the system power consumption. However, we did not target the display in this study because the hardware technology in that area is still rapidly evolving (it is not clear that the display will continue to be the dominant power cost). Moreover, proposed techniques to better manage the power consumption of the display (for instance, back lighting only the portion of the screen containing the cursor) would require extensive hardware changes to be practical. Instead, we focused on managing the disk drive which represents 20% of the power consumption. The disk is a promising candidate for power management because it is a device with which the user does not interact with directly. With proper management by the operating system, the disk may be spun up and down without the user noticing much difference in performance or reliability. From anecdotal observations of disk activity on personal computers, we believed that almost all the power consumed by a disk drive could be eliminated.

Component	Manufacturer & Model	Power (watts)	Percent of Total
Display	Compaq monochrome lite25c	3.5	68%
Disk Drive (105 Mbytes)	Maxtor MXL-105 III	1.0	20%
CPU	3.3V Intel486	0.6	12%
Memory (16 Mbytes)	Micron MT4C4M4A1/B1	0.024	0.5%

Table 1: Breakdown of power consumption by components.

To investigate these issues, we collected traces of file system activity from both personal computers and Unix workstations. We then simulated the effect on power consumption of different disk management strategies for our traces. Among the questions we examined were:

- How long should the disk remain idle after servicing a request before spinning down?
- Can a small, fixed spindown delay approach the optimal energy savings?
- What is the effect of spinning down the disk on the system performance observed by the user?
- What is the effect on power consumption of adding memory as a disk cache?
- Does delaying disk writes to occur in the background save power?
- Are name-attribute caches helpful for reducing power consumption?

The remainder of the paper discusses these issues in more detail. Section 2 gives background on disk power consumption and the assumptions we made in building our simulator. Section 3 outlines how we collected our file traces. Section 4 presents the results of our simulation study, and Section 5 summarizes our conclusions.

2 Background/Simulator Components

2.1 Disk

The recent explosion in the portable computer market has enticed disk drive manufacturers to develop a special breed of drives especially designed for the portable environment. In addition to high shock tolerances, reduced physical volume, and smaller weights, these drives consume less energy and more importantly have a new mode of operation called SLEEP mode. A very significant portion of the energy consumed by a disk drive is spent in preserving the angular momentum of the physical disk platter. A much smaller fraction is spent in powering the electrical components of the drive. By sleeping, a drive can reduce its energy consumption to near zero by allowing the disk platter to spin down to a resting state. This substantial energy reduction is not without its costs. An access to the disk while it is sleeping incurs a delay measured in seconds as opposed to the tens of milliseconds required for an access to a spinning disk.

The disk we simulated is a prototypical next generation low power drive heavily optimized for the portable environment. It has four major modes of operation. OFF mode is when the disk consumes no energy and is incapable of performing any functions except powerup. SLEEP mode is when the disk is powered up but the physical disk platter is not spinning. IDLE mode is characterized by a spinning disk, but the absence of disk activity. A drive attached to a personal computer typically resides in this mode. The last mode of operation is ACTIVE, when the disk platter is spinning and either the disk head is seeking or the disk head is actively reading or writing the disk. This mode consumes the most power, but occurs only for short periods of time in a typical single-user system. Figure 1 shows a state transition diagram for our simulated drive.

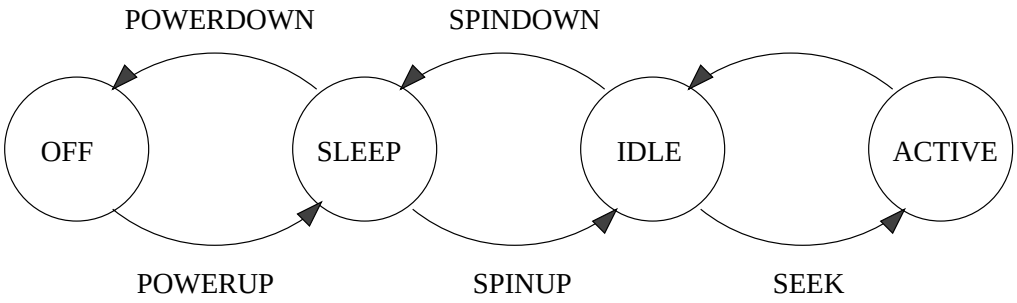

Figure 1: **Disk State Transition Diagram.**

Mode	Power (watts)
OFF	0.0
SLEEP	0.025
IDLE	1.0
ACTIVE	1.95

Table 2: Power consumption of the major disk modes for the Maxtor MXL-105 III.

Transition	Time (seconds)	Power (watts)
POWERUP	0.5	0.025
SPINUP	2.0	3.0
SEEK	0.009	1.95
SPINDOWN	1.0	0.025
POWERDOWN	0.5	N/A

Table 3: Average transition times between major disk modes and their power consumptions for the Maxtor MXL-105 III.

The diagram shows each of the states and the relationships between them. The disk mode transitions which require a non-zero time delay have been labeled for reference. Tables 2 and 3 quantify the power consumption of each mode and the transition times between modes. These numbers are taken from the Maxtor MXL-105 III disk drive [Maxtor].

2.2 Disk Cache

Traditionally, disk caches have been used to increase the performance of file systems. Due to the principle of locality and the large speed differences between DRAM's and disks, the operating system can use a disk cache to give the illusion of a faster disk drive. This performance boost takes on even more importance in the portable computer environment where disks sleep and the speed separation between DRAM's and disks grows even larger.

Performance, however, is not the only reason for including a disk cache in a low power operating system. From a power management viewpoint, disk operations are not only measured in terms of access times, but also in terms of energy costs. Disk operations which are requested while the disk is sleeping will incur a high energy cost to spinup the disk. If the disk cache can satisfy most of these requests, the result will be much smaller energy costs than in a system with no cache. Thus disk caches can reduce energy consumption by filtering read traffic.

The write policy of the disk cache also has impact on the design of low power operating systems. We investigate the impact of a write-back policy for our disk cache because it can make writes asynchronous with user activity, and because disk caches on portables are battery-backed non-volatile RAM. Asynchronous writes yield a measurable speed advantage in distributed and timeshared systems by overlapping computation and I/O. In addition, this policy allows writes to files that are quickly overwritten to be eliminated. More generally, asynchronous writes make sense whenever there is a high probability of long latency disk access as would occur if a disk is sleeping in a low power system. By making writes asynchronous, we can continue processing while the disk spins up for handling the write request. We thereby reduce the overall performance impact of spinning down a disk. In addition, caching writes in memory for short periods of time aids in reducing the number of spinups necessary to service writes.

Our disk cache is a typical LRU block cache with a block size of 4 kilobytes. The cache is parameterized over both the number of blocks and the write delay. A larger cache can be specified by increasing the number of blocks allocated. Greater reliability can be achieved by decreasing the write delay of the cache. The write delay specifies the maximum amount of time a dirty block may spend in the cache. A write delay of zero instantiates a write-through cache.

2.3 Name-Attribute Cache

Studying prior measurements of file system behavior, we noticed that a significant portion of file system activity is involved with reading, translating, and listing the names and attributes of files in the file system. With the goal of minimizing both energy consumption and user delay, we investigate the quantitative impact of incorporating both a name and an attribute cache in a low power system. A recent study into the effectiveness of name and attribute caches was done in [SO92]. They concluded that a twenty-directory name cache had a 97% hit rate with 2% capacity and 1% compulsory misses and that a twenty-entry attribute cache had an 88% hit rate with 5% capacity and 7% compulsory misses. Although these numbers are very good for distributed and timesharing systems, a low power system differs in that it has a much higher miss latency penalty. If a significant percent of the misses occur while the disk is sleeping, then a low power system would pay a large user delay penalty in addition to increased energy consumption. With such high costs, it seems reasonable to investigate how to further reduce the attribute cache miss rate to something near the name cache miss rate. We propose that all files in directories cached in the name cache should also have their attributes cached in memory.

Since we are caching all attributes of files in the name cache, there is no longer any need to distinguish between name and attribute caches. In our system, they have been combined into one cache which we will call the name-attribute cache. In the [SO92] study, 93% of directories had 25 or fewer files. A quick "back of the envelope" calculation shows that with 25 files per directory and 100 bytes of name, attribute, and structure overhead per file, a 20-directory cache consumes less than fifty kilobytes, which is a tiny amount of memory by today's standards.

A rudimentary study of the number of directories referenced in the DOS traces over a four hour period indicates that the number of active directories is well below twenty. With almost all cache misses due to compulsory misses, we assume in this study that the name-attribute cache pays an initial warmup penalty to read in the active directories and then performs perfectly thereafter.

3 Traces

We examine file system traces from two separate platforms. Our measurements of Microsoft DOS file system activity represent most of the applications run on portable computers today. However, in the near future, we expect to see a growing number of Unix-like applications running on increasingly powerful portables. Thus, for completeness, we also examine file system access patterns on a Unix-like platform and compare the results with the DOS traces. To be representative of portable usage with limited battery lifetimes, we used 1 hour and 4 hour long traces.

3.1 DOS Traces

A large portion of the DOS traces were taken from a student and faculty computing facility at City College of San Francisco (CCSF). All of the CCSF traces were running one of five programs: Word Perfect, Lotus 123, Quattro Pro, Paradox, and Question & Answer. These traces were taken over a period of a month and include trace lengths spanning the spectrum from thirty minutes to six hours. To further increase the representativeness of our DOS traces, we had several other trace sources running Windows 3.1, Microsoft Word, Microsoft Excel, Equation Editor, Quicken, and various text editors. These traces were taken over a period of three months. We used 61 one hour DOS traces and 15 four hour DOS traces.

3.2 Sprite Traces

For Unix-like file activity, we extracted segments from trace data collected on the Sprite distributed file system [BHK91]. These traces primarily capture the file activity of Unix programs. The only perceptible difference in the file activity of the programs running on Sprite rather than a Unix file system is that Sprite performs additional name lookups before accessing files. The traces were taken from two different weeks, one in January of 1991 and one in May of 1991. We extracted traces from three different times of the day and from twelve distinct host machines. Each trace was filtered for file system activity due solely to that machine. We used 8 one hour Sprite traces and 7 four hour Sprite traces.

The Sprite traces we used contain information about all open, lseek, and close operations, but no read or write operations except those to concurrently write-shared files. Thus, it was usually necessary to infer read and write operations from changes in the file pointer value reported for lseek and close operations. All file pointer differences in files opened for write or read/write access were attributed to unrecorded write operations. Likewise, file pointer differences in files opened for read access were attributed to unrecorded read operations. This policy resulted in approximately two thirds as many write operations as reads in our traces. We believe that inferring these read and write operations had minimal impact on the integrity of the traces, since files are generally open for very short periods of time. Measurements of the Sprite file system [BHK91] showed that over 90% of the files were open for less than 1/2 second. The traces gave us enough information to infer the number of bytes read or written, the offset of those bytes in the file, and when the operations occurred to within a tolerance of about 1/4 second. It should be noted that the remaining 10% of the files have open-close times which are distributed roughly exponentially. In this paper, we assume that the disk operations for these files occur at close time.

A complete analysis of disk activity would rightfully include the paging activity of the system. Unfortunately for our study, the Sprite traces we examined did not contain paging data and we could not account for these disk references. However, for performance reasons, we expect that the amount of disk activity due to paging is small compared to file system activity.

4 Evaluation

We split the DOS and Sprite traces into one hour and four hour lengths to analyze any differences in reference patterns. Our one hour results were consistent with the four hour results so we present only the latter set in the following sections. The DOS traces were grouped according to applications and analyzed using the same criteria. There were some minor differences in energy consumption characteristics between software vendors, but no notable differences along application domains such as databases or spreadsheets. This result is somewhat surprising and is most likely due to differences in run-time memory management systems. As the differences were minor, we again decided to surpress this distinction by averaging the results.

4.1 Spindown Delay

4.1.1 Spindown Delay Curve

Figures 2 and 3 show the energy consumption of the disk as a function of spindown delay – that is, the length of time we wait for further disk activity before allowing the disk to stop rotating. Figure 3 shows

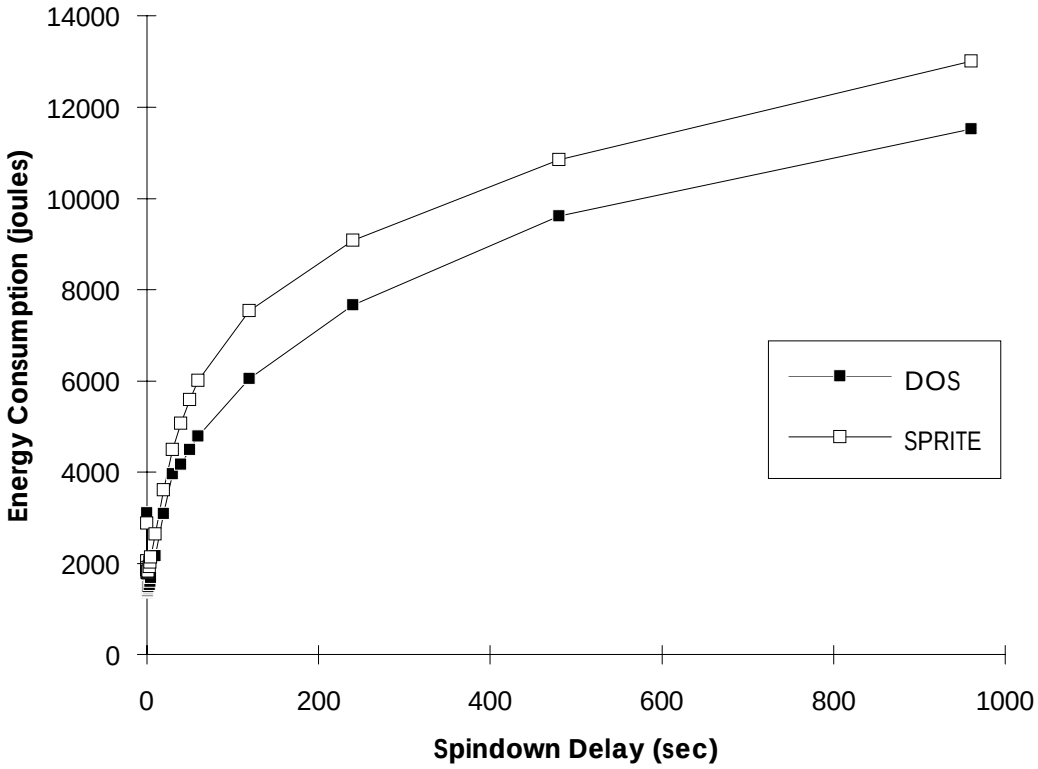

Figure 2: **Spindown Delay vs. Energy Consumption.** This figure shows the effect of varying spindown delay on energy consumption. The simulations were run with a one megabyte disk cache, 30 second write delay, name-attribute caching enabled and 4 hour traces.

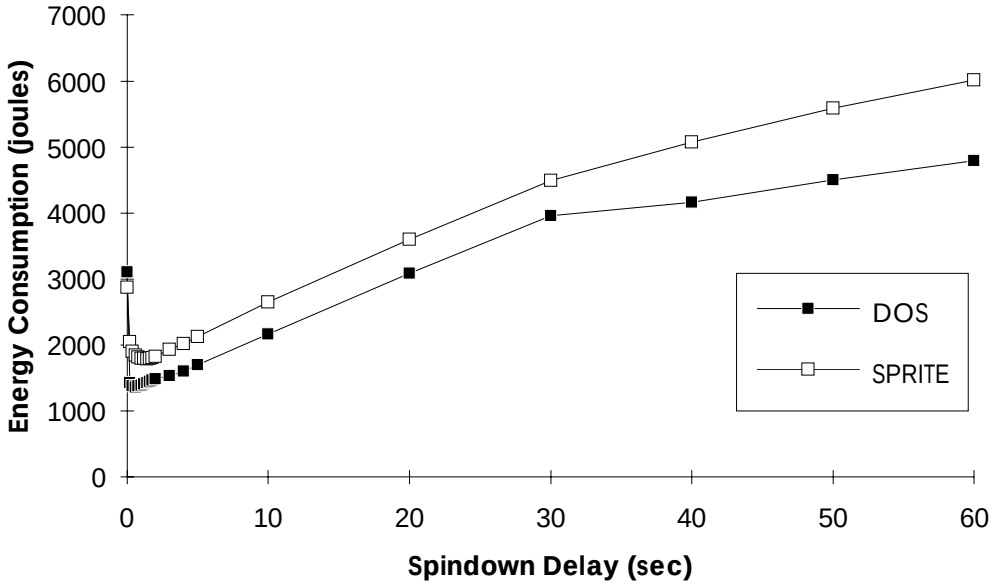

Figure 3: **Spindown Delay vs. Energy Consumption (Minimal Region).** This figure shows the effect of varying spindown delay on energy consumption for small spindown delay values. The simulations were run with a one megabyte disk cache, 30 second write delay, name-attribute caching enabled and 4 hour traces.

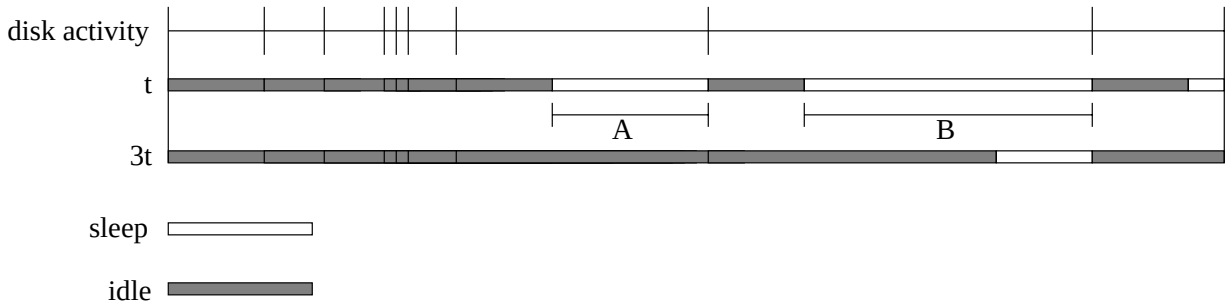

Figure 4: **Effect of Spindown Delay.** This figure illustrates the dual effects of spindown delay on the length of time a disk sleeps.

that the minimum amount of energy is consumed with a spindown delay of around 2 seconds. Using a 2 second spindown delay saves nearly 90% of the 14400 joules that would be consumed if no spindown policy were used. At a spindown delay of zero, there is a sharp increase in energy consumption due to the cost associated with spinning up the disk repeatedly. For our disk drive, the spinup cost is 2 seconds of user delay and 6 joules of energy. The fact that a few second delay is effective confirms the intuition that disk events usually occur in pockets of a few seconds and that it would be a bad idea to always spin down the disk immediately.

The unintuitive result of these figures is the sharp slope to the right of 2 seconds. The steepness of this region indicates that a very small increase in spindown delay beyond a few seconds will yield a very large energy penalty. With the current practice in industry spinning down the disk after 3-5 minutes, our results show that a factor of four in energy savings can be achieved by reducing the spindown delay to 2 seconds. This translates into added latency in accessing the disk, but as we will show in section 4.2, this penalty is small.

It is insightful to compare our 2 second spindown delay policy with the prescient spindown policy (OPTIMAL_DEMAND in [DKM94]) which uses foreknowledge of disk events to optimally decide when to spindown the disk. Using the numbers from Table 2 and 3, we can compute the period of time a disk must remain idle before the cost of spinning up the disk is below that of just idling. The breakeven point (relative to energy) comes out to be 6.2 seconds. If there is no disk activity for greater than 6.2 seconds, then spinning down the disk will save energy. We can approximate the effect of an optimal spindown policy by examining the energy usage and number of spindowns using a fixed 6 second spindown delay. Note that the only difference between optimal and a fixed 6 second delay occurs when the next disk event occurs more than 6 seconds into the future. If the idle time is less than 6 seconds, neither policy will spindown the disk. If the idle time is more than 6 seconds, both will spindown the disk, but optimal will spin down immediately, instead of waiting 6 seconds before spinning down. Thus the energy consumption for the optimal policy is the energy consumption of a fixed 6 second delay policy, minus the energy consumed by the unnecessary idle time before every spindown. As shown in Figure 5, the 6 second delay policy resulted in about 100 spindowns per session. Multiplying the number of spindowns by 6 seconds at 1 watt of power consumption in idle mode yields 600 joules that prescient knowledge saves over our 6 second spindown delay policy. Thus, the prescient algorithm saves only 30% of the energy used by our fixed spindown delay policy. Since the 2 second spindown delay algorithm already saves 90% of the power that would be consumed with no spindown policy, the prescient algorithm can at best save an additional 3%. This shows that attempts to further reduce energy consumption by refining the spindown policy will not yield much benefit.

4.1.2 Spindown Delay Analysis

The shape of the curve in Figure 2 is somewhat baffling at first sight. Why should a small change in the left end of the curve yield such drastic changes in energy consumption while the right end is relatively flat? The answer lies in the fact that spindown delay has two effects on energy consumption. The first

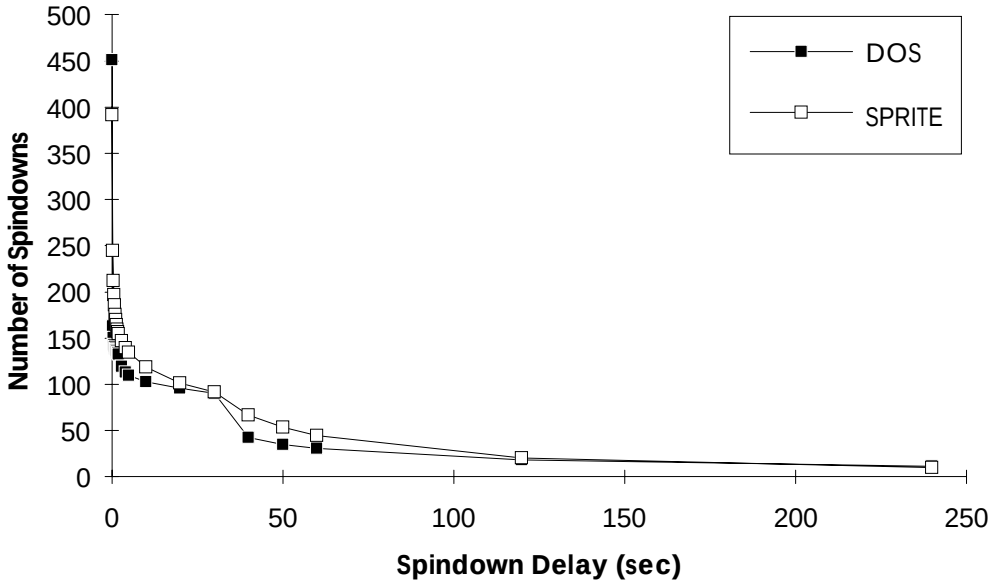

Figure 5: **Number of Spindowns vs. Spindown Delay.** This figure shows the average number of spindowns performed for various spindown delays. The simulations were run with a one megabyte disk cache, 30 second write delay, name-attribute caching enabled and 4 hour traces.

is how often a disk gets to sleep and the second is how long a disk gets to sleep. Figure 4 shows both of these effects. The top line of the figure indicates disk activity and the other two bars illustrate the idle and sleep times of a disk with spindown delay t and 3t. In the region labeled A, the next disk event arrives sufficiently late that the smaller spindown delay is able to take advantage of the inactivity and spindown the disk. This is not the case with a spindown delay of 3t. Here, the next disk event is considered part of the disk activity of the preceding cluster and spinning down of the disk occurs 3t time units after this last event. The goal of choosing a spindown delay value is to make it small enough that it will define clusters of disk activity, but large enough that the amount of energy saved while sleeping is significantly greater than the cost of spinning up the disk. A zero spindown delay defines a cluster as a single disk event while a fifteen minute spindown delay effectively defines a cluster as the entire session. Our analysis shows that a two second spindown delay effectively determines the extent of a disk cluster.

The second and more important effect of spindown delay on energy consumption is the length of time a disk gets to sleep. The region labeled B shows that the disk sleeps for a much longer period of time with the smaller spindown delay. By reducing the delay, we increase the sleep time, which reduces our energy consumption. This energy savings is then compounded by the number of spindowns per session. As shown in Figure 5, a spindown delay of 2 seconds results in approximately 100 spindowns per session. By increasing the spindown delay from 2 seconds to 2 minutes, one can see how this effect is multiplied to yield the steep slope observed on the left end of figure 2.

4.2 User Delay

Section 4.1 showed that it is possible to reduce the energy consumption of a disk drive to a very small fraction of what it uses without power management. In this section, we analyze the tradeoff between energy consumption and user delay. We define user delay as the sum of the spinup delays over the entire 4 hour trace which are synchronous with the user's activities. These are the delays that the user will feel in his interaction with the computer. Asynchronous spinups due to delayed writes from the disk cache are not counted in user delay, as this activity is transparent to the user.

Figure 6 captures the tradeoff between total user delay over the trace and energy consumption. The figure shows that by tolerating a small amount of user delay, large energy savings can be achieved. The

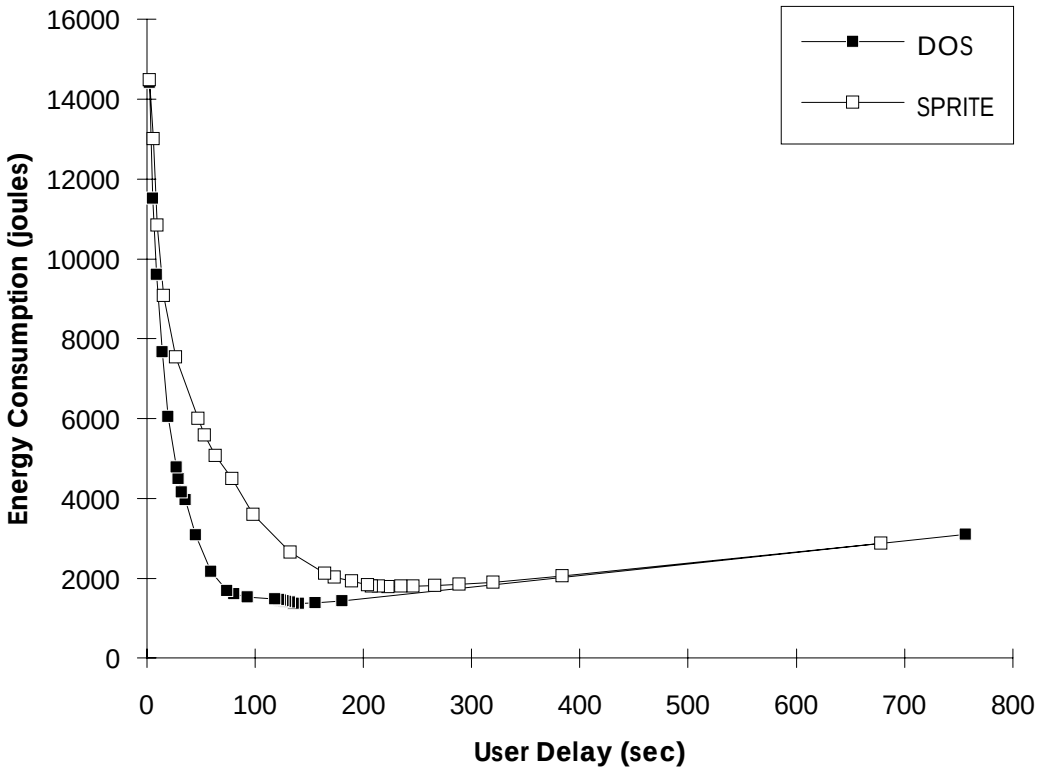

Figure 6: **User Delay vs. Energy Consumption.** This figure shows the tradeoff between user delay and energy consumption. The simulations were run with a one megabyte disk cache, 30 second write delay, name-attribute caching enabled and 4 hour traces. Obtained by varying spindown delay and plotting energy consumption vs. user delay for each data point.

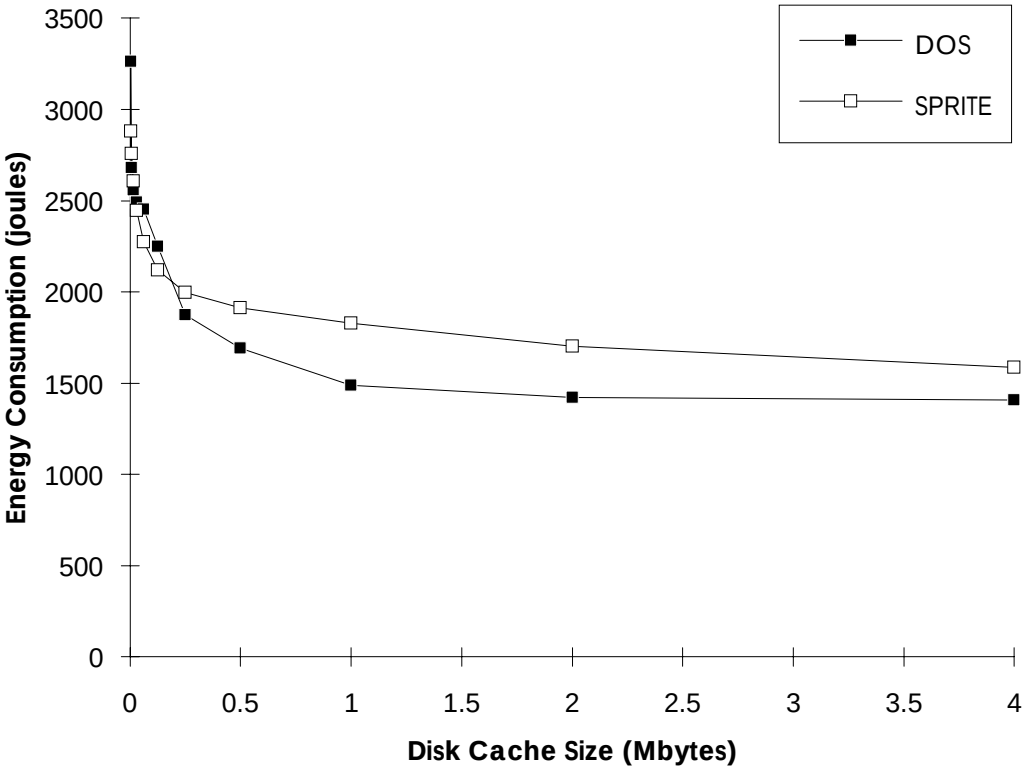

Figure 7: **Disk Cache Size vs. Energy Consumption.** This figure shows the effect of varying disk cache size on energy consumption. The simulations were run with a 2 second spindown delay, 30 second write delay, name-attribute caching enabled and 4 hour traces.

DOS curve bottoms out at around 60 seconds while the Sprite curve bottoms out at around 120 seconds of user delay. These are the delays which would be felt by a user operating with a 2 second spindown delay over a 4 hour period. This amounts to 15-30 seconds of user delay per hour. In other words, the user would have to wait for a 2 second disk spinup 8-15 times per hour. We believe this delay is sufficiently small that its overall effect will be lost in the overhead of the system and application software. Thus a 2 second spindown delay does not impose a significant performance penalty.

4.3 Disk Cache

Contrary to our initial expectations, the disk cache did not exhibit as large an impact on the results as we predicted. Although having a disk cache does aid in reducing energy consumption by filtering the disk traffic, its effect is secondary to the choice of spindown delay.

Figure 7 shows the effects of varying the disk cache size on energy consumption. The figure indicates that a one megabyte cache is sufficient to achieve most of the energy benefits of disk caching. The cost of not having a disk cache is a twofold increase in energy consumption. From a power mangement point of view, trading the energy consumed by one megabyte of DRAM for the additional energy savings of the disk is a wise investment.

Varying the delay in writing dirty blocks to disk has approximately the same effect on energy consumption as varying the disk cache size. As Figure 8 shows, there is approximately a factor of two decrease in energy consumption by enforcing a write delay of 30 seconds. Again, the curve is very steep at the left end and nearly flat at the right end. This indicates that a very small sacrifice in tolerance to lost data will initially yield large energy savings. However, further sacrifices beyond 60 seconds will not yield any additional savings. Thus, the policy implemented by Unix and Sprite of delaying writes for 30 seconds has applicability in the low power environment as well.

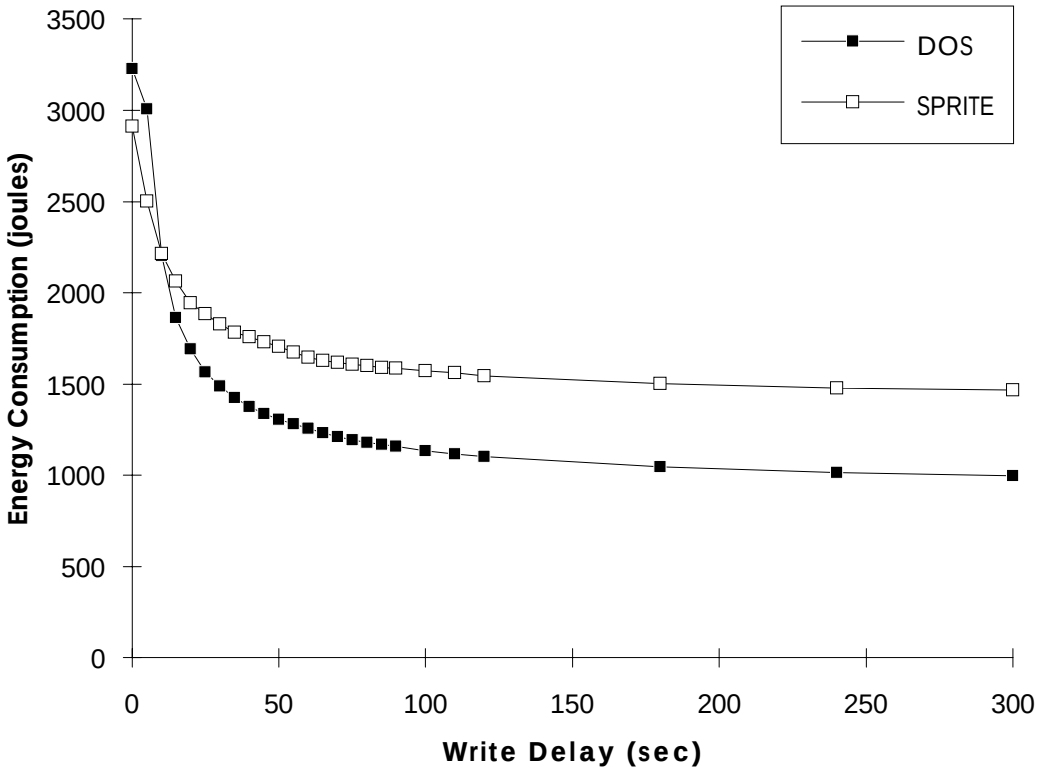

Figure 8: **Write Delay vs. Energy Consumption.** This figure shows the effect of varying write delay on energy consumption. The simulations were run with a 2 second spindown delay, one megabyte disk cache, name-attribute caching enabled and 4 hour traces.

4.4 Spinups

Another area of concern in disk power management is the increased number of disk spinups per session. Increasing the number of spinups per session increases the friction induced wear on the disk-head interface. This in turn decreases the useful lifetime of the disk drive. Figure 5 shows the number of spinups per session under our policies. With a 2 second spindown delay, there are approximately 100 spinups per 4 hour period. Traditional contact start/stop technology disk drives are currently specified at 40,000 start/stops. This gives the moderately disappointing result of 400 four hour sessions before drive wear becomes a problem. However, the portable computer and embedded systems market is currently funding huge efforts into the development of non-contact techniques such as dynamic head loading [PK92] which are targeted specifically for the portable market. These fast, high-shock tolerance drives are expected to populate all future portable computers. The non-contact technology is expected to increase the number of starts/stops of disk drives to one million. With this figure, the expected time to failure under a 2 second spindown delay policy is 10,000 four hour sessions, or about 4 1/2 years of continuous use.

4.5 Name-Attribute Cache

The results of the DOS and Sprite traces turned out to be surprisingly similar from a power management viewpoint. Although Sprite traces consistently consumed more energy and incurred higher user delays than the DOS traces, the shapes of the curves were nearly identical. The only notable point of difference is the relative effectiveness of the name-attribute cache in reducing energy consumption. Table 4 indicates that name-attribute caching is somewhat useful in the DOS environment, but indispensable in the Sprite environment. This difference is due to the presence of large numbers of periodic lookup operations in the Sprite traces. We believe this anomaly is an artifact of the Sprite implementation and is not a general property of a Unix file system. We therefore conclude that a name-attribute cache is moderately successful

Name-Attribute Cache	DOS (joules)	Sprite (joules)
Enabled	1488	1829
Disabled	1800	9149

Table 4: Relative effectiveness of the name-attribute cache under DOS and Sprite. The simulations were run with a 2 second spindown delay, one megabyte disk cache, 30 second write delay and 4 hour traces.

in reducing power consumption, but that its primary contribution to the system is improved performance.

5 Conclusion

We have performed a quantitative analysis of the costs and benefits of spinning down a disk drive as a power management technique. Given an intelligently designed operating system with a one megabyte write-back disk cache and a twenty directory name-attribute cache, we have shown that 90% of the energy consumed by a disk drive can be eliminated with almost no performance or reliability impact. Furthermore, we determined that a spindown delay of 2 seconds minimizes the power consumption of the disk and that deviation from this minimum rapidly increases the energy consumption without significant gains in performance.

6 Acknowledgements

This work was supported in part by the National Science Foundation (CDA-8722788), the Digital Equipment Corporation (the Systems Research Center and the External Research Program), and the AT&T Foundation. Anderson was also supported by a National Science Foundation Young Investigator Award.

We would like to thank Albert Goto who diligently collected and labeled the hundreds of DOS traces taken from the CCSF computing laboratory. This study would have suffered greatly without his efforts.

References

[BADOS92] 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

[BHK91] 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

[Compaq] Compaq User's Manual, 1993

[CDLM93] R. Cáceres, F. Douglis, K. Li and B. Marsh, "Operating Systems Implications of Solid-State Mobile Computers" *Proceedings of the Fourth Workshop on Workstation Operating Systems*, Oct. 1993, 21-27

[DM93] F. Douglis and B. Marsh, "Low-power Disk Management for Mobile Computers", Technical Report MITL-TR-53-93, Matsushita Information Technology Laboratory, April 1993

[DKM94] F. Douglis, P. Krishnan and B. Marsh, "Thwarting the Power-Hungry Disk" *USENIX*, Winter 1994

[Greenawalt94] P. Greenawalt, "Modeling Power Management for Hard Disks", to appear in *Proceedings of the 13th Symposium on Modeling and Simulation of Computer and Telecommunication Systems"* 1994

[LPCMOS] Low Power CMOS Digital Design, IEEE Solid State, April 1992

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

[MDK94] B. Marsh, F. Douglis and P. Krishnan, "Flash Memory File Caching for Mobile Computers", to appear in *Proceedings of the 27th Hawaii Conference on Systems Sciences IEEE* 1994

[PK92] T. Parrish and G. Kelsic, "Dynamic Head Loading Technology Increases Drive Ruggedness" *Computer Technology Review*, Winter 1992

[RW93] C. Ruemmler and J. Wilkes, "UNIX disk access patterns" *Proceedings of the Winter 1993 USENIX Conference* Jan. 1993, 405-420

[SO92] K. Shirriff and J. Ousterhout, "A Trace-Driven Analysis of Name and Attribute Caching in a Distributed System", *USENIX*, Winter 1992

[Wilkes92] J. Wilkes, "Predictive power conservation", Technical Report HPL-CSP-92-5, Hewlett-Packard Laboratories Feb. 1992

Author Information

Kester Li is a graduate student in the Department of Electrical Engineering and Computer Sciences at the University of California at Berkeley. His interests include programming languages and operating systems. He received a B.A. in computer science from UC Berkeley in 1992. Kester Li can be reached at "kesterl@cs.berkeley.edu".

Roger Kumpf is a graduate student in the Department of Electrical Engineering and Computer Sciences at the University of California at Berkeley. His interests include software systems and user interfaces. He received a B.S. in computer science from The Ohio State University in 1992. Roger can be reached by e-mail at "kumpf@cs.berkeley.edu".

Paul Horton is a graduate student in the Department of Electrical Engineering and Computer Sciences at the University of California at Berkeley. He is interested in computation biology and artificial and natural intelligence. He received his B.S. in molecular biology from The University of Washington in 1989 and his M.S. in biophysics from Kyōto University in 1992. He can be reached at "paulh@cs.berkeley.edu".

Thomas Anderson is an Assistant Professor in the Computer Science Division at the University of California at Berkeley. He received his A.B. in philosophy from Harvard University in 1983 and his M.S. and Ph.D. in computer science from the University of Washington in 1989 and 1991, respectively. He won an NSF Young Investigator Award in 1992, and he co-authored award papers at the 1989 SIGMETRICS Conference, the 1989 and 1991 Symposia on Operating Systems Principles, the 1992 ASPLOS Conference, and the 1993 Winter and Summer USENIX Conferences. His interests include operating systems, computer architecture, multiprocessors, high speed networks, massive storage systems, and computer science education. His e-mail address is "tea@cs.berkeley.edu".