# Thwarting the Power-Hungry Disk

*Fred Douglis*
*Matsushita Information Technology Laboratory*
*P. Krishnan*
*Brown University*
*Brian Marsh*
*Matsushita Information Technology Laboratory*

## Abstract

Minimizing power consumption is important for mobile computers, and disks consume a significant portion of system-wide power. There is a large difference in power consumption between a disk that is spinning and one that is not, so systems try to keep the disk spinning only when it must. The system must trade off between the power that can be saved by spinning the disk down quickly after each access and the impact on response time from spinning it up again too often. We use trace-driven simulation to examine these trade-offs, and compare a number of different algorithms for controlling disk spin-down. We simulate disk accesses from a mobile computer (a Macintosh Powerbook Duo 230) and also from a desktop workstation (a Hewlett-Packard 9000/845 personal workstation running HP-UX), running on two disks used on mobile computers, the Hewlett-Packard Kittyhawk C3014A and the Quantum Go•Drive 120. We show that the "perfect" off-line algorithm—one that consumes minimum power without increasing response time relative to a disk that never spins down—can reduce disk power consumption by 35–50%, compared to the fixed threshold suggested by manufacturers. An on-line algorithm with a threshold of 10 seconds, running on the Powerbook trace and Go•Drive disk, reduces energy consumption by about 40% compared to the the 5-minute threshold recommended by manufacturers of comparable disks; however, over a 4-hour trace period it results in 140 additional delays due to disk spin-ups.

## 1 Introduction

The recent trend toward portable, battery-operated computers is motivating advancements in reducing power consumption through both hardware and software approaches. One area that has seen rapid improvement is disks for mobile computers: decreasing scale and increasing density have led to small, lightweight, low-power disks such as the Hewlett-Packard Kittyhawk [9], as well as flash memory devices such as Seagate's IDE-compatible FlashDrive [16]. In recent papers we have examined the effect of using some amount of flash memory as a cache of frequently-accessed disk blocks, in order to keep the disk from spinning as often [13], or as a complete replacement for disk [3]. Until flash is inexpensive[1] and long-lasting enough to be ubiquitous, many notebook and laptop computers will have only DRAM and a hard disk. If the disk drive is used with any frequency, it will have a significant impact on the length of time the computer can operate on a single battery charge.

Spinning down the disk when it is not being used can save power. Most if not all current mobile computers use a fixed threshold to determine when to spin down the disk: if the disk has been idle for some (predetermined) amount of time, the disk is spun down. The disk is spun up again upon the next access. The fixed threshold is typically on the order of many seconds or minutes to minimize the delay

[1] A recent advertisement in the New York Times priced the Hewlett-Packard Omnibook at $1599 with a 40-Mbyte hard disk or $1799 with a 10-Mbyte flash memory card. These prices have dropped a few hundred dollars just since the introduction of the Omnibook, with the gap between them closing, but even so, the added cost of flash is substantial when the relative sizes of the media are considered.

| Machine | CPU Speed (MHz) | Disk Size (MBytes) | Disk State | System Power (W) | Power Savings (W) | % of Total System Power |
|---|---|---|---|---|---|---|
| Zenith Mastersport SLe | 25.0 | 85 | Idle | 10.5 | 1.0 | 9.5 |
| | | | Stopped | 9.5 | | |
| | 6.5 | | Idle | 9.2 | 0.9 | 9.8 |
| | | | Stopped | 8.3 | | |
| Toshiba T3300SL | 25.0 | 120 | Idle | 8.1 | 1.2 | 14.8 |
| | | | Stopped | 6.9 | | |
| | 6.5 | | Idle | 7.3 | 1.1 | 15.1 |
| | | | Stopped | 6.2 | | |
| Dell 320SLi | 20.0 | 120 | Idle | 4.5 | 0.9 | 20.0 |
| | | | Stopped | 3.6 | | |
| | 2.5 | | Idle | 3.2 | 1.0 | 31.2 |
| | | | Stopped | 2.2 | | |

**Table 1:** Power measurements of three typical laptop computers.

from on-demand disk spin-ups. The Hewlett-Packard Kittyhawk C3014A spins down and up again in about three seconds, and its manufacturer recommends spinning it down after about five seconds of inactivity [10]; most other disks take several seconds for spin-down/spin-up and are recommended to spin down only after a period of minutes [5, 18]. In fact, spinning a disk for just a few seconds without accessing it can consume more power than spinning it up again upon the next access. Spinning down the disk more aggressively may therefore reduce the power consumption of the disk, in exchange for higher latency upon the first access after the disk has been spun down.

To understand this tradeoff, we use trace-driven simulation to evaluate different disk spin-down policies for reducing power consumption. We consider threshold policies, which are practical to implement, off-line algorithms that are optimal in terms of power consumption, and predictive on-line algorithms that take past history into account. We find that threshold policies that spin down the disk after 1–10 seconds come close to the power consumption of the optimal off-line algorithm, which reduces the power consumption using manufacturers' recommended thresholds by about half. However, in some cases the threshold algorithms substantially increase the delays incurred by the user. These delays could be avoided if access times could be predicted accurately enough, but predictive strategies that close the gap between the optimal off-line algorithm and current threshold-based algorithms are difficult to construct.

The rest of this paper is organized as follows. Section 2 elaborates on the motivation behind our work, specifically the power consumed by the disk subsystem in current mobile computers. Section 3 discusses various spin-down policies. Section 4 describes the input traces and simulator used in our experiments, and Section 5 reports the results of our simulations. Section 6 discusses related work, and finally, Section 7 concludes the paper.

## 2 Power Consumption

To get some idea of how the disk can affect battery life, we measured the power consumption of the disk on a Dell 320 SLi, a Toshiba T3300SL, and a Zenith Mastersport SLe. This data is shown in Table 1.[2] All three machines are running Mach 3.0 (UX37/MK77). The machines are listed in the relative order of their age. All were purchased in the past two years, and at the time represented the state-of-the-art in low-power notebook design. All three use the Intel SL Superset, which consists of the 386 SL CPU and the 82360 I/O controller. The Zenith and the Toshiba both have a backlit LCD display, while the Dell uses a "triple super-twist nematic, reflective LCD" display.

The measurements were made using an HP 34401A multimeter using customized instrumentation

[2] This table also appears in [13].

software. We varied two parameters: the speed of the CPU and the state of the disk. We controlled the state of both using hot-key bindings supplied by the system manufacturers. The CPU speed was set at the fastest and slowest speeds available. The disk was set to be either "spun-up" or "spun-down."

Varying the clock speed is important because the CPU can consume a large amount of power. Reducing its clock speed when there is no work to be done can significantly reduce the amount of power consumed. In fact, many commercial systems already incorporate "Advanced Power Management" [6] to take advantage of such savings. As a result, it is reasonable to expect that on most systems the CPU will be slowed down when idle. Mobile computers are likely to be used for highly interactive software (such as mailers, news readers, editors, etc.) so it is reasonable to expect a large amount of CPU idle time. When the CPU clock speed is reduced, a spinning disk will consume proportionally more of the total system power.

There are several important things to note about Table 1. First, disk densities are increasing, making it possible to carry more data. Machines are now available with even larger disks than the systems we instrumented. Second, even though disk densities have increased, the power used by the largest disks has stayed about the same, around 1W for an idle spinning disk. Third, the overall system power cost is dropping. The result is that the amount of power consumed by the disk sub-system on these notebook computers has increased from 9% to 31%. Improved recording densities make it possible to store more data on the same physical device, but they do not affect the physical mass. Drives are becoming more efficient, but cost about the same to spin up and to keep spinning. Theoretically, machines could have smaller disks, but in practice, higher recording densities are used to increase the overall capacity of the storage system instead of decreasing its power consumption. With the exception of the smallest and lightest computers, such as the Hewlett-Packard Omnibook [11], the trend seems to be to carry a larger disk with the same mass rather than a smaller disk with the same number of bytes.

Our measurements suggest that proper disk management can improve battery life. In addition, technology trends suggest that the such improvements will become increasingly important. For instance, battery life for the Dell 320 could be improved 20 to 31%, the amount that could be saved if the disk were off all the time. Put another way, a battery that lasts 5 hours could last from 6 to 6.5 hours instead. Of course, turning the disk off can result in increased access latency, so policies for saving power need to balance the two issues. In the next section, we describe different approaches to managing this tradeoff.

## 3 Policies

We investigated two types of algorithms for spinning a disk up and down: *off-line*, which can use future knowledge, and *on-line*, which can use only past behavior. Off-line algorithms are useful as a baseline for comparing different on-line algorithms, and for showing where there is room for potential improvement. On-line algorithms are implementable (though some may consume more memory, processing, or other resources than is feasible). Typically mobile computers spin down their disk based on a simple heuristic; for instance, when it has not been accessed in a predetermined period of time (such as 5 minutes). They spin up the disk when the first access after a spin-down occurs.

### 3.1 Off-line Policies

Spin-up and spin-down policies should minimize both power consumption and response time. Unfortunately, power and time are not always optimized by the same policy. It is easy to see that the optimal policy with respect to response time is not necessarily optimal with respect to power consumption. Leaving the disk spinning all the time will produce the minimal impact on response time, but will waste power if the disk isn't accessed for long periods of time. Likewise, the optimal policy with respect to power may result in a delay when a new request stalls waiting for the disk to spin up.

An off-line policy for spinning down the disk is based on the relative costs of spinning or starting it up. We define $T_d$ as the amount of time the disk must spin before the cost of spinning the disk continuously equals the cost of spinning it down immediately and then spinning it up again just prior to the next access. With future knowledge one can spin down the disk immediately if the next access will take place more than $T_d$ seconds in the future. This will result in the minimal power consumption of all spin-down algorithms,

| Characteristic | Hewlett-Packard Kittyhawk C3014A | Quantum Go•Drive 120 |
|---|---|---|
| Capacity (Mbytes) | 40 | 120 |
| Power consumed, active, (W) | 1.5 | 1.7 |
| Power consumed, idle, (W) | 0.6 | 1.0 |
| Power consumed, spin up (W) | 2.2 | 5.5 |
| Normal time to spin up (s) | 1.1 | 2.5 |
| Normal time to spin down (s) | 0.5 | 6.0 |
| Avg time to read 1 Kbyte (ms) | 22.5 | 26.7 |
| Break-even interarrival time $T_d$ (s) | 5.0 | 14.9 |

**Table 2:** Disk characteristics of the Kittyhawk C3014A and Quantum Go•Drive 120. The Kittyhawk has less capacity than the Go•Drive, but it has significantly lower operating costs, especially the power drawn during disk spin-up and the average spin-up duration. As a result, the break-even point for the Kittyhawk is about a fourth that of the Go•Drive, making a short spin-down threshold much more important for the Kittyhawk. Also, the Kittyhawk spins down in a half a second, while the Go•Drive takes "$<$ 6s" [14].

among policies that have minimum response time. There are, of course, complications beyond this simple threshold; for instance, a disk usually has multiple states that consume decreasing amounts of power but from which it is increasingly costly (in time and power) to return to the active state. Table 2 lists the characteristics of two disk drives for mobile computers, the Hewlett-Packard Kittyhawk C3014A [9] and the Quantum Go•Drive 120 [14], including values for $T_d$.

In fact, the time to spin up the disk once a new request arrives has a substantial impact on response time. An on-line algorithm that spins up the disk when a request arrives if the disk is spun down will cause the request to wait until the disk is ready, typically at least 1–2 seconds. This latency is up to a couple of orders of magnitude greater than normal disk access times, and should be avoided whenever possible. The high spin-up overhead is the reason why typical thresholds for spinning down a hard disk are often on the order of several minutes even if $T_d$ is just a few seconds: if the disk has not been accessed for several minutes then the overhead of a couple of extra seconds before a new request can be serviced is neither unexpected nor unreasonable. In contrast to the on-line approach, an off-line algorithm can not only spin down the disk when that would save power, it can spin up the disk again just in time for the next request to arrive.

### 3.2 Threshold-based Policies

Threshold-based policies are the standard timeout-based algorithms used in most present systems. If the disk is not accessed within a fixed period of time it is spun down. That timeout value may vary depending on environmental characteristics (e.g., running off battery rather than A/C power) or input from the user, but is normally not modified dynamically by the system. The disk is spun up again upon the next access, and the request must wait for spin-up to complete.

### 3.3 Predictive Policies

The predictive policies store historical information and interpret it to predict the next access. There are many possible heuristics for interpreting this data. For instance, Wilkes hypothesized that it would be effective to use a weighted average of a few previous interarrival times to decide when to spin down the disk on a mobile computer. He noted as well that if inactive intervals were of roughly fixed duration, the disk could be spun up in advance of the expected time of the next operation [17]. If access patterns are not so consistent, however, these techniques may not prove to be helpful.

In practice, predictive models may have difficulty beating simple threshold policies because access patterns are not sufficiently regular. If *every* access either came within 100ms of the previous access or after a delay of many seconds, it would be possible to spin down the disk after 100ms passed. But if even a small fraction of accesses take place between 100ms and $T_d$ seconds after their predecessors, having such a quick trigger to spin down the disk could be costly. We are presently evaluating some predictive heuristics to see if they can be applied across a range of workloads, and have some comments on predictive algorithms in Section 5.3.

### 3.4 Taxonomy

Here we describe a taxonomy of disk spin-down policies, considering both the algorithm used to decide when to spin down the disk and the one used to spin it up again. The naming scheme indicates the most salient feature of the particular algorithm; the first part of the name denotes the spin-down policy, while the second part denotes the spin-up policy.

OPTIMAL_OPTIMAL This is the off-line algorithm described in Section 3.1, which uses future knowledge to spin down the disk and to spin it up again *prior* to the next access. It provides the lowest power consumption among policies that have minimum response time. Other off-line algorithms with slightly lower power consumption may exist, but they will suffer increased response times.

OPTIMAL_DEMAND An alternative off-line approach is to assume future knowledge of access times when deciding whether to spin down the disk but to delay the first request upon spin-up. This algorithm will consume about the same amount of power as OPTIMAL_OPTIMAL, but will have poorer response time. This algorithm is relevant because an on-line algorithm may be better at predicting that the next request will occur more than $T_d$ seconds in the future than predicting exactly when the request will occur; i.e., predicting the correct time to spin down the disk may be easier than predicting when to spin it up again.

THRESHOLD_DEMAND The disk is spun down after a fixed period of inactivity and is spun up upon the next access. This is the policy used on most systems at present.

THRESHOLD_OPTIMAL The disk is spun down after a fixed period of inactivity but is spun up just before the next access. If the next access occurs too soon (there is not enough time to spin up the disk before the access) then the access will be delayed. This algorithm is primarily for purposes of comparison and completeness.

PREDICTIVE_DEMAND Spin-down is based on a heuristic that uses information about previous accesses to determine when to spin down the disk. Spin-up is performed upon the next access.

PREDICTIVE_PREDICTIVE Spin-down uses the same heuristic as PREDICTIVE_DEMAND, and spin-up is based on a predictive heuristic as well.

## 4 Methodology

### 4.1 Traces

To evaluate the effect of the disk spin-down policy on power consumption and response time, we used traces from two execution environments: an Apple Macintosh Powerbook Duo 230 and a Hewlett-Packard 9000/845 personal workstation running HP-UX.

The Powerbook traces were gathered at MITL. We collected two traces of approximately two hours of activity and one trace of approximately four hours of activity. One of the two-hour traces has characteristics very similar to the four-hour trace, while the other shows more constant use of the disk. In this paper we report results of simulating the four-hour trace, during which time mostly Microsoft Word, an editor, and Eudora, an electronic mail application, were running.

Disk management on the Macintosh is unusual in several respects. First, its cache behavior is dependent on the size of the cache: a larger disk cache not only increases the number of blocks that can be cached but also increases the maximum size of any given read or write that can be cached. Even with a cache size of 256 Kbytes, the maximum transfer that can be cached is only 8176 bytes [1]. Second, writes that are cached in the buffer cache are later passed to the disk in 512-byte units, which can degrade performance

| | Powerbook | HP-UX |
|---|---|---|
| Duration | 4.1 hours | 7 days |
| Mean interarrival time (s) | 0.4 | 15.6 |
| Standard deviation of interarrival time (s) | 1.9 | 142.9 |
| Maximum interarrival time (s) | 269.2 | 1769.5 |

**Table 3:** Summary of trace characteristics. An important distinction between the Powerbook and HP-UX traces is that the statistics for the Powerbook trace report the interarrival times as seen by the buffer cache while the ones for the HP-UX trace are as seen by the disk.

relative to transferring larger files as a unit. Third, a Macintosh may be configured with a "RAM disk" that behaves like a magnetic disk drive but is stored in DRAM. On the Powerbook Duo the RAM disk is not persistent, so it is useful only for storage of temporary files, other noncritical files that can be copied to disk later, or copies of read-only files such as the System folder. The RAM disk thus allows users to get around the deficiencies of the buffer cache, but only for a specific subset of files.

Our Powerbook trace records reflected access at the file-system level (i.e., above the disk cache). Rather than simulating the Macintosh buffer cache as it is implemented, we simulated a simple LRU write-through buffer cache with each 1-Kbyte block handled separately and no maximum per-file limit. The Macintosh enforces a minimum cache size of 32 Kbytes; we varied the cache size from having no cache at all to a maximum of 1 Mbyte. Because a relatively large cache is essential to eliminating enough disk accesses to make spinning down the disk worthwhile [12], in this paper we report results for the 1-Mbyte cache. Also, in our Powerbook traces, about 2% of accesses went to files on the RAM disk, and we ignored these accesses in the simulator. More experienced Powerbook users might have different access patterns, with more accesses to a RAM disk, but in fact one may consider the 1-Mbyte disk cache to be equivalent to a RAM disk of the same size.

In addition, we used traces from an HP-UX workstation, documented by Ruemmler and Wilkes in a previous USENIX conference [15]. We used the HP-UX traces for three reasons: first, because we did not have the Powerbook traces available initially, and even now have somewhat limited experience with those traces; second, because there are a number of UNIX-based mobile platforms available, so a UNIX trace might be indicative of actual mobile usage patterns; and third, because UNIX does the sort of aggressive caching that is essential for eliminating disk accesses. The HP-UX traces are at the disk level; they represent requests from the HP-UX buffer cache to the disk. As a result, we did not simulate a buffer cache for these traces.

We believe the HP-UX traces should be representative of a mobile environment because the sorts of activities mobile users perform—such as word processing, electronic mail, and spreadsheets—are the same activities as the user in the HP-UX trace would perform in the office. Nevertheless, the HP-UX and Powerbook traces do vary considerably in some respects. Most notably, the HP-UX trace is over a prolonged period of time, with about the same number of accesses as the Powerbook trace had over four hours spread out instead over a week's time. (The original trace consisted of two months' worth of data, and we used the first week's worth.) The HP-UX trace has pauses of up to a half-hour between accesses, so any policy that spins the disk down at all will result in spin-ups after such long pauses. However, there are still periods of activity within that trace that result in large differences between different spin-down policies. Table 3 summarizes some characteristics of the two traces.

### 4.2 Simulator

The simulator consists of roughly 7500 lines of C code (including comments). It is a general storage management simulator that models three levels of a storage hierarchy, nominally DRAM, flash memory, and magnetic disk; for these experiments the size of flash was always set to 0. For the HP-UX trace, the size of the DRAM cache was also set to 0, as described above, so all disk requests from the original trace resulted in disk requests in the simulations.

Each disk is represented by a file specifying a number of parameters, one for each state in which the disk might be (reading, writing, active, idling, spun down/standby, or halted), and one for each state transition. The configuration files specify how long the disk stays in a given state or transition and how much power, in Watts, is consumed during that time. Several of these parameters are shown in Table 2.

We made a number of simplifying assumptions in the simulator. First, a disk access is assumed to take the average time for seek and rotational latency, unless it involves a disk block with a numerical identifier within $\epsilon$ of the previous block accessed. Second, all operations and state transitions are assumed to take the average or "typical" time specified by the manufacturer, if one is specified, or else the maximum time. While these assumptions may affect the specific values of energy and response time produced by the simulator, we do not believe they affect the relative differences in energy consumption and response time from using different spin-down policies.

## 5 Results

We simulated a number of threshold-based policies as well as the OPTIMAL_OPTIMAL and OPTIMAL_DEMAND policies, running on both the Powerbook and HP-UX traces, and using both the Kittyhawk and Go•Drive specifications. (We have also experimented with predictive algorithms; some preliminary results are discussed below in Section 5.3.) In this paper we consider two metrics:

**Energy Consumption** The number of Joules consumed by the disk over the course of the simulation.

**Read-Spin-up Delays** The number of times a read operation was delayed to spin the disk up.

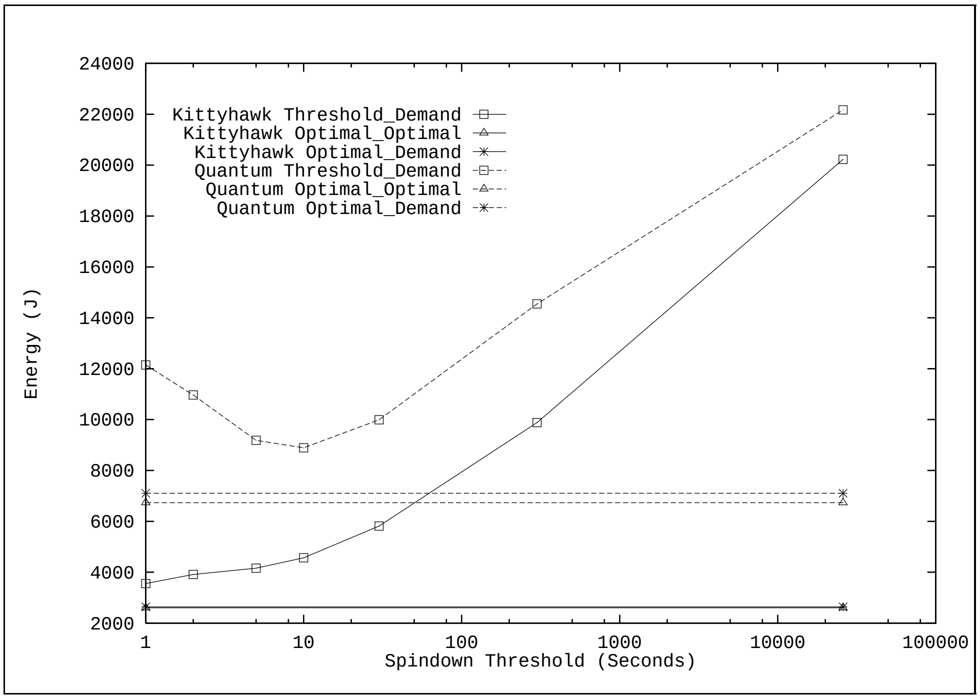
Another metric for the impact on read response time might be the average response time across all reads, but that metric is less satisfying: it considers average delay but not the great discrepancy between operations that have no spin-up delay and those that are delayed. In fact, the actual delay from spin-up varies from about 1 second on the Kittyhawk to 2.5 seconds on the Go•Drive, so the penalty from these undesirable spin-up delays is much greater for the Go•Drive.

Note also that we do not report the impact on write operations, since writes are usually asynchronous, and because even synchronous writes can be decoupled from disk latency with a small amount of nonvolatile memory [2, 15].

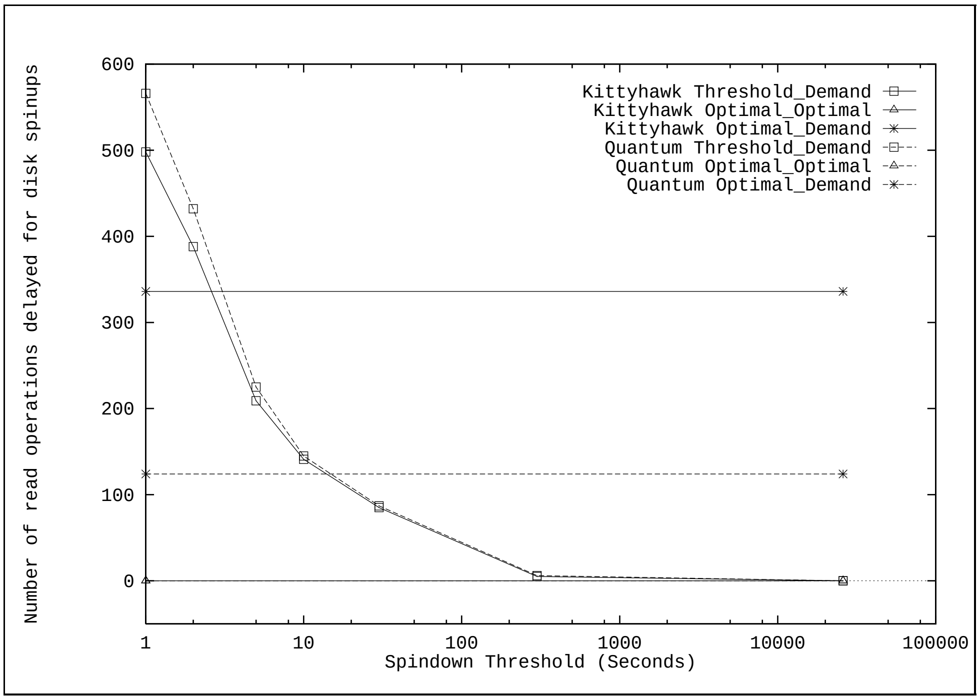
Figure 1 shows the energy consumption and read-spin-up delays for the Powerbook traces, with a 1-Mbyte cache, and Figure 2 shows the same for the HP-UX traces (which has an implicit buffer cache, as discussed above). Both figures show, for both types of disk, several THRESHOLD_DEMAND policies, OPTIMAL_OPTIMAL, and OPTIMAL_DEMAND. For the threshold-based policies, the disk always goes to the idle state after two seconds if it has not already spun down by then, and always goes from the "standby" (spun-down) state to the "halt" state immediately. On the Kittyhawk there is marginal overhead in going from "halt" to "active" relative to going from "standby" to "active," and on the Go•Drive the "standby" and "halt" states are identical.

The most important conclusions one may reach from these figures are:

- The off-line OPTIMAL_OPTIMAL algorithm can reduce disk power consumption by 35–50%, compared to the fixed threshold suggested by manufacturers, without adversely affecting response time. (This compares OPTIMAL_OPTIMAL to the 5-second spindown of the Kittyhawk and the 5-minute spindown of the Go•Drive 120.)
- On-line THRESHOLD_DEMAND algorithms with shorter than recommended thresholds approach the power consumption of OPTIMAL_OPTIMAL but may increase the number of read-spin-up delays substantially.
- The best compromise between power consumption and response time is workload-dependent. For the HP-UX trace on the Kittyhawk, a spin-down threshold of 1s consumes 23% less power than the

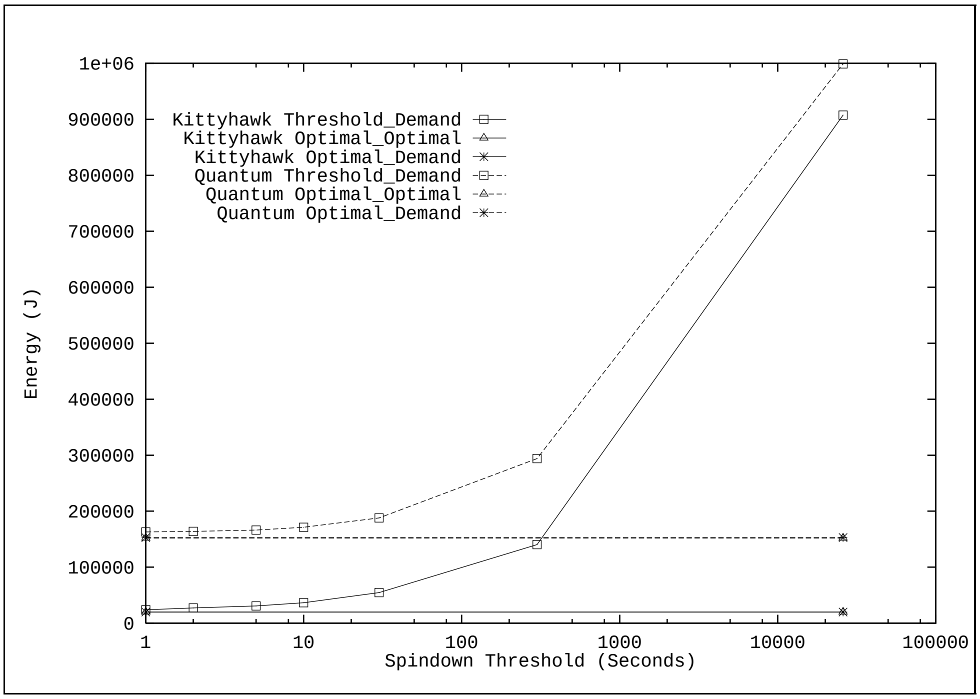



(a) Energy consumption as a function of spin-down time.

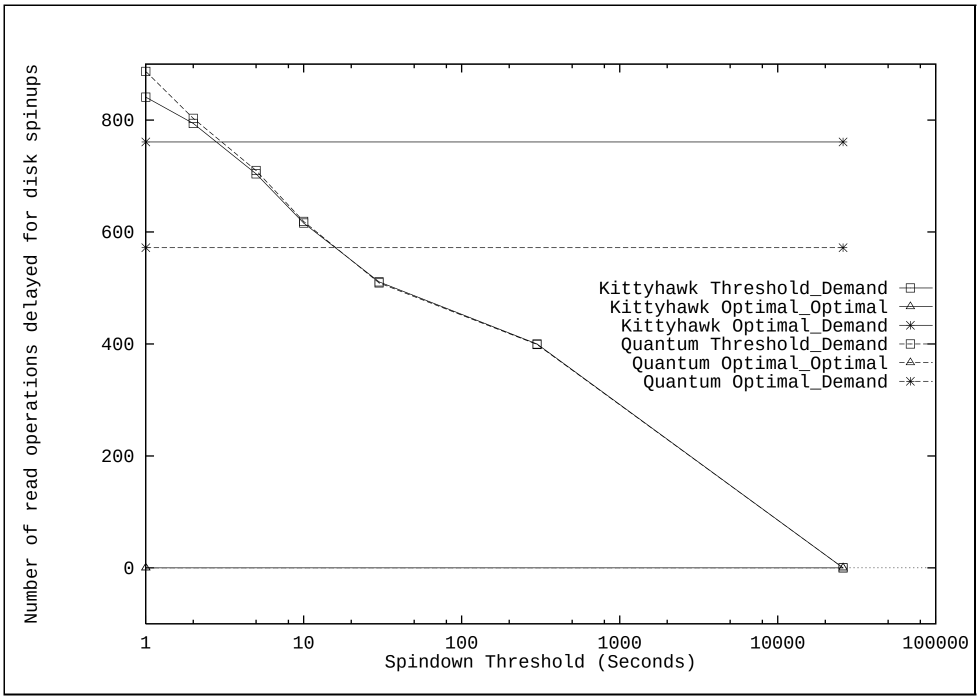



(b) Delays due to spin-up on read accesses, as a function of spin-down time.

**Figure 1:** Results of simulating the 4-hour Powerbook trace on Kittyhawk and Go•Drive 120 disks.




(a) Energy consumption as a function of spin-down time.




(b) Delays due to spin-up on read accesses, as a function of spin-down time.

**Figure 2:** Results of simulating the HP-UX trace on Kittyhawk and Go•Drive 120 disks.

recommended threshold of 5s, and increases delays by 19%, a fair tradeoff. The Powerbook trace on the same hardware shows a 15% improvement in energy with the 1-second threshold but a 150% increase in delays.

- Lastly, the characteristics of the disk make an enormous difference in the appropriateness of an aggressive spin-down policy. The high latency to spin down and spin up the Go•Drive 120, compared to the Kittyhawk, results in a higher value for $T_d$ and, for the Powerbook trace, minimal power consumption for a threshold of 10s rather than 1s for the Kittyhawk.

We discuss each type of algorithm in turn, as well as the impact of disk characteristics.

### 5.1 Off-line Algorithms

With future knowledge of disk activity, one can both reduce energy consumption and delays due to disk spin-up. OPTIMAL_OPTIMAL uses 64% of the energy consumed by the 5-second THRESHOLD_DEMAND policy for the HP-UX trace running on the Kittyhawk; it uses 52% of the energy consumed by the 5-minute policy running on the Go•Drive. For the Powerbook trace, OPTIMAL_OPTIMAL used 62% and 46% of the energy of the recommended thresholds for the Kittyhawk and Go•Drive, respectively. In each case, because the disk was always spinning at the time of the next request, response time would improve as well.

OPTIMAL_DEMAND considers the hypothetical case where one could predict the future well enough to spin down immediately if that would save power, but would not be able to predict the time of the next access precisely. It uses about the same amount of energy as OPTIMAL_OPTIMAL but has a much larger number of read-spin-up delays than the 5-minute THRESHOLD_DEMAND policy, since many more read operations result in the disk spinning up. The number of read-spin-up delays incurred by this algorithm is a lower bound on the number of delays that a THRESHOLD_DEMAND algorithm with a spin-down threshold less than $T_d$ would incur, and an upper bound on the number that a THRESHOLD_DEMAND algorithm with a spin-down threshold greater than $T_d$ would incur.

### 5.2 Threshold-Demand Algorithms

The set of THRESHOLD_DEMAND algorithms reported above demonstrate the tradeoffs between energy and delay that arise with any simple threshold-based technique. The differences between the Powerbook and HP-UX traces show how important the workload is in this regard: for the HP-UX trace on the Go•Drive disk, a 1-second threshold performed best out of this class of algorithms, reducing power consumption 45% (within 7% of optimal) compared to the 5-minute threshold, while for the Powerbook traces a 10-second threshold was better (and even then reduced power by only 11%). This is not surprising given the difference in mean interarrival times described in Section 4.1.

Regardless of the subtle differences in energy consumption between relatively short thresholds of 1, 2, or 10 seconds, it is clear that any of these short thresholds is much more energy efficient than the manufacturers' commonly recommended spin-down threshold of 5 minutes. However, the shorter thresholds introduce more spin-up delays. As a result, the most energy efficient threshold may not be the most desirable one. For example, with the Powerbook trace running on the Kittyhawk, moving from a threshold of 1s to 5s increases energy consumption by 16% but reduces the read spin-up delays by 42%.

### 5.3 Predictive Algorithms

We experimented with heuristics for predicting when to spin down based on past history rather than just the time since the last access. In order to ensure that a bad prediction did not result in keeping the disk spinning indefinitely, we used a threshold as a fallback: if the simulator predicted that an access would occur quickly enough that the disk should not spin down, and the time between accesses passed the threshold value, the disk would then spin down anyway (just like THRESHOLD_DEMAND). To date our results have been disappointing, with the energy consumption and response time from the predictive algorithms generally being slightly worse than the corresponding THRESHOLD_DEMAND algorithm.

The one case in which PREDICTIVE_DEMAND out-performed THRESHOLD_DEMAND was when the Powerbook trace was simulated with no buffer cache. In that case, the energy consumption of PREDICTIVE_DEMAND was roughly halfway between the optimal energy consumption and the best THRESHOLD_DEMAND policy, but all of these were much higher than the energy consumed using a moderate-sized buffer cache. Our conclusion with respect to PREDICTIVE_DEMAND is that the buffer cache tends to increase the entropy of interarrival times as seen by the disk; without a cache, access patterns are more predictable.

We have not yet experimented enough with PREDICTIVE_PREDICTIVE to be able to comment on it definitively. Our impression so far is that predicting the next access time well enough to spin up the disk just ahead of it will be very difficult, and the penalty for predicting incorrectly (spinning up uselessly, then spinning down again, or not spinning up when needed) will outweigh the benefits from "guessing" correctly. However, the area does bear further exploration.

### 5.4 The Impact of Disk Characteristics

The figures above show both the Kittyhawk and the Go•Drive drives using the same sets of traces and spin-down parameters. The lower operating costs for the Kittyhawk result in less power consumed for the same policies, and the faster and less power-intensive spin-up for the Kittyhawk makes it more feasible to quickly spin down the disk.

Since the Go•Drive consumes more power than the Kittyhawk when operating, and takes much longer to spin up and spin down, its overall power consumption is far greater than the same workload on the Kittyhawk. The impact of varying the spin-down threshold is less than for the Kittyhawk as well. Of course, operations on larger disk drives may be expected to consume more power than those on small ones, and spinning them up is especially costly—both in current and time—by comparison. For the foreseeable future, if users wish to store more data on their mobile computer they must be prepared for shorter battery life, poorer response time, or both.

## 6 Related Work

Li, *et al.*, have also investigated the issue of disk drive power management [12]. They used trace-driven simulation to look at a THRESHOLD_DEMAND policy for disk spin-control, and studied important buffer cache parameters. There are three important differences between our work and theirs. The first difference is that we consider multiple algorithms for determining when to spin down the disk: off-line optimal, THRESHOLD_DEMAND, and predictive. In contrast, they focused on THRESHOLD_DEMAND. The second difference is that they did a more detailed analysis of the impact of the buffer cache on power consumption and performance. We simulated the buffer cache only for the Powerbook trace and did not study the impact of different amounts of cache. We did not simulate a buffer cache at all for the HP trace since it is traffic filtered by the HP-UX buffer cache prior to being measured. This filtering made it impossible to experiment with the buffer cache. Finally, they considered the impact of spin-downs on disk reliability; we do not address this issue, relying on their prediction that disk-drive manufacturers will greatly increase the number of spin-up/spin-down cycles a disk can tolerate.

Their basic conclusion is the same as ours: short timeouts on the order of a few seconds greatly reduce power consumption by the disk and do not significantly degrade performance. Of course, there are differences in the exact amount of power saved and the impact on performance. They report that almost 90% of disk power consumption can be eliminated. Our results are less optimistic, with power consumption being reduced by 35–50%. They measure the impact on performance differently from us, making it somewhat difficult to compare results. They found that a 2-second timeout caused 15-30 seconds of delay per hour. With the 4-hour Powerbook trace, a 1-Mbyte buffer cache, and a Kittyhawk disk, we found that 388 read accesses had to wait for spin-up. Since in our simulation the Kittyhawk went immediately from the spin-down state to the halt state, spinning up would take approximately 1.5s, for a total of 153 seconds of delay per hour. This is not surprising, since the mean interarrival time in the Powerbook trace was less than a second. With the HP-UX trace, there were about 840 delays over 168 hours for an average delay of 7.5 seconds per hour. This large difference from the Powerbook simulation is due to the long periods of inactivity during the HP-UX trace.

The other differences in our results reflect the traces used to drive the simulation and the disks that were simulated. Like us they used traces from Unix workstations: in particular the Sprite traces from the recent Berkeley study [7]. Unlike us they used traces from DOS PCs. They simulated a Maxtor drive; we simulated an HP Kittyhawk drive and a Quantum Go•Drive.

As mentioned above in Section 3.3, John Wilkes at Hewlett-Packard proposed a predictive algorithm for disk management [17]. He suggested adjusting spin-down timeouts based on a weighted average of recent interarrival times. Picking the weights may be a difficult task: we attempted to implement this strategy but were unable to out-perform THRESHOLD_DEMAND with the particular algorithms we tried. To the best of our knowledge, no other implementation of this strategy has been attempted.

In other work, Greenawalt [8] did an analytic study of disk management strategies. He assumed a Poisson process for request arrival; this is a questionable assumption given the clustering that tends to occur in real workloads. He considered two synthetic workloads, depending on the interarrival rate. He defined the "critical rate" as the number of accesses per unit time at which it is more power efficient to leave the disk spinning than to spin it down. His analysis is useful as an off-line policy; an on-line policy must be able to respond to and anticipate changes in the request arrival rate, something not addressed in this paper. Finally, like Li, *et al.*, Greenawalt studied the impact of the spin-down timeout on the reliability of the drive, an issue which we do not consider.

Finally, some commercial products have recently begun to address this issue. For example, while the standard Apple Powerbook control panel only allows the user to choose broadly between "maximum conservation" and "maximum performance," the Connectix Powerbook Utilities (CPU) [4] provide fine-grained control over such details as the disk spin-down threshold and processor speed, as well as feedback on the current state of the disk (such as a count-down to when the disk will spin down). Personal experience with CPU shows that a short spin-down delay, on the order of several seconds, does extend battery life at the cost of increased disk spin-ups and correspondingly slow response. The poor response time was sufficient to justify increasing the spin-down threshold from 5s to 15s in order to make the execution environment acceptable.

## 7 Conclusions and Future Work

We have simulated techniques for minimizing the power consumption of hard disks on mobile computers by spinning the disk only when necessary. Our off-line policy, OPTIMAL_OPTIMAL, demonstrates that significant power savings are possible: it can reduce disk power consumption by 35–50%, compared to the fixed threshold suggested by manufacturers. Our THRESHOLD_DEMAND policy demonstrates that it is possible for practical policies to get power savings close to that of the off-line policy. Threshold policies that spin down the disk after 1–10 seconds come within 7–27% of the off-line policy, and consume only 56–86% of the energy consumed by manufacturers' recommended thresholds (5 seconds for the Kittyhawk, and 5 minutes for the Go•Drive 120).

Threshold policies with fast spin-down save energy but degrade response time. We do not yet have enough experience to know what the best metric of degradation might be: in this paper we have compared algorithms based on the number of times a read request is delayed, but as noted above, other metrics are possible.

Finally, the impact of hardware design on software parameters is significant. The Kittyhawk was designed to make the transition between the active and idle states much less prohibitive than the Go•Drive or comparable disk drives. As a result, a shorter threshold is appropriate for the Kittyhawk than for these other disks, though even the Kittyhawk is susceptible to workloads that will result in too much overhead to make frequent spin-downs feasible. Ultimately a better approach than very short spin-down timeouts may be predictive algorithms that exploit past history, but such predictive algorithms may prove to be elusive.

## Acknowledgements

We are grateful to John Wilkes and Hewlett-Packard Company for making their file system traces available to us. We also thank Brian Bershad, Ramón Cáceres, Hank Korth, and Kai Li, who have provided helpful feedback throughout our research; Bill Sproule, who collected the Powerbook traces and provided information about the Macintosh; and Bruce Zenel, who performed the measurements of power consumption reported in Section 2. Lastly, we thank the USENIX referees for their comments.

## References

[1] Applelink. Developer Support:Developer Talk. AppleLink bulletin board, October 1993.

[2] Mary Baker, Satoshi Asami, Etienne Deprit, John Ousterhout, and Margo Seltzer. Non-volatile memory for fast, reliable file systems. In *Proceedings of the Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 10–22, Boston, MA, October 1992. ACM.

[3] Ramón Cáceres, Fred Douglis, Kai Li, and Brian Marsh. Operating Systems Implications of Solid-State Mobile Computers. In *Proceedings of the Fourth Workshop on Workstation Operating Systems*, pages 21–27. IEEE, October 1993.

[4] Connectix Corporation, San Mateo, CA. *CPU Connectix PowerBook Utilities Version 2.0 Addendum*, April 1993.

[5] Dell Computer Corporation. *Dell System 320SLi User's Guide*, June 1992.

[6] Intel Corporation/Microsoft Corporation. *APM Specification Version 1.0.*

[7] Mary Baker et al. Measurements of a distributed file system. In *Proceedings of the 13th Symposium on Operating System Principles*, pages 198–212, Pacific Grove, CA, October 1991. ACM.

[8] Paul Greenawalt. Modeling Power Management for Hard Disks. In *Proceedings of the Symposium on Modeling and Simulation of Computer and Telecommunication Systems*, 1994. To appear.

[9] Hewlett-Packard. *Kittyhawk HP C3013A/C3014A Personal Storage Modules Technical Reference Manual*, March 1993. HP Part No. 5961-4343.

[10] Hewlett-Packard. Kittyhawk power management modes. Internal document, April 1993.

[11] Hewlett-Packard Company, Corvallis Division, Corvallis, OR. *Omnibook 300 Operating Guide*, 1 edition, April 1993.

[12] Kester Li, Roger Kumpf, Paul Horton, and Thomas Anderson. A Quantitative Analysis of Disk Drive Power Management in Portable Computers. In *Proceedings of the 1994 Winter USENIX*, 1994. To appear.

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

[14] Quantum. *Go•Drive 60/120S Product Manual*, May 1992.

[15] Chris Ruemmler and John Wilkes. UNIX disk access patterns. In *Proceedings of the Winter 1993 USENIX Conference*, pages 405–420, San Diego, January 1993.

[16] Seagate Technology. *Datasheet for Seagate ST7XXA Family: 1.8-Inch IDE FlashDrives.* Scotts Valley, CA, 1993.

[17] John Wilkes. Predictive power conservation. Technical Report HPL-CSP-92-5, Hewlett-Packard Laboratories, February 1992.

[18] Zenith Data Systems, Groupe Bull. *MastersPort 386SL/386SLe Owners Manual*, 1991.

## Trademarks

Powerbook is a registered trademark of the Apple Computer Corporation. DECstation is a registered trademark of Digital Equipment Corporation. Kittyhawk and HP-UX are registered trademarks of Hewlett-Packard Company. Word is a registered trademark of Microsoft Corporation. Eudora is a registered trademark of Qualcomm. Go•Drive is a registered trademark of Quantum. UNIX is a registered trademark of UNIX System Laboratories, Inc.

## Author Information

**Fred Douglis** is a research scientist at the Matsushita Information Technology Laboratory. His research interests include mobile and distributed computing, file systems, and user interfaces. He received a B.S. in Computer Science from Yale University in 1984. He received his M.S. and Ph.D. degrees in Computer Science from the University of California, Berkeley in 1987 and 1990. Reach him electronically at douglis@research.panasonic.com. Reach him via USMail at Matsushita Information Technology Laboratory; 2 Research Way, Third Floor; Princeton, NJ 08540–6628.

**P. Krishnan** is a Ph.D. student at Brown University. His research interests include the development and analysis of algorithms, online and prediction algorithms, prefetching and caching, mobile computing, data compression, databases systems and operating systems. He received his B. Tech in Computer Science and Engineering from the Indian Institute of Technology, Delhi, in 1989, and his Masters in Computer Science at Brown University in 1991. He is currently visiting Duke University. Reach him electronically at pk@cs.brown.edu. Reach him via USMail at Box 90129, Duke University; Durham, NC 27708–0129.

**Brian Marsh** is a research scientist at the Matsushita Information Technology Laboratory. His research interests include operating systems, particularly for mobile computers. He received an A.B. in Computer Science from the University of California, Berkeley in 1985. He received his M.S. and Ph.D. degrees in Computer Science from the University of Rochester in 1988 and 1991. Reach him electronically at marsh@research.panasonic.com. Reach him via USMail at Matsushita Information Technology Laboratory; 2 Research Way, Third Floor; Princeton, NJ 08540–6628.