Adaptive Disk Spin-down Policies for Mobile Computers*

Fred Douglis†
AT&T Bell Laboratories
P. Krishnan
Brown University
Brian Bershad
University of Washington

Abstract

Mobile computers typically spin down their hard disk after a fixed period of inactivity. If this threshold is too long, the disk wastes energy; if it is too short, the delay due to spinning the disk up again frustrates the user. Usage patterns change over time, so a single fixed threshold may not be appropriate at all times. Also, different users may have varying priorities with respect to trading off energy conservation against performance. We describe a method for varying the spin-down threshold dynamically by adapting to the user's access patterns and priorities. *Adaptive spin-down* can in some circumstances reduce by up to 50% the number of disk spin-ups that are deemed by the user to be inconvenient, while only moderately increasing energy consumption.

1 Introduction

In today's mobile computers, the hard disk is typically spun down after a fixed period of inactivity in order to conserve energy. When the disk is next accessed, it is spun up again, which can cause a delay of a few seconds. This spin-up delay may be acceptable to the user, who knows that the delay is in exchange for extending battery life, or it may be bothersome. We have developed a method for distinguishing between *undesirable* and *acceptable* spin-up delays and varying the idle-time threshold for spinning down the disk based on the user's tolerance for undesirable delays. We term this method *adaptive disk spin-down*, and henceforth refer to undesirable delays as "bumps" for simplicity.

A good adaptive policy will reduce the number of bumps without adversely affecting energy consumption compared to a fixed-threshold policy; or it will reduce energy consumption without adversely affecting the number of bumps. In the best case, it will improve both of these metrics; however, as explained in Section 5.6, that is difficult to achieve.

The rest of this paper is organized as follows. The next section discusses disk spin-down in greater detail. Section 3 describes adaptive spin-down and defines some terminology. In Section 4, we describe the experiments we performed, and in Section 5 we report their results. Section 6 summarizes related work, and Section 7 concludes.

2 Background

2.1 Disk Spin-Down

The functionality of mobile computers is limited by the amount of time they can operate on a single battery charge. Most mobile computers use magnetic disk drives, which can consume 20–30% of total system power, or more [5]. The power consumed by the disk subsystem can be reduced by spinning the disk only when necessary. All systems of which we are aware use a *fixed-threshold policy* to spin down the disk: if the disk has not been accessed in T seconds it is spun down. The disk is spun up again the next time it is accessed, which delays the access by 1–2s or more while the disk readies itself.

In general, a spin-down policy has two conflicting goals: reducing energy consumption and preserving interactive response. Spinning down the disk after a short period of inactivity can decrease energy consumption but will also result in more delays due to spin-up. Thus the fixed threshold T is typically on the order of many seconds or minutes to minimize the delay from on-demand disk spin-ups. Another reason for a large fixed threshold is that is possible to consume so much energy due to the spin-ups that overall energy consumption increases with a shorter threshold. After being spun down, the disk must stay spun down for a period of time in order to amortize the spin-up overhead. The break-even point, T_d, de-

*This work was performed in part at the Matsushita Information Technology Laboratory of Panasonic Technologies, Inc.
†Email: douglis@research.att.com

pends on the time and energy consumed in each state and can be statically calculated based on the parameters for a disk [5]. A fixed threshold shorter than T_d can in some cases increase energy consumption, and the closer the threshold gets to spinning down the disk immediately, the greater the likelihood of increasing rather than decreasing energy consumption.

The Hewlett-Packard Kittyhawk C3014A spins down and up again in about 3s, and its manufacturer recommends spinning it down after about 5s of inactivity [8]; most other disks take several seconds for spin-down/spin-up and are recommended to spin down only after a period of about 5m [3, 19]. The Quantum Go•Drive in most Macintosh PowerBooks takes approximately 5s to spin up, and the PowerBook by default allows the drive to be spun down after 30s to 15m of inactivity, or not at all. Commercial products such as Connectix PowerBook Utilities [2] allow finer-grained control over the threshold, however.

The main intuition underlying the variations among disk spin-down policies is identifying periods of inactivity at disk that are "sufficiently large." Fixed-threshold policies wait a constant period of time to be sure that the period of inactivity is large enough. Compared to thresholds of 30s or more usually recommended by manufacturers, aggressive fixed-threshold spin-down policies—those that spin down the disk after a relatively short period of inactivity—often consume less energy because they observe inherent characteristics of disk interarrival times when typical activities are performed by the user [5, 14]. Usually, either the disk is accessed repeatedly in a short time-span, or it is idle longer than $T_d + \Delta$, where Δ is the time one waits before spinning down the disk. The greater the disk interarrival times, the more effective an aggressive spin-down policy can be.

2.2 Caching

A simple method that increases the interarrival times at disk and helps nearly[1] any disk spin-down policy to decrease energy consumption is to reduce the number of activities at disk via caching or buffering of I/O. Both DRAM caching and SRAM buffering affect which I/Os go through to the disk. DRAM serves as a cost-effective cache for read-only data and can dramatically reduce energy consumption and improve performance [5, 14]. It does have the potential to cause additional spin-up delays compared to a configuration with less DRAM caching, because misses in the DRAM cache are more likely to find a spun-down disk.

SRAM can absorb writes to disk, which decouples disk latency from application performance when a synchronous write can be performed to SRAM rather than to disk [1]. It can reduce energy consumption by avoiding writes to disk completely if the same blocks are frequently overwritten. If we assume that SRAM is completely recoverable in case of a system crash or other failure, I/Os to SRAM need go to disk only when SRAM is full. In this case, writes to disk can sometimes be completely eliminated (for example, when data blocks are overwritten), and the disk may spin down when it otherwise would have been accessed for writing. As with DRAM caching, subsequent operations (reads or writes) may be delayed by a disk spin-up.

One feature of deferring writes to disk indefinitely is that if the disk is currently spun down, a write that fits in the SRAM buffer need not spin up the disk at all. While many disks use SRAM as a write buffer, we know of only one, the Quantum Daytona, that buffers writes to a spun-down disk rather than spinning it up again. Because aggressive spin-down policies and relatively short spin-up delays will result in the disk being spun down more often than on past systems, and deferring small writes will be extremely important, we consider a Daytona-style disk drive in this study. We show the impact of deferred writes later in this paper by comparing it to a policy that quickly writes blocks from SRAM to disk.

Although the discussion in this paper focuses on mobile computers, which have limited battery life on a single charge, desktop computers can benefit from these techniques as well. Manufacturers are striving to provide low-power desktop machines [7]; spinning down a disk on a desktop computer is one necessary aspect of the EPA Energy Star Computers Program [9]. Adding SRAM to a workstation can improve performance and reduce server load [1].

2.3 Evaluation Metrics

There are several possible metrics by which one may evaluate a spin-down policy. One simple metric is *least energy*, which optimizes energy savings with no regard to spin-up delays. At the other extreme is the policy that minimizes spin-up delays. In the absence of future knowledge of accesses, the latter policy is one that never spins down the disk, minimizing delays but usually consuming substantially more energy.

Numerous metrics lie between these two extremes. One possible method is to distinguish between spin-up delays that the user finds acceptable and those that severely inconvenience the user. Towards this end, we in-

[1] There are situations when caching can eliminate just enough I/Os that the disk spins down due to inactivity, but not enough to amortize the cost of the spin-down. In this case, the overhead from spinning up the disk can increase over-all energy consumption.

troduce the notion of *undesirable spin-ups*, or *bumps*, and formalize a possible definition of bumps in Section 3.1. As an example, a user who must wait for the disk to spin up when accessing the computer for the first time in 30m should find a short delay an acceptable cost of saving the energy needed to spin the disk for the entire time. By comparison, most users will be irritated if they must wait a few seconds for the disk to spin up when it has only been idle a few seconds. Compared to the energy-optimal spin-down threshold, a small increase in the threshold T can substantially decrease the number of spin-ups relative to the increase in energy consumption.

3 Adaptive Spin-down

We define *adaptive spin-down* as a policy that monitors the spin-down threshold and adjusts it to keep a balance between energy consumption and bumps. In this paper we consider a policy that attempts to keep the number of bumps within a tolerable range (it need not necessarily be zero). The user defines what is acceptable and what is undesirable. We first refine the notion of an undesirable spin-up and then describe how to adjust the spin-down threshold dynamically.

3.1 Undesirable Spin-ups

There are many different ways of defining acceptable spin-ups; in this paper, we define our measure of acceptability as a function of the ratio ρ between the spin-up delay δ and the idle time I of the disk prior to the spin-up. A spin-up delay is *unacceptable* (i.e., a bump) if $\delta > \rho I$. A value of $\rho = 0$ indicates that all spin-up delays are considered bumps.

A way to ensure that there are *no* bumps is to use a fixed spin-down threshold $T = \delta_M / \rho$, where δ_M is the maximum delay possible (including any overhead due to spinning down the disk first, if the I/O occurs just as the disk starts to spin down). In other words, one would wait until the disk has been idle long enough that spinning it up again cannot be perceived as an inconvenience. While this policy will satisfy any metric that tries to minimize the number of bumps, it may consume much more energy than a policy that allows a small number of bumps.

3.2 Threshold Adjustment

Here we describe a software approach to adaptive disk spin-down. *Threshold adjustment* is at the heart of adaptive spin-down. The adjustment may take place at various times: when a bump occurs, when an acceptable spin-up occurs, or when other information suggests the need to change the threshold.

- When a bump occurs, the threshold was too short and should be increased.
- When a spin-up is acceptable, the threshold was long enough. It can possibly be decreased without increasing the number of bumps.
- There may be other times when the threshold should be changed even though no spin-up has occurred. For instance, if the disk is idle I seconds and the current spin-down threshold T is just greater than I, i.e., $I < T < (1+\epsilon)I$, then the disk will not be spun down. But a slight variation in the time of the next disk I/O could cause the disk to be spun down, after being idle for more than T seconds, only to be spun up again immediately. Thus if there is a *close call*, in which the spin-down threshold was barely high enough to avoid a situation in which the disk would be spun down and back up again in a short time, it may be appropriate to increase the threshold.

There are many possible approaches to, and enhancements of, adaptive spin-down. They include:

Rate of adjustment By how much is the threshold adjusted when a spin-up occurs? The formula for adjusting the threshold can be arbitrarily complex. One example is to add two different values, α_a and β_a respectively,[2] to T when undesirable or acceptable spin-ups occur. One would expect that $\alpha_a > 0$, $\beta_a < 0$, and $\alpha_a > |\beta_a|$. In other words, when a bump occurs, the spin-down threshold should be increased by enough to make future bumps significantly less likely; when an acceptable spin-up takes place, the spin-down threshold should be decreased, but more gradually.

Another related method is to multiply T by α_m and β_m respectively. Here one expects that $\alpha_m > 1$ and $1 > \beta_m \geq 1/\alpha_m$. This assumption is examined more closely in Section 5.2.

Restrictions on threshold In order to avoid pathological behavior, adjustments to the spin-down threshold may apply only within certain ranges. The minimum spin-down threshold may be 0s (spin the disk down immediately upon each access), or it may be positive in an attempt to keep the adjustment process from "overdoing" its compensating behavior. The minimum value is especially important if ρ is high, since there is an implicit assumption above that the threshold can be decreased whenever an acceptable spin-up occurs. This assumption will not

[2] The subscript a denotes an additive adjustment and the subscript m denotes a multiplicative adjustment.

	MAC			DOS			HP		
Applications	Finder, Excel, Newton Toolkit			Framemaker, Powerpoint, Word			email, editing		
Duration	3.5 hours			1.5 hours			4.4 days		
Number of distinct Kbytes accessed	22000			16300			32000		
Fraction of reads	0.50			0.24			0.38		
Block size (Kbytes)	1			0.5			1		
Mean read size (blocks)	1.3			3.8			4.3		
Mean write size (blocks)	1.2			3.4			6.2		
Inter-arrival time (s)	Mean	Max	σ	Mean	Max	σ	Mean	Max	σ
	0.078	90.8	0.57	0.528	713.0	10.8	11.1	30min	112.3

Table 1: Summary of trace characteristics. The statistics apply to the 90% of each trace that is actually simulated after the warm start. Note that it is not appropriate to compare performance or energy consumption of simulations of different traces, because of the different mean transfer sizes and durations of each trace. (This table is reproduced from [4].)

hold if spinning down too quickly results not only in many spin-up delays (which the user has said are acceptable) but also an increase in over-all energy consumption.

Similarly, bumps may increase the threshold indefinitely, or they may be ignored after some point. As mentioned above, there is no point to increasing the threshold beyond δ_M/ρ if one is trying to minimize bumps.

Time of adjustment The method described above adjusts the threshold upon every spin-up event, as well as after "close calls." This approach can be modified, for example, to decrease the spin-down threshold only when several spin-ups in a row are acceptable.

Variable penalties Thus far, the description of adaptive spin-down has assumed that either a spin-up is acceptable, or it is not. There is in fact a continuum of degrees of acceptance: spinning up just after the disk has spun down is worse than spinning up just before the point at which the spin-up delay would be deemed acceptable. In fact, the *worst* type of spin-up is one that occurs before the disk has completely spun down, since the delay will be greater (the disk must typically be fully spun down before the order to spin up can be issued). An improvement on the method above, therefore, is to increment the spin-down threshold by a greater amount when the delay due to a spin-up is especially egregious than when it is barely above the user's threshold.

4 Experiments

The effectiveness of adaptive policies depends on a number of factors: workload, hardware characteristics (i.e., disk parameters, DRAM size, and SRAM size), user perception (i.e., the acceptability ratio), and threshold adjustment ranges and modifiers. We performed several experiments to quantify the effect of these factors. To keep the study manageable, we fixed DRAM at 1 Mbyte and SRAM at 32 Kbytes except where noted.

To study workload, we simulated both adaptive and fixed-threshold spin-down policies on three traces used in a previous study of mobile storage management: a 3.5-hour Macintosh PowerBook trace, a 1.5-hour Windows 3.1 trace, and a 4.4-day HP-UX trace [4]. Table 1, reproduced from that study, summarizes information about the traces. In addition, the MAC trace has a distinctive quality that has a significant impact on the effectiveness of the SRAM write buffer: it has a very high locality of write accesses, with 36% of writes going to just one 1-Kbyte block and 24% of writes going to another. Note as well that the HP trace contains disk-level rather than file-level accesses, below the level of the buffer cache, so we do not simulate a DRAM buffer cache when considering the HP trace.

With respect to hardware characteristics, we consider two magnetic disks: a Western Digital Caviar Ultralite CU140, and a Quantum Go•Drive. The CU140 is available with lightweight mobile systems such as the Hewlett-Packard OmniBook 300 PC, and typically spins up the disk in about 1s. However, the disk may take a maximum of 5s to respond after spinning down [17]. Our simulator considers average behavior, so we model the

Additive		Multiplicative	
α_a	β_a	α_m	β_m
2.00	-1.00	1.5	0.5
5.00	-1.00	1.5	0.75
1.00	-0.50	2	0.75
1.00	-0.25	1.25	0.9
2.00	-0.25	1.5	0.9

(a) Adjustment values. The left two columns list the additive values studied in this paper, while the right two list the multiplicative values.

Starting value	Minimum value $T_{\min}$	Maximum value $T_{\max}$
5	2	10
5	2	30
10	5	30
30	10	∞

(b) Ranges of spin-down threshold studied in this paper. Maximal values used for a given configuration depend on the value of δ_M for the disk and are set to $\min(T_{\max}, \delta_M/\rho)$. If the nominal starting value is greater than the maximum it is adjusted to be the mean of the minimum and maximum.

Table 2: Parameters for adaptive spin-down (times in seconds). The cross-product of the sets of parameters was used to drive the simulations; that is, each of the 5 combinations of (α_a, β_a) and 5 combinations of (α_m, β_m) in Table (a) is used with each of the 4 sets of values in Table (b), giving 40 sets of adaptive parameters to drive the simulator.

CU140 by charging 2.5s to spin down the CU140 and 1s to spin it up again. The Go•Drive is a bigger disk and takes 6s to spin down and 2.5s to spin up [15].

We varied the acceptability ratio among three values: 0.02, 0.05, and 0.2. Putting aside for a moment the issue of a request arriving at the disk while it is spinning down, a ratio of 0.05 means that a CU140 would have to be idle for 20s for a spin-up delay of 1s to be acceptable, while for the Go•Drive idle time would normally have to be 50s for the 2.5s delay to be acceptable.[3] Based on subjective personal experience, these idle-time requirements seem like a fair trade-off between delay and energy savings, and we use the ratio of 0.05 as the canonical example of the adaptive approach. A ratio of 0.02 is more restrictive, requiring long idle times to reduce the number of bumps, while a ratio of 0.2 is forgiving: for the CU140, the system need be idle only 5s for a spin-up to be acceptable, though again, accessing the disk while it is spinning down increases the threshold to as much as 17.5s.

Finally, we varied the spin-down policies themselves. The fixed-threshold policies used spin-down thresholds of 2, 5, 10, 30, and 300 seconds. The adaptive ones used additive or multiplicative modifiers within ranges that approximated the fixed-threshold policies. Table 2 shows (a) the modifiers and (b) the starting, minimum, and maximum spin-down thresholds allowed. The values were chosen largely by trial and error; a more formal method of finding an appropriate set of parameters for a given hardware configuration and workload would be useful. Note that the ($\alpha_m = 1.5$,$\beta_m = 0.5$) pair contradicts the assumption in Section 3 about the increment upon a bump adjusting the threshold more rapidly than the decrement upon an acceptable spin-up; this issue is considered in Section 5.2 below.

For DRAM and SRAM caching, we used the following approach. Except for the HP trace, which has an implicit DRAM buffer cache, we simulate a 1-Mbyte DRAM buffer cache, which is used to satisfy read requests when possible. One tenth of each trace is simulated before statistics are recorded, in order to prime that cache. All writes go into SRAM; if all blocks in SRAM contain data that are not also stored on disk, a disk write must first take place. We use a cleaning policy similar to the flash memory cleaning policy described in [4], in which the simulator attempts to keep a fraction of the SRAM buffer "clean" at all times, but only if the disk is spinning. In the experiments reported here, SRAM blocks are written to disk either when there is no room for new data, or when the disk is already spinning and fewer than 5% of SRAM is available for new writes; blocks are written out until the next user I/O occurs or 10% of SRAM is free. We consider a more aggressive write policy in Section 5.5.

5 Results

To evaluate different sets of parameters, we plot for each trace the count of bumps against energy consumption. We consider general comparisons of adaptive and fixed-threshold policies in Section 5.1, variations in the adjustment values in Section 5.2, variations in the acceptability ratio in Section 5.3, disk parameters in Section 5.4, and DRAM and SRAM sizes in Section 5.5.

[3] If the disk is accessed just after spinning down, the delay could be as high as 3.5s for the CU140 and 8.5s for the Go•Drive, resulting in minimum idle times of 70s and 170s respectively when $\rho = 0.05$.

5.1 Adaptive versus Fixed Thresholds

WINDOWS

Figure 1 shows the effect of spin-down policies on energy consumption and bumps, using the Windows trace on the CU140 disk with $\rho = 0.05$. For this trace/hardware configuration, a fixed threshold of 2s consumes the least energy; this follows from previous studies [5, 14]. However, the short threshold results in over 50 bumps over a 1.5 hour period.

Increasing the fixed threshold or moving to an adaptive policy can decrease the number of bumps in exchange for higher energy consumption. At the extreme case, a spin-down threshold of 30s results in no bumps, but also an increase of 48% in energy consumption. Compared to the 2s threshold, a fixed threshold of 10s decreases bumps by two-thirds and increases energy by just 15%.

Figure 1 demonstrates that adaptive policies span a roughly linear range of points between the fixed-threshold policies of 5s and 30s thresholds, depending on the parameters of the adaptive policy. Generally they are in the “desirable region” (below or to the left of comparable fixed-threshold points).

For instance, consider an adaptive policy that varies its threshold between 5 and 30s, increasing the threshold by $\alpha_a = 2$s upon a bump and reducing it by $\beta_a = 1$s upon an acceptable spin-up. (This point is marked by the solid arrow in Figure 1.) This configuration increases energy by just 8% compared to the energy-optimal 2s fixed threshold, and decreases the number of bumps by 65%.

As another example (shown as the dashed arrow), consider an adaptive policy that varies the threshold between 5s and 30s, multiplies the threshold by $\alpha_m = 1.5$ on a bump, and multiplies it by $\beta_m = 0.5$ on an acceptable spin-up. Compared to the 5s fixed threshold policy, this adaptive policy consumes just 0.5% more energy while reducing bumps by a third. Each of the other multiplicative policies within the 5–30s range consumes slightly more energy and encounters slightly fewer bumps, except for the ($\alpha = 1.25, \beta = 0.9$) point, which both consumes more energy and encounters more bumps. This point is shown by the unfilled circle above and to the right of the point shown by the dashed line.

MACINTOSH

Figure 2 shows the Macintosh trace running on the CU140, with $\rho = 0.05$. In this case, except for a couple of points, adaptive spin-down does not provide a demonstrable improvement beyond the ability to interpolate and choose an arbitrary point between different fixed thresholds. With a fixed threshold of 5s, the disk consumes about 6250J and hits 370 bumps. One multiplicative adaptive policy (indicated by the arrow), varying between 2–10s, consumes 6300J (1% more) and reduces bumps to 290 (22% less). But many of the adaptive points cluster near the fixed-threshold curve, consuming about 10% less energy with a small increase in bumps, relative to the curve defined by the simulated fixed-threshold points.

HP

Figure 3 shows the HP trace running on the CU140, with $\rho = 0.05$. At the point where the fixed spin-down threshold is low (2–10s), the adaptive policies follow the fixed curve closely. However, this figure shows that rather than a fixed 10s threshold, one should use an adaptive policy (varying between 10–70s, $\alpha_a = 2$, $\beta_a = -0.2$, indicated by the arrow) that would reduce bumps from around 200 to 100 with only a 3% increase in energy. Moving to a fixed 30s threshold reduces bumps by a similar amount but increases energy more (18%).

5.2 Adjustment Values

Figures 1–3 have shown basic comparisons between adaptive policies, grouped into ranges within which their thresholds are allowed to vary, and fixed-threshold policies. By comparison, Figure 4 graphs bumps against energy consumption for the Windows trace with the adaptive policies grouped by adjustment value rather than range (as was done in Figure 1) . The additive policies are shown with solid marks and the multiplicative ones are shown with unfilled marks.

In this case, the multiplicative policies tended to result in points closer to the upper left region of the graph, i.e., similar to a fixed threshold of 2–10s, while the additive policies tended more toward the center of the graph (comparable to 5–30s). The clear triangles, which show modifiers that multiply the threshold by $\alpha_m = 1.5$ upon a bump and by $\beta_m = 0.5$ upon an acceptable spin-up, support the argument made in Section 3 that the penalty for a bump should outweigh the adjustment when a bump does not occur. When the threshold range is restricted to be low (2–10s or 5–10s), the ($\alpha_m = 1.5$,$\beta_m = 0.5$) policy increases bumps without a corresponding decrease in energy, compared to both additive and other multiplicative parameters. When the range is closer to the right of the graph, the distinction is not as clear, but compared to the ($\alpha_m = 1.5$,$\beta_m = 0.9$) policy, the ($\alpha_m = 1.5$,$\beta_m = 0.5$) policy increases bumps by 30% while reducing energy just 2%.

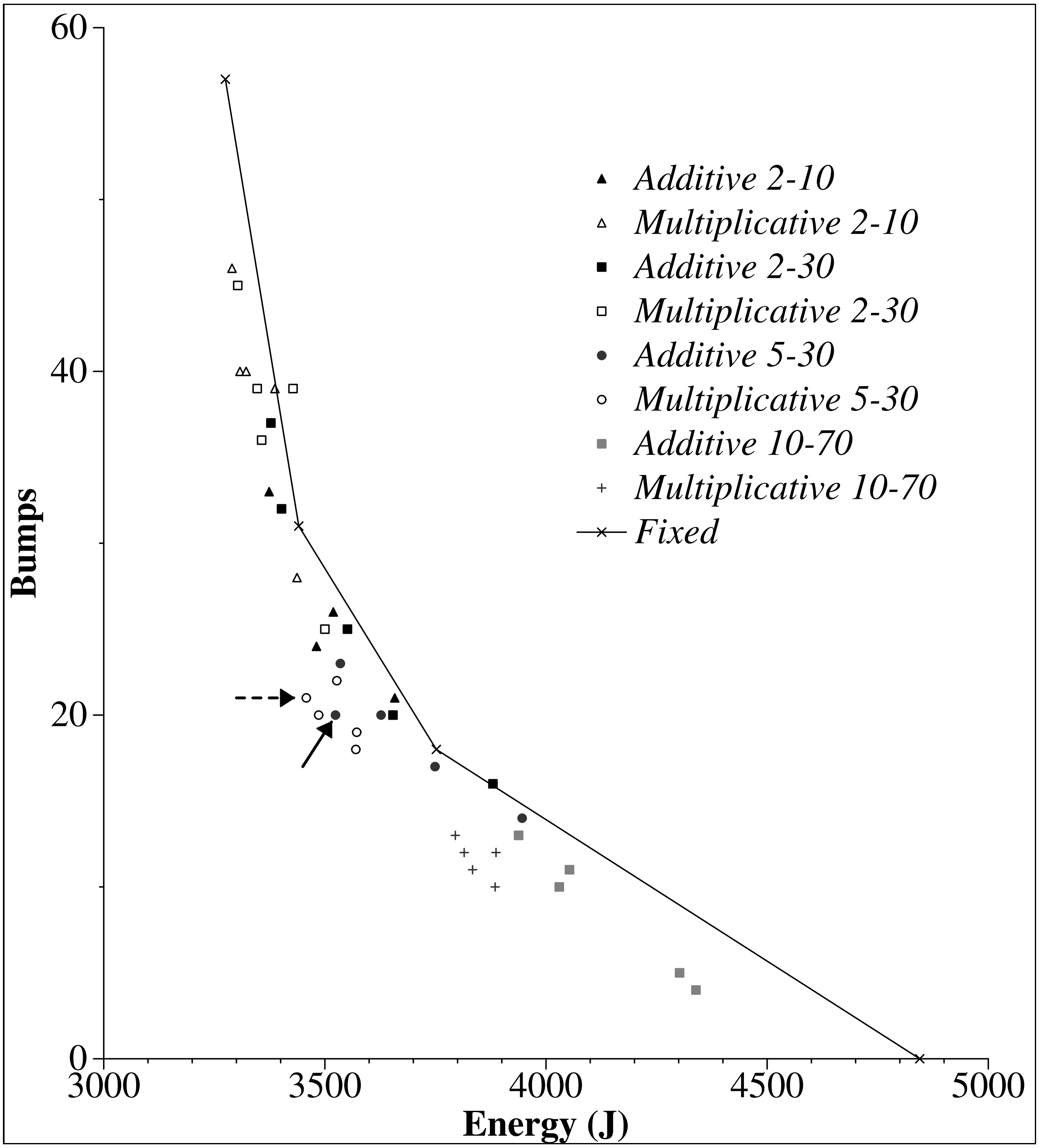


Figure 1: Simulation results comparing bumps and energy consumption, for adaptive and fixed-threshold spin-down policies and an acceptability ratio of 0.05, run on the **Windows** trace and a CU140 disk. In this and subsequent figures, a line connects **x**-marks that represent fixed-threshold policies. Here they show thresholds of 2s, 5s, 10s, and 30s (increasing from left to right). A fixed threshold of 300s, not shown, consumes about 10600J with 0 bumps, compared to 4800J and 0 bumps for the 30s threshold. The adaptive configurations varied in the minimum and maximum spin-down threshold allowed (e.g., 2–30s) and the adjustments to the threshold as shown in Table 2. They are grouped by their allowable ranges, and here the effect of differences in adjustment values within a range is not shown. The arrows point out specific points discussed in the text.

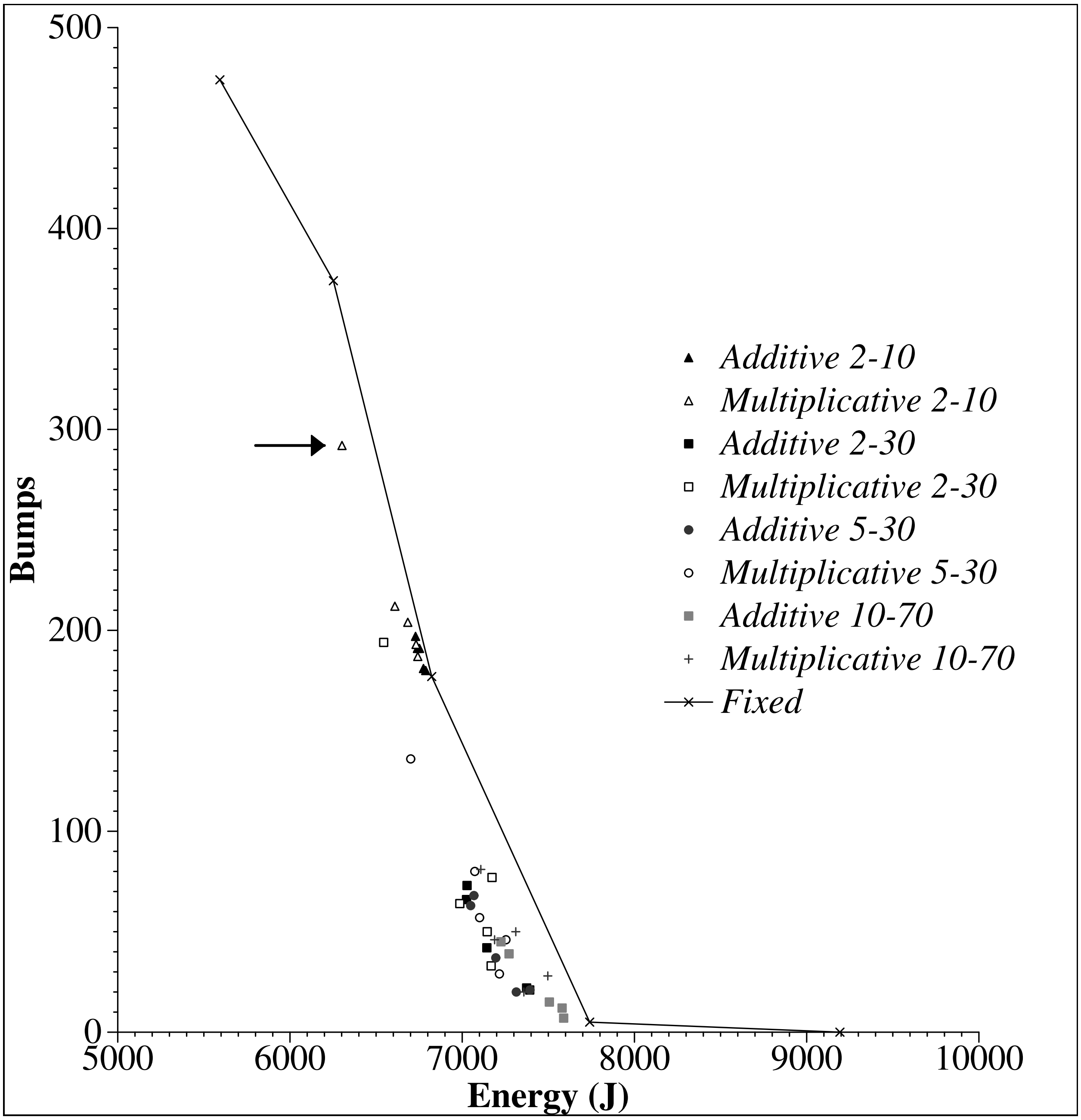


Figure 2: Simulation results comparing bumps and energy consumption, for adaptive and fixed-threshold spin-down policies and an acceptability ratio of 0.05, run on the **Macintosh** trace and a CU140 disk. The line with **x**-marks represents fixed-threshold policies using thresholds of 2s, 5s, 10s, 30s, and 300s (increasing from left to right).

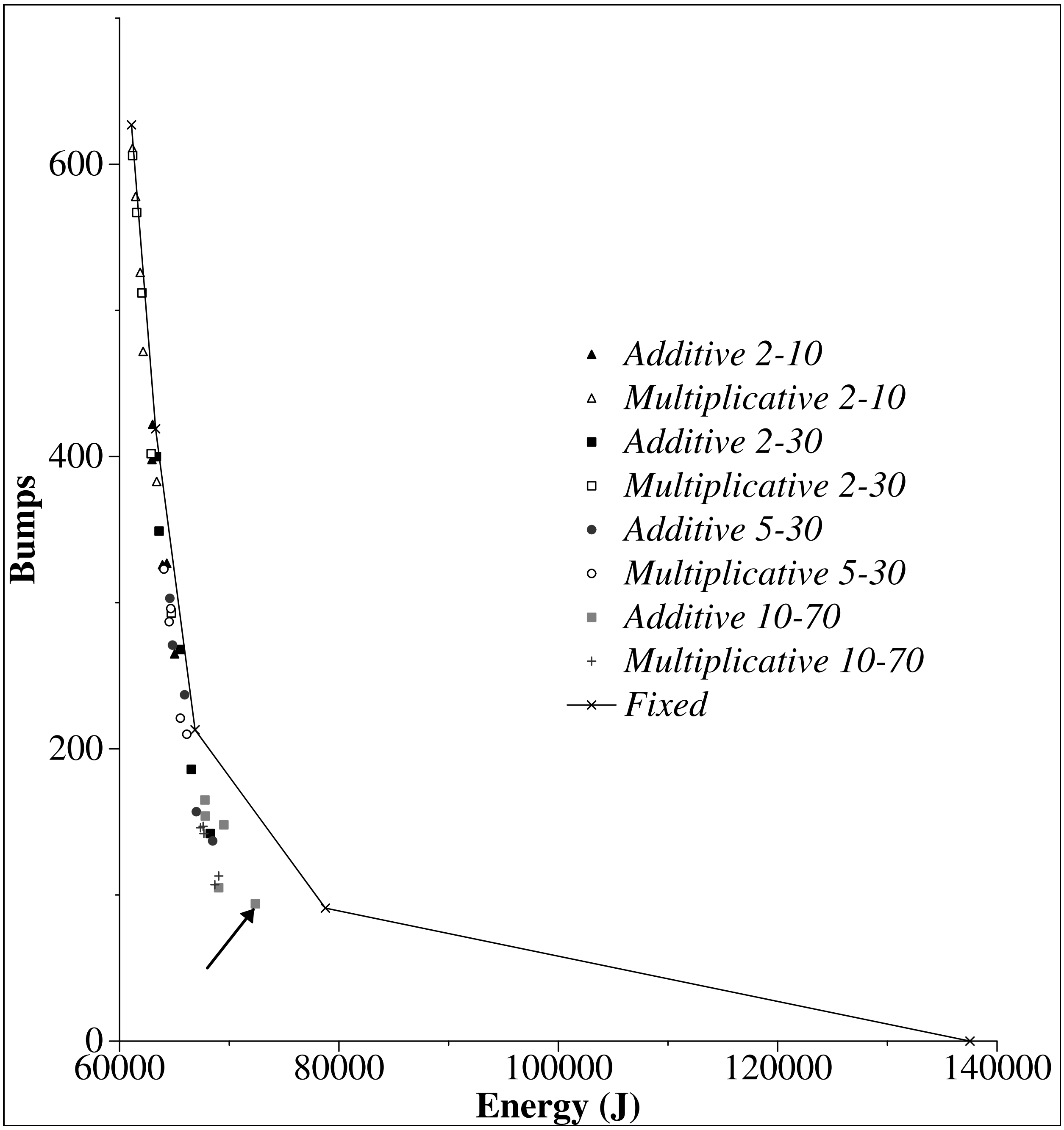


Figure 3: Simulation results comparing bumps and energy consumption, for adaptive and fixed-threshold spin-down policies and an acceptability ratio of 0.05, run on the **HP** trace and a CU140 disk. A line connects **x**-marks that represent fixed-threshold policies with thresholds of 2s, 5s, 10s, 30s, and 300s (increasing from left to right).

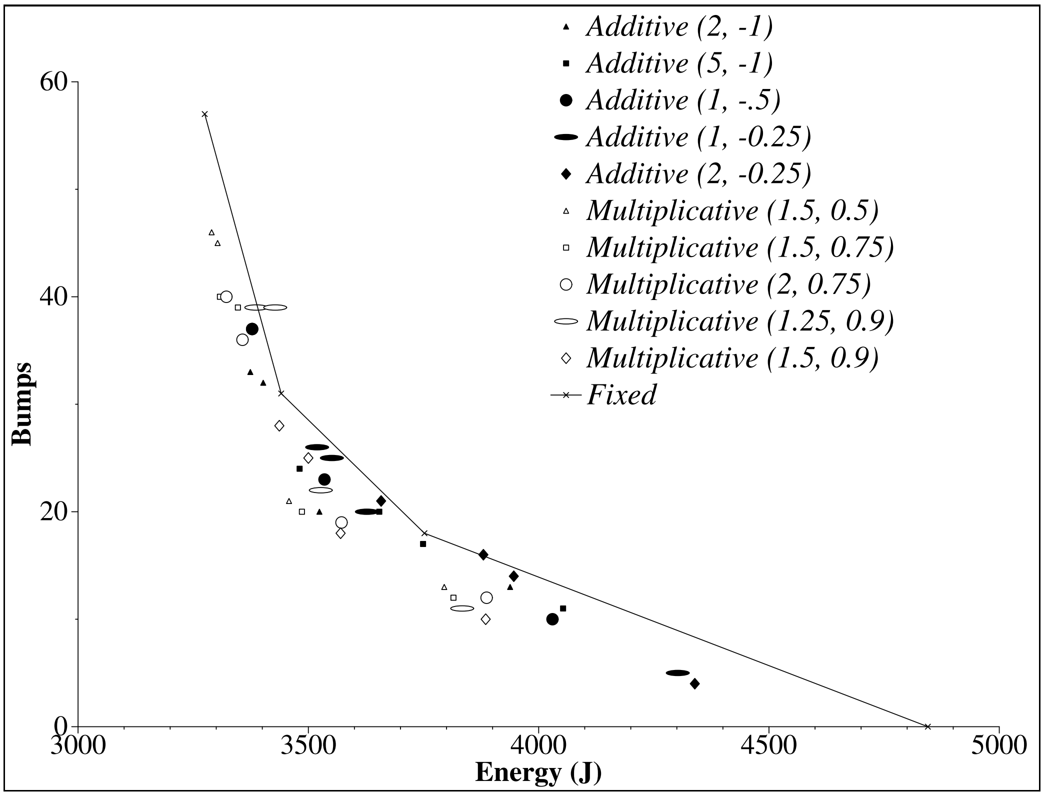


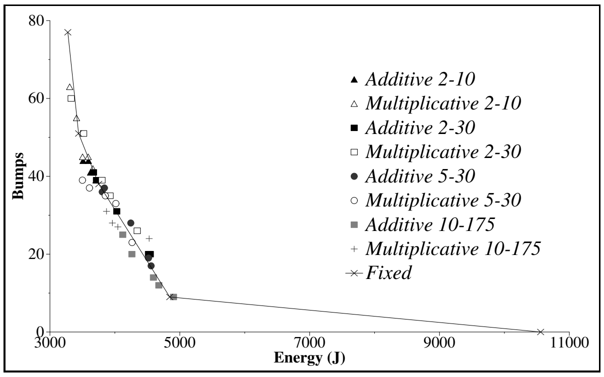
Figure 4: Simulation results comparing bumps and energy consumption, for adaptive and fixed-threshold spin-down policies and an acceptability ratio of 0.05, run on the **Windows** trace and a CU140 disk. Here the points are grouped by the *adjustment values* rather than the ranges over which the adaptive policies varied. The line connecting **x**-marks represents fixed-threshold policies with thresholds of 2s, 5s, 10s, and 30s (increasing from left to right); the 300s threshold is omitted. The two values following the policy name specify α and β.

5.3 Varying ρ

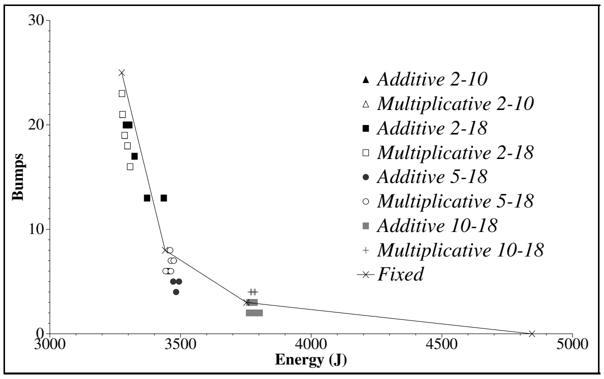
The graphs in Section 5.1 compare adaptive and fixed-threshold policies when the acceptability ratio is 0.05. Figure 5 graphs bumps versus energy consumption for each trace with $\rho = 0.02$ and $\rho = 0.2$. Each pair of graphs for a single trace may be compared with each other and with the graph for $\rho = 0.05$ given previously. (The level of detail in these graphs permits the reader to discern trends among the different traces and policies, but not individual points. Points of interest are discussed in the text, and the raw data for the graphs are available as discussed at the end of the paper.)

For the Windows trace, decreasing ρ results in the adaptive points clustering more closely along the line connecting the fixed-threshold points, but the points farthest to the left (corresponding to fixed thresholds of 2–5s) show adaptive points with 20% reductions in bumps and no increase in energy. Increasing ρ results in the greatest advantage for the adaptive policies: for example, a 2-second fixed threshold results in 3280J and 25 bumps, while one adaptive policy results in 3310J (1% more) and 16 bumps (36% less). A 5-second threshold (3440J, 8 bumps) similarly compares to an adaptive one (3480J, 4 bumps). Of course, the graph shows that there are other adaptive points that are closer to the fixed-threshold results.

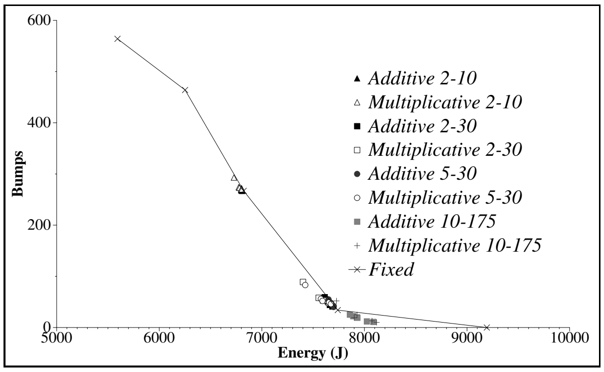
The Macintosh and HP traces show similar effects from changes in the acceptability ratio. Decreasing the ratio, thereby making bumps more common, results in adaptive policies coming fairly close to the fixed-threshold policy. In each case, the jump from 30s to 300s decreases the number of bumps to 0 at a substantial increase in energy consumption, while the adaptive points track the original curve (with fixed thresholds ranging from 2s to 30s). Increasing the acceptability ratio, however, results in adaptive points that represent significant decreases in the number of bumps without significant increases in energy consumption. The Macintosh trace is of particular interest, because the number of bumps barely drops when the fixed threshold moves from 5s to 10s,



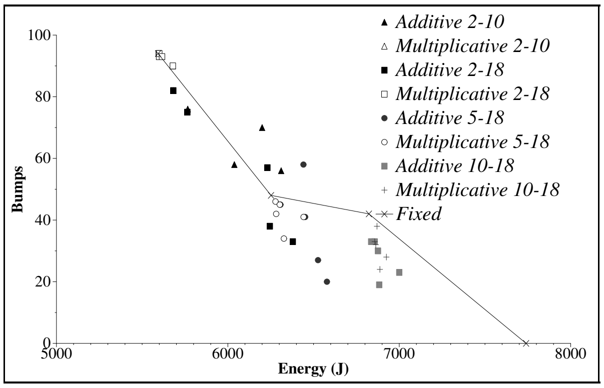
(a) Windows, $\rho = 0.02$



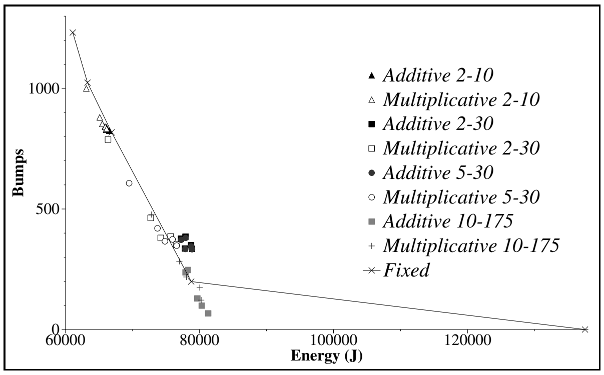
(b) Windows, $\rho = 0.2$



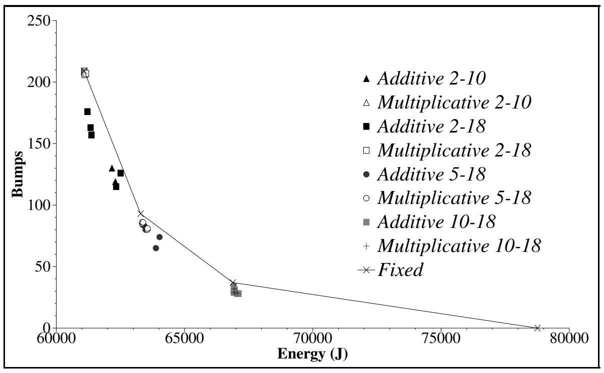
(c) Macintosh, $\rho = 0.02$



(d) Macintosh, $\rho = 0.2$



(e) HP, $\rho = 0.02$



(f) HP, $\rho = 0.2$

Figure 5: Simulation results varying ρ, for each trace. Other parameters are as described in the preceding figures. The scale of each axis varies to show the graphs in the greatest detail. For each trace, when $\rho = 0.2$, there are no bumps with a fixed 30s threshold, and the 300s fixed-threshold run is omitted.

though several adaptive points closely follow the curve one would extrapolate from the 2s and 5s thresholds.

5.4 Varying the Disk

The simulations presented above are based on the Caviar Ultralite CU140, which spins down and up quickly by comparison to some others, such as the Quantum Go•Drive. Figure 6 graphs bumps versus energy consumption for the Macintosh trace with the Go•Drive when $\rho = 0.05$, and it shows an interesting effect of the interaction between disk interarrival times and spin-up costs. For this configuration, the fixed-threshold policy with minimal energy consumption used a 30-second spin-down threshold, rather than 2s. As expected, with a long fixed threshold, the number of bumps is relatively low (34 in this case). Moving to a 10-second threshold increases energy by 4% and bumps by nearly a factor of 8. The increase in energy is a result of the overhead of spinning up the Go•Drive: the break-even point T_d for this disk is 14.9s [5], and spinning it down after 10s of inactivity reduces energy only if it will not be accessed for *another* 15s. Shorter thresholds of 2s and 5s also increased energy consumption by 7% and 9% respectively, compared to the 30s threshold, and increased bumps by factors of 17 and 14 respectively. On the other hand, the adaptive policies generally did about as well as the 30s fixed-threshold policy, or better. The ones that were constrained to spin down within 2–10s increased bumps by as much as 100% without a substantial drop in energy consumption, but the adaptive policies that ranged from 10–175s reduced the number of bumps by up to a third with only a 2–3% increase in energy consumption.

The Macintosh trace is anomalous in that a small (32-Kbyte) write buffer and moderate (1-Mbyte) DRAM cache absorbs enough I/Os to allow the disk to spin down with a short threshold, but not so many that it isn't likely to spin up again quickly relative to T_d. The Windows trace, shown in Figure 7, is more typical, and has a distribution of adaptive points similar to the CU140. For example, one adaptive point represents a 1% increase in energy consumption and a 43% decrease in bumps, compared to the 2s threshold, and a 0.5% increase in energy and 14% decrease in bumps compared to the 5s threshold.

5.5 Caching Effects

As mentioned in Section 2.1, both DRAM caching and SRAM buffering can reduce energy consumption and improve performance.

Figure 8 graphs bumps versus energy consumption for the Macintosh trace, similar to Figure 2 in all ways except DRAM size, which is increased here to 2 Mbytes. Adding DRAM beyond the first Mbyte does not appreciably affect either the fixed-threshold or adaptive policies.

Figure 9 shows the effect of SRAM size on the Macintosh trace. Comparing Figure 9(a), with no SRAM buffer at all, to Figure 2, with a 32-Kbyte SRAM buffer, demonstrates that for this trace a small SRAM buffer dramatically reduces energy,[4] but at the cost of a large number of bumps to get the best energy savings. Both graphs show that adaptive policies improve moderately over fixed-threshold policies, without SRAM size being a great factor. Figure 9(b) indicates that a large SRAM buffer further reduces both energy consumption and bumps, with adaptive policies continuing to show improvements over fixed ones.

We have mentioned that writes to SRAM are deferred when possible. Figure 10 shows the effect on the HP trace of writing SRAM blocks to disk more "aggressively," ignoring the need to spin up the disk, and writing all data through to disk quickly (not just 10% of the SRAM buffer). Here, all modified data goes through to disk at the earliest possible moment, as long as there are not other user I/Os taking place. This figure, when compared to Figure 3, demonstrates that deferring disk writes reduces power consumption but generally results in more bumps when the spin-down threshold is small. This is not surprising: since the writes are not clustered as much, more of them follow a brief period of inactivity. For this particular trace, which has periodic disk writes resulting from the UNIX 30-second *sync* policy, a 30s fixed-threshold policy behaves especially poorly, while the adaptive policies do not suffer from this anomalous behavior.

Changing the SRAM write-back policy for the Windows and Macintosh traces does not generate any pathological thresholds; in fact, for the Windows trace, which is only 20% writes, it has a minimal effect. Figure 11 shows the same experiment as the Macintosh run in Figure 2, but without the deferred writes. For the fixed 2s threshold, energy consumption increases by 67% without deferred writes, but bumps decrease by 80%. But to get down to nearly no bumps requires over 10,000J, compared to under 8000J in the deferred-write case.

5.6 Discussion

The preceding analyses demonstrate that the effectiveness of adaptive policies depends on many factors. Generally speaking, if the user's goal is to minimize energy while paying at least some attention to spin-up delays, the adaptive policies that permit the threshold to go as

[4] This contrasts with the result reported in [4], which did not buffer writes to a spun-down disk as effectively as possible, and showed only about a 20% improvement due to SRAM buffering.

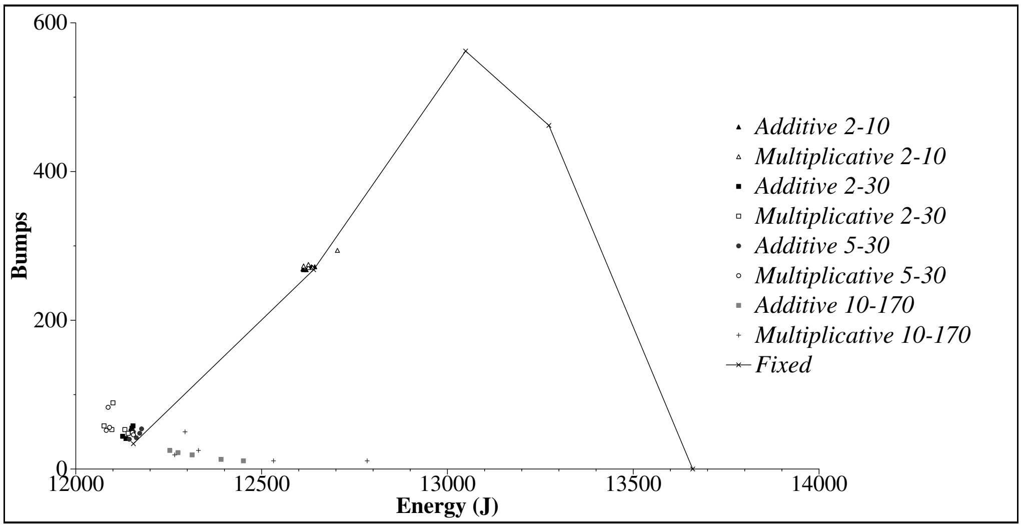


Figure 6: Simulation results comparing bumps and energy consumption, for adaptive and fixed-threshold spin-down policies and an acceptability ratio of 0.05, run on the **Macintosh** trace and a **Quantum Go•Drive** disk. A line connects **x**-marks that represent fixed-threshold policies, but with anomalous behavior: this time the thresholds are 30, 10s, 2s, 5s, and 300s respectively, increasing from left to right.

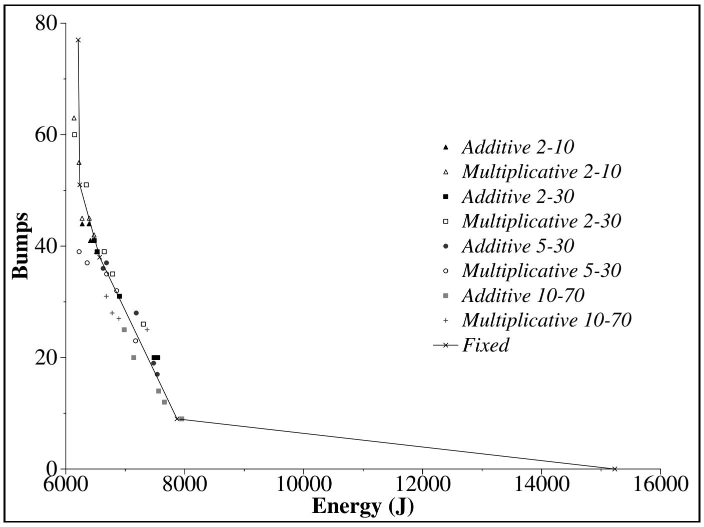


Figure 7: Simulation results comparing bumps and energy consumption, for adaptive and fixed-threshold spin-down policies and an acceptability ratio of 0.05, run on the **Windows** trace and a **Quantum Go•Drive** disk. The line with **x**-marks represents fixed-threshold policies using thresholds of 2s, 5s, 10s, 30s, and 300s (increasing from left to right).

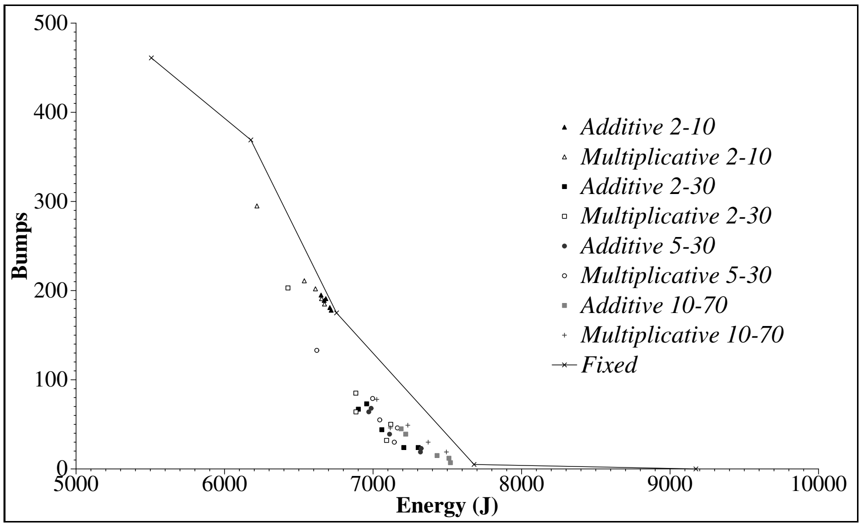
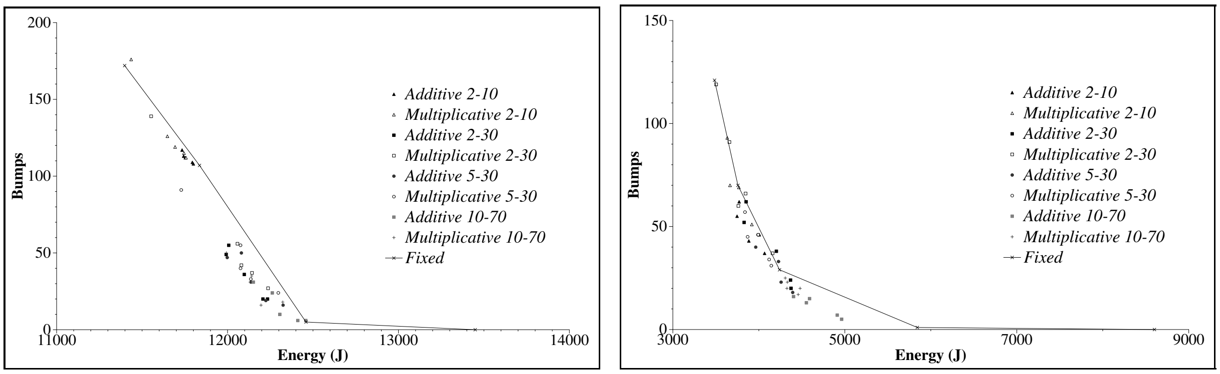


Figure 8: Simulation results comparing bumps and energy consumption, for adaptive and fixed-threshold spin-down policies and an acceptability ratio of 0.05, run on the **Macintosh** trace, a CU140 disk, and **2 Mbytes** of DRAM. The line with **x**-marks represents fixed-threshold policies using thresholds of 2s, 5s, 10s, 30s, and 300s (increasing from left to right).



(a) No SRAM. Here, the point with a fixed 2-second spin-down is omitted, as it consumes more energy than a 5s or 10s threshold and results in over 600 bumps.

(b) 1 Mbyte of SRAM.

Figure 9: Simulation results varying SRAM size for the **Macintosh** trace running on the CU140 with $\rho = 0.05$. The line with **x**-marks represents fixed-threshold policies using thresholds of (2s), 5s, 10s, 30s, and 300s (increasing from left to right).

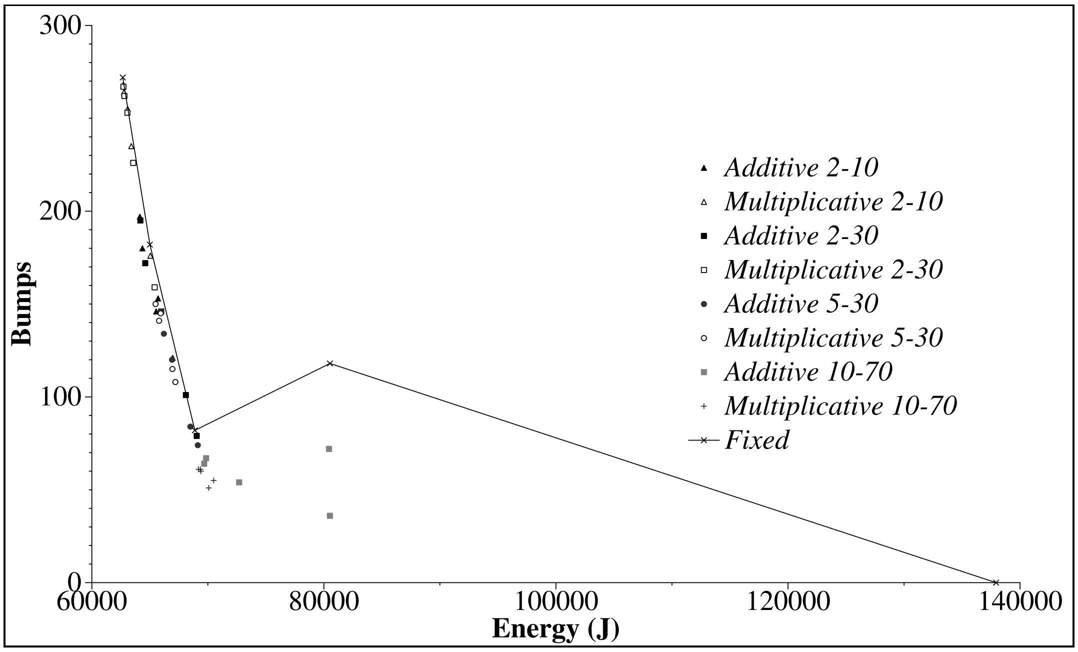


Figure 10: Simulation results for the **HP** trace running on the CU140 with $\rho = 0.05$ and writes to SRAM going through immediately to disk. The line with **x**-marks represents fixed-threshold policies using thresholds of 2s, 5s, 10s, 30s, and 300s (increasing from left to right). Spinning down the disk after 30s of inactivity results in greater energy consumption and more bumps than a 10s threshold, due to periodic writes.

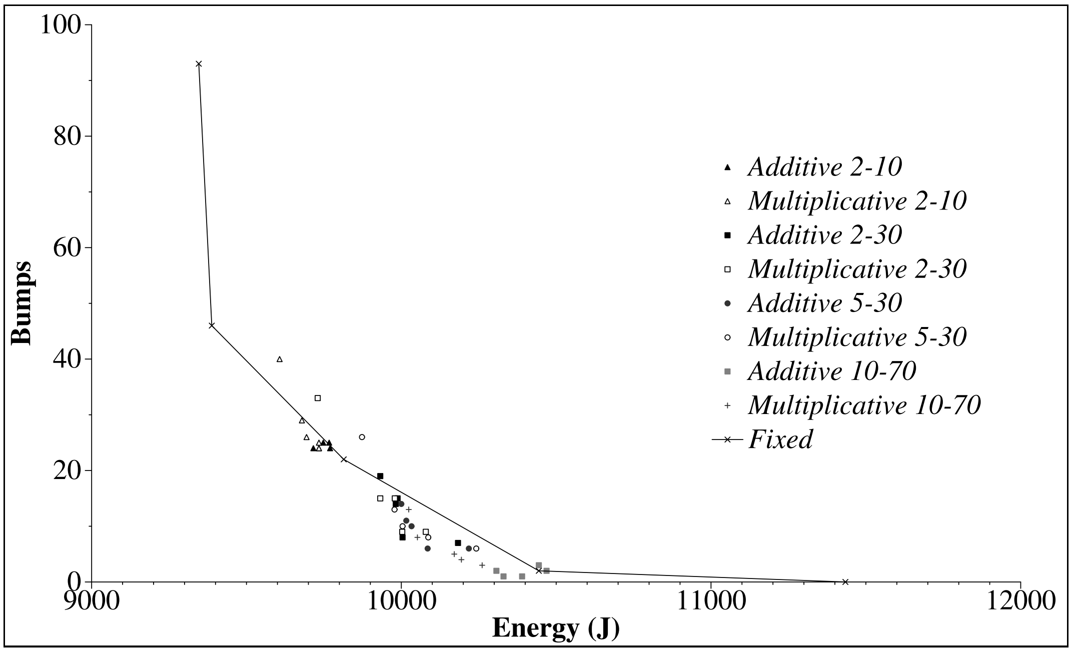


Figure 11: Simulation results for the **Macintosh** trace running on the CU140 with $\rho = 0.05$ and writes to SRAM going through immediately to disk. The line with **x**-marks represents fixed-threshold policies using thresholds of 2s, 5s, 10s, 30s, and 300s (increasing from left to right).

low as 2s are useful. They encounter fewer bumps than a fixed-threshold policy of 2–5s but for only a little more energy. The more willing the user is to tolerate spin-up delays, the more effective adaptive policies can be.

We mentioned in the introduction that it is hard to improve both bumps and energy consumption using these techniques. To reduce energy while simultaneously reducing the number of bumps compared to a fixed threshold policy, we would need to spin down the disk almost immediately whenever a spin-up would be acceptable. Intuitively, this requires a sophisticated prediction of acceptable spin-ups; prediction of interarrival times at disk for use in disk spin-down has proven to be difficult [5], though it warrants further study.

Instead, we have described in this paper a technique to vary the threshold dynamically to gain on one of these metrics, bumps, by compromising on the other, energy consumption. Users of mobile computers can already make this tradeoff simply by varying the fixed spin-down threshold, but changes in workload over time can result in changes to the proper place to make this tradeoff. By varying the threshold based on recent history instead of a static parameter, adaptive policies can react to these changes. At times, an adaptive policy may avoid cases when the disk spins for a while, and then spins down just before an access. If these cases can be avoided, the adaptive policy will save both energy and bumps. In other cases, when comparing an adaptive policy to a fixed-threshold one, there will be times when by spinning down the disk earlier some energy is saved in exchange for a bump that the fixed-threshold policy avoided, or vice-versa. One metric or the other will increase relative to the fixed-threshold policy.

6 Related Work

Most of the prior work in this area has focussed on what threshold to use for the fixed-threshold policy [5, 7, 14]. Generally speaking, a spin-down threshold of 2–5s consumed the least energy of all fixed thresholds, but resulted in several spin-up delays per hour.

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 [18]. If access patterns are not so consistent, however, these techniques may not prove to be helpful.

Golding, et al., studied idle-time detection and prediction in a more general framework [6]. They considered a number of prediction methods, including arithmetic and geometric adjustments of a predicted interval. Although they note the applicability of their taxonomy to powering down components on portable computers, they reported the effect of different methods only in the context of the TickerTAIP simulation system [16]. Also, they separate the prediction of *when* an idle period will arrive from the prediction of its *duration*. Here, our spin-down threshold is a prediction of how long the system should wait within an idle period before deciding that the remaining duration of the idle period is long enough to justify spinning down the disk.

On-line optimization of when to spin down the disk is similar to detecting how long to hold a virtual circuit open [12] or whether to spin on a lock or context switch [10], and can be modeled by a sequence of rent-to-buy decisions [13]. Our adaptive spin-down policies are similar to the "random walk" method described by Karlin, et al., which they reported performed almost as well as the optimal on-line policy and was more efficient than other adaptive policies they studied. A "profiling" approach (modeling the distribution) works better than a "random walk" approach for the virtual circuit problem [11], and we plan to explore profiling in the context of this problem.

7 Summary and Future Work

Adaptive disk spin-down allows the user of a mobile computer to trade off the energy consumption gained by spinning down a hard disk against the inconvenience of spinning the disk up again. It uses recent history to adjust the threshold for spinning the disk down, based on the user's specification. This specification is made in terms of the amount of time the disk must have been idle, compared to the amount of time the user is delayed by a spin-up.

Simulations with three sets of trace data and two magnetic disks indicate that adaptive spin-down offers advantages over a fixed threshold. In some configurations, adaptive policies eliminate up to a third of all undesirable spin-ups, or more, with only a marginal increase in energy consumption, such as 3%. Others represent intermediate points that could be obtained by varying a fixed-threshold policy, and still others represent undesirable points that are worse than fixed-threshold policies.

There are several directions that may be explored further. No single set of parameters (increments or ranges) is uniformly the best across each configuration, or uniformly undesirable; obviously, the differences in adjustment parameters (additive versus multiplicative, values and criteria for adjustment, and threshold bounds) should be further considered. The taxonomy described by Golding, et al. [6] would be helpful in this regard. The definition of a "bump" may warrant further elaboration, for example to take the energy savings from spinning down the disk into account (so a disk that has just been spun down and then spins up again would constitute a bump regardless of when the user last accessed it). Finally, these policies should be implemented in real systems and tested empirically.

Acknowledgments

W. Sproule of MITL gathered the Macintosh traces used in this study. R. Caceres, M. F. Kaashoek, K. Li, and B. Marsh helped with our understanding of the policies and algorithms. S. Keshav helped formulate optimization strategies. B. Krishnamurthy, D. Milojičić, D. Presotto, A. Skarra, J. Trotter, and the anonymous referees provided comments on earlier drafts.

Availability

The raw data for the graphs in this paper are available via anonymous *ftp* from *research.att.com* in the directory *dist/douglis/mlic95*.

References

[1] 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.

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

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

[4] Fred Douglis, Ramón Cáceres, Brian Marsh, Frans Kaashoek, Kai Li, and Joshua Tauber. Storage alternatives for mobile computers. In *Proceedings of the First Symposium on Operating Systems Design and Implementation*, pages 25–37. USENIX Association, November 1994.

[5] Fred Douglis, P. Krishnan, and Brian Marsh. Thwarting the Power Hungry Disk. In *Proceedings of 1994 Winter USENIX Conference*, pages 293–306, San Francisco, CA, January 1994.

[6] Richard Golding, Peter Bosch, Carl Staelin, Tim Sullivan, and John Wilkes. Idleness is not sloth. In *Proceedings of the USENIX 1995 Technical Conference on UNIX and Advanced Computing Systems*, pages 201–212, New Orleans, LA, January 1995.

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

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

[9] B. Johnson. *EPA Energy Start Computers Program*. Environmental Protection Agency, 1993. Basic Information Package.

[10] Anna R. Karlin, Kai Li, Mark S. Manasse, and Susan Owicki. Empirical studies of competitive spinning for a shared-memory multiprocessor. In *Proceedings of 13th ACM Symposium on Operating Systems Principles*, pages 41–55. Association for Computing Machinery SIGOPS, October 1991.

[11] S. Keshav. Personal Communication, 1994.

[12] S. Keshav, C. Lund, S. J. Phillips, N. Reingold, and H. Suran. An empirical evaluation of virtual circuit holding time policies in IP-over-ATM networks. *IEEE Journal on Selected Areas in Communications*, 1995. To appear.

[13] P. Krishnan, P. Long, and J. S. Vitter. On-line rent-to-buy in probabilistic environments. Technical Report CS–1995–08, Duke University, 1995.

[14] 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*, pages 279–291, San Francisco, CA, 1994.

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

[16] Chris Ruemmler and John Wilkes. An introduction to disk drive modeling. *IEEE Computer*, 27(3):17–28, March 1994.

[17] Western Digital. *Caviar UltraLite CU140 Technical Reference Manual*, May 1993.

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

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