# Adaptive Hard Disk Power Management on Personal Computers

*Yung-Hsiang Lu*
Computer System Laboratory, Stanford
luyung@pampulha.stanford.edu

*Giovanni De Micheli*
Computer System Laboratory, Stanford
nanni@galileo.stanford.edu

**Abstract**

Dynamic power management can be effective for designing low-power systems. In many systems, requests are clustered into *sessions*. This paper proposes an adaptive algorithm that can predict session lengths and shut down components between sessions to save power. Compared to other approaches, simulations show that this algorithm can reduce power consumption in hard disks with less impact on performance or reliability.

## 1. Introduction

The increasing popularity of portable electronics and the concept of green computers have generated a need for low-power computer design. In a computer, a hard disk can consume more than one fifth of the total power [4] [8]. Studies show that hard disks will keep consuming a significant portion of power in the near future [7] [9]. Although stopping plate spinning can reduce power consumption, this approach has three problems: a decrease in performance while waiting for the plates to spin up, extra energy while accelerating the plates, and higher failure rates which increase with the number of spin up-down cycles, typically tens of thousands of cycles [5] [13]. A desirable power management algorithm should save energy while providing high performance and low failure rates.

Disk shutdown algorithms can be classified into two categories: predictive and stochastic schemes [1]. The former are based on prediction of idle periods while the latter use stochastic system models and solve the optimization problems. Examples of the predictive schemes include [3] and [8], while [2] and [10] use the stochastic approach. Although these methods are effective in several applications, they do not consider the bursty nature of disk accesses. Traces [11] show that accesses are clustered, or bursty, with varying time between two accesses in the same cluster. Thus, none of these approach can accurately model requests.

*This paper is published in the Ninth Great Lakes Symposium on VLSI 1999 p 50-53.*

Disk shutdown algorithms can be implemented (in hardware or software) as part of a dynamic power management methodology [12]. The contribution of this paper is two-fold: (1) it presents a new algorithm for power management in hard disks and (2) it reports the result of a simulator specifically designed for comparing different power management schemes. The new algorithm is based on the concept of *sessions*, which can cope with the non-stationarity of system requests. Our method is a heuristic that takes into account non-stationary requests and hard disk reliability.

The method we present in this paper divides disk requests into sessions. Requests close in time belong to the same session; those that are far from each other without any other request in between are divided into different sessions. Since inter-session periods do not have disk activities, they are ideal candidates for spinning down the plates. Shutting down a disk inside a session, however, will cause serious performance degradation because the spin-up delay can be a significant portion of the time between requests. We develop an adaptive algorithm to dynamically predict session lengths and shut down a disk between sessions. Compared with other existing approaches through extensive simulations, our algorithm shows low energy consumption with less impact on performance or reliability.

## 2. Disk Accesses and Definition of Sessions

Figure 1 shows a disk access trace on a personal workstation for one day [11]. This figure suggests that accesses are clustered into groups, called sessions. A session starts with an access separated from the previous one by a long period of inactivity.

A threshold $\tau$ is used to separate sessions. If the time of two consecutive accesses differs by more than $\tau$, they belong to two distinct sessions. A smaller $\tau$ may divide adjacent accesses into two sessions while a larger value

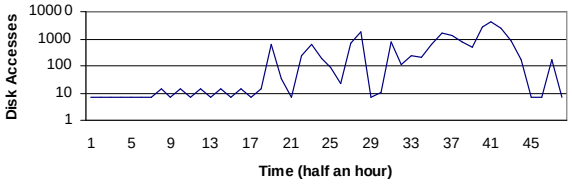


Figure 1: Disk Accesses on Thursday

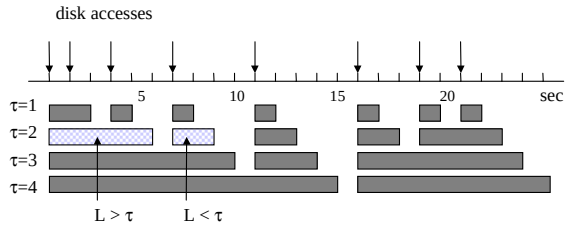


Figure 2: Sessions for Different Threshold

can combine them into one. Figure 2 shows an example of sessions with different $\tau$. In this figure, disk accesses are shown by arrows on the time axis. Each gray bar indicates the span of one session, from the first access to $\tau$ after the last access. For example, the last two accesses occur at t = 19 and t = 21. They are classified into the same session if $\tau$ is greater than two. If there were an access at the 20th second, three accesses would be classified into the same session, even when $\tau$ equals one. The session *length*, $L$, is the time between the first and the last accesses in one session. The lengths of the first two sessions are three and zero when $\tau$ = 2. This example shows that L can be larger or smaller than $\tau$. If a power manager shuts down a disk after it is idle for $\tau$ seconds when the ratio $\frac{E}{\tau}$ is rather small, a significant amount of energy is wasted for the last $\tau$ seconds. Conversely, a large $\frac{E}{\tau}$ indicates that multiple sessions are combined into one. The *intermission* is the time between the last access of the previous session and the first access of the next session. For example, when $\tau$ = 2 the intermissions are three, four, five, and three.

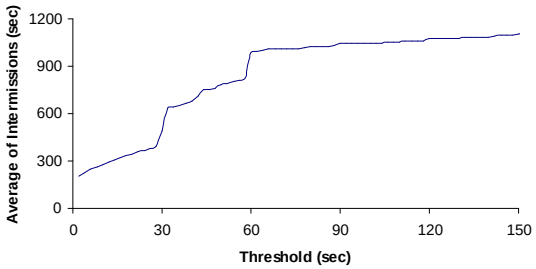


Figure 3: Intermission for Different $\tau$ Values

Although the one-day trace of disk accesses clearly shows the bursty nature, it is too short to derive an appropriate

| (unit: second) | mean | median | standard deviation |
|---|---|---|---|
| intermission | 986 | 872 | 668 |
| length | 59 | 48 | 51 |
| time between access in a session | 1.2 | $\ll 1$ | 6.0 |

Table 1: Statistical Properties When $\tau = 60$

value for $\tau$. Instead, we used a one-week trace to compute an appropriate value for $\tau$. Figure 3 shows the average length of the intermissions for different $\tau$. We chose sixty seconds for $\tau$ because it is the knee of the curve. A much larger $\tau$ will combine a lot of sessions while a much smaller $\tau$ will divide one session into multiple ones. In Section 4, we will show that this value is effective for a nine-week trace. Therefore, on-line derivation of $\tau$ is unnecessary due to the adaptivitiy of our algorithm. Table 1 shows the statistics when $\tau$ is 60. As expected, the average session length is much smaller than the length of intermissions. This $\tau$ value rarely combines sessions while keeping clustered accesses in one session. The standard deviation is compatible to the mean. This suggests that the individual session length varies widely and an adaptive algorithm is required to adjust session length prediction.

## 3. Adaptive Disk Shutdown Algorithm

A disk model with two states is used in our study. When the disk is in the spinning state, it can serve IO requests right away. On the other hand, when the disk is in the sleeping state, IO requests have to wait for the plates to spin up. A power manager (PM) changes the disk into the sleeping state whenever this transition is beneficial under the three conflicting goals: low power consumption, high performance, and a low failure rate. Figures 4 and 5 show our algorithm. The three intermediate states are enclosed by dashed lines. A sleeping disk wakes up only when a request arrives. The PM checks the IO request queue periodically. If the queue is nonempty and the disk is sleeping, the PM issues a spin-up command. If the plates are already spinning, they stay spinning. Meanwhile, the PM *increases* the predicted length of the current session. If the queue is empty and the disk is sleeping, the disk stays in the sleeping state. The difficulty arises when the queue is empty while the plates are spinning. Instead of immediately issuing a spin-down command, the PM *decreases* the predicted session length by an adjustment parameter. This parameter can affect the performance of the algorithm. If it is too large, the algorithm is too sensitive

to the variations in the time differences of two consecutive accesses. If it is too small, the predicted length is adjusted too slowly and the algorithm becomes a fixed-duration timeout scheme. The PM issues a spin-down command only when the queue has been empty long enough compared to the predicted session length. By dynamically adjusting the prediction, the algorithm can shut down the disk earlier for a shorter session while keeping the plates spinning for a longer session.

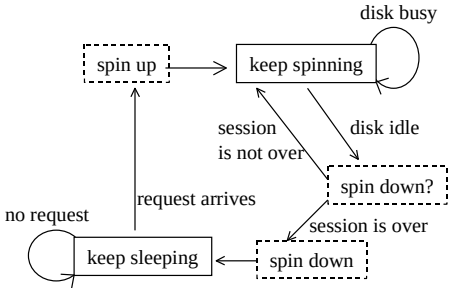


Figure 4: State Transition Diagram

```
/* PL/AL: predicted/actual session length */
/* a: attenuation factor */
/* SE: predicted session end time */
/* Th: threshold; inc: increment constant */
switch(state) {
 case spinUp:
   state = spinning; PL = a * PL + (1-a) * AL;
   SE = now + PL; break;
 case spinDown:
   state = sleeping; break;
 case sleeping:
   if (a request arrives) { state = spinUp; }
   break;
 case ?spinDown:
   if ((now >= SE) && ((now - SE)/PL) < Th1)
     { state = sleeping; }
   else { state = spinning; }
   break;
 case spinning:
   if (a request arrives) {
     if ((now >= SE) && ((now - SE)/PL < Th2))
       /* almost ready to shut down; defer SE */
       { PL += inc1; SE += inc2; }
   } else {
     state = ?spinDown; PL -= inc1; SE -= inc2; }
   break;
}
```

Figure 5: Adaptive Algorithm

## 4. Comparing Power Management Algorithms

Table 2 shows the disk model in our simulation. We have developed a dynamic power management analysis tool to simulate five control algorithms:

1. Adaptive algorithm 1: proposed in this paper. The initial prediction is 60 seconds and the adjustment parameter is 1.25 seconds because simulations show that this value can balance the sensitivity and adaptivity mentioned in Section 3. The prediction for the next session is 0.7 * previous prediction + 0.3 * actual length of the last session in order to adjust for workload changes.
2. Adaptive algorithm 2 [3]: The minimum length is 2 seconds and the other parameters are chosen according to the suggestion by the authors of [3] as $(\alpha_a, \beta_a, \delta_M, \rho) = (2, -0.25, 1.0, 0.05)$
3. Adaptive algorithm 3 [6] : The lower bound for the predicted idle time is 1 second and $(\alpha, c, S_{th}) = (0.5, 2, 4.75)$.
4. Fixed-timer algorithm: We used two seconds [8], and five minutes as commonly seen on desktop computers.
5. Greedy algorithm: shut down the disk ten milliseconds after serving each access. If another request arrives during the spin-up delay, the second request will also be served before the shutdown.

| | | | |
|---|---|---|---|
| spinning | 1.5 W | sleeping | 0.3 W |
| spin down | 1.0 W<br>1.0 sec | spin up | 2.5 W<br>1.0 sec |
| spinning failure | $1 \cdot 10^{-6}$ / hr | switching failure | $3 \cdot 10^{-5}$ |

Table 2: Disk Model

Our adaptive algorithm differs from [3] and [6] because it (1) estimates the length within each session, (2) predicts lengths instead of changing acceptable amounts of idle time, (3) assumes equal probability for longer or shorter sessions than the average length, and (4) is less sensitive to exceptionally long idle periods compared to Hwang's algorithm. In our simulation, all algorithms check the request queue every second.

Table 3 shows the result of running these algorithms for a nine-week trace with 385,213 disk accesses on a personal workstation [11]. The last two rows are normalized. We compare the following items.

1. Consumed energy.
2. Number of state changes. This is used for predicting the disk lifetime. We assume each up-down cycle increases the failure probability by $3 \times 10^{-5}$ and one spinning hour increases it by $1 \times 10^{-6}$. The lifetime is the time when the failure probability reaches one-half. A large number of state changes reduces the lifetime and implies more requests have to wait for the plates to spin up.
3. Total spinning time and average duration.
4. Product of 1 and 2. A smaller number is better because fewer requests are affected due to spinning-

| | adaptive 1 | adaptive 2 | adaptive 3 | fixed (2s) | fixed (5m) | greedy |
|---|---|---|---|---|---|---|
| E (energy, J) | 2603507 | 1870586 | 1859128 | 1846958 | 4778338 | 1791558 |
| ratio | 1.45 | 1.04 | 1.04 | 1.03 | 2.67 | 1.00 |
| S (switch cycle) | 6493 | 21485 | 22352 | 23330 | 3306 | 33609 |
| ratio | 1.96 | 6.50 | 6.76 | 7.06 | 1.00 | 10.12 |
| life time (week) | 317 | 93.6 | 90.0 | 86.0 | 626 | 58.7 |
| ratio | 5.4003 | 1.5945 | 1.5332 | 1.4651 | 10.6644 | 1.0000 |
| spinning (sec) | 789852 | 135357 | 123280 | 129590 | 2613312 | NA |
| mean spinning time | 121.6 | 6.3 | 5.5 | 4.7 | 790 | NA |
| mean sleeping time | 716.7 | 247.0 | 238.0 | 228.6 | 855 | 160.9 |
| E × S | 1.07 | 2.54 | 2.63 | 2.73 | 1.00 | 3.81 |
| E / S (efficiency) | 7.52 | 1.63 | 1.56 | 1.49 | 27.11 | 1.00 |

Table 3: Comparison of Power Management Algorithms (NA: not applicable)

up delay while energy consumption is also small.

5. Efficiency, the ratio of the first two items. A large number means that a higher portion of energy is used to keep plates spinning to reduce delay.

This table shows that four algorithms (adaptive 2, 3, two-second fixed and greedy) have similar results because all of them are dominated by the *intra-session* behavior. They also suffer from short lifetimes – less than two years. The widely-used timer of five minutes consumes much more energy because the average session length is one minute. Although we derive the value of $\tau$ from a one-week trace, simulations show that it is equally applicable to this nine-week trace. This suggests that on-line adjustment of $\tau$ is unnecessary because the algorithm can dynamically adjust for workload changes. Our algorithm consumes 45% more energy than the other two adaptive algorithms but it provides a six-year lifetime, generally long enough for a personal computer. The last two rows show that our algorithm also provides a small ES product and a five-times higher efficiency.

## 5. Conclusions

We have proposed an algorithm for dynamic power management. This algorithm adaptively adjusts its prediction of future requests based on the notion of session. We have performed extensive simulations on controlling the power states of hard disks and have shown that this algorithm can reduce energy consumption with longer sleeping duration, less performance impact, and reasonable lifetimes.

## 6. Acknowledgements

This work is supported by MARCO and ARPA. We would like to thank Luca Benini and Alessandro Bogliolo at DEIS-Università di Bologna for their valuable comments and John Wilkes at HP Lab for providing the disk traces.

## 7. References

[1] L. Benini, A. Bogliolo, and G. D. Micheli. Dynamic power management of electronic systems. In *International Conference on Computer-Aided Design*, pages 696–702, 1998.

[2] E.-Y. Chung, L. Benini, A. Bogliolo, and G. D. Micheli. Dynamic power management for non-stationary service requests. In *Design Automation and Test in Europe*, 1999.

[3] F. Douglis, P. Krishnan, and B. Bershad. Adaptive disk spin-down policies for mobile computers. In *Computing Systems*, volume 8, pages 381–413, 1995.

[4] F. Douglis, P. Krishnan, and B. Marsh. Thwarting the power-hungry disk. In *USENIX Winter Conference*, pages 293–306, 1994.

[5] P. Greenawalt. Modeling power management for hard disks. In *International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, pages 62–6, 1994.

[6] C.-H. Hwang and A. C. Wu. A predictive system shutdown method for energy saving of event-driven computation. In *International Conference on Computer-Aided Design*, pages 28–32, 1997.

[7] Intel. Mobile power guide '99.

[8] K. Li, R. Kumpf, P. Horton, and T. Anderson. A quantitative analysis of disk drive power management in portable computers. In *USENIX Winter Conference*, pages 279–292, 1994.

[9] J. R. Lorch and A. J. Smith. Software strategies for portable computer energy management. *IEEE Personal Communications*, 5(3):60–73, June 1998.

[10] G. A. Paleologo, L. Benini, A. Bogliolo, and G. D. Micheli. Policy optimization for dynamic power management. In *Design Automation Conference*, pages 182–187, 1998.

[11] C. Ruemmler and J. Wilkes. Unix disk access patterns. In *USENIX Winter Conference*, pages 405–20, 1993.

[12] M. B. Srivastava, A. P. Chandrakasan, and R. W. Brodersen. Predictive system shutdown and other architecture techniques for energy efficient programmable computation. *IEEE Transactions on VLSI Systems*, 4(1):42–55, March 1996.

[13] S. Udani and J. Smith. Power management in mobile computing. Technical report, University of Pennsylvania, MS-CIS-98-26, 1998.