Hui Miao
Project Report for 290S
Prof. Scott Brandt

Predictive Power Conservation based on the Idle Time Pattern from Disk Access Data

Abstract

We examined the power consumption of various dynamic disk spindown policies with the goal to save energy for mobile computing. Based on a trace from HP Laboratories, we find that the proposed adaptive 2-competitve algorithm can outperform the share algorithm by 15% or so when the disk spindown cost is low. The relative power performance of different disk spindown techniques are presented and compared. We focus on algorithms that explicitly exploit the burst nature of disk access pattern.

1. Introduction

With the limited battery life, most mobile computing device has to be designed in an energy efficient way in order to extend its usage. Screen display and disk accounts for most of the power consumption for a mobile computer. It is reported that disk device usually costs about one third of total power consumption. Therefore, by effectively predicting the disk access pattern, the disk spin down and up can be controlled in order to save energy.

2. Related Work

The screen saver adopts a predictive approach for energy conservation. When there is no keystroke after a while, the screen display will be turned off. The same approach can be applied to disk subsystem as well. You may also change the CPU speed depending on the particular application. For example, Word processing may tolerate a low CPU speed. We will focus on predictive power conservation for its simplicity and low implement cost.

By predicting the idle time correctly, the disk can spin down and save energy. But a wrong prediction may consume more energy, as it is costly to spin down and back up the disk again. We denote the cost of shut down the disk and then back it up as the spindown cost, S, measured in terms of keep-spinning-equivalent time. Accelerating the disk plate to back it up is very costly. The back up of disk will also delay the disk response time. Usually a fixed time out strategy is adopted on most computers. The disk simply sit idle for a fixed time, say 1 minute, if there is no new request coming, the disk shut itself down until new request comes in to back itself up.

It is observed that disk access is bursty and doesn't follow a statistical distribution. It is not easy to set this as an optimization problem and solve explicitly. Adaptive machine learning algorithm can be useful to increase the accuracy of predicting idle time. Helmbold et al [2] found that ***share algorithm*** can do much better than a fixed time out strategy by adaptively changing the weight of 100 time-out candidate "experts" to get the final prediction. They have two parameters for control of how rapidly the weights of incorrect experts are reduced and how fast a poorly predicting expert recovers when it starts to predict well [3].

The share though has proved to do much better than the fixed time-out policy [2], it has some ***drawbacks*** as well. The computation of weights involves heavy floating-point computation and could be time-consuming and thus lowers the response time of disk requests, the additional hardware to support share algorithm implementation is added cost. Moreover, the training of parameters will have to be done manually in order to figure out the best parameters in the share algorithm, there is no self-tuning involved here. It is not practical to use just one set of parameters to fit all disk access patterns. Finally the algorithm has a serious drawback that the weights constantly shrink towards zero, weights have to be rescaled periodically.

A simple ***2-competitive algorithm*** is to set the fixed waiting time exactly to the spindown cost. When there is no new requests coming in, the disk will keep spinning for a fixed time equal to the spindown cost (measured in terms of keep-spinning-equivalent time), the disk shuts itself down when there is still no request arriving. This algorithm is very straightforward and cheap to implement. It can be proved that the worst-case cost bound is two times the optimal energy cost with perfect foresight about the timing of future request arrival.

However, this 2-competitive algorithm is ***static*** and doesn't adapt to the burst disk access pattern. That is, the well-observed burst nature of the disk access pattern was not exploited to make the disk spinning policy more efficient. The major inefficiency is that when there are few requests coming in, the disk keeps spinning for S seconds instead of shutting it immediately.

Here, we propose a new ***adaptive 2-competitve*** to improve this inefficiency. When the arriving requests are very frequent and clustered together, it is more likely that the next request will come shortly, so we keep the disk spinning for S seconds, but the arriving requests are very sparse, it is more likely that next request will be far away, so we shut down the disk immediately instead of spinning idle for S seconds. We find this simple improvement of 2-competitive algorithm can save energy 13% to 57% compared to static 2-competitive algorithm under different spindown cost scenario. The cost gain comes from the fact the burst nature of the disk access pattern has been utilized in our decision making process, the additional piece of information helps to reduce cost.

3. Experimental Results

We present the trace-driven simulation results showing that comparative performance of various proposed algorithms.

3.1 The Trace Data

The trace data is from HP Lab [1]. Cello is a timesharing system used by a small group of researchers at HP Lab to do compilation, simulation, editing and mail, these workload may not be representative for a typical mobile computer user, but it is one we have at hand. The trace is collected from 92.4.18 to 92.6.20. The disk idle time sequence is distilled from the trace data, which is provided by Web Ryan at UCSC. Total observations are 200K, it is measured in microseconds.

Figure 1. CDF of a Typical Sample Data

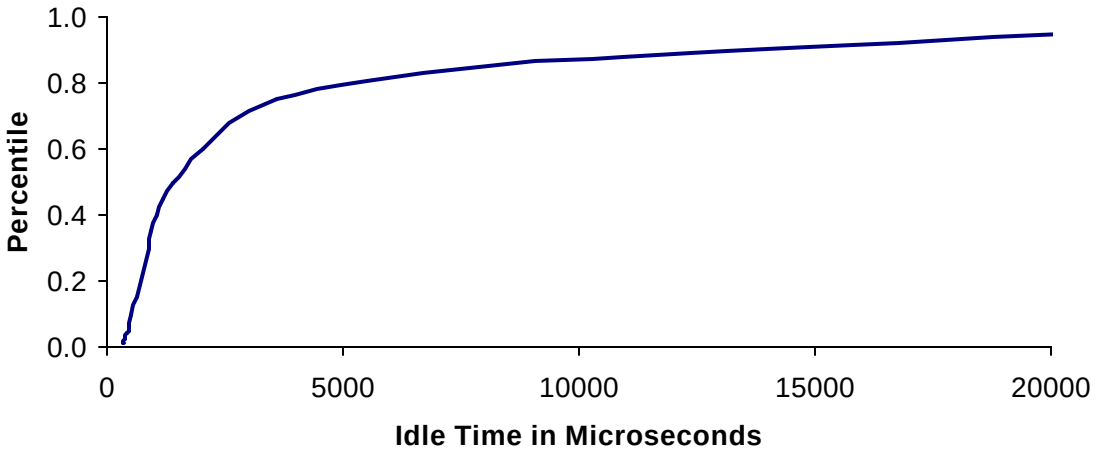


Table 1 Summary of the CDF of the Experimental Data

Percentile	*95%*	*90%*	*85%*	*75%*	*60%*	*50%*	*45%*	*35%*	*25%*	*20%*	*15%*	*10%*
Idle Time	20707	13126	7887	3596	2044	1396	1186	938	820	715	631	514

From the CDF of the idle time sequence data, we can find that most idle time is short, three quarters of total observations are less than 3.6 seconds. That is, most disk requests are quite clustered together. More than 10% of total requests are spaced out more than 1 minute. The distribution of the data is heavy tailed and skewed toward a small idle time around 1 second. The spindown cost S is depending on the physical structure of the disk device, its reported value range from 4 to 15 seconds. It is obvious that most idle time periods are much smaller than spindown cost S. For example, with 2-competitve algorithm, actual disk spindown is rather rare. Given a spindown 13, the disk is only spinning down 10% between jobs.

3.2 The Methodology: Adaptive 2-Compititve Algorithm

In order to illustrate the algorithm, let's first define some statistics similar to Helmbold [2] in the following:

Table 2. Definition of Statistics

Energy used by time-out = {Idle time, if idle time < time-out, no spindown
Time-out + S, if idle time > time-out, spindown after S }

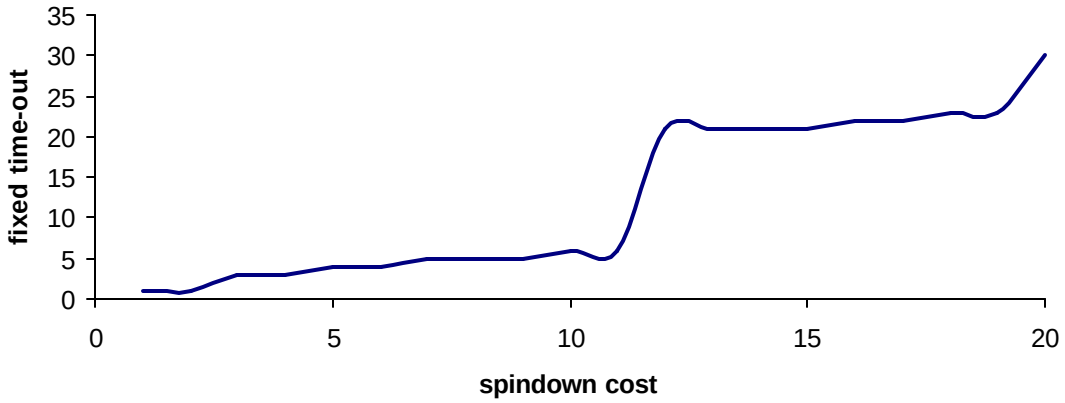
Energy used by optimal = {Idle time, if idle time < time-out, no spindown
S, if idle time > time-out, spindown immediately}

$$\text{Energy Waste \%} = \frac{(\text{Energy used by time-out} - \text{Energy used by optimal})}{\text{Optimal Energy}}$$

With perfect foresight, we can calculate the optimal energy, but this is impractical since we don't know the future data given the past data. This can be used as a benchmark to compare the cost of various strategies. We use energy waste compared to ideal situation as metrics to compare these methods.

The ex post best fixed time-out strategy is defined as adopting one fixed time-out by peeking into the future data, various time-out value is checked to find out the best time out. This policy is not practical since we don't have knowledge about future in reality. But it can also be used for comparison purpose.

Figure 2. Best fixed time out under different spindown cost



Given the trace data, we find that the optimal fixed time-out is close to spindown cost when the spindown cost is low. When the spindown cost is high, the optimal fixed time-out value is sensitive to the distribution of the data and it could be bigger than S. The energy consumption could be 70% to 25% more than the optimal ideal value.

The details of share algorithm can be found in [2], the time-out is varied dynamically based on the prediction. When the real idle time at time ***t*** can be fairly predicted based on past pattern, and then the decision can be made more efficiently. We use the same parameters as in [2], since the parameters are learned from a similar dataset.

With the goal of designing practical and simple algorithm in mind, we designed this adaptive 2-compititive algorithm. All we need is a disk spindown cost, it is a structural parameter that can be estimated in advance and hardly change very often when the disk is installed. Then three most recent past idle time value are retained for decision-making. If the disk did not spin down two out of three times, we consider the disk is in a ***busy mode*** and let the disk wait for the next request while keeping the disk running for at most S seconds. If there are few requests coming in, it is more likely that the disk is in a ***sparse mode*** and waiting has little value, so shut it down immediately. The two modes basically respond to the burst nature of the disk access pattern.

Of course, this method is rudimentary. With better statistical characterization of the burstiness and the queue length, we can define the transition between these two modes better. We can define a better statistical detector to switch between these two modes. When it is in a sparse mode, there is little incentive to waiting S second instead of shutting down immediately. This is improvement of 2-competitive static algorithm.

The penalty of mispredicting a sparse mode at time (*t*) will incur a cost equal to S minus idle time at time (*t*). This penalty cost will increase as the spindown cost increase, so with high spindown cost, this algorithm will need a high accuracy of prediction. In other words, spindown may be too often compared with optimal ideal case. As we observed in the data summary section, spindown will be rare with a high spindown cost since only 10% idle time will be bigger than 13 seconds in our data.

Table 3. Description of Adpative 2-Competitive Algorithm

On each trial the algorithm:

If two out of previous three trials, disk spindown did not happen, (Busy Mode)

Then Adopt C-2 Competitive algorithm
Shut down the disk only after S seconds of waiting idle

Else shut down the disk immediately without waiting S (Sparse Mode)

3.3 Experimental Results

With 200K idle periods, we have simulated various algorithms to compare their performance. With the limited computation resource, we don't have the opportunity to try more algorithms. It takes about 11 hours to run the share algorithm on my machine.

We find that the adaptive 2-competitve algorithm outperforms other strategies by a big margin when the spindown cost is low. When spindown cost is 2 seconds, adaptive 2-competitive cost wastes 36% of energy compared to optimal ideal scenario, while share algorithm consumes 45% and the rest consumes 55% or more.

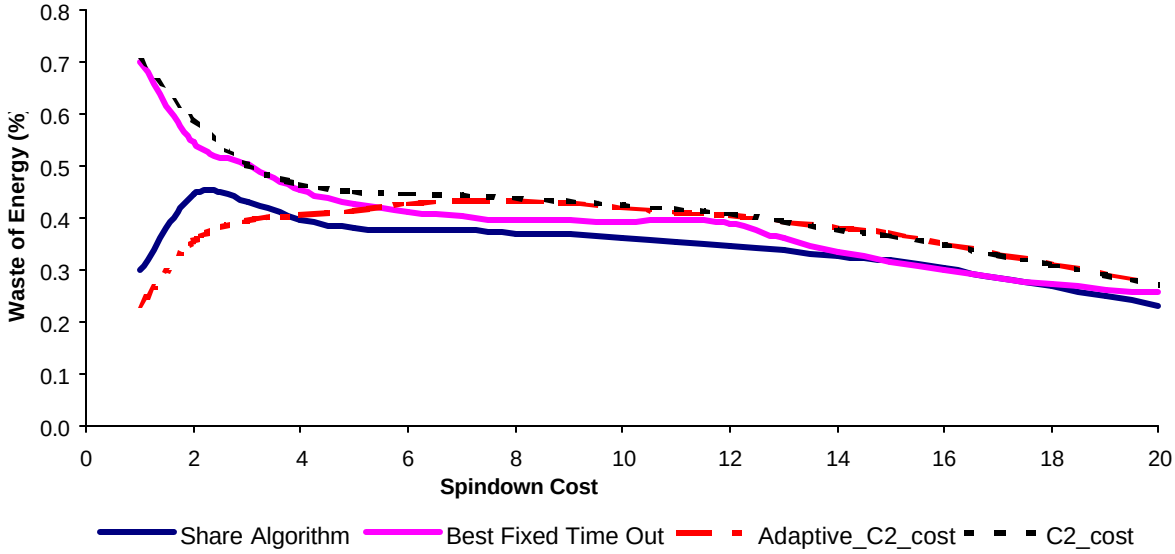
Table 4 Energy Waste of Different Algorithm compared to Optimal Cost

Spindown Cost	Share Algorithm	Best Fixed Time Out	Adaptive_C2_cost	C2_cost
1	30%	70%	23%	70%
2	45%	55%	36%	59%
3	43%	50%	40%	50%
4	40%	45%	41%	47%
5	38%	43%	41%	45%
6	38%	41%	43%	45%
7	38%	40%	43%	44%
8	37%	40%	43%	44%
9	37%	40%	43%	43%
10	36%	39%	42%	43%
11	35%	40%	41%	42%
12	35%	39%	41%	41%
13	34%	36%	39%	39%
14	33%	34%	38%	38%
15	32%	32%	37%	37%
16	30%	30%	35%	35%
17	28%	28%	33%	33%
18	27%	27%	31%	31%
19	25%	26%	29%	29%
20	23%	26%	27%	27%

However, when the spindown cost is big, the power consumption of these algorithms converges, while share algorithm still leads but with a small margin. With high restart cost for a disk, more complicated techniques only does little better than the simple static 2-competitive algorithm.

With better statistical characterization of the bursty disk access pattern, we can do better by predicting the traffic pattern. Similar studies have also been found in network traffic. The network traffic are found to be burst as well, there are study to design high throughput queuing policy to meet the burst traffic. There are some interesting discussions on why the traffic is burst. Future research can learn something useful in that field as well.

Figure 3. Waste of Energy Compared to Optimal, in Percentage



4. Conclusion

We have shown that adaptive 2-competitive algorithm can outperform share algorithm by 8% to 20% while the spindown cost is small, but its performance converges toward static 2-competitive algorithm when spindown cost increases. We need a better statistical detector to switch between sparse mode and busy model in order to minimize the misprediction.

Share algorithm still dominates other existing approaches by saving15% energy when the spindown cost is large. Share algorithm performs always better than the impractical optimal fixed time-out algorithm. Compared to optimal ideal power consumption, share algorithm consumes more than 23% to 45% energy. This suggests that there is still room to improve the predictive power consumption techniques.

Reference:

[1] C. Ruemmler and J. Wilkes, “UNIX Disk Access Patterns”, Proceedings of the Winter 1993 USENIX Conference (San Diego, CA), January 1993
[2] D. Helmbold, D. Long and B. Sherrod, “A Dynamic Disk Spin-down Technique for Mobile Computing”, University of California, Santa Cruz, 2001
[3] M. Herbster and M.K. Warmuth, “Tracking the Best Expert”, in Proceedings of the Twelfth International Conference in Machine Learning, Tahoe City, CA 1995
[4] Mark E. Crovella and Azer Bestavros, ``Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes," in IEEE/ACM Transactions on Networking, December 1997.