Adaptive Disk Spin-Down Algorithms in Practice

Timothy Bisson, Scott A. Brandt
{tbisson,sbrandt}@cs.ucsc.edu

1 Introduction

Significant research has been done to minimize disk energy consumption with excellent timeout based disk spin-down algorithms [1, 3, 4, 5]. However, in practice, performance is limited with such algorithms due to spin-up latency, file-system logging, and page-cache write-back policies. Spin-up latency significantly decreases user interactivity while file-system logging and dirty-page write-back policies decrease the potential energy savings.

We have implemented the dynamic disk spin-down algorithm of Helmbold *et al.* [1] in the Linux Kernel, herein referred to as DSA. The algorithm uses multiple experts each predicting with fixed timeouts for its overall dynamic timeout calculation. A weight is associated with each expert and used to restrict its contribution to the overall timeout calculation. A new timeout is calculated after every idle period. An expert's weight is reduced by considering its energy consumption relative to an oracle policy's energy consumption. We modified the DSA algorithm to address its spin-up latency overhead and call it Modified DSA. The aim of our work addresses spin-up latency in the DSA algorithm by reducing the total number of spin-ups during dynamically determined periods of user interactivity.

2 Spin-up Latency

The main difficulty with an efficient disk spin-down algorithm is spin-up latency. We measured the spin-up latency of our hard-disk [2] and found it to be 4.492s on average with .0036 variance. In an efficient disk spin-down algorithm, timeout computations should approach and remain at a few seconds, since the ratio of idle to spin-down energy use may be in the low teens. Therefore, users of such algorithms may suffer from spin-up latency every few seconds, provided idle periods are also a few seconds.

Our workload consists of a 24-hour trace on our personal development machine. We show the first 10 hours in our graphs as it consists of use by an actual user. We used this trace in order to generate the same disk accesses for each algorithm, and because it is should be representative of typical workstation use. Our energy calculations are based on the specifications provided byt the disk manufacturer [2]. We are currently implementing a mechanism to directly record the actual energy consumption.

Modified DSA reduces the number of spin-downs when system activity is determined to be interactive. Our enhancement performs additional weight reduction to experts when their predicted timeout and the idle period is less than an *interactive threshold*. Weight reduction increases with continuous idle periods less than the interactive threshold. The development of this feature is still in progress.

Figure 1 shows energy use with 4 different algorithms over a period of 10 hours. We see the 30 second timeout algorithm uses 3.5 times more energy than the oracle after 10 hours of operation. We also see the DSA algorithm uses 1.5 more energy than the oracle, while the modified DSA only uses 1.8 as much energy.

Figure 2 shows the spin-up latency per disk request sampled at 10 minute intervals. The 30 second timeout algorithm endures the least latency as it spin-downs the least number of times relative to the other algorithms. Our modified DSA algorithm has a lower spin-up latency than both the Oracle and the original DSA algorithm. This is due to experts with lower timeouts than the idle_period being progressively punished while the idle period is less than the interactive threshold.

Figure 1: Energy consumption of the four spin-down algorithms

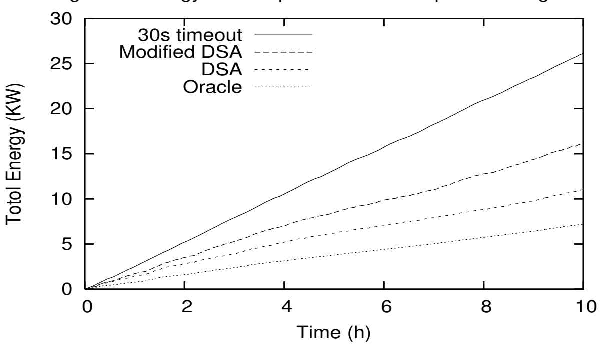
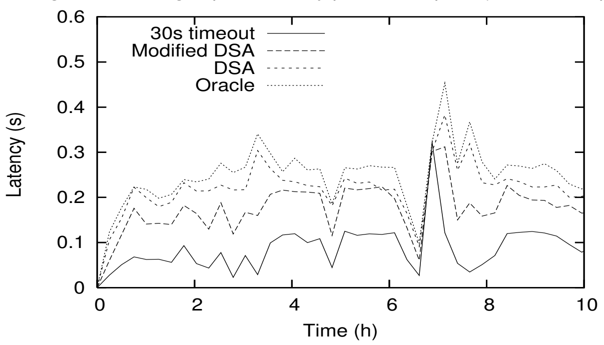


Figure 2: AVerage spin-u latency per disk request (10 min sampling)



3 Conclusion

Our contribution consists of implementing the DSA algorithm in the Linux Kernel, which to our knowledge has not previously been done. We also adapted the algorithm to address periods of user interaction with the system by punishing experts with low timeouts when the system is interactive, addressing one of the main drawbacks of adaptive spin-down algorithms. We have observed that periodic writes due to file-system logging prevent optimal energy consuption and unecessary spin-ups. We are currently considering a solution to this whereby logged data is temporarily stored in nonvolotile memory, such as flash, while a disk is spun-down, and then dumped to disk at the next spin-up.

References

[1] Western Digital. Specifications for the 120gb special edition drive (wd120jb, wd1200pb). Dec 2003.

[2] F. Douglis, P. Krishnan, and B. Bershad. Adaptive disk spin-down policies for mobile computers. In *In Proceedings of the second USNIX Symposium on Mobile and Location Independent Computing*, April 1995.

[3] D. Helmbold, D. Long, and B. Sherrod. A dynamic disk spin-down technique for mobile computing. In *In Proceedings of the 2nd ACM International Conference on Mobile Computing*, November 1996.

[4] P. Krishnan, P.M. Long, and J.S. Vitter. Adaptive disk spindown via optimal rent-to-buy in probabilistic environments. In *In Proceedings of the 12th International Conference on Machine Learning*, July 1995.

[5] G. De Micheli Y. Lu. Adaptive hard disk power management on personal computers. In *IEEE Great Lakes Symposium on VLSI*, March 1999.