A Survey of Power Management Techniques in Mobile Computing Operating Systems

Gregory F. Welch
Department of Computer Science
University of North Carolina at Chapel Hill
Chapel Hill, NC 27599-3175

welchg@cs.unc.edu

Abstract

Many factors have contributed to the birth and continued growth of mobile computing, including recent advances in hardware and communications technology. With this new paradigm however come new challenges in computer operating systems development. These challenges include heretofore relatively unusual items such as frequent network disconnections, communications bandwidth limitations, resource restrictions, and power limitations. It is the last of these challenges that we shall explore in this paper—that is the question of what techniques can be employed in mobile computer operating systems that can reduce the power consumption of today's mobile computing devices.

1. Introduction

Batteries are typically the largest single source of weight in mobile computing devices. While reduction of the physical dimensions of batteries is certainly a possibility, such efforts alone will reduce the amount of charge retained by the batteries. This will in turn reduce the amount of time a user can use the computing device before being forced to re-charge the batteries. Such restrictions tend to undermine the notion of mobile computing. However if we can succeed in reducing the basic consumption of individual components of mobile computing devices, we have the luxury of either reducing the battery dimensions while retaining the original charge characteristics, or increasing the battery charge time while retaining the original battery dimensions, or some combination of both.

Many physical components are responsible for ongoing power consumption in a mobile computing device. For example, Table 1 below lists the top power-consuming components of the Sharp PC 6785 as presented by Forman et al. [1]. This data is indicative of most of today's mobile computing devices, and highlights the areas requiring attention for power reduction.

Immediately we notice that the top three items are simply the base CPU and memory, operating at three different clock rates. The backlight for the LCD screen is the next largest consumer, and the hard drive motor follows the backlight. While it is certainly reasonable to expect that a mobile operating system might manage the intensity of an LCD screen backlight, we choose in this paper to survey only various power management techniques aimed at controlling the power consumption of the CPU and the hard drive.

Table 1: Power consumption of various components of the Sharp PC 6785 shown in order of decreasing power from top to bottom. (Data courtesy of Forman et al. [1].)

Component	Power (Watts)
base system (2MB, 25 MHz CPU)	3.650
base system (2MB, 10 MHz CPU)	3.150
base system (2MB, 5 MHz CPU)	2.800
screen backlight	1.425
hard drive motor	1.100
math co-processor	0.650
floppy drive	0.500
external keyboard	0.490
LCD screen	0.315

1.1 CPU Power Consumption

In general, the power consumed by the CPU and memory is related to the clock rate, the supply voltage, and the capacitance of the devices being switched (e.g. the transistors). The reduction in CPU and memory power consumption as the clock rate decreases (cf. Table 1) is a result of the switching characteristics of the logic gates in today's typically CMOS* VLSI circuits. When a complementary transistor pair in such VLSI circuits switches states, a brief short-circuit between the power supply and ground occurs, resulting in wasted energy. Unfortunately, most of the gates in a CMOS CPU will switch states on *every* clock cycle. Thus the higher the CPU clock rate, the more frequently the gates are switching, and the more energy or power is wasted.

In addition, the wasted power is related to the operating voltage of the components—the voltage appearing across each complementary transistor pair. The power wasted by logic gates during the brief short-circuit switching transitions is equal to the supply voltage squared divided by the resistance of the short-circuit path, i.e. $P = V^2/R$. Because the switching resistance is generally fixed, the wasted power is proportional to the square of the operating voltage.

* Complementary Metal Oxide Semiconductor circuits typically consist of logic gates that employ pairs of N and P-type transistors connected in series between the power supply and ground in a complementary fashion.

Furthermore, in order to accomplish useful work, transistors are typically connected together to form “chains” of logic, with each transistor or pair driving the gates of succeeding transistors. The larger the gate capacitance of the succeeding transistors, the more energy required to charge them during a transition. Thus the switching power is also proportional to the gate capacitance of any driven transistors.

The total power required by the CPU and memory is therefore proportional to CV^2F where C is the total capacitance of the wires and transistor gates, V is the supply voltage, and F is the clock frequency. While C can only be affected during chip design, newer chips are beginning to make it possible to vary F and V at run-time, in order to achieve linear and quadratic (respectively) savings in power. Thus a mobile operating system can adjust these parameters in cooperation with scheduling policies to reduce the CPU’s consumption of energy. This is the topic of the first paper we will look at in section 2, “Scheduling for Reduced CPU Energy” by Weiser et al.

1.2 Hard Drive Power Consumption

The power consumption of a hard drive in a mobile computing device can also be controlled by judicious scheduling of drive “spindown” events*. Such control is becoming more and more important as other components of mobile computing devices reduce their power consumption. These other reductions are resulting in overall drops in power consumption, with the result being that the relative amount of power consumed by disks has increased from 9% to 31% in some cases [5]. Disk drive power management is the topic of the two papers we will explore in section 3, “A Quantitative Analysis of Disk Drive Power Management in Portable Computers” by Li et al., and “Thwarting the Power-Hungry Disk” by Douglis et al.

1.3 Power and Wireless Communication

As somewhat of a supplement to secondary storage, periodic wireless broadcasts can sometimes be used to simultaneously disseminate information, e.g. a large common database, to a large number of mobile computing devices. While this technique can in a sense provide some of the functionality of a local hard drive, the organization of and access to broadcast data is very different from that of a hard drive. In section 4 we will examine “Energy Efficient Indexing on Air”, where the authors present and evaluate two methods for organizing and accessing such broadcast data.

2. “Scheduling for Reduced CPU Energy” [3]

As was discussed in the introduction of this survey, the power consumed by the CPU in a mobile computing device is significant. In fact, if the CPU clock frequency and supply voltage can be controlled, linear and quadratic savings in power can be realized (see section 1.1). This is the main motivation for the work presented by Weiser et al. in their paper on “Scheduling for Reduced CPU Energy”. In particular, because lowering the supply voltage results in a quadratic power savings, this is the authors’ preferred method for power reduction. As the authors note however, the supply

* Transitions from a drive’s *idle* state to its *sleep* or *off* state where it consumes no energy [4].

voltage cannot be lowered (dynamically) without also reducing the clock speed. Therefore the authors consider the CPU clock rate to be linearly adjusted with the supply voltage. Incidentally, it is the authors' assumption throughout their paper that such dynamic adjustments to the supply voltage and clock rate are indeed possible. As this hardware technique is becoming more popular, this is not an unreasonable assumption.

In this paper, the authors present and evaluate three algorithms for adjusting the CPU clock speed under the control of the operating system. The evaluation of the algorithms was accomplished through trace-driven simulations. The trace data was collected from a number of UNIX workstations over several working days, under otherwise normal conditions. During the trace collection, the workstations were used for a variety of applications such as e-mail, simulations, word-processing, etc. The collected traces were then used in off-line simulations of the three algorithms, allowing for flexible but consistent comparisons.

The basic approach of the algorithms is to balance CPU usage between periodic bursts of high CPU utilization and the remaining periods of idle time. For example, assume we have a task that normally results in a 50 millisecond (ms) burst of CPU usage at the full clock rate, followed by a 50 ms idle period. If that task can be replaced by a 100 ms period of CPU usage at half the normal clock rate, without affecting the user adversely, then such a scheme can be used to conserve power. This can be accomplished by dynamically adjusting (slowing) the CPU speed, effectively "stretching" activities from busy periods into subsequent adjacent idle periods.

The CPU speed adjustment decisions made in the simulations were based on the measured amounts of run time and idle time in various segments of the traces. In particular, all three algorithms rely on the assumption that the operating system *sleep* events that normally result in CPU idle time can be classified into two categories, "hard" and "soft" sleep events. Hard sleep events result in idle time during which the CPU speed cannot be reduced. For example, in the case of a disk wait, the sleep is typically issued from within the kernel's biowait() routine. On the other hand, soft sleep events result in idle time during which the CPU speed can be reduced. For example, it should be possible to slow and therefore delay completion of a sleep event that occurs while waiting for the next user keypress. The reason for such classifications was to be "fair" about assessments of which idle times were candidates for use in balancing periods of high and low activity.

Although CPU speed adjustments during small windows might affect job ordering, the authors did not explicitly reorder trace data events in the simulations to "correct" for any such occurrence. Their justification for this was twofold. First, significant reordering will generally only occur if unusually high (unreasonable) loads are offered to the CPU. Second, in their simulations a slowed CPU is ramped back up to full speed with increasing loads, thus restoring the original full speed conditions of the traces during periods of high demand.

The three algorithms presented are named OPT, FUTURE, and PAST. The three algorithms are similar in spirit to those typically used in (evaluating) page replacement policies for virtual memory systems. OPT is an impractical and undesirable algorithm. It is completely optimistic (and impractical) in the sense that it assumes complete knowledge of the future work to be done in an interval. It is undesirable because it adjusts the CPU rate over an *entire* trace, severely impacting the run times of user jobs and ignoring the relative importance of events such as keystroke response or network communications. OPT is however useful as a benchmark for performance. The FUTURE algorithm is similar to OPT in that it peers into the future, but it does so for only short windows of time, and optimizes over only those windows. Like OPT, FUTURE is unrealistic because it peers into the future. Unlike OPT, it is practical because it only optimizes over short windows, hence the impact on time-critical work is minimized. The PAST algorithm is a realistic (practical) version of FUTURE. It is realistic because instead of peering into the future, it looks to a short window in the past for information about CPU usage. It is desirable because like FUTURE it only monitors activity over a short window of the trace. It is less reliable because it assumes the activity in the next (future) window will be like that in the current (past) window. If the adjustments were optimistic (the activity level was higher than expected) and a window ended with work left to be done, that work was added to the work in the following window. The amount of such "carry-over" work was used as a measure of penalty incurred with different window lengths under this scheme.

The authors conclude that it is indeed more profitable (in terms of power savings) to spread out work when possible over periods of slower CPU clock rates, rather than contending with bursts of high-speed activity followed by wasted idle time. They conclude that in their simulations a PAST implementation with a 50 ms window would have saved up to 50% of the power in a 3.3 volt CPU, and up to 70% in a more aggressive (in terms of power savings) 2.2 volt CPU. They note that the energy savings are dependent on the window length, and also the minimum CPU clock rate used. Specifically, the interval between adjustments must not be too small or CPU usage becomes "bursty", and the minimum clock rate used must not be too slow or the assumptions about the future windows become less reliable resulting in higher penalties from excess or carry-over work. On the other hand, windows must not be too large or the interactive response of the system will be impacted. As a compromise between power savings and response time, the authors suggest a shorter 20 or 30 ms window.

3. Disk Drive Power Management

A common scheme for reducing the power consumed by hard drives in mobile computers is to "spindown" or turn off drives during extended periods of non-use. The papers discussed in the following two subsections evaluate schemes for determining when to perform a spindown.

3.1 "A Quantitative Analysis of Disk Drive Power Management in Portable Computers" [4]

It is common for manufacturers of hard disk drives for mobile computers to recommend a fixed time threshold for spinning-down the hard drives, where the threshold is generally on the order of 3-5 minutes. In this paper, the authors perform a quantitative analysis of the pros & cons of hard drive spindown as a power reduction technique. The basic premise of this work is that a fine-grained approach to spinning-down the hard drive is more appropriate than the coarse-grained approach pursued by manufacturers, given what has found to be normally very scattered usage patterns[*].

Like the work done by Weiser et al. described in section 2, the authors of this work used a large collection of operating system traces for their evaluations. For this work (paper) the authors collected traces from both Microsoft DOS machines (which the authors claim to be representative of portable machines today), and from the Sprite file system for Unix-like activity. The traces were collected over a series of months, and lasted from one to four hours in an attempt to be representative of mobile computer usage.

With the trace data in hand, the authors proceeded to analyze various spindown thresholds. For example, when simulating various spindown delays (time waited for further disk activity before spinning-down) they found that there was minimal consumption with a delay of approximately two seconds. This is interesting in light of the 2-3 minutes recommended by most manufacturers. They noticed (as seems intuitive) large consumption with a zero delay, as the drive would have to repeatedly spin-up. There was also a steady increase as the period was extended beyond two seconds. They found that with a two second spindown, users would have conserved 90% of the power otherwise used in the traces with *no* spindown. When compared to the 3-5 minute recommendations of most manufacturers they realized a factor of four in savings.

The authors also looked at some further optimizations, to see if the savings of their simple two second threshold could be improved upon by more intelligent methods. In particular, they performed some simulations using a prescient spindown policy, an optimal policy that uses foreknowledge of disk usage to determine when to perform a spindown. Using previous results, they determined that (for their simulated hard drive) 6.2 seconds of idle spinning was equal to a spinup in terms of power consumption. Therefore they used six seconds as the optimal threshold—if any future events would occur within the six seconds, *no* spindown was performed. If the next disk event was more than six seconds away, then a spindown *was* performed. Even under these optimal (unrealistic) circumstances, they found only an additional 3% savings in power.

In trying to explain the apparent optimality two second threshold, the authors identified two points concerned with management of the hard drive. The first is how *often* a drive gets to sleep, the second is how *long* it sleeps when it does. The frequency of sleeping is directly related to what is considered a “cluster” of disk activity. At the one extreme with a zero delay, every disk event is treated as its own cluster of activity and sleep is “induced” after every activity, thus requiring a spinup at every activity. At the other extreme with a very large (or infinite) delay, all disk events end up belonging to the same cluster, and hence the disk never sleeps but idles during many periods of inactivity. Their data sheds light on the second point also—with shorter delays, the disk slept for longer periods of time, saving more energy.

Of course the more frequent spindown introduces more user delay as each initial post-sleep activity requires that the user wait for a drive spinup. The authors analyzed the trade-off between energy consumption and user delay and found that (in their traces) a two second spindown delay would have resulted in 8-15 times per hour that the user would have had to wait for a spinup. With a two second spinup time, this translates to 15-30 seconds of user delay per hour. The authors expressed their opinion that this was not unreasonable when balanced against the potential power savings.

[*] A fine-grained approach implies control on the order of seconds, as opposed to minutes for a coarse-grained approach.

The authors also examined some related interests and concerns. They simulated the use of a disk cache, and found (unsuspectedly) that there was little effect on the power consumption. They also examined the effect of frequent spinups on the life of today's hard drives, and found that their two second delay would tend to shorten the lives of today's disks. However, they pointed out that manufacturers are implementing new technologies & methods that will offer approximately 25 times as many spinups in a disk's lifetime or approximately 4 1/2 years of continuous use. Finally they also examined the use of a name-attribute cache and found that it would have helped to some extent in terms of power consumption reduction, but was more valuable in terms of performance.

The optimal two second delay findings of Li et al. were further reinforced by the work of Douglis et al. in the next paper we will look at.

3.2 "Thwarting the Power-Hungry Disk" [5]

In work similar to that by Li et al., Douglis et al. studied the effects of hard drive spindown and spinup strategies on power consumption. By using similar trace driven simulation techniques, the authors of this paper arrived at conclusions very similar to those of Li et al. These authors however consider the read spinup delays (writes were considered to be performed asynchronously) to have a more serious impact than Li et al. did. In a slight twist on the subject, these authors classified disk management policies into two groups: off-line policies and on-line policies. Off-line policies are those that assume knowledge of future use, thus being unrealistic but optimal in that times of optimal spinup and spindown can be precisely determined. Such off-line policies are useful for generation of benchmarks. On-line policies on the other hand rely only on past knowledge, and are thus realistic (implementable) but not optimal.

The authors list a taxonomy of six hard drive spindown/spinup policies The naming of the policies indicates both the spinup and spindown policy. For example, the OPTIMAL_OPTIMAL algorithm can uses future knowledge to know precisely when to spindown the drive for maximum efficiency, and when to spinup the drive to be ready for the next operation. Such an algorithm is certainly a member of the off-line class. Demand-based spinup policies are included. For example, one of the six algorithms uses the THRESHOLD_DEMAND policy we have already seen in the previous paper. In this algorithm, the spindown is performed after a fixed time period (threshold) passes with no disk activity, and the spinup is performed only when the next disk operation "demands" it (thus generally introducing user delay). Unlike the previous authors, these authors also consider predictive algorithms, e.g. PREDICTIVE_PREDICTIVE, where spinup and spindown times are predicted in the hopes of maximizing sleep time (after spindowns) and minimizing the user-delay impact normally associated with a demand spinup. Such a predictive policy was suggested by John Wilkes at Hewlett-Packard Labs [6]. Wilkes proposed using a weighted average of past activity durations to predict both spindown and spinup delays. Douglis et al. experimented with such algorithms, but expressed disappointment with the results.

Not unlike Li et al., these authors concluded that significant savings (up to 85% of the savings resulting from the manufacturer's suggested delays) could be achieved by using the practical THRESHOLD_DEMAND algorithm, with a threshold of between one and ten seconds. And while they express concern at the user delay resulting from short thresholds, they also point out that in many drives (such as one of their simulated drives) the sleep-to-active times are shrinking continually, with some existing drives at one second. As this situation improves, the user-delay impact will be further minimized.

4. “Energy Efficient Indexing on Air” [7]

The periodic broadcast of data over wireless communication channels can be considered in a sense a supplement to a mobile user’s secondary storage. While accessing local hard drives consumes significant power, the reception of wireless broadcast data is a relatively low-power operation. When appropriate, periodic data broadcasts can be used to disseminate large volumes of data, e.g. a large common database, to an easily scalable population of listeners. However, even though wireless reception is a low-power operation when compared to hard drive access, it is not “free”. In addition, because the data arrives at the mobile computer in a unique fashion, the schemes to organize and access the data must be unique. In this final paper, the authors introduce and evaluate two very interesting approaches for organizing and accessing data that is broadcast periodically over wireless communication channels.

The motivation behind searching for and exploiting unique organization and access methods stems from the potential savings in power resulting from being able to wait for expected incoming data while in a “doze” mode [2]. When the mobile computer is receiving, it (its CPU) must be in the “active” mode. As was argued in section 1.1, the power consumed by the CPU and memory (in active mode) is not trivial. As pointed out by Imielinski et. al, the ratio of power consumption in active mode to that in doze mode is on the order of 5000 for the Hobbit chip from AT&T [7]. The question is how to organize the data to be broadcast so that it can be accessed by a mobile receiver in a manner that provides for optimal switching between active and doze modes.

Due to the dynamic nature of mobile computing in terms of wireless communication cell migration, changing information content, and the multiplexing of many different files over the same communication channels, the authors propose broadcasting the directory of a broadcasted data file along with the data file in the form of an index. Without an index, the client would have to filter (listen to) the entire broadcast in the worst case, or half the broadcast on average. This is undesirable because such filtering requires the mobile unit to be in its active mode, consuming power unnecessarily. Therefore every broadcast (channel) contains all of the information needed—the file and the index. Again, the question is how to organize the data to be broadcast for optimal access by a mobile receiver.

For use in evaluating potential methods of organization and access, the authors introduce the two parameters *access time* and *tuning time*. The access time is the average time between identification of desired data in the index portion of the broadcast, and download of the data in the data file portion of the broadcast. The tuning time is the amount of time spent by a mobile client actually listening to a broadcast channel. The goal of the authors was to find algorithms for allocating the index together with the data on a broadcast channel, and to do so in a means that struck a balance between the optimal *access time* algorithm and the optimal *tuning time* algorithm.

The first organization method, called “(1,*m*) Indexing”, broadcasts the entire index *m* times (equally spaced) during the broadcast of one version of the data file. In other words, the entire index is broadcast every 1/*m* fraction of the data file. In the second method, “Distributed Indexing”, the (1,*m*) method is improved upon by eliminating much of the redundancy in the *m* broadcasts of the data file index. Their key observation is that only certain portions of the index tree are necessary between broadcasts of particular segments of the data file. Specifically, the periodically broadcasted index segments need only index the data file segment that follows it. By using fixed-sized “buckets” of data (both index and file data) the two methods allow the mobile client to “doze” for deterministic amounts of time, only to awake just prior to the next necessary listening event, what the authors call a “probe”.

In their evaluations, the authors found that both schemes achieve tuning times that are almost as good as that of an algorithm that had optimal tuning time. In terms of access time, both algorithms exhibit a savings that is a respectable compromise between the two extremes of an algorithm with optimal access time and an algorithm with optimal tuning time. The Distributed Indexing scheme is always better than the (1,*m*) scheme.

In examples of practical implementations, the authors again compare their algorithms to the extreme cases of an optimal tuning time algorithm, and an optimal access time algorithm. In one example for the (1,*m*) algorithm, they show a per query reduction of power by a factor of 120 over the optimal access algorithm, but a 45% increase in access time. For the same (1,*m*) example, they found that the power consumption was very similar to that of the optimal tuning algorithm, but that the access time had improved to 70% of that in the optimal tuning algorithm.

In looking at an example of the distributed indexing scheme, they found a per query power reduction of 100 times smaller than that of an optimal access algorithm, while the access time increased by only 10%. Again when compared to an optimal tuning algorithm, they found similar power consumption, but again improved access time of 53% of that for an optimal tuning algorithm.

The conclusion of the authors is that by using their distributed indexing scheme in periodic broadcasts, a savings of 100 times less energy can be realized. The fruits of this savings can of course be used for other purposes such as extended battery life, or extra queries.

5. Conclusion

The advent of mobile computing has introduced new challenges for the designers of computers and computer operating systems. One such significant challenge is to reduce power consumption in ways that adversely affect the user the least. Savings in power consumption can be “spent” by extending battery life, and (or) reducing the physical size of batteries thus reducing the physical size of mobile computing devices.

We have briefly surveyed several proposed power conservation techniques. These techniques together focus on what are typically the main culprits in terms of power consumption in mobile computing: CPU/memory devices, secondary storage devices such as hard drives, and wireless communication devices used to supplement secondary storage where appropriate. While hardware advances have and will likely continue to reduce the power consumption of these devices, we have seen that efficient operating system techniques can significantly reduce power consumption without considerably affecting the perceived performance.

Acknowledgments

This paper was originally written as an assignment for an advanced operating systems course at the University of North Carolina at Chapel Hill in the spring of 1995. Dr. Prasun Dewan, the instructor for the course, deserves thanks for teaching the course, incorporating the original writing assignment, and for offering useful starting points. Many thanks also to Dr. Anupam Joshi at Purdue University for valuable pointers to relevant work. Finally, thanks to Dr. Kevin Jeffay at UNC for valuable comments on the original survey.

References

[1] Forman, G.H., and J. Zahorjan, 1994. “The Challenges of Mobile Computing,” University of Washington, Computer Science and Engineering, Technical Report UW CSE 93-11-03

[2] Pitoura, E., and B. Bhargava, 1993. “Dealing with Mobility: Issues and Research Challenges,” Purdue University, Department of Computer Sciences, Technical Report CSD-TR-93-070

[3] Weiser, M., B. Welch, A. Demers, and S. Shenker. “Scheduling for Reduced CPU Energy,” *USENIX Association, First Symposium on Operating Systems Design and Implementation (OSDI)*

[4] Li, K., R. Kumpf, P. Horton, and T. Anderson, 1994. “A Quantitative Analysis of Disk Drive Power Management in Portable Computers,” *Proc. of Winter 1994 USENIX Conference*, January 1994

[5] Douglis, F., P. Krishnan, and B. Marsh, 1994. “Thwarting the Power-Hungry Disk,” *Proc. of Winter 1994 USENIX Conference*, January 1994

[6] Wilkes, J., 1992. “Predictive Power Conservation,” Hewlett Packard Laboratories, Technical Report HPL-CSP-92-5

[7] Imielinski, T., S. Viswanathan, and B. Badrinath. “Energy Efficient Indexing on Air,” *Proc. of the International Conference on Management of Data—ACM-SIGMOD*, May 1994