5.9 POWER MANAGEMENT

The first general-purpose electronic computer, the ENIAC, had 18,000 vacuum tubes and consumed 140,000 watts of power. As a result, it ran up a nontrivial electricity bill. After the invention of the transistor, power usage dropped dramatically and the computer industry lost interest in power requirements. However, nowadays power management is back in the spotlight for several reasons, and the operating system is playing a role here.

Let us start with desktop PCs. A desktop PC often has a 200-watt power supply (which is typically 85% efficient, that is, loses 15% of the incoming energy to heat). If 100 million of these machines are turned on at once worldwide, together they use 20,000 megawatts of electricity. This is the total output of 20 average-sized nuclear power plants. If power requirements could be cut in half, we could get rid of 10 nuclear power plants. From an environmental point of view, getting rid of 10 nuclear power plants (or an equivalent number of fossil fuel plants) is a big win and well worth pursuing.

The other place where power is a big issue is on battery-powered computers, including notebooks, laptops, palmtops, and Webpads, among others. The heart of the problem is that the batteries cannot hold enough charge to last very long, a few hours at most. Furthermore, despite massive research efforts by battery companies, computer companies, and consumer electronics companies, progress is glacial. To an industry used to a doubling of the performance every 18 months (Moore's law), having no progress at all seems like a violation of the laws of physics, but that is the current situation. As a consequence, making computers use less energy so existing batteries last longer is high on everyone's agenda. The operating system plays a major role here, as we will see below.

There are two general approaches to reducing energy consumption. The first one is for the operating system to turn off parts of the computer (mostly I/O devices) when they are not in use because a device that is off uses little or no energy. The second one is for the application program to use less energy, possibly degrading the quality of the user experience, in order to stretch out battery time. We will look at each of these approaches in turn, but first we will say a little bit about hardware design with respect to power usage.

5.9.1 Hardware Issues

Batteries come in two general types: disposable and rechargeable. Disposable batteries (most commonly AAA, AA, and D cells) can be used to run handheld devices, but do not have enough energy to power laptop computers with large bright screens. A rechargeable battery, in contrast, can store enough energy to power a laptop for a few hours. Nickel cadmium batteries used to dominate here, but they gave way to nickel metal hydride batteries, which last longer and do not pollute the environment quite as badly when they are eventually discarded.

Lithium ion batteries are even better, and may be recharged without first being fully drained, but their capacities are also severely limited.

The general approach most computer vendors take to battery conservation is to design the CPU, memory, and I/O devices to have multiple states: on, sleeping, hibernating, and off. To use the device, it must be on. When the device will not be needed for a short time, it can be put to sleep, which reduces energy consumption. When it is not expected to be needed for a longer interval, it can be made to hibernate, which reduces energy consumption even more. The trade-off here is that getting a device out of hibernation often takes more time and energy than getting it out of sleep state. Finally, when a device is off, it does nothing and consumes no power. Not all devices have all these states, but when they do, it is up to the operating system to manage the state transitions at the right moments.

Some computers have two or even three power buttons. One of these may put the whole computer in sleep state, from which it can be awakened quickly by typing a character or moving the mouse. Another may put the computer into hibernation, from which wakeup takes much longer. In both cases, these buttons typically do nothing except send a signal to the operating system, which does the rest in software. In some countries, electrical devices must, by law, have a mechanical power switch that breaks a circuit and removes power from the device, for safety reasons. To comply with this law, another switch may be needed.

Power management brings up a number of questions that the operating system must deal with. They include the following. Which devices can be controlled? Are they on/off, or do they have intermediate states? How much power is saved in the low-power states? Is energy expended to restart the device? Must some context be saved when going to a low-power state? How long does it take to go back to full power? Of course, the answers to these questions vary from device to device, so the operating system must be able to deal with a range of possibilities.

Various researchers have examined laptop computers to see where the power goes. Li et al. (1994) measured various workloads and came to the conclusions shown in Fig. 5-1. Lorch and Smith (1998) made measurements on other machines and came to the conclusions shown in Fig. 5-1. Weiser et al. (1994) also made measurements but did not publish the numerical values. They simply stated that the top three energy sinks were the display, hard disk, and CPU, in that order. While these numbers do not agree closely, possibly because the different brands of computers measured indeed have different energy requirements, it seems clear that the display, hard disk, and CPU are obvious targets for saving energy.

5.9.2 Operating System Issues

The operating system plays a key role in energy management. It controls all the devices, so it must decide what to shut down and when to shut it down. If it shuts down a device and that device is needed again quickly, there may be an annoying delay while it is restarted. On the other hand, if it waits too long to shut

Device	Li et al. (1994)	Lorch and Smith (1998)
Display	68%	39%
CPU	12%	18%
Hard disk	20%	12%
Modem		6%
Sound		2%
Memory	0.5%	1%
Other		22%

Figure 5-1. Power consumption of various parts of a laptop computer.

down a device, energy is wasted for nothing.

The trick is to find algorithms and heuristics that let the operating system make good decisions about what to shut down and when. The trouble is that "good" is highly subjective. One user may find it acceptable that after 30 seconds of not using the computer it takes 2 seconds for it to respond to a keystroke. Another user may swear a blue streak under the same conditions. In the absence of audio input, the computer cannot tell these users apart.

The Display

Let us now look at the big spenders of the energy budget to see what can be done about each one. The biggest item in everyone's energy budget is the display. To get a bright sharp image, the screen must be backlit and that takes substantial energy. Many operating systems attempt to save energy here by shutting down the display when there has been no activity for some number of minutes. Often the user can decide what the shutdown interval is, pushing the trade-off between frequent blanking of the screen and using the battery up quickly back to the user (who probably really does not want it). Turning off the display is a sleep state because it can be regenerated (from the video RAM) almost instantaneously when any key is struck or the pointing device is moved.

One possible improvement was proposed by Flinn and Satyanarayanan (1999). They suggested having the display consist of some number of zones that can be independently powered up or down. In Fig. 5-2, we depict 16 zones using dashed lines to separate them. When the cursor is in window 2, as shown in Fig. 5-2(a), only the four zones in the lower righthand corner have to be lit up. The other 12 can be dark, saving 3/4 of the screen power.

When the user moves the cursor to window 1, the zones for window 2 can be darkened and the zones behind window 1 can be turned on. However, because window 1 straddles 9 zones, more power is needed. If the window manager can sense of what is happening, it can automatically move window 1 to fit into four zones, with a kind of snap-to-zone action, as shown in Fig. 5-2(b). To achieve this reduction from 9/16 of full power to 4/16 of full power, the window manager has to understand power management or be capable of accepting instructions from some other piece of the system that does. Even more sophisticated would be the ability to partially illuminate a window that was not completely full (e.g., a window containing short lines of text could be kept dark on the right hand side).

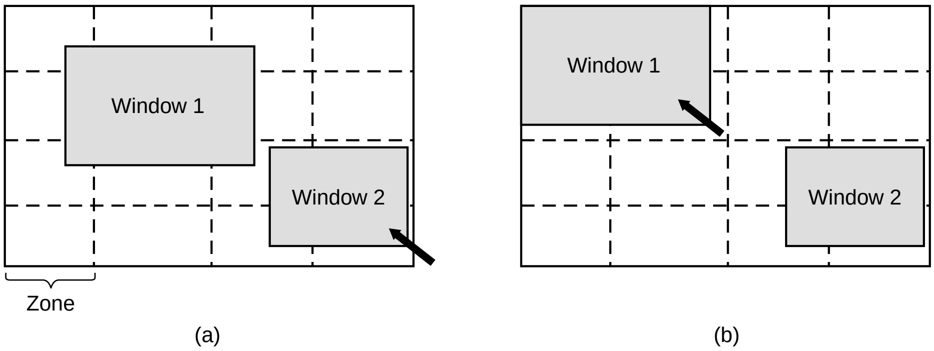

Figure 5-2. The use of zones for backlighting the display. (a) When window 2 is selected it is not moved. (b) When window 1 is selected, it moves to reduce the number of zones illuminated.

The Hard Disk

Another major villain is the hard disk. It takes substantial energy to keep it spinning at high speed, even if there are no accesses. Many computers, especially laptops, spin the disk down after a certain number of minutes of activity. When it is next needed, it is spun up again. Unfortunately, a stopped disk is hibernating rather than sleeping because it takes quite a few seconds to spin it up again, which causes noticeable delays for the user.

In addition, restarting the disk consumes considerable extra energy. As a consequence, every disk has a characteristic time, T_d, that is its break-even point, often in the range 5 to 15 sec. Suppose that the next disk access is expected to some time t in the future. If $t < T_d$, it takes less energy to keep the disk spinning rather than spin it down and then spin it up so quickly. If $t > T_d$, the energy saved makes it worth spinning the disk down and up again much later. If a good prediction could be made (e.g., based on past access patterns), the operating system could make good shutdown predictions and save energy. In practice, most systems are conservative and only spin down the disk after a few minutes of inactivity.

Another way to save disk energy is to have a substantial disk cache in RAM. If a needed block is in the cache, an idle disk does not have to be restarted to satisfy the read. Similarly, if a write to the disk can be buffered in the cache, a stopped disk does not have to restarted just to handle the write. The disk can remain off until the cache fills up or a read miss happens.

Another way to avoid unnecessary disk starts is for the operating system to keep running programs informed about the disk state by sending it messages or signals. Some programs have discretionary writes that can be skipped or delayed. For example, a word processor may be set up to write the file being edited to disk every few minutes. If the word processor knows that the disk is off at the moment it would normally write the file out, it can delay this write until the disk is next turned on or until a certain additional time has elapsed.

The CPU

The CPU can also be managed to save energy. A laptop CPU can be put to sleep in software, reducing power usage to almost zero. The only thing it can do in this state is wake up when an interrupt occurs. Therefore, whenever the CPU goes idle, either waiting for I/O or because there is no work to do, it goes to sleep.

On many computers, there is a relationship between CPU voltage, clock cycle, and power usage. The CPU voltage can often be reduced in software, which saves energy but also reduces the clock cycle (approximately linearly). Since power consumed is proportional to the square of the voltage, cutting the voltage in half makes the CPU about half as fast but at 1/4 the power.

This property can be exploited for programs with well-defined deadlines, such as multimedia viewers that have to decompress and display a frame every 40 msec, but go idle if they do it faster. Suppose that a CPU uses x joules while running full blast for 40 msec and $x/4$ joules running at half speed. If a multimedia viewer can decompress and display a frame in 20 msec, the operating system can run at full power for 20 msec and then shut down for 20 msec for a total energy usage of $x/2$ joules. Alternatively, it can run at half power and just make the deadline, but use only $x/4$ joules instead. A comparison of running at full speed and full power for some time interval and at half speed and one quarter power for twice as long is shown in Fig. 5-3. In both cases the same work is done, but in Fig. 5-3(b) only half the energy is consumed doing it.

In a similar vein, if a user is typing at 1 char/sec, but the work needed to process the character takes 100 msec, it is better for the operating system to detect the long idle periods and slow the CPU down by a factor of 10. In short, running slowly is more energy efficient than running quickly.

The Memory

Two possible options exist for saving energy with the memory. First, the cache can be flushed and then switched off. It can always be reloaded from main memory with no loss of information. The reload can be done dynamically and quickly, so turning off the cache is entering a sleep state.

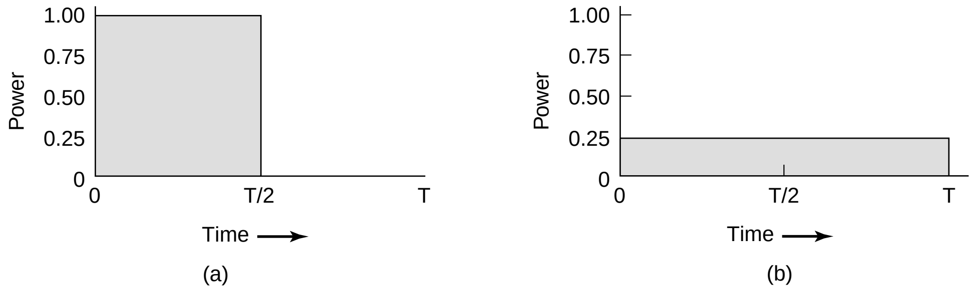

Figure 5-3. (a) Running at full clock speed. (b) Cutting voltage by two cuts clock speed by two and power consumption by four.

A more drastic option is to write the contents of main memory to the disk, then switch off the main memory itself. This approach is hibernation, since virtually all power can be cut to memory at the expense of a substantial reload time, especially if the disk is off too. When the memory is cut off, the CPU either has to be shut off as well or has to execute out of ROM. If the CPU is off, the interrupt that wakes it up has to cause it to jump to code in a ROM so the memory can be reloaded before being used. Despite all the overhead, switching off the memory for long periods of time (e.g., hours) may be worth it if restarting in a few seconds is considered much more desirable than rebooting the operating system from disk, which often takes a minute or more.

Wireless Communication

Increasingly many portable computers have a wireless connection to the outside world (e.g., the Internet). The radio transmitter and receiver required are often first-class power hogs. In particular, if the radio receiver is always on in order to listen for incoming email, the battery may drain fairly quickly. On the other hand, if the radio is switched off after, say, 1 minute of being idle, incoming messages may be missed, which is clearly undesirable.

One efficient solution to this problem has been proposed by Kravets and Krishnan (1998). The heart of their solution exploits the fact that mobile computers communicate with fixed base stations that have large memories and disks and no power constraints. What they propose is to have the mobile computer send a message to the base station when it is about to turn off the radio. From that time on, the base station buffers incoming messages on its disk. When the mobile computer switches on the radio again, it tells the base station. At that point any accumulated messages can be sent to it.

Outgoing messages that are generated while the radio is off are buffered on the mobile computer. If the buffer threatens to fill up, the radio is turned on and the queue transmitted to the base station.

When should the radio be switched off? One possibility is to let the user or the application program decide. Another is turn it off after some number of seconds of idle time. When should it be switched on again? Again, the user or program could decide, or it could be switched on periodically to check for inbound traffic and transmit any queued messages. Of course, it also should be switched on when the output buffer is close to full. Various other heuristics are possible.

Thermal Management

A somewhat different, but still energy-related issue, is thermal management. Modern CPUs get extremely hot due to their high speed. Desktop machines normally have an internal electric fan to blow the hot air out of the chassis. Since reducing power consumption is usually not a driving issue with desktop machines, the fan is usually on all the time.

With laptops, the situation is different. The operating system has to monitor the temperature continuously. When it gets close to the maximum allowable temperature, the operating system has a choice. It can switch on the fan, which makes noise and consumes power. Alternatively, it can reduce power consumption by reducing the backlighting of the screen, slowing down the CPU, being more aggressive about spinning down the disk, and so on.

Some input from the user may be valuable as a guide. For example, a user could specify in advance that the noise of the fan is objectionable, so the operating system would reduce power consumption instead.

Battery Management

In ye olde days, a battery just provided current until it was drained, at which time it stopped. Not any more. Laptops use smart batteries now, which can communicate with the operating system. Upon request they can report on things like maximum voltage, current voltage, maximum charge, current charge, maximum drain rate, current drain rate, and more. Most laptop computers have programs that can be run to query and display all these parameters. Smart batteries can also be instructed to change various operational parameters under control of the operating system.

Some laptops have multiple batteries. When the operating system detects that one battery is about to go, it has to arrange for a graceful cutover to the next one, without causing any glitches during the transition. When the final battery is on its last legs, it is up to the operating system to warn the user and then cause an orderly shutdown, for example, making sure that the file system is not corrupted.

Driver Interface

The Windows system has an elaborate mechanism for doing power management called **ACPI** (**Advanced Configuration and Power Interface**). The operating system can send any conformant driver commands asking it to report on the capabilities of its devices and their current states. This feature is especially important when combined with plug and play because just after it is booted, the operating system does not even know what devices are present, let alone their properties with respect to energy consumption or power manageability.

It can also sends commands to drivers instructing them to cut their power levels (based on the capabilities that it learned earlier, of course). There is also some traffic the other way. In particular, when a device such as a keyboard or a mouse detects activity after a period of idleness, this is a signal to the system to go back to (near) normal operation.

5.9.3 Degraded Operation

So far we have looked at ways the operating system can reduce energy usage by various kinds of devices. But there is another approach as well: tell the programs to use less energy, even if this means providing a poorer user experience (better a poorer experience than no experience when the battery dies and the lights go out). Typically, this information is passed on when the battery charge is below some threshold. It is then up to the programs to decide between degrading performance to lengthen battery life or to maintain performance and risk running out of energy.

One of the questions that comes up here is how can a program degrade its performance to save energy? This question has been studied by Flinn and Satyanarayanan (1999). They provided four examples of how degraded performance can save energy. We will now look at these.

In this study, information is presented to the user in various forms. When no degradation is present, the best possible information is presented. When degradation is present, the fidelity (accuracy) of the information presented to the user is worse than what it could have been. We will see examples of this shortly.

In order to measure energy usage, Flinn and Satyanarayanan devised a software tool called PowerScope. What it does is provide a power usage profile of a program. To use it, a computer must be hooked up to an external power supply through a software-controlled digital multimeter. Using the multimeter, software can read out the number of milliamperes coming in from the power supply and thus determine the instantaneous power being consumed by the computer. What PowerScope does is periodically sample the program counter and the power usage and write these data to a file. After the program has terminated the file is analyzed to give the energy usage of each procedure. These measurements formed the basis of their observations. Hardware energy saving measures were also used and formed the baseline against which the degraded performance was measured.

The first program measured was a video player. In undegraded mode, it plays 30 frames/sec in full resolution and in color. One form of degradation is to abandon the color information and display the video in black and white. Another form of degradation is to reduce the frame rate, which leads to flicker and gives the movie a jerky quality. Still another form of degradation is to reduce the number of pixels in both directions, either by lowering the spatial resolution or making the displayed image smaller. Measures of this type saved about 30% of the energy.

The second program was a speech recognizer. It sampled the microphone to construct a waveform. This waveform could either be analyzed on the laptop computer or sent over a radio link for analysis on a fixed computer. Doing this saves CPU energy but uses energy for the radio. Degradation was accomplished by using a smaller vocabulary and a simpler acoustic model. The win here was about 35%

The next example was a map viewer that fetched the map over the radio link. Degradation consisted of either cropping the map to smaller dimensions or telling the remote server to omit smaller roads, thus requiring fewer bits to be transmitted. Again here a gain of about 35% was achieved.

The fourth experiment was with transmission of JPEG images to a Web browser. The JPEG standard allows various algorithms, trading image quality against file size. Here the gain averaged only 9%. Still, all in all, the experiments showed that by accepting some quality degradation, the user can run longer on a given battery.