Effects of Software on Computer Power Usage Report 2

Arnold Pereira and Lincoln Roop

Department of Electrical & Computer Engineering
Carnegie Mellon University
Pittsburgh, PA 15213
{arnoldp, lroop}@andrew.cmu.edu

Abstract

As more and more computing work is pushed out to mobile devices, power consumption becomes increasingly important to deliver the balance of performance and battery life end users need. We intend to study the effects that different software programs, including operating systems and applications have on computer power consumption. Further, equipped with this knowledge we aim to optimize the system for power consumption depending on the current workload. Our goal in this project is to develop a set of recommendations as to which software programs are the most power-aware for a given task, as well as to develop a "low-power" power management mode that can disable unnecessary tasks to yield improved runtime above that currently attained by hardware power management tactics such as Dynamic Voltage & Frequency Scaling or reducing display brightness.

1. Introduction

Typical computer power management strategies rely on reducing power usage by removing power from unused devices and dynamically scaling operating voltage and clock frequency. That is to say, these approaches are primarily hardware-centric. While these approaches are able to offer significant performance improvement in many usage cases, they only take into consideration total system load. This would be an adequate approach for single-purpose computers, however general-purpose computers running modern multitasking operating systems run many concurrent processes, each contributing their own share of work and power consumption.

By studying different applications, including operating system components, we can identify tasks which are significant contributors of power consumption. In some cases, it may be possible to disable or reduce the execution frequency of less important tasks that consume a significant amount of power. Furthermore, by comparing the power performance of multiple applications designed to perform the same task, we can make recommendations as to what software packages are the most power-friendly.

A lot of traditional applications such as word processing, picture editing, video editing and storage are being moved to the cloud as cloud computing technology improves and becomes more widely accepted. This represents a shift in power consumption trends from the client to the server. However, internet browsers also consume energy, which will have to be compared vis-à-vis the energy consumption of traditional client-side software which performs the same task (for instance Google Docs vs. LibreOffice). We assume the software choice for a user is based only on power and performance and not on GUI. However, GUI is correlated with processor workload and so, on power consumption.

A lot of processes run in the background, both in Windows and Linux. It is a popular trend nowadays for software developers to include automatic updaters or quick launchers for programs which run in the background. Not only does this have an effect on the performance, but considering these run for the entire uptime of the system, their power consumption could accumulate.

Along the same lines, operating systems have various hardware drivers that run to interact with the various hardware or Plug and Play devices on the system. In a particular boot and shut down cycle, it is a possibility that one or many software drivers will not perform any particular task for the user.

2. Prior Work

At a broader level, Windows and Linux operating systems have been compared for their power consumption at different frequency levels and at different loads. The reasons for the difference have not been explored clearly and hence optimization for work load based power consumption could be developed further. Comparisons are often done for servers as power consumption often influences choice of operating system. However, studies for personal computers have not been conducted to the same extent. Linux has seen development of PowerTOP which finds unnecessary programs consuming power in idle mode. Tickless Idle is another project that eliminates the periodic timer tick to save power.

In the referenced paper by Mahesri and Vardhan(2004), a laptop was analyzed for component-wise

breakdown of power consumption. For a laptop, they found that the major source of power is the processor, followed by the CD-R/RW, LCD backlight and 802.11 wireless.

Work has also been done at the operating system level by IBM for RedHat Linux operating system. Using the CPUfreq tool, the in-kernel governors (for regulating CPU clock speed) are selected after getting information about the C and P states of the system. The C states deal with the sleep modes while the P states deal with the voltage and frequency in the active mode of the processor. Most of this work pertains to the hardware level and how to optimize it for the processes it runs, we wish to focus on the software layer.

3. Testing Methodology

Since we are to study the effects of software on the hardware platform provided, benchmarking is required to compare various software programs and to analyze their usage. Currently, Linux benchmarks are primarily focused on various operations like multimedia or CPU performance. However, for our application we need to determine power effects: For example, we wish to compare the power performance of audio or video playback within a standard application versus within a browser. To the best of our knowledge, there does not exist any 'energy benchmark' for operating systems today. To a certain extent, we will be benchmarking software programs on the issue of. Differences in consumption occur due to variations in processor usage, memory access, cache utilization, disk cache utilization, etc.

We are using a Dell OptiPlex 755 desktop as our test platform. This machine uses a 2.67 GHz Intel Core 2 Quad CPU which supports a limited range of frequency scaling steps (2.0, 2.33, and 2.67 GHz). We will implement our power monitoring by making use of the internal power management systems integrated into the machine, as well as conventional electronics test equipment. By making use of conventional test equipment as well as the computer's internal power management data, we can verify that the computer is providing accurate power consumption data as well as monitor the power consumption of individual components if necessary. Hardware monitoring will be done by measuring the DC voltage and current. Current can be measure by introducing a high power, high precision, low value resistor in the path of the current. By measuring voltage across the resistor and dividing it by the resistor value, we can derive the supply current.

The conventional test equipment used consist of an AC plugmeter and multiple digital multimeters. The AC plugmeter is placed between the PC's power cord and the wall outlet and wirelessly delivers statistics regarding AC power consumption to a computer for data collection. The digital multimeters we are using are Hewlett-Packard model 34401A 6 ½ digit bench multimeters. These multimeters can be remotely controlled by a computer over RS-232 or GPIB, we will use this functionality to collect DC voltage and current statistics for the individual power supply output rails automatically. This automation will enable us to produce a very accurate average power consumption calculation. A block diagram of the DC monitoring system is given in Figure 1.

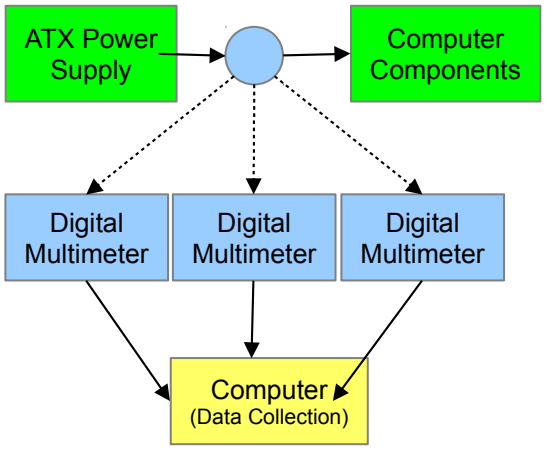

Figure 1: DC Monitoring System

The AC plugmeter is a wireless device that returns the RMS value of the AC current and voltage drawn by the device which is connected into it. It also returns a power value, which is not calibrated to Watts but is a linear value. In addition, it transmits the AC frequency and on-time of the device, which is not relevant to our requirements. The results obtained from the plug meter are tabulated in Figure 2. The objective of the AC data is to find out the effect of varying system power consumption on the AC mains which is the power which is finally billed. However, transformer inefficiencies in the SMPS (Switched Mode Power Supply) have the potential to dampen the changes in DC system power consumption. This would not be in the case of laptops, as they run directly on batteries.

Our testing of operating system components will be performed on a Linux operating system (Ubuntu 10.04 LTS), as its open-source and modular design enables us to operate the system disabling key components that would be difficult or impossible to disable on a Windows platform. We will test application software (Mozilla Firefox, Matlab, or LibreOffice for instance) as well as comparing the results of different applications which serve the same purpose (Firefox vs Google Chrome, etc). In order to deliver accurate results, we will test typical usage cases for each application. For instance, applications such as web browsers are not typically processing large quantities of data (although they very well may be in certain cloud computing or web-based tasks), where a scientific application such as Matlab does exactly that.

	Workload	Reported Power
1.	General	12-13
2	With CR Read	15
3	Continuous HDD access (du command)	17-18
4	Stress Benchmark -8 threads	25
5	Stress Benchmark – 25 threads	26

Figure 2: Data collected from AC plugmeter

We will also be using benchmarking software, though not yet decided, to find the trends in operating systems and in applications. Once the trends are established, we hope to develop a 'Browser-oriented' mode which would disable or kill unnecessary processes which run in the background and have less than 10% utilization. We believe that this will deliver significant power savings to many users due to the extreme proliferation of web-based applications in modern consumer computing.

With respect to our project, we shall be discussing tools in Linux for power monitoring and control along with the results obtained:

3.1. PowerTOP

PowerTOP is a Linux tool to monitor the wake up requests generated by various system processes. This includes the Linux scheduler, the interrupts raised by 802.11 WLAN driver, keyboard, mouse and other peripheral devices. A key note here is that these frequent interrupts prevent the CPU from remaining in the deep sleep state for an extended period of time.

CPU performance states are defined by P-states while its operating states are defined by C-states. These performance states will have different CPU operating frequencies to take reduce CPU performance and power consumption. The number of P states varies for different processor generations. For example, our test hardware system has an Intel Core 2 Quad with 4 cores having three P states defined by three frequencies- 2.00GHz, 2.33GHz and 2.67GHz. Different C –states correspond to different techniques employed to save power to a varying degree. C0 is the CPU running state. C1 is when the clock supply to certain/ all modules are cut off. In C2, the clock supply to the external peripherals as well as the internal clock supply is cut off. C3 and above states cut off clock supply as well as reduce voltages to a low value or zero, depending on the processor architecture.

In our tests, we ran PowerTOP on the OptiPlex workstation. Some of the major causes of interrupts have been summarized in Figure 3. Further exploration is required into interrupts caused by different types of the same hardware/ software. For example, we have to conduct tests by connecting mobile phones from various manufacturers or running different operating systems.

	Source	Interrupt %	Scope for power consumption reduction
1	Kernel Scheduler	44-90	Time quantum to run scheduler can be varied
2	Input- Mouse	7-15%	N/A
3	USB storage(Idle)	4-6%	Kill monitoring process
4	Mobile phone connected via USB (idle)	4-6%	Kill monitoring process
5	Flash player plugin(idle)	10-25%	Kill process

Figure 3: Major Causes of Interrupts

NOTE: Idle means that the process is not required or the user does not wish to access the device or data at that instant

3.2. CPUfreq

CPUfreq is a subsystem in Linux kernel which provides an operating system CPU frequency control. It has a corresponding daemon called CPUfreqd. There are five governors in CPUfreq: performance, powersave, ondemand, conservative and userspace. These modes are described in the following paragraph:

CPUfreqd has a configuration file cpufreqd.conf which can be modified by a root user. This file decides which of the governor is active in a given period of time. For example, a user can define performance governor to be applicable when the AC supply is active, or the powersave governor to be active when the battery is less than 40% with no AC supply. Userspace governor defines profiles for common applications on the system. Ondemand and conservative are the dynamic governors. Two variables are defined here by CPUfreq; namely up_threshold and down_threshold. Both the dynamic governors change the frequency to the next available frequency. Generally, the upper and lower threshold are defined as 98 and 95 % respectively. So if the CPU utilization exceeds 98%, the ondemand governor changes the frequency of the CPU to the next available one. Conservative will do the same, except require the CPU utilization criteria to remain true for a longer time, i.e. more delay in changing.

We shall be modifying the configuration files to achieve our custom power save governor.

Isolating the effects of processes on the power being consumed is a concern. Adding or deleting processes would cause a slight change in power consumption. However, we must also take into account the contribution by the monitoring process: CPUfreqd itself has some processor and memory footprint. CPU footprint can be reduced by reducing the frequency of the governor checks.

4. Challenges

From the preliminary results we obtained from the plug meter and PowerTOP, we would like to study how the Linux scheduler quantum affects the power consumption pattern, currently our results show that it raises the highest number of interrupts. In the idle state if this value can be reduced, it could potentially reduce the power consumption further. This would require recompiling the kernel to change its scheduler quantum, which is currently 1ms. Two specific challenges here are the corresponding unresponsiveness and how to dynamically change this time quantum.

A second challenge we anticipate is how to automate traditionally interactive tasks such as browsing, watching videos, Facebook, Google search, etc. and eliminate variation caused by background tasks or the automation itself. We feel that the best solution to this issue is simply to not automate. It has been suggested that developing 'scripts' that we or others we enlist to help with our data collection would follow. These scripts will be designed to simulate typical ways in which end users interact with computers, and if followed carefully and results averaged out over several trials should yield results similar to an ideal automated solution (one which simulates identical usage patterns across all runs and different applications without contributing any power consumption variation that would be difficult to filter out).

Each script will be designed to take 15 to 30 minutes of user time, as we believe this will be a long enough period to collect useful data without being too long to be feasible. During the session, a user (one of us or someone we enlist to help) will operate our workstation, performing tasks one would commonly use a computer for. Our automated power consumption monitoring system will periodically collect power consumption data, allowing us to calculate the average power usage of the workstation during the usage session. Inconsistencies in data collected will be averaged out through the conduction of many such sessions; we believe this is a suitable method to produce useful experimental data while at the same time not requiring the usage scripts to be incredibly repetitive and thus difficult to perform consistently.

Project Schedule

We have divided the work needed to complete this project into several milestones as described below:

- October 25: Prepare hardware test bench environment – OS installed on computer, power monitoring software working properly.
- November 11: Prepare software benchmarks, finish usage scripts for evaluating interactive applications such as web browsers. Make necessary hardware modifications for DC power monitoring.
- November 18: Collect and analyze power usage data from applications.
- December 1: Develop a power management technique based on data from software power management. Vary Linux scheduler quantum and develop custom CPUfreq governor.

References

1. Aqeel Mahesri and Vibhore Vardhan, “Power Consumption on a Modern Laptop”, *Power-Aware Computer Systems* (2005), pp. 165-180
2. R. Ayoub, U. Ogras, E. Gorbatov, Y. Jin, T. Kam, P. Diefenbaugh and T. Rosing. OS- Level Power Minimization Under tight performance constraints in general purpose systems. *In proceedings ISLPED, 2011*
3. Daniel P. Bovet, Marco Cesati. Understanding the Linux Kernel
4. “Chrome and Firefox power consumption in Linux”, www.benchmarkreviews.com
5. Projects, www.lesswatts.org
6. Jennifer Hopper, “Reduce Linux Power Consumption”, 2009
7. “Linux captures the green flag, beats Windows 2008 Power Saving Measures”, www.networkworld.com