Effects of Software on Computer Power Usage

Arnold Pereira and Lincoln Roop

Electrical & Computer Engineering Department
Carnegie Mellon University

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.

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).

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.

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.

Technical Details:

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 will use a standard x86-based notebook computer as our test platform. We will implement our power monitoring by making use of the internal power management systems integrated into modern notebook computers, 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 battery 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.

Our testing of operating system components be performed on a Linux operating system, 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 on both Windows and Linux platforms, allowing us to compare results for the same application running on Windows or Linux if versions exist for both operating systems, (Mozilla Firefox, Matlab, or LibreOffice for instance) as well as comparing the results of different applications which serve the same purpose. 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.

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.

Challenges:

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. If we can assume that the monitoring application's power consumption remains reasonably constant under different situations, then we can eliminate its effect from our results by simply comparing change in power consumption rather than absolute power consumption.

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. It may be the case 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 would 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).

Project Schedule

- October 6: Prepare hardware testbench environment – OS installed on computer, power monitoring software working properly and verified with multimeter.
- October 20: Prepare software benchmarks, develop method for evaluating interactive applications such as web browsers.
- November 10: Collect data from benchmarks
- December 1: Develop a power management technique based on data from software power management.

References

1. Aqeel Mahesri and Vibhore Vardhan, "Power Consumption on a Modern Laptop", *Power-Aware Computer Systems* (2005), pp. 165-180
2. "Chrome and Firefox power consumption in Linux", www.benchmarkreviews.com
3. Projects, www.lesswatts.org
4. Jennifer Hopper, "Reduce Linux Power Consumption", 2009
5. "Linux captures the green flag, beats Windows 2008 Power Saving Measures", www.networkworld.com