Green Tracker: A Tool for Estimating the Energy Consumption of Software

Nadine Amsel
Donald Bren School of Information and Computer Sciences
University of California, Irvine
6210 Donald Bren Hall
Irvine, CA 92697-3425
namsel@uci.edu

Bill Tomlinson
Donald Bren School of Information and Computer Sciences
University of California, Irvine
5068 Donald Bren Hall
Irvine, CA 92697-3425
wmt@uci.edu

CHI 2010, April 10–15, 2010, Atlanta, Georgia, USA.
ACM 978-1-60558-930-5/10/04.

Abstract
The energy consumption of computers has become an important environmental issue. This paper describes the development of Green Tracker, a tool that estimates the energy consumption of software in order to help concerned users make informed decisions about the software they use. We present preliminary results gathered from this system's initial usage. Ultimately the information gathered from this tool will be used to raise awareness and help make the energy consumption of software a more central concern among software developers.

Keywords
Software, sustainability, green computing, green IT

ACM Classification Keywords
H5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous.

General Terms
Measurement, performance

Introduction
In 2007, a Gartner report estimated that information and communication technologies (ICTs) were responsible for 2% of global carbon emissions [5]. A

PC Energy Report sponsored by 1E [8] states that an average-sized company with approximately 10,000 PCs wastes at least $165,000 in electricity costs each year due to computers being left on overnight. The Report states that the company could reduce atmospheric CO_2 content by 1,381 tons by turning these computers off [8]. However, many IT professionals believe that computers (and the businesses that surround them) operate more effectively if always left on. The energy consumption of computers is a problem that needs to be addressed; one way of doing so is to make computer software more energy efficient.

While computer science researchers are becoming more aware of the environmental impact of their actions, sustainability has yet to play a major role in the development of most software systems. Carla Schlatter Ellis [2] argues that "reducing energy consumption should be raised to first-class status among performance goals when software is being designed" (pg. 1). Our goal with Green Tracker is to do just this: to raise awareness that software systems do have differing levels of energy consumption. Additionally, Ellis argues that "performance measurement studies should regularly include appropriate energy metrics in addition to the more traditional ones (e.g. time, bandwidth)" (pg. 2).

The Green Tracker system attempts to close the gap between energy consumption and software engineering described by Ellis by providing information about the energy consumption of existing software systems. For typical computer usages, such as browsing the internet, creating text documents, and listening to music, there are many software programs from which to choose. By informing users of the impact of their software choices, Green Tracker can encourage users to use those software systems that are the most environmentally sustainable.

A second motivation behind the Green Tracker system is to collect data about software energy consumption across different computer configurations. By implementing a software-based energy meter, we can deploy Green Tracker to users with different computers and different configurations and collect the results. By doing so, we can disseminate information more broadly about the environmental impacts of various software systems.

Related Work

There has been previous research on estimating the energy consumption of software. Seo, et al. [7] created a framework for estimating the energy consumption of pervasive Java-based software systems. The framework estimation is based on the energy consumption of software architecture's components and connectors. This can be a useful framework for early estimation of the energy consumption of a software system's chosen architecture. This framework is primarily focused on software systems still in the development stages for the purpose of extending battery life. Green Tracker is intended for more commercial software systems that are already deployed.

Similar to [7], Gupta and Singh [3] studied the architecture and protocols of the Internet and suggested improvements to increase energy efficiency. According to the authors, allowing subcomponents to sleep or retreat to lower energy states at times when they are not being used could reduce energy

consumption. The authors conclude that this could be an effective strategy with a few changes to Internet architecture and current protocol specifications. Further research into the success of this strategy could offer suggestions for designers to build more energy efficient software.

There has also been some research on the environmental impacts of pervasive computing. According to Jain and Wullert [4], "software will play an increasing role in the environmental impacts of pervasive computing" (pg. 265). The authors argue that with hardware becoming faster and cheaper, more functionality will be implemented in software. This suggests that the effect of software on the energy consumption of computers should be an increasing concern among computer users as well as software engineers.

Chetty, et al. [1] studied the power consumption of computers by observing how people in their homes use built-in power settings. The authors determined that the economic incentives for individuals are not enough to motivate people to change their behavior. However they argue that the collective saving of energy might be a better motivation for people. These findings could also be applied to software energy consumption: while the decrease in energy consumption of one user switching from Firefox to IE may not be significant, the collective energy savings could be significant and thus provide a motivation to users to make the change. Additionally, Chetty, et al. found that users do wish to be informed of how much energy is being consumed by their computers. We hope to provide this information through Green Tracker.

Methods

In order to estimate the energy consumption of any deployed software application, Green Tracker collects information about the computer's CPU. When Green Tracker is first installed, it performs some initial "benchmarking" tasks. The benchmarking tasks are performance-based, which is done by comparing software systems in the class based on energy consumption. Users are prompted to specify which classes of software (that is, software systems that perform essentially the same function) they want to test and asked to specify which software systems in each class are currently installed on their computer. Green Tracker then seeks to determine the average CPU used by of each of the software systems in the specified class. It first opens a specified software system (for example, Mozilla Firefox) and then, at a specified interval, saves a text file with the following items: timestamp, process ID, CPU usage, and command name. From this text file, the CPU usage of that software system is extracted and averaged over the period of time that it was collected (for example, it could collect the average CPU usage of Mozilla Firefox over a period of five minutes).

While CPU data is being collected, the software system currently running automates typical usage patterns. For example, Internet browsers are launched with the URL of a website that uses Javascript to rotate automatically between web pages with HTML content, Flash content, and a Youtube video. Other software systems, such as word processing software and audio software, can be automated by using AppleScript, an automation language for Mac OS X. Scripts that simulate typical usage of word processing software (typing and saving) and audio software (playing an

mp3) have already been implemented. For all systems other than Internet browsers, Green Tracker is responsible for both launching the software system in question and collecting its CPU data, and launching AppleScript to automate usage patterns.

Once the average CPU is calculated for one system, the same process is repeated for all other systems that perform the same function (e.g. Internet Explorer and Google Chrome). When all the systems in one class have been tested, Green Tracker creates a chart comparing the CPUs across all the software systems (Figure 1). Green Tracker also determines which system used the least amount of CPU and thereby recommends which one is the most environmentally sustainable. The results of the CPU tracking (the average CPU data, a chart of the data created using Google Charts, and recommendations) are sent to the user via email. If the user agrees to allow the developers to collect anonymous usage statistics from these benchmarking tests, the information is also emailed to the developers.

Results and Evaluation

Preliminary results show that software systems in the same class have different levels of energy consumption. All tests were performed on an iMac desktop computer with Mac OS X installed. The Internet browsers tested were Mozilla Firefox 3.0.10 and Safari 4.0.3. We found that, on this system, the average CPU power of Firefox was lower than Safari. The average CPU of Firefox over 5 trials was 28.63% while Safari was 30.98%.

We also tested word processing software using Microsoft Word 2007 and Open Office Writer 3.1.1. We used an AppleScript to automate typing in Word and Writer for 5 minutes. The average CPU of Word over 5 trials was 5.2% while Open Office Writer was 3.65%.

Finally, we tested audio software using iTunes 9.0.2 and VLC Media Player 1.0. We used an AppleScript to automate playing an mp3 for 4 minutes. The average CPU of iTunes over 5 trials was 4.27% while VLC Media Player was 4.43%.

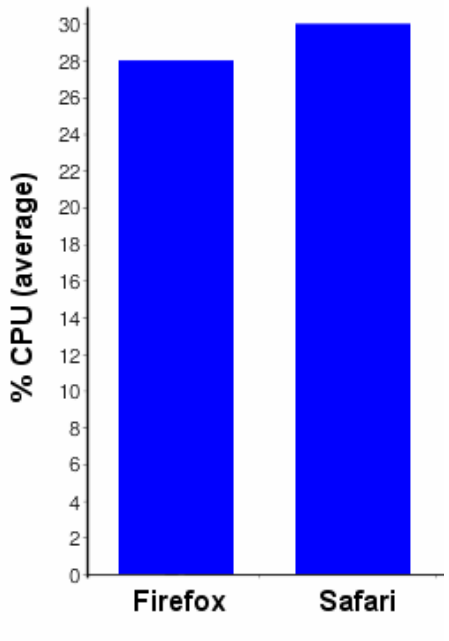

Figure 1. An example of the Green Tracker Google Chart results comparing Firefox and Safari Internet browsers.

While a software system's CPU usage is often not the only aspect of computer energy consumption (for example, disk access may also consume significant

power), it is nevertheless a key component of the system's total energy usage. The energy consumption of other software systems has previously been measured in this way. For example, the website for Notepad++, a text-editor for Windows, specifically states that it tries to reduce CO_2 emissions by "using less CPU power, [allowing the PC to] throttle down and reduce power consumption, resulting in a greener environment" [6].

Some may argue that it is unreasonable to criticize the energy consumption of already existing software systems, many of which were probably designed before energy consumption of computers became an important issue. However Ellis [2] argues that "even for [software systems] that are not explicitly aimed at energy conservation, it is legitimate and valuable to raise the question of their impact on energy consumption" (pg. 5). The results presented in this paper are suggestions that we hope will be used as initial steps towards creating more sustainable software.

We also recognize the concern that creating more efficient software may increase the usage of such software, a phenomenon known as "induced demand," and therefore possibly offset or even outweigh the overall benefits. However we hope that creating awareness about this phenomenon will encourage users to simply substitute more energy efficient software for their existing systems rather than increase their overall usage.

Future Work

Beyond affecting software users' behavior, we hope that pointing out the inefficiencies of software systems will encourage IT companies to re-evaluate their products and work to reduce the environmental impact of them. By understanding which software systems are the most energy efficient, we can also work towards understanding what makes one system more sustainable than another. This leads to the new research field of sustainable software engineering, which aims to establish software engineering techniques that reduce the environmental impacts of software. By using sustainable software engineering practices, companies may be able to reduce the environmental footprint of the software they produce.

Our initial plans to deploy Green Tracker to users involves posting the code on a website and encouraging interested users to download the code and run it on their machines. If the user agrees, any results from the benchmarking tests will be emailed both to the user and to the authors. We hope that advertising on campus and through the media via a press release will create awareness about the availability of our tool and encourage users to download Green Tracker. If the pleasure of contributing to research is not enough to incentivize users to submit their Green Tracker results, we may provide a monetary incentive to participants.

In future deployments of Green Tracker to various types of computers with different software configurations, we anticipate that there will be some recurring trends in the results. This will help us determine which software systems are consistently the most efficient across different configurations. Future work will involve studying the most sustainable software systems to create suggestions for software developers to design more energy efficient systems.

Conclusion

In this paper we have described Green Tracker, a tool for estimating the energy consumption of currently installed software systems. Green Tracker first performs benchmarking tests to determine which software systems are the most efficient given the user's current computer configuration. This information is then presented to the user in the form of a chart comparing the CPU usage of software systems in the same class. The results are then used to make suggestions to the user about which software systems to use. By running Green Tracker on an Apple iMac desktop and testing internet browsers, word processing software, and audio software we demonstrated that the system was able to find differences in the energy consumption of software in each of these classes.

We consider the development of this tool to be critical for establishing which software systems are the most environmentally sustainable. Subsequently we aim to create awareness about the potential environmental hazards associated with software and to improve software engineering techniques to reduce the energy consumption of software.

Acknowledgements

The authors wish to thank the Social Code Group, the students in the Fall 2009 ICS 295: Environmental Issues in Information Technology class, the Donald Bren School of Information and Computer Sciences, and the California Institute for Telecommunications and Information Technology. This material is based upon work supported by the National Science Foundation under Grant No. 0644415 and by the Alfred P. Sloan Foundation.

References

[1] Chetty, M., Brush, A.B., Meyers, B.R., and Johns, P. It's not easy being green: understanding home computer power management. Proc. of the 27th international conference on Human factors in computing systems, ACM (2009), 1033-1042.

[2] Ellis, C.S. The Case for Higher-Level Power Management. *Proc. of the Seventh Workshop on Hot Topics in Operating Systems*, IEEE Computer Society (1999), 162.

[3] Gupta, M. and Singh, S. 2003. Greening of the internet. In *Proc. 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications* (August 25 - 29, 2003). SIGCOMM '03. ACM, New York, NY, 19-26.

[4] Jain, R. and Wullert II, J. Challenges: environmental design for pervasive computing systems. *Proc. of the 8th annual international conference on Mobile computing and networking*, ACM (2002), 263-270.

[5] Mingay, S. (2007). Green IT: A New Industry Shock Wave. Retrieved August 1, 2009, from http://www.ictliteracy.info/rf.pdf/Gartner_on_Green_IT.pdf.

[6] Notepad++. http://notepad-plus.sourceforge.net/uk/site.htm.

[7] Seo, C., Malek, S., and Medvidovic, N. 2008. Estimating the Energy Consumption in Pervasive Java-Based Systems. In *Proc. 2008 Sixth Annual IEEE international Conference on Pervasive Computing and Communications* (March 17 - 21, 2008). PERCOM. IEE Computer Society, Washington, DC, 243-247.

[8] U.S. PC Energy Report 2007, sponsored by 1E. Climate Savers Computing, 2007. http://www.climatesaverscomputing.org/docs/Energy_Report_US.pdf.