Green Mining: Investigating Power Consumption Across Versions

Abram Hindle
Department of Computing Science
University of Alberta,
Edmonton, CANADA
abram.hindle@ualberta.ca

***Abstract*—Power consumption is increasingly becoming a concern for not only electrical engineers, but for software engineers as well, due to the increasing popularity of new power-limited contexts such as mobile-computing, smart-phones and cloud-computing. Software changes can alter software power consumption behaviour and can cause power performance regressions. By tracking software power consumption we can build models to provide suggestions to avoid power regressions. There is much research on software power consumption, but little focus on the relationship between software changes and power consumption. Most work measures the power consumption of a single software task; instead we seek to extend this work across the history (revisions) of a project. We develop a set of tests for a well establish product and then run those tests across all versions of the product while recording the power usage of these tests. We provide and demonstrate a methodology that enables the analysis of power consumption performance for over 500 versions of Firefox 3.6; we show that software change does induce changes in power consumption. This methodology and case study are a first step towards combining power measurement and mining software repositories research, thus enabling developers to avoid power regressions via power consumption awareness.**

***Keywords*-power; power consumption; mining software repositories; dynamic analysis; sustainable-software**

I. Introduction

Until recently, the effect of software, such as desktop applications and server software, on power consumption has been ignored under the tacit assumption that software engineers could use all available resources; but with the advent of contexts such as mobile and cloud computing it is clear that the design and implementation of software has a significant impact on power consumption and software engineers should be aware of it.

Within these contexts resources are limited: smart-phones/mobile devices have slower CPUs and limited battery life; cloud computing nodes are resource limited in terms of CPU speed, memory size, disk I/O [1], heat, and network bandwidth, all of which use power; software services consume power by simply being available. It is estimated [2], [3] that computer power usage alone costs many mid-sized businesses more than $100000 per year, producing over 1000 tonnes of CO_2! Thus resource conservation results in more availability, more battery-life, and smaller power bills.

Current power consumption research tends to focus on CPU usage and ignores the actual patterns of power consumption induced by software evolution and change [4]–[6]. Thus we propose *green mining*, a research agenda that aims to help developers understand power consumption issues related to their code, based on a corpus of software changes associated with power consumption. *Green mining* rests on two pillars, *mining software repositories* research, to find actual software changes, and *instrumentation* combined with *dynamic analysis*, in order to measure and correlate change to power usage. *Mining software repositories* (MSR) research is about the analysis of artifacts found within the huge corpus of software-repositories, such as the version control systems of open-source projects [7]. *Instrumentation* and *dynamic analysis* will be used to investigate the factors and resource utilization of changing software. We will look at each change in a version control system and dynamically measure its effect on power consumption. We plan to compile these patterns of software changes in order to answer questions about software power consumption. **Green mining leverages historical information extracted from the corpus of publicly available software to help provide software power consumption advice.**

Green Mining models how software maintenance impacts a system's power usage. It aims to help software engineers reduce the power consumption of their own software by estimating the impact of software modifications on power consumption. This is a novel extension of *mining software repositories research* into the realm of software-based power consumption as previous research does not address the effect of software change on power consumption. Our research questions include:

- Does Software Change relate to power consumption?
- What information needs to be gathered to estimate power consumption? Is CPU measurement enough?

This work has a potential environmental impact: deployed software will consume less resources and thus have a direct reduction on the CO_2 emissions [3]. This work is industrially relevant because vendors interested in GreenIT [3], such as Apple, Microsoft, Intel, and IBM, have already begun investing in power-management documentation and tools [8]–[12].

In this paper we will address the first steps towards *green*

mining: measuring the power use of multiple versions of a software system, investigating the effects of software change on power use, and creating a corpus/database of this power use. This paper's contributions include:

- Proposing *Green Mining*: mining software repositories research oriented towards power consumption;
- Describing a methodology for generating datasets to be used in *Green Mining*;
- A case study of power consumption and power regressions within 509 versions of Firefox 3.6 that confirms that power consumption does change over the lifetime of a product;
- A preliminary model of power consumption confirming that power consumption is more than just CPU use.

II. Previous Work

Measurement: Power measurement and modelling is industrially relevant and a concern of multiple vendors [8]–[11]. Consumer interest in power management is demonstrated by the Linux-oriented Phoronix site that has benchmarked power usage of Linux distributions [13]. GreenIT [3] is a primarily industrial movement for more power-aware computation.

In terms of benchmarking and monitoring that is relevant to this work Gurumurthi et al. [14] and Amsel et al. [5] have produced tools that simulate a real systems power usage, and benchmark individual applications' power usage. Gupta et al. [10] describe a method of measuring and correlating the power consumption of applications on Windows Phone 7. Tiwari et al. [4] modeled the power consumption of individual CPU instructions.

We argue that not all power consumption is CPU-bound. Authors investigating the power consumption of peripherals include: Lattanzi et al. [15] have modeled the power usage of WiFi adapters, while Greenwalt [1] measured and modeled the power consumption of hard-drives.

Optimization: Measuring power consumption is not enough, acting on it is necessary. Fei et al. [6] and Selby [16] have tuned source code transformations and compiler optimizations to reduce power-use. Power usage research has not leveraged the big-data corpus-based approaches used in MSR research.

Mining Software Repositories: MSR [7] provides us with tools to mine the software changes that we need. Shang et al. have investigated performance over versions of software [17]. To date there has not been much work, other than Gupta's work [10], on combining MSR techniques with power performance.

Summary: These articles demonstrate the interest in power consumption caused by software in industry and academia. *Green mining* demonstrates novelty by combining MSR research and power consumption.

III. Methodology

The *Green Mining* vision is to replicate this methodology over a large corpus of available software, version per version, revision per revision, in order to produce a huge corpus of software change correlated with power consumption behaviour. This would allow us to estimate the power consumption of software changes even without compilation or testing.

To measure the power consumption effects of software change on one product we must develop a test that can be run across the range of versions of the product, then per each version we must run the tests multiple times, each time recording the power usage. Then once all tests for all versions are recorded we can analyze the results.

In this study we focused on Firefox 3.6 from 2009 to the end of 2010, this resulted in 509 different versions of 3.6 ranging from alphas, betas and stable releases. We also added 39 different versions prior to 3.6.

A. Developing The Test

We have to run the software so we must develop a test that observes the running software. The test should exercise the parts of the code that are important to the stakeholders involved.

Our Firefox tests were meant to simulate the browsing behaviour of a mobile-user. For a set of web-pages that we remotely hosted, we would have Firefox load the page and then scroll through the webpage. To scroll the webpages we used X11::GUITest, a GUI testing framework. This allowed us to record a realistic browsing sessions consisting of scrolling using the arrow keys, page-up and page-down keys as well as mouse movements. This test was supposed to simulate generic browsing and reading behaviour. We would load 4 different web-pages, 2 Wikipedia pages that we mirrored and 2 versions of the NYAN-Cat webpage which featured GIF animation and client side Javascript animation. We would run through 4 tests in about 6 minutes. During our testing we discovered that Firefox often checks if it is the current default browser, so we had to modify the GUI test harness to cancel this action otherwise no scrolling would happen.

B. Measuring Test Performance

While we could just measure the power consumption of each test we would be wasting a chance to build models that could estimate power consumption. So per each test we also record other measurements such as disk transactions per second, CPU utilization and memory utilization.

In order to measure power consumption, in this paper, we rely on the *Watts Up? Pro*, a hardware device that measures wall socket power use (watts, kWh, amps, power-factor, volts, etc.), and reports power measures per second. We also use SAR [1] to record system activity information (CPU, Disk,

[1]SAR and Sysstat http://pagesperso-orange.fr/sebastien.godard/

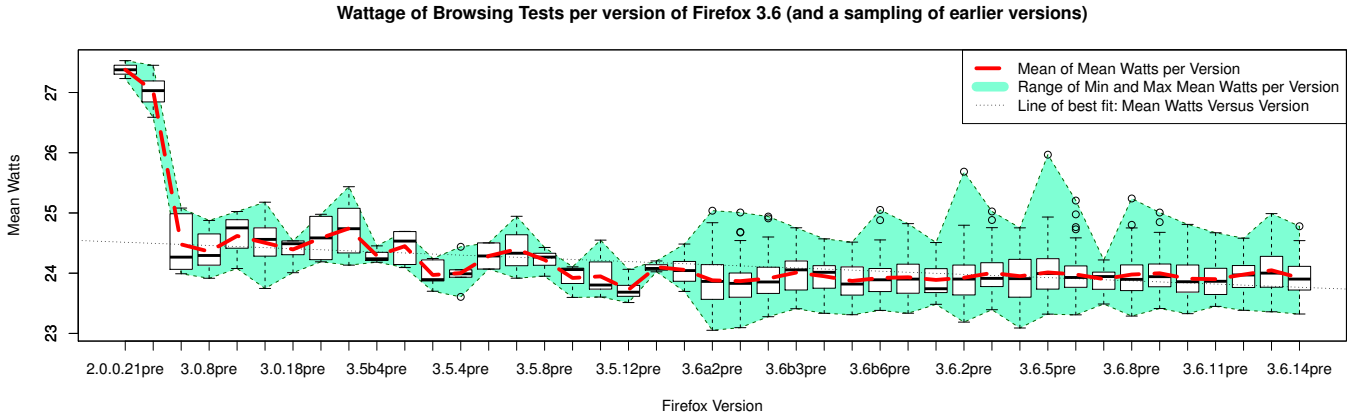

Figure 1. This is a graph of the distributions of mean wattage (power use) of different versions of Firefox. The green-blue area is the range between the minimum and maximum mean wattage for that version. The red line is the mean wattage and the box-plots depict the distribution of mean wattage per test per Firefox version. Note this plot depicts over 509 builds of Firefox 3.6 from alpha to stable versions. The dotted line with a negative slope is a line of best fit on the means; its slope indicates a decrease in power use across versions. The ranges for the earlier versions are smaller because we tested less instances of each of the earlier versions.

Memory, Network, etc.). We combine both of these measures and synchronize them with timestamps. The performance logger is started before a test and terminated after a test. To provide a baseline, idle measurements with the performance monitor need to be taken, much like a scale that is zeroed when a container is placed upon it.

C. Per Version Measurement

Each version of the program needs to be checked out, built and tested using the tests we previously developed. In this study we used Firefox nightly builds from Mozilla. Each binary we tested was the accumulation of revisions for that day on that branch (mozilla-1.9.2 for Firefox 3.6).

IV. Case Study

The motivation behind the test-case of the case study was to observe if changes to Firefox 3.6 induced power regressions when a user started the browser and browsed multiple web-pages. We feel this is a legitimate use case scenario for a mobile user who browses web-pages using a laptop on battery power, thus *our tests used the actual Firefox GUI.*

The hardware we used in the case study included an IBM Lenovo X31 Thinkpad laptop running Ubuntu 11.04, with its battery removed, plugged into a *Watts Up? Pro* device for power monitoring. We did not use the battery because we did not have a method of recharging the battery automatically.

We proceeded to download all 2009 and 2010 Firefox 3.6 nightly compilations from the Mozilla FTP server. These were compilations of Firefox 3.6 built every evening from the HEAD of the branch (Mozilla-1.9.2). We had 509 different binaries and we ran each binary against the mobile browsing use case scenario. As each test ran we recorded local resource statistics from the computer as well as the power usage measurements from the *Watts Up? Pro* device.

We ran each test approximately 3 times on different days at different times (this avoids cache issues). Once these runs were complete we analyzed the results.

Figure 1 shows the mean, minimum and maximum mean wattage of each test run. The test runs took all about the same time, so showing Watt-Hours is not necessary as the relative sizes are similar. What we can see in the graph is that different versions of Firefox 3.6 behave slightly differently but most stay close to a mean of 24 to 25 mean watts. This tells us that Firefox 3.6 is probably stable while prior versions prior exhibited higher power usage.

Since the idle usage is from 19 to 20 watts during logging we can see that Firefox's browsing activity in this case was costing an average of 4W to 5W. More if animation was being used. Power peaks occurred during our tests when Firefox was starting up as it needed to check the disk for a default profile and if no profile existed, make one. The animation seemed to have caused a higher overall average consumption for the NYAN Cat tests. The variance before 3.6 stable came out is higher, this is likely due to development effort dedicated to HTML5 features and improving Javascript performance. 3.6's power performance was relatively stable as most of the changes were for stability and for security which was not exercised by our tests.

A. Discussion and Future Work

Finally something does not correlate with LOC! In this study change affected power consumption but its effect is not clear. For instance lines of code (LOC) did not correlate with mean watts. Firefox 3.6b1 had 578 KLOC of C and C++ code, while Firefox 3.6.14 had 28% more code, 745 KLOC (180 KLOC added, 101 KLOC removed, 79 KLOC net increase). During Firefox 3.6, since the mean wattage distribution did not change much we can argue that LOC and change in LOC is not correlated with power consumption.

This invites more future investigation into what properties of changes are power related.

Is power consumption just the CPUs fault? We created a stateless regression model of watts based on these test runs. We found we could make models (relevant only to this particular test) with 3 independent variables, that were statistically significant and did not suffer from multi-collinearity. The three variables were % user-time per second (user space CPU usage), transactions per second (disk hits), and KB Active (memory that was recently active). This model of the tests predicted Watts with an R^2 of 0.38 (0.36 with only CPU) and a Spearman-rho correlation of 0.648. We achieved similar performance when faults per second could be swapped with KB active. Regardless, these 3 measures show that the disk, memory, and the CPU are all related and relevant to the power use of a software system. As future work we seek to generalize these models to support a wider range of tests, so that we can avoid instrumenting the hardware with power measurement devices.

Cost: our records show that the tests used approximately 2.5 Kilowatt Hours. As of writing, in Edmonton 1 kWh is worth about 14 cents, thus we used at least 35 cents worth of electricity in this study. Firefox cost us nearly 6 of those cents. We estimate, in this case, that Firefox usage compared to idle usage would reduce battery-life by a **third**.

Measurement Issues: we only measured each Firefox binary 3 times as we were limited by time. The *Watts Up? Pro* produces a reading every second with a granularity of $.1W$, which misses smaller changes. Measurement was sensitive to settings of the laptop, which were kept consistent.

V. Conclusions

In conclusion, we demonstrated with our case study of over 500 versions of Firefox 3.6, tested against a realistic web-browsing UI test, that power consumption can vary between versions, that power consumption was not correlated with size metrics such as LOC, and that power use was not just the CPU's fault: disk and memory use mattered too.

These kinds of results will be useful in the future to estimate power consumption, debug power consumption regressions, reduce power use on phones and computers, and reduce power consumption costs for businesses. In the future we want to evaluate power consumption **per each revision** to the source code, which we could not do in this study due to time constraints. We plan to leverage these kinds of measurements in order to increase developer awareness of power consumption and provide developers with an idea of the power consumption impact of their code. For a popular product like Firefox, if we could help prevent a power consumption regression of even just 1 Watt-hour, in terms of users we could save millions of Watt-hours.

Acknowledgments

I would like to thank Andrew Neitsch, Philippe Vachon, Andrew Wong, Eleni Stroulia, Ken Wong, Jim Hoover, Mario Nascimento, Denilson Barbosa, Daniel German, Michael Godfrey, and Taras Glek.

References

[1] P. Greenawalt, "Modeling power management for hard disks," in *MASCOTS '94., Proceedings of the Second International Workshop on*, Jan 1994, pp. 62 –66.

[2] Alliance to Save Energy, "PC Energy Report 2007: United States," http://www.climatesaverscomputing.org/docs/Energy_Report_US.pdf, 1E, Tech. Rep., 2007.

[3] S. Murugesan, "Harnessing Green IT: Principles and Practices," *IT Professional*, vol. 10, no. 1, pp. 24–33, 2008.

[4] V. Tiwari, S. Malik, A. Wolfe, and M. Tien-Chien Lee, "Instruction level power analysis and optimization of software," *The Journal of VLSI Signal Processing*, vol. 13, pp. 223–238, 1996.

[5] N. Amsel and B. Tomlinson, "Green tracker: a tool for estimating the energy consumption of software," in *Proceedings of the 28th of the International Conference of Extended Abstracts on Human factors in Computing Systems*, ser. CHI EA '10. New York, NY, USA: ACM, 2010, pp. 3337–3342.

[6] Y. Fei, S. Ravi, A. Raghunathan, and N. K. Jha, "Energy-optimizing source code transformations for operating system-driven embedded software," *ACM Trans. Embed. Comput. Syst.*, vol. 7, pp. 2:1–2:26, December 2007.

[7] H. Kagdi, M. L. Collard, and J. I. Maletic, "A survey and taxonomy of approaches for mining software repositories in the context of software evolution," *J. Softw. Maint. Evol.*, vol. 19, no. 2, pp. 77–131, 2007.

[8] C. Gray, "Performance Considerations for Windows Phone 7," http://create.msdn.com/downloads/?id=636, 2010.

[9] Apple Inc., "iOS Application Programming Guide: Tuning for Performance and Responsiveness," http://ur1.ca/696vh, 2010.

[10] A. Gupta, T. Zimmermann, C. Bird, N. Naggapan, T. Bhat, and S. Emran, "Energy Consumption in Windows Phone," Microsoft Research, Tech. Rep. MSR-TR-2011-106, 2011.

[11] Intel, "LessWatts.org - Saving Power on Intel systems with Linux," http://www.lesswatts.org, 2011.

[12] IBM, "IBM Active Energy Manager," http://www.ibm.com/systems/management/director/about/director52/extensions/actengmrg.html, 2011.

[13] M. Larabel, "Ubuntu's power consumption tested," http://www.phoronix.com/scan.php?page=article&item=878, Oct 2007.

[14] S. Gurumurthi, A. Sivasubramaniam, M. Irwin, N. Vijaykrishnan, and M. Kandemir, "Using complete machine simulation for software power estimation: the SoftWatt approach," in *Proc. of 8th Int. Symp. High-Performance Computer Architecture*, Feb 2002.

[15] E. Lattanzi, A. Acquaviva, and A. Bogliolo, "Run-Time Software Monitor of the Power Consumption of Wireless Network Interface Cards," in *Integrated Circuit and System Design. Power and Timing Modeling, Optimization and Simulation*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2004, vol. 3254, pp. 352–361.

[16] J. W. A. Selby, "Unconventional applications of compiler analysis," Ph.D. dissertation, University of Waterloo, 2011.

[17] W. Shang, Z. M. Jiang, B. Adams, A. E. Hassan, M. W. Godfrey, M. N. Nasser, and P. Flora, "An exploratory study of the evolution of communicated information about the execution of large software systems," in *WCRE*, 2011, pp. 335–344.