

What Do Programmers Know about Software Energy Consumption?

Candy Pang and Abram Hindle, University of Alberta

Bram Adams, Polytechnique Montréal

Ahmed E. Hassan, Queen's University

// A survey revealed that programmers had limited knowledge of energy efficiency, lacked knowledge of the best practices to reduce software energy consumption, and were unsure about how software consumes energy. These results highlight the need for training on energy consumption. //



WITH THE rising popularity of mobile computing and the advent of large-scale cloud deployments, the nonfunctional requirement of minimizing software energy consumption

has become a concern. For mobile devices, energy consumption affects battery life and limits device use. For datacenters, energy consumption limits the number of machines that

can be run and cooled. According to an IDC white paper, “Today, for every \$1.00 spent on new hardware, an additional \$0.50 is spent on power and cooling, more than double the amount of five years ago. Datacenters at their power and cooling thresholds are unable to support new server deployments, a fact that severely limits the expansion of IT resources.”¹

Unfortunately, the demand for energy-efficient computing isn’t reflected in the education, training, or knowledge of programmers. Programmer training often focuses on methodologies such as object-oriented programming and nonfunctional requirements such as performance. Performance optimization is often considered a substitute for energy optimization because a faster system likely consumes less energy. Although this is a step in the right direction, it’s insufficient and sometimes even incorrect. For instance, parallel processing might improve performance by reducing calculation time. However, saving and restoring execution context, scheduling threads, and losing locality might end up consuming more resources than sequential processing.²

A previous analysis based on energy-related questions on Stack-Overflow (<http://stackoverflow.com>) showed that programmers had many such questions but rarely got appropriate advice.³ To gain more tangible evidence of and concrete insight into this problem, we surveyed programmers to gauge their knowledge of software energy consumption and efficiency. In particular, we addressed four questions. Are programmers aware of software energy consumption? What do they know about reducing it? What’s their level of knowledge about it? What do they think causes spikes in it?

The Survey

An anonymous online survey comprised 13 questions in four phases (full survey details, data, and analysis can be found online⁴).

Phase 1 involved three questions regarding respondent demographics:

- How many years of programming experience do you have?
- How would you rank your programming skill—beginner, intermediate, or advanced?
- In what programming language are you most proficient?

Phase 2 involved eight quantitative questions. The first two evaluated the respondents' knowledge of software energy consumption (using the common term "power consumption"⁵):

- For desktop computers, rank the software power consumption of the CPU, hard drive, memory, network, and screen and GPU.
- For mobile devices, rank the software power consumption of the CPU, data storage device, memory, network, and screen and GPU.

The next six yes/no questions gathered information about the respondents' experience with software energy consumption:

- Do you take power consumption into account when developing software?
- Is minimizing power consumption a requirement or a concern of your software?
- Have users complained about your software's power consumption?
- Have you modified your software to reduce power consumption?

- Have you measured your software's power consumption? If yes, how do you measure it? (If the respondent answered yes, the questionnaire provided additional space for a text response.)
- Would power consumption be one of your decision factors when choosing a mobile development platform?

Phase 3 involved two qualitative questions, allowing respondents to further express their knowledge and experience regarding software's power consumption:

- What software functions have higher power consumption?
- How would you improve your software's power efficiency?

Phase 4 involved optional qualitative follow-up interviews.

We posted survey invitations in numerous programming-related Reddit (www.reddit.com) subgroups (subreddits) between 20 August and 4 September 2013. We received 122 responses and conducted four follow-up interviews.

The Results

The survey respondents identified themselves as programmers. Of the 122 respondents, 37 (30 percent) used C or C++ and 84 (69 percent) used C#, Java, JavaScript, Perl, PHP, Python, or Ruby. According to the November 2014 TIOBE Index (www.tiobe.com/index.php/tiobe_index), these languages accounted for 54 percent of existing programs and likely represented more than half the software running in datacenters. In addition, Java is the primary language for Android and BlackBerry, whereas C# is the primary language for Windows Phones.

Programmers Have Limited Awareness of Software Energy Consumption

Our survey results show that the programmers rarely addressed energy efficiency and that users rarely requested it. Only 22 respondents (18 percent) claimed to take energy consumption into account when developing software. Only 17 respondents (14 percent) considered minimizing energy consumption a requirement. Twenty-six respondents (21 percent) said they modified software to reduce energy consumption.

One interviewee indicated that clients "care first and foremost about speed of development, and secondly about reasonable quality and performance." This suggests that the lack of attention to software energy consumption is an issue of priorities.

These results show that these programmers either were unaware of energy efficiency or weren't asked to address it. An interviewee mentioned that "1 watt would be a lot of power for a mobile phone, [but] it's absolutely negligible in comparison to other household appliances." That 1 watt might be negligible on the personal level, but on the global level, energy consumed by all mobile devices and datacenters multiplies. In 2006, 6,000 US datacenters reportedly consumed 61 billion kilowatt-hours of energy costing US\$4.5 billion.⁶

Similarly, software users and clients were unaware of software energy consumption. Only 4 respondents (3 percent) reported that their users complained about their software's energy consumption.

Our results confirmed Hammad Khalid and his colleagues' finding that mobile-application users have low awareness of resource use.⁷ Their results showed that resource-related complaints (application re-

views), including energy consumption complaints, ranked last out of 12 types of user complaints in terms of frequency. Although users don't complain frequently about resource consumption, Khalid and his colleagues' results also show that resource-related complaints negatively affect users. So, despite the low frequency of such complaints, they're highly troubling. If customers and clients aren't asking for energy-efficient software, programmers are less likely to address the energy efficiency of software. Hence, appropriate public education and expanded awareness among clients and programmers about software energy consumption is needed.

Programmers Lack Knowledge of Reducing Software Energy Consumption

To reduce software energy consumption, programmers must start by measuring the energy consumption of their software. Only 12 respondents (10 percent) said they did this. Fifteen respondents (12 percent) indicated that you can measure software energy consumption through a power meter, the battery, the power supply, resource measurement, software tools, and CPU time.

These results show that these programmers lacked knowledge of how to accurately measure software energy consumption. Most of the suggested methods measure the overall hardware energy consumption, not the fine-grained energy consumption of the software. In addition, mobile-device batteries don't accurately report the actual energy use.⁵ Ding Li and his colleagues also found that programmers used "typical practices in energy measurement studies ..., [which] have limitations that could introduce inaccuracy."⁸

Measuring software energy consumption is a challenge. One interviewee stated that "one has to have a proper understanding of the entire system [to] make an informed [energy consumption] analysis." Pro-

spectively, were aware that less computation and reduced polling can reduce energy consumption. The results show that the respondents' ideas about how to best reduce software energy consumption var-

The demand for energy-efficient computing isn't reflected in the education, training, or knowledge of programmers.

grammers have to understand the interactions between high-level and low-level components to really analyze the root causes of software energy consumption. The survey and interviews showed that most of the respondents had difficulty measuring and optimizing software energy consumption even when the appropriate tools were available.

Another interviewee admitted, "It's more often the hardware rather than the software that we are interested in when we talk about energy consumption." Although few respondents measured the energy consumption of their software, 79 (65 percent) of them considered energy consumption as a factor when choosing a mobile development platform. Many respondents relied on choosing the right platform and hardware to ensure the energy efficiency of their software. However, they rarely addressed software energy consumption.

Figure 1 summarizes how the respondents would improve energy consumption. Nineteen respondents (16 percent) were aware that better algorithms lead to better energy efficiency, which was the most popular suggestion. Only 11 (9 percent) and 8 (7 percent) of the respondents, re-

spectively, were aware that less computation and reduced polling can reduce energy consumption. The results show that the respondents' ideas about how to best reduce software energy consumption var-

Programmers Lack Knowledge of Software Energy Consumption

Figure 2 summarizes the respondents' rankings of the software energy consumption for desktop computer and mobile-device components.

For desktop computer components, we expected these rankings (from highest to lowest consumption):

1. CPU,
2. hard drive,
3. screen and GPU,
4. network, and
5. memory.

For mobile-device components, we expected these rankings:

1. screen and GPU,
2. CPU,
3. network,
4. hard drive, and
5. memory.

We based these rankings on current conventional wisdom, backed up by experiments we had performed

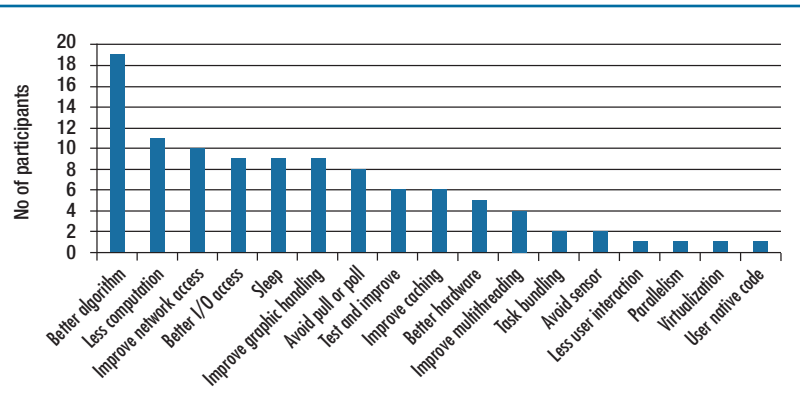


FIGURE 1. Respondents' responses regarding ways to improve software energy efficiency. The respondents' answers varied widely, indicating the need for increased education on software energy consumption and efficiency.

over the past three years and recent studies.^{5,8,9} Because these rankings weren't written in stone, they could differ among specific hardware. We were focusing on the consistency of rankings across all respondents, independent of our expected ranking. For desktop computers, only 1 respondent (1 percent) ranked the components in our expected order. For mobile devices, 12 respondents (10 percent) ranked the components in our expected order.

Using Spearman's rank correlation, we compared the respondents' rankings with the expected ranking. Generally, positive correlations closer to 1 indicate stronger agreement. If two rankings completely match, their correlation is 1. If they're the inverse of each other, the correlation is -1. If they're unrelated, a correlation near 0 is possible.

For desktop computers, the average correlation between the respondents' rankings and the expected ranking was 0.48, indicating a medium level of agreement. For mobile devices, the average correlation was 0.75, indicating a much stronger agreement. The correlation's standard

deviation was 0.25 for desktop computers and 0.20 for mobile devices.

We also used Spearman's rank correlation to compare the respondents' rankings against each other (interagreement), regardless of the expected ranking. The correlation was 0.3 for desktop computers and 0.6 for mobile devices. So, respondents had less internal agreement on the energy consumption of desktop computer components than on the consumption of mobile-device components. The correlation's standard deviation was 0.48 for desktop computers and 0.32 for mobile devices. This implies that respondents agreed less about the energy consumption of desktop hardware components and more about the energy consumption of mobile-device components.

In other words, considerable disagreement existed on whether a particular component consumed more energy than another. One explanation might be that different types of programmers make different assumptions about the energy consumption of hardware components. For example, game programmers interact

mostly with the screen and GPU, so they're more likely to identify the screen and GPU as the most energy-consuming components. Programmers might blame the most obvious component without understanding how software consumes energy.

Furthermore, programmers might focus overly on their users' on-screen experience—that is, on what's observable. The respondents overwhelmingly ranked the screen and GPU as the highest-energy-consuming components: 82 respondents (67 percent) for desktop computers and 95 respondents (78 percent) for mobile devices. It is true, though, that the screen and GPU often consume the most energy on mobile devices.

The overall results show that programmers lack consistent knowledge regarding the energy consumption relationship between software and hardware. Nonetheless, programmers have more consistent knowledge about software energy consumption on mobile devices than on desktop computers. So, it might be more effective to develop education and awareness programs and guidelines for specific domains (for example, mobile devices and gaming).

Programmers Are Unaware of Software Energy Consumption's Causes

Gustavo Pinto and his colleagues mined StackOverflow data to identify seven causes of unnecessary software energy consumption:³

- unnecessary resource use,
- faulty GPS behavior,
- background activities,
- excessive synchronization,
- background wallpapers,
- advertisements, and
- high GPU use.

Our results closely matched the results of Pinto and his colleagues. Taken as a group, the respondents identified most of Pinto's seven causes. For example, in Figure 3, "Network" refers to unnecessary resource use, "Sensor" refers to faulty GPS behavior, "Threading" refers to background activities, "Pulling or polling" refers to excessive synchronization, and "Graphics" refers to background wallpapers and high GPU use. The respondents didn't identify advertisements.

However, individually, only 19 respondents (16 percent) identified network data access as a cause of high energy consumption. The low identification rate matches the observation of Li and his colleagues.⁸ Six respondents (5 percent) identified pulling or polling for excessive synchronization. Only 4 (3 percent) identified sensor use, which can leave the GPS turned on for too long. Thirty-five (29 percent) identified graphics as a cause of high energy consumption for functions such as background wallpapers and animations. In short, the numbers show that only a few of the respondents could identify the causes of high energy consumption.

Limitations

One limitation was our use of social media to recruit survey respondents, instead of directed emails or phone calls. Nevertheless, through social media, we were able to reach programmers in the field whom we otherwise couldn't have reached. More than half a million users have subscribed to programming-related subreddits, which often have more than 1,000 concurrent users. To avoid having our survey invitation tagged as spam and to avoid negative reactions, we posted it to one relevant sub-

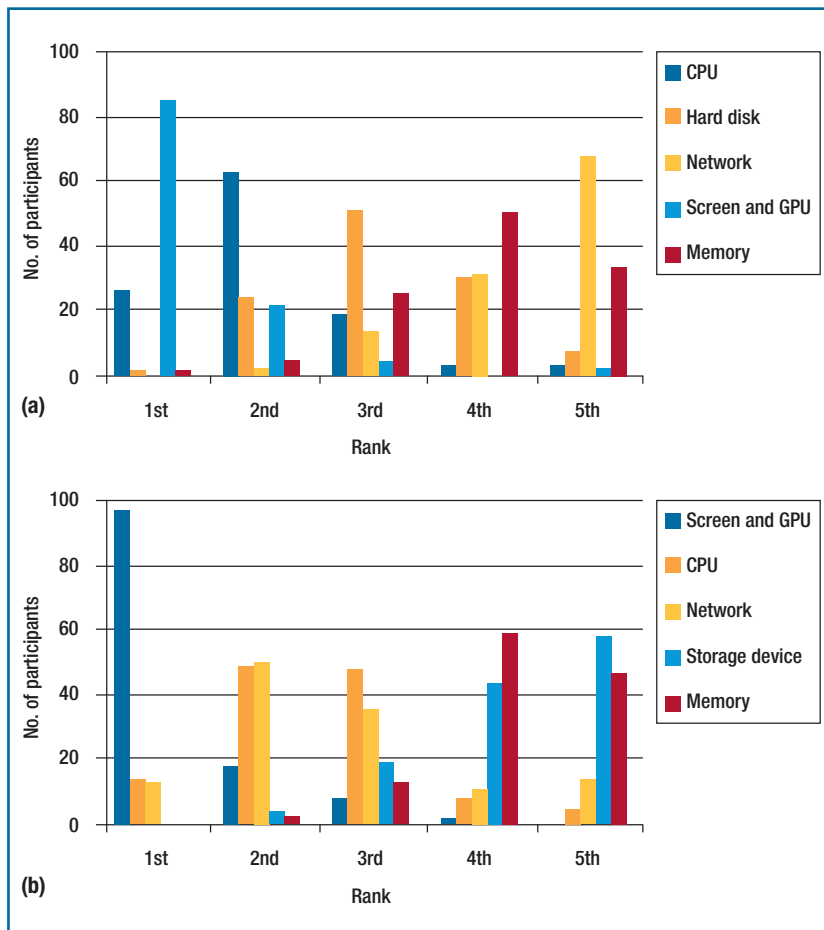


FIGURE 2. Respondents' rankings of energy consumption for the components of (a) desktop computers and (b) mobile devices. Considerable disagreement existed on whether a particular component consumed more energy than another.

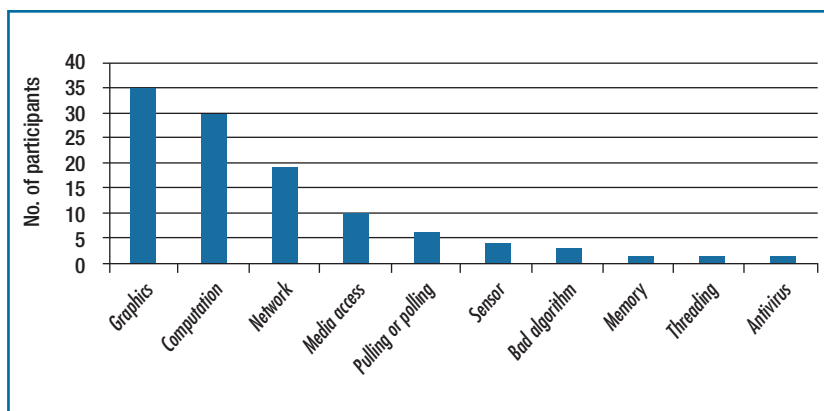


FIGURE 3. Respondents' responses regarding causes associated with high energy consumption. Only a few of the respondents could identify those causes.



CANDY PANG is a PhD student in computing science at the University of Alberta. Her research interests are enterprise application design, implementation, maintenance, and education. She was an application architect in multiple Government of Alberta ministries for more than 13 years. Pang received her BSc and MSc in computing science from the University of Alberta. She's a member of ACM and is an Information Systems Professional and Information Technology Certified Professional. Contact her at cspang@ualberta.ca.



ABRAM HINDLE is an assistant professor of computing science at the University of Alberta. His research focuses on problems with mining software repositories, improving software-engineering-oriented information retrieval with contextual information, and software maintenance's impact on software energy consumption. Hindle received a PhD in computer science from the University of Waterloo. Contact him at abram.hindle@ualberta.ca; <http://softwareprocess.ca>.



BRAM ADAMS is an assistant professor at Polytechnique Montréal, where he heads the Maintenance, Construction, and Intelligence of Software lab. His research interests include software release engineering, software integration, software build systems, software modularity, and software maintenance. Adams received a PhD in computer science engineering from Ghent University. He is one of the organizers of the International Workshop on Release Engineering and is a member of IEEE. Contact him at bram.adams@polymtl.ca.



AHMED E. HASSAN is the Canada Research Chair in Software Analytics and the Natural Sciences and Engineering Research Council of Canada / BlackBerry Software Engineering Chair at the School of Computing at Queen's University. His research interests include mining software repositories, empirical software engineering, load testing, and log mining. Hassan received a PhD in computer science from the University of Waterloo. He spearheaded the creation of the International Conference on Mining Software Repositories and its research community. Hassan also serves on the editorial boards of *IEEE Transactions on Software Engineering*, *Empirical Software Engineering*, *Computing*, and *PeerJ Computer Science*. Contact him at ahmed@cs.queensu.ca.

reddit at a time. This process significantly lengthened the data-gathering period and limited the number of respondents.

Another limitation is that we didn't control for the development process that the respondents used. Many development processes exist. An enterprise might employ the full COBIT (Control Objectives for Information and Related Technology; www.isaca.org/cobit) development cycle. A mid-size shop might use agile processes. A one-man startup might do whatever is necessary. In a formal process, programmers might not have the opportunity to specify the functional or nonfunctional requirements. But surveying a statistically significant number of programmers for each process type would have been difficult. So, we surveyed a board range of programmers. Future studies need to investigate the energy consumption knowledge of IT workers in different roles.

A third limitation is that we focused on programmers' software energy consumption knowledge. Software has many nonfunctional requirements (for example, memory use, performance, security, and usability); energy consumption is just one of them, albeit the least-studied one. However, although nonfunctional requirements might affect each other, mixing other criteria into our study might have blurred the results.


The programmers in our study lacked knowledge and awareness of software energy-related issues. More than 80 percent of them didn't take energy consumption into account when developing software. Nevertheless, most of them considered software energy consumption to be important

when choosing a mobile development platform.

The fact that only 3 percent of the respondents received complaints about software energy consumption might suggest that users are unaware of it. As Chenlei Zhang and his colleagues argued, the creation of benchmarks and reporting mechanisms (similar to Energy Star) that inform users of software energy efficiency can significantly increase user awareness.¹⁰ Increased user awareness will, in turn, motivate programmers to measurably enhance their software's energy efficiency. As one Reddit respondent commented, the "survey has at least made me consider ... possible costs of doing things."

Pinto and his colleagues identified eight strategies to reduce energy consumption through software modification:³

- minimizing IO,
- bulk operations,
- avoiding polling,
- hardware coordination,
- concurrent programming,
- lazy initialization,
- race to idle, and
- efficient data structure.

These strategies should be part of programmers' education. In addition, development tools can be created to identify unnecessary energy consumption and suggest how to reduce it. Educators could develop slides, videos, projects, and assignments as part of an undergraduate curriculum for energy efficiency and sustainability. 

References

1. J. Scaramella and M. Eastwood, *Solutions for the Datacenter's Thermal Challenges*, white paper, IDC, Jan. 2007; http://ecoinfo.cnrs.fr/IMG/pdf/idc_WHITE_PAPER_-_Solutions_for_the_Datacenter_s_Thermal_Challenges_.pdf.
2. B. Götz et al., *Java Concurrency in Practice*, Addison-Wesley, 2006.
3. G. Pinto, F. Castor, and Y.D. Liu, "Mining Questions about Software Energy Consumption," *Proc. 11th Working Conf. Mining Software Repositories*, 2014, pp. 22–31.
4. C. Pang, "Technical Summary: What Do Developers Know about Software Energy Consumption and Power Use?," 2013; <http://webdocs.cs.ualberta.ca/~hindle1/2014/green-programmers>.
5. A. Hindle, "Green Mining: A Methodology of Relating Software Change to Power Consumption," *Proc. 9th Working Conf. Mining Software Repositories (MSR 12)*, 2012, pp. 78–87.
6. P. Kurp, "Green Computing," *Comm. ACM*, vol. 51, no. 10, 2008, pp. 11–13.
7. H. Khalid et al., "What Do Mobile App Users Complain About?," *IEEE Software*, vol. 32, no. 3, 2015, pp. 70–77.
8. D. Li et al., "An Empirical Study of the Energy Consumption of Android Applications," *Proc. IEEE Int'l Conf. Software Maintenance and Evolution*, 2014, pp. 121–130.
9. A. Banerjee et al., "Detecting Energy Bugs and Hotspots in Mobile Apps," *Proc. 22nd ACM SIGSOFT Int'l Symp. Foundations of Software Eng.*, 2014, pp. 588–598.
10. C. Zhang, A. Hindle, and D.M. Germán, "The Impact of User Choice on Energy Consumption," *IEEE Software*, vol. 31, no. 3, 2014, pp. 69–75.



Selected CS articles and columns are also available for free at <http://ComputingNow.computer.org>.



Want more know more about the Internet?

This magazine covers all aspects of Internet computing, from programming and standards to security and networking.

www.computer.org/internet