The Perils of Energy Mining: Measure a Bunch, Compare just Once

A Tale of TWO HTTPs

Shaiful Chowdhury [Chowdhury2015] came into the lab and proclaimed, "HTTP/2.0 has significantly better energy performance than HTTP/1.1!" "That's excellent," I exclaimed, "Do you know why?"

The next week Shaiful came back to the lab and said, "I think I was wrong, I don't think there's a difference." He went on to explain that he was happy with the results of the first experiment but something was bothering him.

To test HTTP/1.1 in Firefox he had to disable HTTP/2.0 in Firefox's settings, thus his energy consumption test of Firefox for HTTP/1.1 had to do more work than his HTTP/2.0 tests. He investigated the CPU state of the HTTP/2.0 and HTTP/1.1 tests and found that the rigorous button pushing used to disable HTTP/2.0 put the CPU into a higher state: high frequency, and more voltage. Thus the CPU state for the HTTP/1.1 started higher and was consuming more than the HTTP/2.0 tests.

Shaiful's solution was to inject some idle time into the HTTP/1.1 test, allowing the CPU to lower its state. He then ensured via manual inspection that the idling did drop down the CPU state. Afterwards both tests produced results that were comparable; it turns out that on a fast network there's not much difference between HTTP/1.1 and HTTP/2.0.

The entire experiment was threatened by attributing the change in energy consumption to HTTP/1.1 and HTTP/2.0 code in Firefox, when in fact it was caused by our HTTP/1.1 test inducing the CPU to have a different state than the HTTP/2.0 tests. Misattribution of the root causes of energy consumption is just one of many perils one faces when engaged in Green Mining, energy-aware mining, and software energy consumption measurement.

Let's ENERGISE your Software Energy Experiments

We want to help prevent experimental accidents when measuring the energy consumption of software systems. The most difficult aspect of measuring and mining software energy consumption is to juggle all of the confounds and potential threats to validity one faces. The first 2 questions any software energy miner should ask themselves are "What do I want to measure?" and "What am I actually measuring?". The intent is to create a test or benchmark that will allow us to compare energy consumption of the task or program in question.

In prior work [Hindle2012] we propose a methodology of repeated measurement and comparison of such a test:

1. Choose a product and task to test;
2. Decide on granularity and level of instrumentation;
3. Choose which versions of the software to test;
4. Develop a test case for the task to be tested;
5. Configure the testbed environment to reduce background noise;
6. For each combination of version, task and configuration repeat multiple times:
    1. Setup the testbed to record energy consumption measurements;
    2. Run the test;
    3. Compile and store the recorded data;
    4. Cleanup the testbed.
7. Analyze and evaluate.

This methodology can be combined with a simple mnemonic, **ENERGISE** 1. ENERGISE is a checklist of issues, built from prior experience[Hindle2014], that one should consider when measuring software energy consumption:

- **Environment** -- prepare a stable testbed for energy measurement.
- **N-versions** -- run a test across more than 1 version of the software.
- **Energy or power** -- do we care about total energy consumed of a task or the per second cost of running a service?
- **Repeat** -- 1 run is not enough, we need to run our tests multiple times to address background noise.
- **Granularity** -- what level of measurement, how often, and what level of invasiveness of instrumentation?
- **Idle** -- how do applications and service react with no load: is energy being wasted?
- **Statistics** -- repeat measures call for summary statistics and comparing distributions.
- **Exceptions** -- errors happen, how do we address with them or notice them?

Environment

The environment is the testbed and the system that will run the software under test. Environments should be representative of realistic platforms and scenarios, yet balanced against noise such as third-party apps or traffic, unneeded applications, or other users. Generally environments should be as controlled as possible. Even temperature can affect energy measurements.

N-Versions

The first step to any successful attempt at optimization is to measure system performance before optimization. If we are measuring software energy consumption to determine the impact of changing the code we should measure the system before the modification to allow for comparison. Our work in Green Mining has shown that software does indeed change in energy performance over time and measuring just 1 version of the software might not be representative of the versions before or after. It is recommended that multiple versions of the

software are measured.

Energy or Power

Energy is the cost of work or the capacity to do work, energy is typically measured in joules (J). Power is the rate of energy consumption, essentially the derivative of energy consumption, measured in watts (W) where 1 watt is equal to 1 joule second.

When you measure a task that a system executes, ask your system, does this task have a clear beginning or end? Does this task continuously run? Tasks that do not to run continuously, such as sharpening an image or compressing a video file can be characterized by the energy consumed. Where as a task that runs continuously, such a sharpening video images of a surveillance web-camera or streaming video compression, is better characterized by its workload, its power, the rate of energy consumption.

Repeat!

While we already recommended measuring multiple versions of software, something even more important is to repeat your measurements. Modern computers are very noisy and active systems with lots of background processes and lots of state. They have many peripherals and services. It is hard to guarantee what tasks are executing on a modern system and what the current environment is like. If you're running a test using WIFI your own cellphone could affect the experiment. Furthermore we're measuring physical phenomena: energy consumption. Our measurement equipment, our testbeds, or energy measurement devices all have error in them, error is inherent in physical measurement thus we need to take multiple measurements so we can rely upon statistics to give us a more clear picture of what we measured.

Granularity

When designing a energy consumption test one has to choose the level of granularity. Do you want to measure method calls, system calls, CPU use, memory use, processes, etc. and at what frequency? Many measurement devices and ICs, such as the Watts Up? Pro device or the TI INA-219 IC, are sampling at thousands of times per second, integrating the results and reporting back to you at a fraction of that rate. If you only have 1 second of granularity from a Watts Up? Pro you won't be measuring the cost of a method call unless you explicitly make a benchmark that repeatedly calls it. If you need method call level measurements, the instrumentation overhead will be high.

Granularity is a concern if you measure just a single process or component or if you measure the whole system. An example scenario is if you asked the sound-card to play a sound for 1 second. The request to play a sound might return in 1ms to the process, for the next second the OS, the sound card driver and sound card will be interacting, playing the requested 1 second of audio. If we measure at the process level we miss the induced cost on the system of the software, the software commanded work but we are not measuring that work. If we measure at the entire system level we will be including a lot of other processes and drivers that will be irrelevant.

Idle Measurement

Not all applications or tasks have idle behaviour, but many user facing applications and services run continuously or intermittently in the foreground or background. What programs do when they aren't in direct use is important because usually their idle operation is a often waste of resources, unless work is being executed during idle time. An efficient process can stay asleep until it receives input. The operating system's scheduler is very good at sleeping and waking processes.

Idle use often characterizes baseline energy consumption for an application is. If executing tasks and idleness use the same amount of energy perhaps the idle components can be optimized to use less energy. Measuring idle consumption is useful to determine what impact changes might have on the implementation of non-functional requirements.

Statistical Analysis

Repeat measurements cause many problems: there is no longer 1 measurement, there are many. Thus summary statistics can be employed to describe the distribution of measurements. There is one saving grace, energy measurements are of physical phenomena and the Normal (or Gaussian) distribution often model the errors within natural measurements well. Thus the variance and mean are two reasonable descriptors of our multiple runs. This means we can use parametric tests such as the Student's T-test to compare two distributions of measurements to see if they are statistically significantly different. Without statistical tools such as the T-test we would not have much confidence to determine if distribution's were different or not based on random chance.

Exceptions

To err is human and to throw uncaught exceptions is to execute code. Mobile devices and modern computers still suffer from crashing software. Exceptions happen, core dumps occur, sometimes apps decide to update, sometimes the network goes down, sometimes a remote site is unavailable. Often the software under test is just inherently buggy and only half of the test runs will complete. When developing tests one should instrument the tests with auditing capabilities such as screenshots to enable postmortem investigations. Furthermore outliers should be investigated and potentially re-run. You wouldn't want to attribute a difference in energy consumption to an erroneous test.

Summary

In summary, there are many confounds that one faces when measuring software energy consumption. First and foremost, energy consumption is a physical process and energy consumption measurement requires repeated measurement and statistical analysis. Thus remember and use the ENERGISE mnemonic to help evaluate energy measurement scenarios: environment, N-versions, energy or power, repeated measurement, granularity, idle measurement, statistical analysis, and exceptions.

Footnotes

British/Canadian spelling.

References

[Chowdhury2015] Shaiful Alam Chowdhury, Varun Sapra , and Abram Hindle. "Is HTTP/2 More Energy Efficient Than HTTP/1.1 for Mobile Users?" , PeerJ Preprints, https://peerj.com/preprints/1280/ (https://peerj.com/preprints/1280/) , 2015.

[Hindle2012] Hindle, Abram. "Green mining: A methodology of relating software change to power consumption." In Proceedings of the 9th IEEE Working Conference on Mining Software Repositories, pp. 78-87. IEEE Press, 2012.

[Hindle2014] Hindle, Abram, Alex Wilson, Kent Rasmussen, E. Jed Barlow, Joshua Charles Campbell, and Stephen Romansky. "Greenminer: A hardware based mining software repositories software energy consumption framework." In Proceedings of the 11th Working Conference on Mining Software Repositories, pp. 12-21. ACM, 2014.