Exploring the Energy Consumption of Highly Parallel Software on Windows

Mads Hjuler Kusk*, Jeppe Jon Holt*
and Jamie Baldwin Pedersen*
Department of Computer Science, Aalborg University, Denmark
*{mkusk18, jholt18, jjbp18}@student.aau.dk

February 28, 2023

Abstract

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

1 Introduction

Research questions:

- What is the best measuring instrument for Windows?
- *How well does microbenchmarks represent a realistic usecase compared to macrobenchmarks?*
- How does parallelism affect the energy consumption
- How does P-cores and E-cores affect the execution of parallelism in a process, versus only P-Cores?
- *How can measuring instruments be calibrated to better fit a ground truth*

Hypothsis:

- The expectation when using different compilers is that the energy consumption of the code will be similar, but with some deviations. Resulting from the individual compilers implementation.
- The expectation for the temperature is that it in itself would not effect the performance unless the cpu starts to thermal throttle. Though the heat of the cpu is directly related to the electrical resistance meaning that it would be less efficient in terms of joule per computation.
- We do not expect the background process to have a large impact on the energy consumption, this mostly because the non-essential background process are executed rarely and for very brief durations. We do expect that the results will become more consistent without
- We know that IPG and LHW are very similar and we expect Windows RAPL driver to be similar as well.
- We expect that using parallelism there is not a correlation between execution time and energy consumption.
- We expect if we can get the right thread to run on a E core that it could improve energy consumption slightly depending on the size of the benchmark. Where the larger the benchmark the bigger the improvement in energy consumption.
- We expect the calibration to be very situation specific, where the calibration might improve the measurements in some cases, but not in others.

2 Related Work

2.1 Previous Work

This paper builds upon the knowledge gathered in our previous work *"A Comparison Study of Measuring Instruments"*[1] where different measuring instruments were compared to explore whether a viable software-based measuring instrument was available for Windows. It was found that Intel Power Gadget (IPG) and Libre Hardware Monitor (LHM) on Windows have similar correlation to hardware-based measuring instruments as Intel's Running Average Power

Limit (RAPL) has on Linux. This chapter builds upon the related work chapter of the previous work and as such will not be repeated, however, it will be expanded upon.

2.2 Parallel Software

In [2], the energy consumption for sequential and parallel genetic algorithms are explored, where one research question aims to explore the impact on the energy consumption when using a variable number of cores. In this study they find that a larger number of cores in the execution pool results in a lower running time an energy consumption, and concludes that parallelism can help reduce the energy consumption. Parallelisms ability to reduce the energy consumption is argued to be due to the large number of cores working to solve the problem simultaneously, where the combination of more cores, more parallel operations per time unit will require less energy.

When considering parallel software, [2] also find asynchronous implementations to use less energy. The reason for this is because there are no idle cores waiting for data in asynchronous implementations, while in synchronous implementations cores can be blocked during runtime, while waiting for responses from other cores.

In [3], the behavior of parallel applications and the relationship between execution time and energy consumption are explored. They test four different language constructs which can be used to implement parallelism in C#. Furthermore, they use varying amounts of threads and a sample of micro- and macro-benchmarks. [3]

They found that workload size has a large influence on running time and energy efficiency and that a threshold limit for the workload must be reached for there to be improvements when changing a sequential program into a parallel one. Additionally, it was found that execution time and energy consumption of parallel benchmarks do not always correlate. Comparing micro- and macro-benchmarks the findings remain consistent, although the impact becomes low for the macrobenchmarks due to their being an overall larger energy consumption. Furthermore, they have included some recommendations, which should be considered:[3]

- Shield cores: Avoid unintended threads running on the cores used in the benchmarking
- PowerUp: Can be used to ensure that benchmark is not optimized away during compilation
- Static clock: Make the clock rate of the CPU as static as possible
- Interrupt request: Avoid interrupt requests being sent to cores used in the benchmarking
- Turn off CPU turbo boost
- Turn off hyperthreading

2.3 Compilers

In [4] the language C++ and different compilers are explored and compared to find the impact of using different coding styles and compilers, where the goal is to find a balance between performance and energy efficiency. The different coding styles introduced explore the impact of splitting CPU and IO operations and interrupting the CPU-intensive instructions with sleep statements. The C++ compilers used in [4] include MinGW GCC, Cygwin GCC, Borland C++, and Visual C++, and the energy measurements are performed using Windows Performance Analyses (WPA). All compilers are used with default settings, and no optimizations were chosen. This decision was made based on works like [5], where it was found how mainstream compilers will apply multiple optimizations to the final code, where these optimizations in the worst case will result in worse performance and increased energy consumption. The issue of optimizations being very machine dependent was also shown in [6], where analysis and optimizations were done on a Texas Instruments C6200 DSP CPU, they found that a large portion of the energy is used by fetching instruction. They address this by introducing a fetch packet mechanism, and also find loop-unrolling to reduce energy consumption. While these optimizations decrease the energy consumption for the Texas Instruments C6200 DSP CPU, they note that for other CPUs varying results are expected. A similar conclusion is also found in [4], where they find that when choosing a compiler and coding style the energy reduction depends on the nature of the target machine and application. Based on the test case used, this being an election sort algorithm, they find the best performance with the Borland compiler, and the lowest energy with the Visual C++ compiler. When considering the coding styles, they find that both separating IO and CPU operations and interrupting the CPU-intensive instructions with sleep statements also decrease the energy consumption.

3 Method

3.1 Measuring Instruments

This section present the different measuring instruments utilized in this work. The measuring instruments utilized in the previous work will only be briefly introduced, however more detail can be found in our previous work[1]. In this paper, four software-based measuring instruments and one hardware-based measuring instrument is used, where the hardware-based measuring instrument represents the ground truth.

Running Average Power Limit Intel's RAPL is a commonly used software-based measuring instrument seen in the literature.[1] It uses model-specific-registers (MSRs) and Hardware performance counters to calculate how much energy the processor uses. The MSRs RAPL uses include *MSR_PKG_ENERGY_STATUS*, *MSR_DRAM_ENERGY_STATUS*, *MSR_PP0_ENERGY_STATUS* and *MSR_PP1_ENERGY_STATUS*. Which corresponds to the power domains, PKG, DRAM, PP0, and PP1 which are explained in our previous work[1]. RAPL has previously only been directly accessible on Linux and Mac. In our previous work we found that RAPL had a high correlation of 0.81 with our ground truth on Linux.[1]

Intel Power Gadget IPG is a software tool created by Intel, which can estimate the power of Intel processors. It contains a command line version called Powerlog which allows accessing the energy consumption using callable APIs. It uses the same hardware counters and MSRs as RAPL[7], therefore it is expected to observe similar measurements to that of RAPL. Which is also shown in our previous work where we found that IPG had a high correlation of 0.78 with our ground truth on Windows. We also found that IPG had a high correlation of 0.83 with RAPL, although the measurements is on different operating systems.[1]

Libre Hardware Monitor LHM is a fork of Open Hardware Monitor, where the difference is that LHM does not have a UI. Both projects are open source. LHM can use the same hardware counters and MSRs as RAPL and IPG and as such can measure the power domains PKG, DRAM, PP0, and PP1. Since it uses the methods to read energy consumption, a similar measurement is expected between LHM and IPG. We found that LHM correlated 0.76 with our ground truth on Windows. LHM was also found to have a high correlation of 0.85 with IPG.[1]

AC Current Clamp Severing as our ground truth measurement is our hardware-based measuring setup which is comprised of an MN60 AC clamp that is connected to the phase of the wire that goes into the PSU. It is also connected to an Analog Discovery 2 which is used as an oscilloscope which in turn is then connected to a Raspberry Pi 4. This setup allows us to continuously measure and log our data. For more detail see our previous work[1].

Scaphandre One measuring instrument not used in our previous work is Scaphandre[8]. Scaphandre is described as a monitoring agent which can measure energy consumption and is made for Linux where it can use Powercap RAPL which is a Linux kernel subsystem where data can be read from RAPL. It also has the functionality of measuring the energy consumption of some virtual machines, specifically Qemu and KVM hypervisors. A driver also exists which allows for installing RAPL on Windows.[9] Doing so allows using Scaphandre on a Windows computer where the sensor is RAPL which is utilizing the MSRs to update its counters. The Windows version of Scaphandre has some limitations but is able to report the energy consumption of the power domain PKG, using the MSR *MSR_PKG_ENERGY_STATUS*. Furthermore, it can also give an estimation of the energy consumption for individual processes. It does so by storing CPU usage statistics alongside the values of the energy counters. Then it is able to calculate the ratio of the CPU time for each Process ID (PID). With the calculated ratio a new calculation is made to get the subset of the energy consumption which is estimated to belong to a specific PID. A Linux exclusive feature is that the monitoring system Prometheus can be used with Scaphandre to get the energy consumption of an application which consist of several PIDs.

4 Experiments

5 Results

6 Discussion

7 Conclusion

Acknowledgements

8 Future Works

References

1. Holt, J., Kusk, M. H. & Pedersen, J. B. *A Comparison Study of Measuring Instruments* (Aalborg University Department of Computer Science, 2023).
2. Abdelhafez, A., Alba, E. & Luque, G. A component-based study of energy consumption for sequential and parallel genetic algorithms. *The Journal of Supercomputing* **75,** 1–26 (Oct. 2019).
3. Lindholt, R. S., Jepsen, K. & Nielsen, A. Ø. *Analyzing C# Energy Efficiency of Concurrency and Language Construct Combinations* (Aalborg University Department of Computer Science, 2022).
4. Hassan, H., Moussa, A. & Farag, I. Performance vs. Power and Energy Consumption: Impact of Coding Style and Compiler. *International Journal of Advanced Computer Science and Applications* **8** (Dec. 2017).
5. Lima, E., Xavier, T., Faustino, A. & Ruiz, L. *Compiling for performance and power efficiency* in (Sept. 2013), 142–149.
6. Cooper, K. & Waterman, T. Understanding Energy Consumption on the C62x (Jan. 2004).
7. Mozilla. *tools/power/rapl* https://firefox-source-docs.mozilla.org/performance/tools_power_rapl.html. 24/02/2023.
8. Hubblo. *Scaphandre* https://github.com/hubblo-org/scaphandre. 23/02/2023.
9. Hubblo. *windows-rapl-driver* https://github.com/hubblo-org/windows-rapl-driver. 23/02/2023.