Green Metrics Tool: Measuring for fun and profit

Geerd-Dietger Hoffmann
Green Coding Solutions
Berlin, Germany
didi@green-coding.io

Verena Majuntke
HTW
Berlin, Germany
verena.majuntke@htw-berlin.de

Abstract

The environmental impact of software is gaining increasing attention as the demand for computational resources continues to rise. In order to optimize software resource consumption and reduce carbon emissions, measuring and evaluating software is a first essential step. In this paper we discuss what metrics are important for fact base decision making. We introduce the *Green Metrics Tool* (GMT), a novel framework for accurately measuring the resource consumption of software. The tool provides a containerized, controlled, and reproducible life cycle-based approach, assessing the resource use of software during key phases. Finally, we discuss GMT features like visualization, comparability and rule- and LLM-based optimisations highlighting its potential to guide developers and researchers in reducing the environmental impact of their software.

1 Introduction

The significance of the environmental impact of software and hardware has grown in recent years [12]. With the increasing demand for computational resources [10], the reduction of such and associated carbon emissions becomes progressively important. Currently 8–10% of total electricity production [11] is used by the information and communication infrastructure. With new AI technology this is projected to further increase [8].

In order to optimize software, the measurement and evaluation of resource usage is the first essential step. In this paper we (1) discuss metrics which are required to accurately assess the resource usage of software. (2) We present a novel framework, the *Green Metrics Tool*, specifically designed to facilitate the precise measurement of metrics across different operating systems and along the entire software life cycle such as installation, runtime, and removal. (3) Lastly, we discuss the visualization of data and optimization recommendations the GMT provides to assess and improve resource usage. All tools and systems are published open source[18] under the GNU Affero General Public License[1].

2 Related Work

The measurement of resource consumption and performance has been an area of active research since the inventions of computers. Early approaches primarily focused on hardware-level measurements, utilizing specialized equipment to monitor energy usage in real-time [7][22]. However, these methods were often impractical for widespread adoption due to the need for expensive hardware and the complexity of setup [4][20]. Other tools like Greenframe.io [2], Kepler [3], Scaphandre [14] are geared towards assessing the general energy consumption but do not focus on fine grained benchmarks.

LOCO '24, December 3, 2024, Glasgow, Scotland, UK
2024.

3 Challenges and Metrics

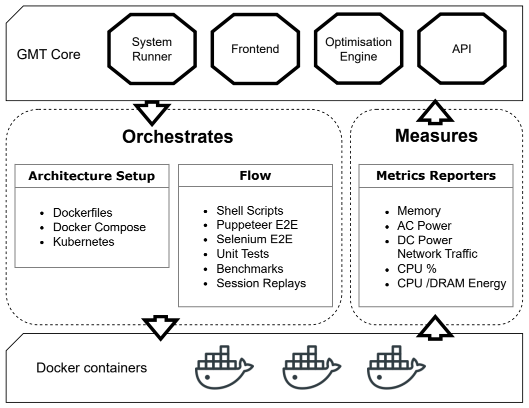
The accurate measurement of software resource consumption is a complex task due to the inherent variability in system behavior and the information required to assess environmental impact. The precision of any measurement is highly dependent on the system environment, i.e., hardware specifications, operating system configuration, and background processes. While CPU energy consumption is often the primary measurement focus - likely due to the availability of tools such as Intel's Running Average Power Limit (RAPL) - this metric alone is insufficient for a holistic assessment as not all problems are CPU bound. In our analysis we identified 15 metrics, e.g., thermal characteristics, disk/network activity, memory utilization, and execution time to name a few. Each of these parameters contributes to a fuller understanding of the environmental impact of software. Moreover, certain system-level inefficiencies, such as resource overprovisioning, have to be considered. Accurate measurements are essential for distinguishing the effects of code edits, as inaccuracies can lead to incorrect conclusions and misguided optimization efforts.

4 System Architecture

The core objective of the GMT is to enable highly accurate reproducible measurements. The architecture overview is shown in Figure 1. To achieve reproducibility and ensure a controlled and isolated environment, the tool employs containerization technology (Docker [15]) to orchestrate the measurement environment. By using containers, the GMT standardizes the testing scenario, reducing the variability introduced by different system configurations and environments enabling precise measurements for all parts of the system, including network traffic. A configuration is used, modeling how the tool is deployed, shown as *Architecture Setup*. Furthermore, the specification of a *usage scenario* modeling how the system is used (shown as *Flow*) is required. Since this is also configured in a container, it can be tailored to whatever is most suitable for the software being benchmarked like *curl*[21], *Puppeteer*[9] or *Playwright*[16]. The tool also offers an interactive mode which is meant for logging resource consumption in deployed systems. This is used to benchmark running AI systems, for example[13]. The tool also includes a *web fronted* and an *optimisation engine* which are described in section 5. Further there is a comprehensive *API* to extract the benchmark data.

The GMT utilizes a range of *Metrics Reporters* which are small, specialized, configurable programs that log performance metrics depending on the machine the benchmark is executed on. These reporters collect data on energy consumption, CPU utilization, memory usage, various temperature sensors and other relevant parameters, providing a comprehensive view of the software's environmental impact. Notably, the GMT is designed to minimize its own

Figure 1: The GMT Architecture



interference during the benchmarking process. While the benchmark is running, the tool refrains from performing any processing, writing the collected data directly to a file. This design choice was empirically validated to impose minimal overhead ($< 1\%$), ensuring that the measurement process itself does not skew the results.

To support sharing specialized measurement hardware, large-scale periodic benchmarking and avoiding the pitfalls of consumer hardware, the Green Metrics Tool offers a cluster deployment option. This enables benchmarking jobs to be run on certain signals like a git commit or periodic measurements of resource consumption over the development time. It also enables the use of expensive specialized hardware, required for precise measurements, as multiple people can share the same physical machine.

4.1 System Calibration and Pre-Measurement Checks

To further enhance measurement accuracy, the GMT performs system checks before initiating the benchmarking process. One critical factor considered is the CPU temperature, as fluctuations in temperature can significantly impact energy consumption readings. It is also important to account for temperature changes over time as this overhead needs to be accounted for through cooling. Additionally, the GMT can disable CPU features, such as Turbo Boost and dynamic frequency scaling as they can introduce variability in performance metrics. Another critical aspect of the GMT's methodology is a calibration script which measures the baseline resource utilization and temperature of the system in its idle state, establishing a reference point for subsequent measurements.

4.2 NOP Linux and Interrupt Reduction

To mitigate the impact of operating system-induced variability on measurements, the GMT integrates with NOP Linux [19], a specialized Linux flavour designed to minimize system OS activity. Operating system interrupts can disrupt the measurement process by consuming CPU cycles. This alters energy consumption patterns as most measurement devices report on a per core or whole machine level. NOP Linux disables these services providing a more stable environment for the measurements while keeping the system as close to an off the shelf system as possible.

4.3 Lifecycle Assessment and Comprehensive Measurement

The GMT's approach to software measurement encompasses the entire software life cycle, from initial installation to eventual removal. This holistic view allows developers to make data-driven decisions that reduce the environmental impact of their software along the life cycle. The life cycle stages considered by the GMT include **Baseline**, which is the measurement of the system's idle state to establish a reference point; **Installation**, involving the evaluation of the resource utilization during the software installation and build process; **Boot**, assessing resource characteristics during the software's startup phase; **Idle**, measuring the software's behavior when it is running but not actively being used; **Runtime**, analyzing the software's resource consumption during active use; and **Removal**, evaluating the impact of the software's uninstallation on the system.

5 Data Visualization and Optimization Recommendation

The GMT includes an interface for data visualization and comparison allowing users to explore collected data in detail, facilitating comparisons between different versions or configurations. The comparison view is particularly useful for identifying changes that may have led to resource improvements or regressions. By providing a clear and intuitive visualization, the GMT empowers developers to make informed decisions about how to optimize their software.

The GMT also offers optimization recommendations based on the collected data. These recommendations are generated after the analysis of the software across the different life cycle stages. For this purpose, a rule based system looks at the values and flags abnormalities like over provisioning of resources, long boot times, low IPC counts, high page faults, etc. A configurable external LLM (Llama[5], ChatGPT[17], Mistral[6], etc.) is prompted for improvement recommendations for code segments which have shown to have a high resource usage. In this step the code segment is copied and a prompt is created queering the LLM to try to "improve" this segment. There is also the option for an LLM to "rate" code segments and suggest improvements. This is done by prompting the LLM with the respective code segments.

6 Conclusion and Future Work

The Green Metrics Tool represents a significant advancement in the field of sustainable software development. By providing precise, reliable measurements and minimizing interference during the benchmarking process, the GMT enables developers to optimize their software based on accurate and comprehensive data. The integration of containerization, NOP Linux, and extensive pre-measurement checks ensures that the tool can deliver consistent and reproducible results. Moreover, the GMT's lifecycle-based approach to software assessment and its robust data visualization capabilities make it an invaluable resource for developers seeking to reduce the environmental impact of their software. Future work includes extending the GMT to have more recommendations, more reporters and finer grain analytics. As the demand for sustainable software continues to grow, tools like the GMT will play a crucial role in helping developers meet these challenges.

References

[1] 2007. GNU Affero General Public License v3.0. https://www.gnu.org/licenses/agpl-3.0.en.html.

[2] 2024. Greenframe - Measure the Carbon Impact of Your Web Applications. https://www.greenframe.io/.

[3] 2024. Sustainable Computing - Measuring and Reducing the Environmental Impact of Computing. https://sustainable-computing.io/.

[4] David Abdurachmanov, Peter Elmer, Giulio Eulisse, Robert Knight, Tapio Niemi, Jukka K Nurminen, Filip Nyback, Gonçalo Pestana, Zhonghong Ou, and Kashif Khan. 2015. Techniques and tools for measuring energy efficiency of scientific software applications. *Journal of Physics: Conference Series* 608, 1 (apr 2015), 012032. https://doi.org/10.1088/1742-6596/608/1/012032

[5] Meta AI. 2024. LLaMA Language Model. https://ai.facebook.com/blog/large-language-model-llama-meta-ai/. Accessed: 2024-09-25.

[6] Mistral AI. 2024. Mistral AI Official Website. https://mistral.ai. Accessed: 2024-09-25.

[7] Frank Bellosa. 2000. The benefits of event: driven energy accounting in power-sensitive systems. In *Proceedings of the 9th Workshop on ACM SIGOPS European Workshop: Beyond the PC: New Challenges for the Operating System* (Kolding, Denmark) *(EW 9)*. Association for Computing Machinery, New York, NY, USA, 37–42. https://doi.org/10.1145/566726.566736

[8] Adrien Berthelot, Eddy Caron, Mathilde Jay, and Laurent Lefèvre. 2024. Estimating the environmental impact of Generative-AI services using an LCA-based methodology. *Procedia CIRP* 122 (2024), 707–712. https://doi.org/10.1016/j.procir.2024.01.098 31st CIRP Conference on Life Cycle Engineering.

[9] Google Developers. 2024. Puppeteer Documentation. https://pptr.dev/. Accessed: 2024-09-25.

[10] World Economic Forum. 2024. Data growth drives ICT energy innovation. https://www.weforum.org/agenda/2024/05/data-growth-drives-ict-energy-innovation/ Accessed: 2024-09-27.

[11] Erol Gelenbe. 2023. Electricity Consumption by ICT: Facts, trends, and measurements. *Ubiquity* 2023, August, Article 1 (Aug. 2023), 15 pages. https://doi.org/10.1145/3613207

[12] Udit Gupta, Young Geun Kim, Sylvia Lee, Jordan Tse, Hsien-Hsin S. Lee, Gu-Yeon Wei, David Brooks, and Carole-Jean Wu. 2021. Chasing Carbon: The Elusive Environmental Footprint of Computing. In *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. 854–867. https://doi.org/10.1109/HPCA51647.2021.00076

[13] Geerd-Dietger Hoffmann and Verena Majuntke. 2024. Improving Carbon Emissions of Federated Large Language Model Inference through Classification of Task-Specificity. (2024).

[14] hubblo org. 2024. Scaphandre: An Energy Consumption Monitoring Agent. https://github.com/hubblo-org/scaphandre.

[15] Docker Inc. 2024. *Docker*. https://www.docker.com/

[16] Microsoft. 2024. Playwright Documentation. https://playwright.dev/. Accessed: 2024-09-25.

[17] OpenAI. 2024. ChatGPT. https://openai.com/chatgpt/. Accessed: 2024-09-25.

[18] Green Coding Solutions. 2024. Green Metrics Tool: A Tool to Measure the Environmental Impact of Software. https://github.com/green-coding-solutions/green-metrics-tool.

[19] Green Coding Solutions. 2024. NOP Linux: Reducing Interrupts for Accurate Energy Measurements. https://www.green-coding.io/blog/nop-linux/

[20] Shekhar Srikantaiah, Aman Kansal, and Feng Zhao. 2008. Energy Aware Consolidation for Cloud Computing. In *USENIX HotPower'08: Workshop on Power Aware Computing and Systems at OSDI* (usenix hotpower'08: workshop on power aware computing and systems at osdi ed.). USENIX. https://www.microsoft.com/en-us/research/publication/energy-aware-consolidation-for-cloud-computing/

[21] Daniel Stenberg. 2024. curl – Command Line Tool and Library for Transferring Data with URLs. https://curl.se/. Accessed: 2024-09-25.

[22] Vivek Tiwari, Sharad Malik, Andrew Wolfe, and Mike Tien-Chien Lee. 1996. *Instruction Level Power Analysis and Optimization of Software.* Springer US, Boston, MA, 139–154. https://doi.org/10.1007/978-1-4613-1453-0_9