SimplyEnergi: Visualizations and Analysis for EnergiBridge

Group 19

Aline Mol
5034590
Technical University of Delft
Delft, The Netherlands
a.l.mol@student.tudelft.nl

Duyemo Anceaux
4919327
Technical University of Delft
Delft, The Netherlands
d.anceaux@student.tudelft.nl

Max de Groot
5574870
Technical University of Delft
Delft, The Netherlands
m.r.degroot-1@student.tudelft.nl

Kris van Melis
5070619
Technical University of Delft
Delft, The Netherlands
k.p.vanmelis@student.tudelft.nl

Abstract—Due to the large impact software has on the environment, it is crucial that software developers can gain insights into the sustainability of their software. Current solutions are either solely focused on data collection, specific to a domain, or not versatile enough regarding energy metrics. We introduce SimplyEnergi, a system which allows users to easily generate visualizations from a diverse set of energy-related metrics. Furthermore, the application supports standard statistical analysis of the metric measurements. The system is an extension of EnergiBridge, which is an application that allows users to collect data of the energy-related metrics on different CPU, GPU and operating system configurations. Qualitative validation with multiple testing datasets shows that SimplyEnergi reliably generates accurate visualizations.

Index Terms—Sustainable Software, Energy Monitoring, Green IT

I. Introduction

With the rise of AI, the impact of the IT sector on global energy consumption is growing rapidly. The World Economic Forum estimates that the energy consumption of data centers and communication centers accounts for 2-3% of total global energy consumption today and that this is expected to increase even further [1]. Due to pressures on the environment from the emission of greenhouse gases, reducing energy usage of the IT sector is vital. Improving energy efficiency of software at the design stage can have significant contributions towards this goal.

However, it is difficult to analyze the energy efficiency of programs through static analysis, especially when analyzing across different hardware set-ups or software like compilers. Accessible tools that can both measure and visualize the energy consumption of programs can offer a good solution to aid in the design of more energy efficient software. By offering a quick graphical insight, hotspots of energy consumption can be identified and comparisons of energy consumption between different software and hardware configurations can be made.

Programs that measure energy consumption along with other performance metrics already exist. An open-source example is EnergiBridge [2], a command-line application that records energy usage over time and saves the results in CSV format. Although EnergiBridge is effective at collecting measurements, it places the responsibility of data analysis and visualization entirely on the user, which can be time-consuming and technically demanding.

To address this challenge, we propose SimplyEnergi; a visualization tool for EnergiBridge that automates the pre-processing and visualization of its measurements, using Grafana for its visualizations[1]. SimplyEnergi lowers the bar of entry for analysis capabilities for users of EnergiBridge by offering a more versatile and user-friendly visualization tool. This extension enables users to easily and quickly visualize a variety of metrics related to energy efficiency.

In addition to visualizations, the tool supports basic analytical features such as significance testing and statistical summaries. These enhancements are designed to streamline the energy analysis process, reduce the manual effort required, and promote more effective energy efficiency studies. By providing users with energy metric visualizations and analysis tools, SimplyEnergi delivers an efficient and accessible way to review the energy efficiency of their software, with the ultimate goal of aiding in the design of more sustainable software.

II. Related Work

Despite the growing need for easy generation of accessible and flexible visualizations of energy consumption

[1]Grafana observability tool: https://grafana.com

data, few tools exist which fully accomplish this. Most existing solutions either focus solely on data collection or are tailored to specific domains, such as machine learning. General-purpose, user-friendly tools that automate both pre-processing and visualization remain scarce. In this section, we discuss the most relevant related efforts and discuss them in relation to our goal of creating a more versatile and accessible energy visualization tool.

A recent paper central to our work is one by Sallou et al. [3], who introduce EnergiBridge as a cross-platform measurement utility tool for energy related metrics to facilitate analysis of energy consumption. While EnergiBridge is mostly concerned with data collection, we propose a system that takes the burden of visualizing and analysing the data EnergiBridge provides away from users.

A relevant effort is EnergiReporter [4]; an open-source application designed to visualize energy data collected from EnergiBridge. EnergiReporter has taken important steps towards simplifying the interpretation of energy usage by providing ready-made visualizations. However, it primarily focuses on visualizing power consumption over time and does not support more advanced features, such as comparing multiple experiments within a single plot, or analyzing a broader range of measurement types. To address these limitations, SimplyEnergi allows users to generate more flexible and versatile visualizations.

Machine learning is the field within IT where energy consumption is increasing the most. That is why a group of researchers developed EnergyVis [5] as an interactive energy consumption tracker for machine learning models. EnergyVis enables users to track CO2 emissions and compare energy usage between different pre-loaded machine learning models. In addition to its visual capabilities, EnergyVis also supports live tracking via a back-end system that monitors CPU and GPU usage during training. This tool aims to increase awareness on the cost of training machine learning models. As EnergyVis is tightly coupled with machine learning workflows, the very domain-specific structure might not align with our more general use cases.

The work by White et al. [6] extends the XDMoD High-Performance Computing (HPC) monitoring platform by integrating power consumption metrics into its analysis dashboard. Their system enables time-series and statistical visualizations of energy usage per job, allowing users to correlate power demand with performance indicators like energy consumption and memory usage. Their focus is on HPC environments, and the approach focuses on embedding energy metrics into specific monitoring workflows. Our work builds a more flexible and general-purpose visualization for energy usages.

III. System Design

Various aspects are considered during the design process of the application. This section highlights our design choices by first explaining our design philosophy and elaborating on the energy metrics we choose to visualize. Then, a typical user workflow is defined and implementation details that realize these features are discussed.

A. Design Philosophy

The primary motivation for creating the system is the need for easy and intuitive interpretations of data produced by EnergiBridge, as the large amounts of data may appear unclear to users, causing any primary analysis to be time and focus-intensive. We therefore prioritize runtime, usability, and accessibility in the system to provide meaningful visualizations of metrics, keeping in mind the trade-off in expressiveness of the visualizations.

Another aspect considered in the design is flexibility since the columns in output data produced by EnergiBridge depend on hardware specifics, such as the type of GPU and CPU. Hence, the system should be flexible to allow data from any hardware configuration to be processed by the application, as long as the input data originates from EnergiBridge.

Furthermore, the system should also be extensible and open source to enforce open science principles and allow for maintainability and further development based on how EnergiBridge progresses. To facilitate this, the code is written in a modular fashion and documentation is included at both function and module level. The codebase for SimplyEnergi is publicly available on GitHub[2].

B. Measurements for Visualization

The CSV file that EnergiBridge outputs contains a wide range of different measurements related to energy consumption, on which visualizations generated are based. We refer to the types of data that can be recorded within EnergiBridge, such as CPU power or total memory, as measurements and the aggregated values, such as median and quartiles, as metrics. The types of measurements that can be visualized are categorized into two types: time-independent and time-dependent. For both categories we explain which measurement types we visualize, motivated by their relevance to sustainability. All example visualizations depicted in this section are generated from sample data obtained in our previous project where we compare sustainability metrics of different extension configurations within Visual Studio Code[3].

1) Time-independent measurements: Measurements independent from time can give an impression about a trial at a single glance and are useful when tasks as a whole should be considered. These include CPU statistics, such as the total energy consumption and peak power usage. An example where measuring the total energy consumption is of interest could be to compare the value for two different versions of a system to determine whether the

[2]GitHub repository containing SimplyEnergi's codebase: https://github.com/krisvanmelis/energibridge-analytics

[3]GitHub repository containing Visual Studio Code energy consumption data: https://github.com/idealine2/CS4575-SSE-gr19-project1

new version of the system is more or less energy efficient. Meanwhile, high peaks in power consumption can indicate which programs and software are more intensively used and may indicate inefficiencies. In EnergiBridge, there is the possibility to generate these measurements for the CPU as a whole and for each core. For ease in navigating, the statistics are organized in collapsible sections per core or for the whole CPU. Figure 1 shows how total energy consumption and peak power usage are presented within SimplyEnergi.

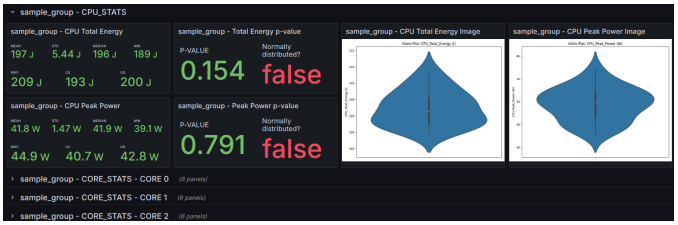

Fig. 1: Example of the panel generated by SimplyEnergi of general time-independent statistics.

2) Time-dependent measurements: Time-dependent measurements include those where variations due to specific sequential events are of interest. Time-based data may reveal how the system's performance fluctuates throughout the duration of an experiment, highlighting resource demand spikes and inefficiencies, allowing identification of corresponding software components.

A measurement that typically comes to mind for measuring a system's efficiency or performance visualized over time is power consumption. Moments of peak power consumption can highlight certain processes within a task and indicate targets for efficiency improvements while averages can indicate typical or idle energy usage. Shifting peaks may be beneficial when using intermittent sources such as solar panels, which may vary in their energy output throughout the day. Using this measurement while programming for sustainability can therefore help determine whether different strategies are effective at shifting or reducing this peak, and to help coordinate supply and demand of electricity. SimplyEnergi considers both the total power consumption and the power consumption per CPU core, when this is available (AMD CPUs).

Another metric chosen to visualize is the memory and swap memory usage over time. Efficient memory usage reduces the need for larger, more power-demanding hardware, leading to lower energy consumption and material waste. On the other hand, enough memory could reduce the number of costly disk operations that need to be performed, further linking to power consumption. For both, we decided on only displaying the median, due to a low amount of variance in the data.

Aside from overall CPU measurements, EnergiBridge is also able to visualize measurements per core or logical processor when these are available from the EnergiBridge data. This may help in analyzing programs that run on multiple cores simultaneously to see if resource demand is spread evenly or could be optimized. For example, it may help in initial investigation for undervolting similar to what Koutsovasilis et al. do in their paper where undervolting is explored for improving energy efficiency [7]. By examining voltage and power consumption side-by-side, developers can identify opportunities where undervolting might be safely applied without compromising program stability or responsiveness.

Since each experiment group may contain multiple trials, data at a specific point in time is defined by the median of that trial. Moreover, the quartiles of the data are depicted to indicate variance accross trials. The median is chosen above the mean as taking a mean is more sensitive to anomalous trials that may occur due to other background processes. For the same reason, the upper and lower quartiles are displayed instead of the minimum and maximum values. Aside from these three metrics, the preprocessing stage does still calculate the mean, minimum and maximum values as well as the standard deviation for ease in later analysis if wished for. When visualizing the data for the cores, we decided to only display the median. This is because all cores are visualized in the same graph to facilitate easy comparison, although this can also lead to visual clutter.

Visualizations over time look similar for all metrics, apart from single measurements and measurements per core. Figures 2 and 3 depict example visualizations of CPU power and CPU power per core.

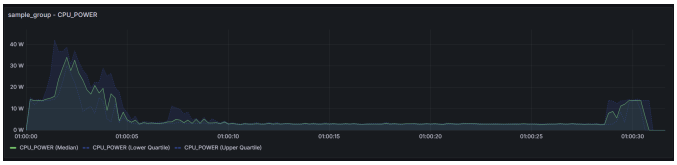

Fig. 2: Visualization generated by SimplyEnergi of CPU power over time for sample data.

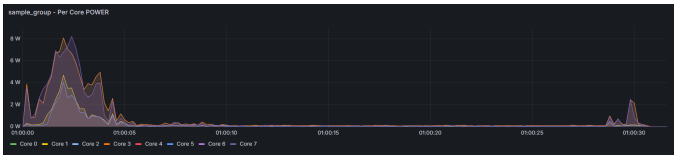

Fig. 3: Visualization generated by SimplyEnergi of CPU power over time per core for sample data.

Having described all metrics we visualize, there are also numerous metrics that are not visualized by SimplyEnergi. An example of this would be CPU temperature. Most of these were also not as useful to visualize, such as CPU frequency, which is expected to remain relatively constant. However, we must mention that CPU temperature, which is currently not supported would have been interesting to visualize as temperature gives an indication of energy efficiency. Due to EnergiBridge not supporting CPU

temperature for all platforms, it is also not included in SimplyEnergi.

3) Comparing groups: In the interest of cross-configuration analysis, SimplyEnergi is able to compute statistics to compare two different groups of trials. The measurements available for comparison are the total energy, peak power, power over time, memory and swap usage over time, which all give some indication of sustainability, as explained in Sections III-B1 and III-B2. Since we allow users to add multiple experiments to the same dashboard, a user can also compare more than two groups.

Supplementary to these basic comparisons, SimplyEnergi also supports the generation of violin plots of the total energy and peak power for two groups of trials. The choice for inclusion of these measurements are the same as mentioned previously. Violin plots offer a quick method to show data distributions, this is useful for these metrics because qualitative analysis on whether the data is normal can be done, and differences in averages are easy to compare across groups. An example of how violin plots are generated can be seen in Figure 4.

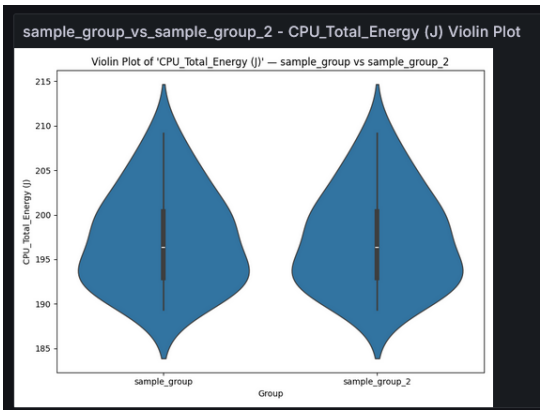

Fig. 4: Example violin plot generated by SimplyEnergi.

To facilitate scientific research with sustainability metrics, the system can also calculate the p-value of the metrics between two groups. To calculate this value, we used Welch's t-test if the data for both groups is normally distributed, else the Mann–Whitney U test is used. We test for normality using the Shapiro test. When the p-value of a single dataset is smaller than or equal to 0.05, it is assumed the dataset is normally distributed. For these simple numeric values, the panel shows whether the difference between the two groups is statistically significant. Firgure 5 shows an example of a statistics panels. comparing two groups

Finally, for comparison purposes, it is also possible to visualize measurements of two groups in the same graph. We chose to do this by plotting power over time, as is shown in Figure 6.

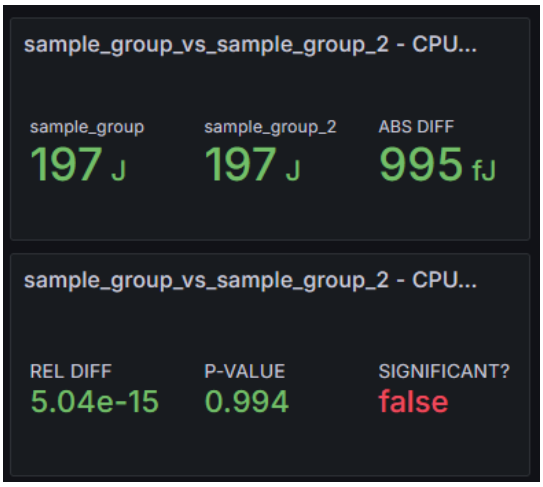

Fig. 5: Example statistics comparing two groups generated by SimplyEnergi.

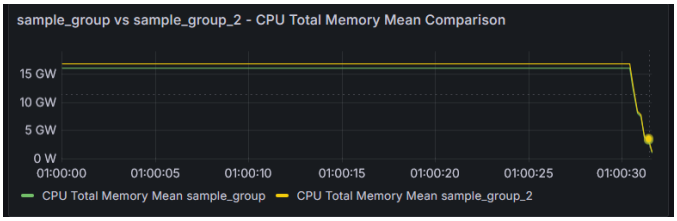

Fig. 6: Comparison of total energy statistics for two experimental groups.

C. User Workflow

So far, various functionalities of the system have been described. This subsection explains the workflow a user goes through when generating visualizations. Figure 7 shows the user-interface, along with numbers that represent steps in the workflow. The steps a user goes through to generate visualizations are as follows:

1) A user chooses a name for their experiment (a dashboard within Grafana).
2) A user chooses which experiment groups (data) to use for the experiment.
3) A user chooses their experiment type (e.g. visualization over time).
4) A user selects which metric, or measurement types to visualize.
5) Button to add an experiment configuration.
6) Table showing all configured experiments.
7) Button to generate visualizations for the defined. Pressing the button redirects a user to the Grafana dashboards generated.

D. Implementation Details

There are two primary components that the user interacts with. First, we have the user interface used to load the data and define visualizations that the user wants to see from a pre-defined set of measurement types.

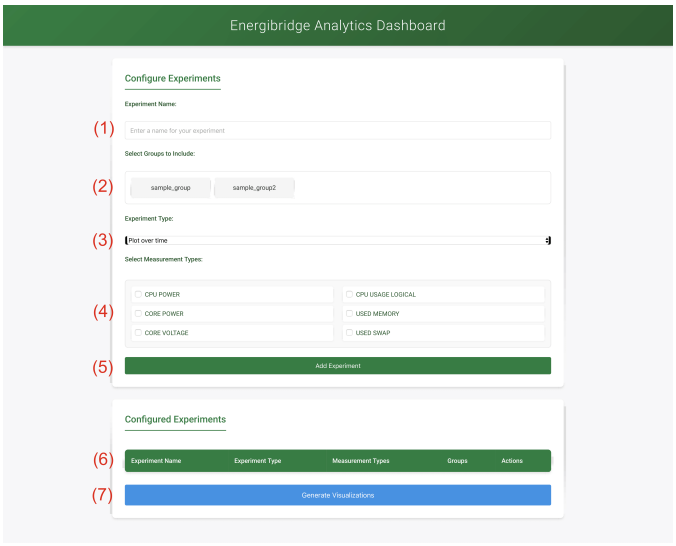

Fig. 7: User-interface overview of SimplyEnergi.

Second, the user interacts with a Grafana dashboard to see the visualizations that they have selected. We decided to use Grafana for the purpose of visualizations as it provides extensive out-of-the-box functionality to visualize data, while also giving users the ability to customize visualizations.

The entire system is built using Python 3.13 and the framework Flask[4]. Flask provides both a simple way to set up a user interface and a back-end for data processing.

Data pre-processing is necessary because EnergiBridge does not have a constant CSV format in which it outputs data, as columns depends on the user's hardware. In the pre-processing stage, columns of the CSV file that EnergiBridge outputs are renamed so that they are standardized across different hardware.

Additionally, the pre-processing stage aggregates the data of all trials into a single group of metrics per unit time, described in Section ??. This standardization procedure facilitates the generation of visualizations in Grafana. Violin plots are also generated during this stage for potential later use because Grafana does not have built-in options for this type of graph. The violin plots are stored as PNG files and loaded into Grafana in a similar manner to the CSV data.

To load these CSV datasources into Grafana the system uses the Infinity plugin for Grafana[5]. A Grafana dashboard (collection of visualizations) is configured through a configuration file in JSON format. When a user clicks the button to generate visualizations in the GUI, this JSON file is generated in the back-end based on inputs the user provided (e.g. input data and measurement types). After the dashboard is configured, the user is redirected to Grafana, where they can see the visualizations generated based on their data.

Both the GUI and Grafana are hosted on their own Docker container. In order for Grafana to have access to the CSV data a user uploads, the CSV data is hosted on an Nginx server, which also has its own Docker container. Currently, the containers are set up to host locally. However, due to the modular design these containers could be hosted on an actual server in the future so that the system can be accessible on a website, for example.

IV. Validating the capabilities of SimplyEnergi

To investigate whether the application accurately generates visualizations outside the sample data used during development, we upload data from another work, EnergyDebug [8], which generates graphs based on data collected with EnergiBridge. We compare the graphs that SimplyEnergi generates with those in the paper of EnergyDebug. Note that this is not a complete validation protocol, but it does give an indication of the accuracy of the system. We compare the visualizations for two different kinds of graphs between our proposed tool and EnergyDebug.

The first of these visualizations is the one for plotting data over time. Figure 8 shows the graph from the EnergyDebug paper, while Figure 9 shows the visualization generated by our tool. Though lines plotted roughly show the same trends, there is a notable difference in the scaling for this plot. The plot of our tool is a bit wider, which results in a less sharp drop in the end. Also, in the visualization done by SimplyEnergi, the peaks at the start and end of the dataset are not as clear which likely due to this scaling.

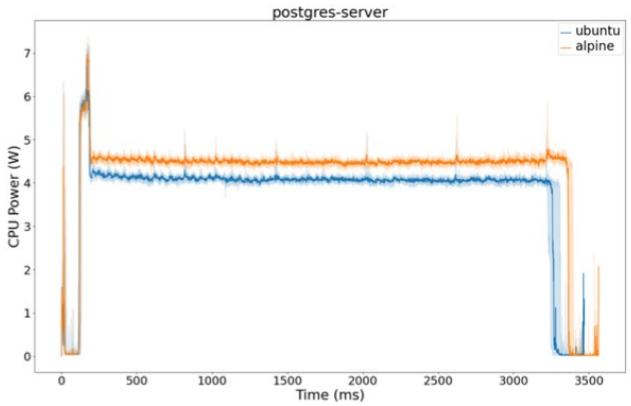

Fig. 8: Graph of CPU power over time, adapted from Energy Debug [8].

The second type of visualization for which we make the comparison is that of the violin plot of total energy used when using Ubuntu versus Alpine. Figure 12 shows two violin plots generated by EnergyDebug. Our tool generated violin plots using the same data, which can

[4]Documentation of the Flask Python framework: https://flask.palletsprojects.com/en/stable/

[5]Infinity plugin for Grafana: https://grafana.com/docs/plugins/yesoreyeram-infinity-datasource/

Fig. 9: Graph showing CPU power over time generated using SimplyEnergi.

be seen in Figures 10 and 11. Both of the violin plots our tool created are identical to those from EnergyDebug.

Our tool created a visualization for power over time similar to those of EnergyDebug, and created identical violin plots depicting the total amount of energy used. These two factors indicate the validity of the capability of our tool, assuming the graphs in the EnergyDebug report were generated correctly.

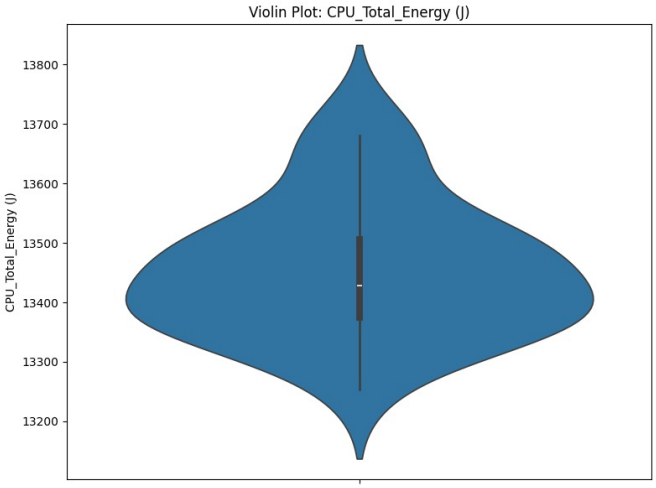

Fig. 10: Violin plot of CPU energy for alpine configuration, adapted from [8].

V. Discussion

Despite the promising results of the qualitative analysis,SimplyEnergi still has a number of improvements that can be made. First, we highlight the system's strengths and weaknesses, after which future work to improve the system is given.

A. Evaluation

A major strength of our design lies in its versatility across different measurements and its flexibility in terms of hardware and operating systems. The system is built to automatically pre-process the input data and compute a range of, in some cases aggregated, statistics. Examples of this are the mean, median, and standard deviation, which are calculated for all available metrics. This automation significantly reduces the manual efforts typically required to prepare the data for analysis. As a result, users can easily generate custom visualizations based on these

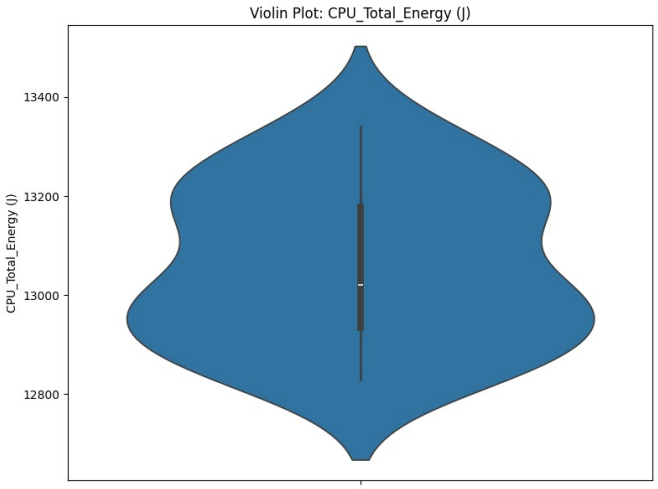

Fig. 11: Violin plot of CPU energy for ubuntu configuration, adapted from [8].

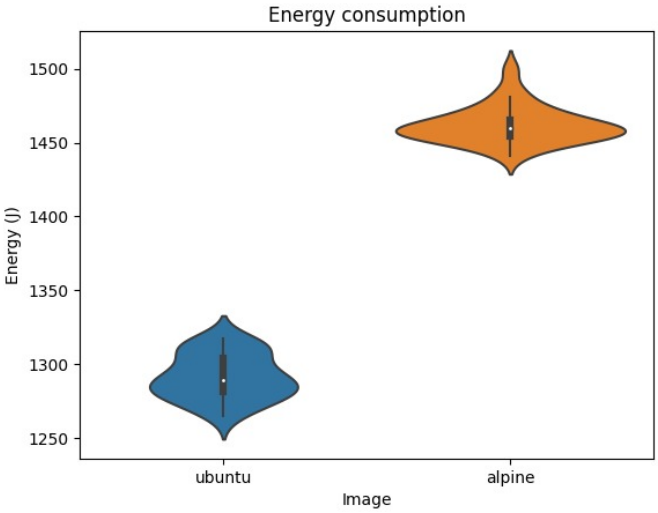

Fig. 12: Violin plot of CPU energy for alpine and ubuntu configurations generated using SimplyEnergi, using data from [8].

general statistics without having to write additional code or perform separate calculations.

Another key strength of the tool is its ease of use, particularly when leveraging the set of pre-configured visualizations. The entire system can be launched with a single command from the command line, streamlining the setup process. By providing simplicity, the barrier to entry is lowered, although it still requires some technical expertise. More importantly, the simplicity makes the process of analyzing the energy consumption measurements less time-consuming.

A possible area for improvement is the experience after clicking the Generate Visualization button. It can take some time for the new Grafana dashboard to be fully

generated and loaded. However, users are immediately redirected to Grafana during this process, resulting in not seeing any available dashboard or a previous version. While this is easy to work around by simply refreshing the page, it can be confusing for first-time users.

Moreover, while the violin plot gives insight into the distribution of the data, they are currently rendered as static images within Grafana, due to Grafan not supporting violin plots natively yet. The lack of interactivity from having an image loaded limits the user's ability to explore the data more deeply. Integrating interactive violin plots, potentially through custom plugins or embedding libraries, could significantly elevate the analytical experience

It is also worth mentioning that certain measurement types that EnergiBridge calculates, such as CPU temperature, are specific to certain hardware or operating systems. Some of these exclusive measurement types are selectable in the GUI, like the energy usage per core, but the system will not successfully generate visualizations if that measurement type is not included in the selected CSV files.

B. Future work

While our current system focuses on post-experiment visualization of data collected by EnergiBridge, future work could explore the implementation of real-time graphing capabilities using data streams to enable live monitoring of energy usage. One way to achieve this would be through deeper integration with EnergiBridge's data collection pipeline to stream measurements directly into the visualization layer. Alternatively, EnergiBridge could be configured to run in short epochs, periodically updating the visualizations after each interval to simulate a near real-time experience. This enhancement would provide immediate feedback during program execution and could be especially useful for interactive development and performance tuning.

Furthermore, not all measurement types exclusive to specific hardware/software configurations are handled consistently. The energy per core is accepted by the UI, but the CPU temperature is not given as an option. In a future version of the application, a consistent solution for exclusive measurement types should be implemented. A possible solution is to always show them as an option in the UI, but present users with a disclaimer mentioning the specific configurations that support those measurement types. Additionally, users should be warned with an error message when trying to generate visualizations with unsupported measurement types.

While we propose a tool, there are no clear insights into how user-friendly it is. To formally asses this, a user-study could be conducted. Feedback from such a study could highlight lackluster features, reveal unmet user needs, and guide improvements to both the interface and overall user experience. Other research that could be done in the future may include analyzing how energy visualization tools affect developer behavior and decision-making. This could then be used to help validate the real-world impact of the proposed tool.

VI. Conclusion

The goal of this work is to satisfy the demand for software that allows users to quickly and easily generate visualizations of various measurements related to energy consumption of software and processes. For this purpose, we introduce SimplyEnergi, an extension of EnergiBridge. The application provides a simple user interface where users can upload their data and select visualizations to generate, which can be viewed in Grafana dashboards. SimplyEnergi inherits EnergiBridge's flexibility regarding hardware and operating systems by applying a preprocessing stage to data uploaded by users.

We establish that the system accurately generates versatile visualizations by uploading data from other projects and comparing the resulting graphs to those of the original works. Despite its current limitations, SimplyEnergi already makes a significant contribution towards facilitating the analysis of energy-related measurements. In conclusion, the tool developed in this work provides a usable, flexible, and versatile solution, and takes a step towards more sustainable software.

References

[1] B. Valkhof, E. Kemene, and J. Stark. "Data volume is soaring. Here's how the ICT sector can sustainably handle the surge." (2024-05-22, 2024), [Online]. Available: https://www.weforum.org/stories/2024/05/data-growth-drives-ict-energy-innovation/.

[2] J. Sallou, L. Cruz, and T. Durieux, EnergiBridge: Empowering software sustainability through cross-platform energy measurement, version 1.0.0, Dec. 2023. doi: 10.48550/arXiv.2312.13897. [Online]. Available: https://github.com/tdurieux/EnergiBridge.

[3] J. Sallou, L. Cruz, and T. Durieux, EnergiBridge: Empowering software sustainability through cross-platform energy measurement, 2023. doi: 10.48550/arXiv.2312.13897. arXiv: 2312.13897 [cs.SE].

[4] T. Penning, B. Marcelis, and M. de Koning, EnergiReporter: An open-source app for analyzing energy data files, GitHub, 2023. [Online]. Available: https://github.com/tpenning/EnergiReporter.

[5] O. Shaikh, J. Saad-Falcon, A. P. Wright, et al., "EnergyVis: Interactively tracking and exploring energy consumption for ML models," in Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems, ser. Chi Ea '21, Yokohama, Japan and New York, NY, USA: Association for Computing Machinery, 2021, isbn: 978-1-4503-8095-9. doi: 10.1145/3411763.3451780.

[6] J. P. White, M. Innus, R. L. Deleon, M. D. Jones, and T. R. Furlani, "Monitoring and analysis of power consumption on hpc clusters using xdmod," in Practice and Experience in Advanced Research Computing 2020: Catch the Wave, 2020, pp. 112–119.

[7] P. Koutsovasilis, K. Parasyris, C. D. Antonopoulos, N. Bellas, and S. Lalis, "Dynamic Undervolting to Improve Energy Efficiency on Multicore X86 CPUs," IEEE Transactions on Parallel and Distributed Systems, vol. 31, no. 12, pp. 2851–2864, Dec. 1, 2020, issn: 1045-9219, 1558-2183, 2161-9883. doi: 10.1109/TPDS.2020.3004383. [Online]. Available: https://ieeexplore.ieee.org/document/9123555/ (visited on 04/04/2025).

[8] E. B. Roque, L. Cruz, and T. Durieux, Unveiling the energy vampires: A methodology for debugging software energy consumption, 2024. doi: https://doi.org/10.48550/arXiv.2412.10063. arXiv: 2412.10063 [cs.SE].