From YAML to Insight: Interactive Visualization of hws Hardware Samples

1st Tim Thüring
IPVS
University of Stuttgart
Stuttgart, Germany
Tim.Thuering@ipvs.uni-stuttgart.de

2nd Alexander Van Craen
IPVS
University of Stuttgart
Stuttgart, Germany
Alexander.Van-Craen@ipvs.uni-stuttgart.de

3rd Dirk Pflüger
IPVS
University of Stuttgart
Stuttgart, Germany
Dirk.Pflueger@ipvs.uni-stuttgart.de

In recent years, energy efficiency has become an additional optimization goal for application developers alongside peak performance. When faced with such a challenge, a holistic view of how the application interacts with the hardware is crucial. However, even simple questions can quickly become cumbersome to answer: Which GPU consumed the most power? Does the GPU still reach its maximum frequency after switching from FP32 to FP64? The situation becomes even more complex when working with heterogeneous hardware ranging from CPUs to GPUs from different manufacturers. This is why the hws library by Breyer et al. [1] was developed to provide an easy-to-use way to capture hardware metrics for CPUs and GPUs from various manufacturers, including NVIDIA, AMD, and Intel. hws makes tracking power consumption, clock frequencies, memory, and other metrics straightforward. However, especially when working in distributed multi-GPU settings, even short sampling periods can yield hundreds of time series with millions of data points. As hws exports this data in YAML format, it has been up to the user to transform the large volumes of data gathered by hws into insights by developing custom visualization and post-processing workflows for each problem. Therefore, in this poster, we present a companion visualization tool that completes the hws workflow from sampling to data exploration by generating interactive, standalone HTML reports. This allows users to explore and filter the hws data to easily understand how their application interacts with the underlying hardware. Our visualization tool continues the ease-of-use paradigm of hws by keeping setup effort to a minimum. A single Python script takes the hws output YAML file(s) as input and generates an interactive Plotly-based [2] .html report file that can be viewed in the user's browser. To reduce system memory usage when displaying larger reports, only a fixed number of plots are active at a time. Unlike trace visualizers such as Vampir [3] for, e.g., Score-P [4] traces, or Paraver [5], which require a dedicated viewer, our tool needs only a browser. The resulting HTML report is a self-contained file and, as such, can be easily shared with collaborators. Next, we detail the visualizer's key features.

The authors acknowledge support by the state of Baden-Württemberg through bwHPC.



Fig. 1. Main tab of the hws visualization tool, highlighting the filter section, custom views, energy dashboard, and custom plots tabs.

A. Filtering

The sidebar panel highlighted in Fig. 1 allows the user to apply various filters to restrict the sampled series to those of interest. The user can choose to visualize only the time series for a subset of MPI ranks, devices, metric groups (e.g., all memory metrics), or individual samples (e.g., GPU temperature).

B. Energy Dashboard

The energy dashboard summarizes aggregated energy and power metrics across all devices and MPI ranks. Fig. 2 shows an overview section of the dashboard, presenting the energy consumption of six GPU nodes, broken down by the contributions of their 24 NVIDIA H100 GPUs. This allows the user to quickly verify whether the energy distribution is even across all nodes and GPUs, or to identify any imbalances. The dashboard also reports the average, minimum, and maximum power drawn by each device.

C. Compare Mode

The Python script can take multiple .yaml files from multiple hws runs as input and overlay all time series in the interactive HTML report. This is especially useful for quickly analyzing the effects of code or hardware configuration changes. Fig. 3 showcases the compare mode, overlaying the clock frequencies of two sustained GEMM benchmark



Fig. 2. Energy dashboard showing the total energy consumption of six nodes equipped with four NVIDIA H100 GPUs each during a sustained GEMM benchmark.

runs on an AMD MI210 GPU with FP32 (blue series) and FP64 (orange series) precision. The visualization shows at a glance that the FP64 run resulted in lower clock frequencies throughout.

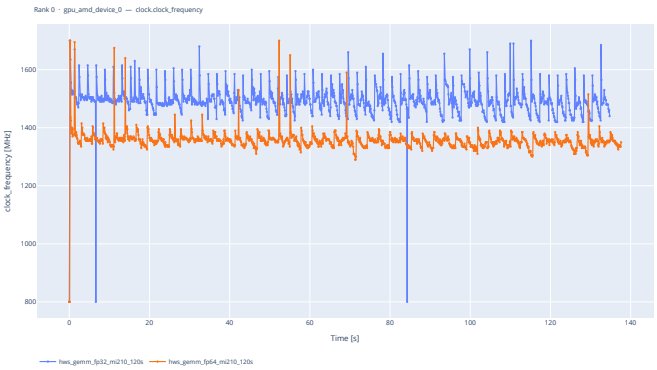


Fig. 3. Clock frequencies of sustained GEMM benchmark runs with FP32 (blue) and FP64 (orange) precision on an AMD MI210 GPU visualized with the compare mode.

D. Custom Views and Custom Plots

Custom views, preconfigured when creating the HTML report, show only subsets of samples in separate tabs at the top (Fig. 1). To gain further insights by comparing metrics, our visualization tool offers an interactive plot builder that lets users overlay multiple series of compatible metrics. For example, the user could overlay the GPU and CPU memory consumption or the power draw of all GPUs in the system into one plot for an intuitive comparison.

In summary, the hws visualizer closes the gap between hws's simple hardware sampling and fast extraction of insights for developers. Both the visualizer[1] and hws[2] are open source on GitHub. To demonstrate our tool's usefulness, our e-poster will feature embedded HTML reports from multiple hws-tracked scenarios, letting visitors explore them directly on a touchscreen. Interactive quiz questions will further encourage visitors to explore the scenarios with our tool.

AI Use Disclosure

Generative AI tools, including Grammarly, DeepL, Perplexity, and Claude, were employed to enhance the clarity, grammar, and overall coherence of the manuscript. All technical content, data analyses, and research findings were conceived and developed independently by the authors. AI-assisted outputs were carefully reviewed, verified, and edited by the authors to ensure factual accuracy, interpretive rigor, and scholarly integrity. The final manuscript reflects the authors' original intellectual contributions and analytical work.

References

[1] M. Breyer, A. Van Craen, P. Domanski, and D. Pflüger, "hws: A tool for monitoring hardware metrics across diverse vendors: A case study on hyperparameter optimization algorithms," in *SC24-W: Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2024, pp. 1815–1825.
[2] Plotly, "Plotly," https://plotly.com, 2026, visited on 2026-09-23.
[3] A. Knüpfer, H. Brunst, J. Doleschal, M. Jurenz, M. Lieber, H. Mickler, M. S. Müller, and W. E. Nagel, "The Vampir performance analysis tool-set," in *Tools for High Performance Computing*, M. Resch, R. Keller, V. Himmler, B. Krammer, and A. Schulz, Eds. Berlin, Heidelberg: Springer, 2008, pp. 139–155.
[4] A. Knüpfer, C. Rössel, D. an Mey, S. Biersdorff, K. Diethelm, D. Eschweiler, M. Geimer, M. Gerndt, D. Lorenz, A. Malony, W. E. Nagel, Y. Oleynik, P. Philippen, P. Saviankou, D. Schmidl, S. Shende, R. Tschüter, M. Wagner, B. Wesarg, and F. Wolf, "Score-P: A joint performance measurement run-time infrastructure for Periscope, Scalasca, TAU, and Vampir," in *Tools for High Performance Computing 2011*, H. Brunst, M. S. Müller, W. E. Nagel, and M. M. Resch, Eds. Berlin, Heidelberg: Springer, 2012, pp. 79–91.
[5] BSC, "Paraver," https://tools.bsc.es/paraver, 2026, visited on 2026-09-23.

[1] https://github.com/SC-SGS/hws_visualizer
[2] https://github.com/SC-SGS/hardware_sampling