Fine-Grained Application Energy and Power Measurements on the Frontier Exascale System

Oscar Hernandez
Wael Elwasif
oscar@ornl.gov
elwasifwr@ornl.gov
Oak Ridge National Laboratory
Oak Ridge, TN, USA

Abstract

The increasing complexity and power/energy demands of heterogeneous exascale systems, such as the Frontier supercomputer, present significant challenges for measuring and optimizing power consumption in applications. Current tools either lack the resolution to capture fine-grained power and energy measurements, fail to validate in-band measurements against out-of-band power sensors, or cannot integrate this information with application performance events in a scalable manner. This paper introduces a novel open-source performance toolkit that integrates extended PAPI components with Score-P plugins to enable in-band, fine-grained power and energy measurements, while also supporting validation using power meter measurements for both CPUs and GPUs. One key contribution is the ability to perform millisecond-level power and energy measurements for AMD MI250X GPUs, mapping them to application performance events within a single trace and measurement system that scales. Our toolkit combines coarse-grained measurements from `cray_pm` counters with high-resolution metrics from `rocm_smi` and `RAPL`, converting GPU instantaneous accumulated energy into power to capture both transient and steady-state power behavior, a capability often missed by out-of-band and monitoring tools. By mapping these metrics to specific application regions, developers can identify energy hotspots, address inefficiencies in GPU kernel execution, and validate in-band measurements against external measurements. We demonstrate the effectiveness of this approach through case studies using benchmarks such as GPU `rocblas_sgemm`, BLIS `c_blas_dgemm`, and rocHPL, highlighting the variability of the measurements and the impact of transient power spikes on kernel-level efficiency.

CCS Concepts

• **Computing methodologies → Simulation evaluation**; **Parallel programming languages**; • **Computer systems organization → Parallel architectures**.

Keywords

Power and energy measurements, HPC applications, Performance Tools, AMD GPUs, Exascale computing systems

ACM Reference Format:
Oscar Hernandez and Wael Elwasif. 2025. Fine-Grained Application Energy and Power Measurements on the Frontier Exascale System. In *Proceedings of Cray User Group 2025 (CUG '25)*. ACM, New York, NY, USA, 12 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

With exascale HPC systems like Frontier pushing the envelope of parallel performance, coupled with the slowdown of Moore's law, measuring and managing power consumption becomes a key challenge for sustainability and energy efficiency. The current exascale systems, such as Frontier, are based on heterogeneous nodes with multiple GPUs and a CPU, each exposing power and energy data through a variety of sources and interfaces. As a result, integrating and interpreting this data in a meaningful and low-overhead way presents both a technical challenge and an opportunity for fine-grained application optimizations. The problem is further exacerbated by hardware variability across thousands of nodes and accelerators, where some GPUs may perform slower or consume more power than others. This is also influenced by how applications are mapped to these compute resources, as studied on large-scale HPC systems [30], highlighting the need to capture the energy and power signatures of applications at scale and to relate this information to their performance characteristics.

Historically, power measurements have been categorized based on their source as either in-band or out-of-band. In-band measuring techniques, such as Running Average Power Limit (RAPL) on x86 processors or `rocm_smi` on AMD GPUs, expose energy or power counters directly to the application through APIs that read driver-level monitoring files or model-specific hardware registers (MSR). These sources offer high-resolution measurements but require careful validation of their data quality and offer low- or mid-overhead access to avoid interfering with application execution. Validating the accuracy of their information can be challenging, as some methods rely on models, approximations, or running averages, with limited documentation on how these values are generated. Out-of-band methods, such as motherboard-level or system controller power sensors, provide external, non-intrusive measurements, but

CUG '25, New York City, NY
2025. ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM
https://doi.org/XXXXXXX.XXXXXXX

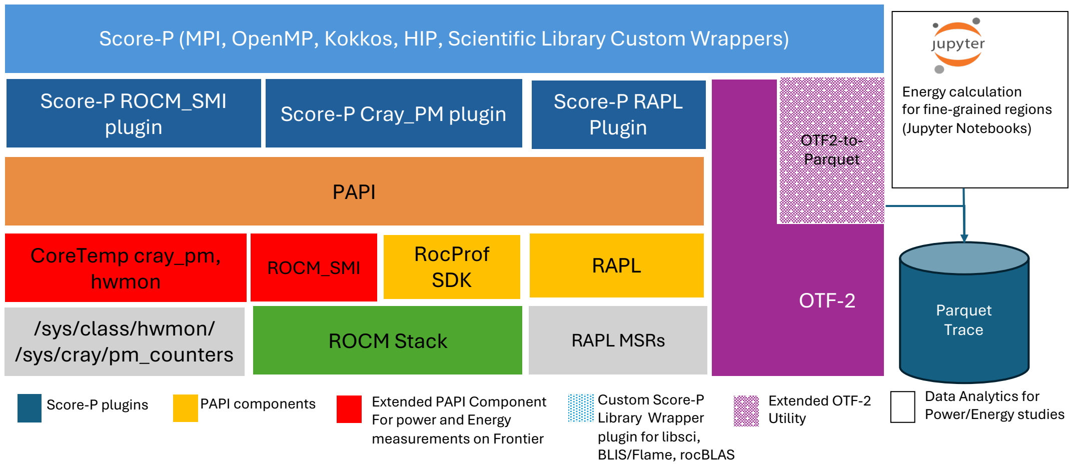

Score-P (MPI, OpenMP, Kokkos, HIP, Scientific Library Custom Wrappers)
Score-P ROCM_SMI plugin
Score-P Cray_PM plugin
Score-P RAPL Plugin
PAPI
CoreTemp cray_pm, hwmon
ROCM_SMI
RocProf SDK
RAPL
/sys/class/hwmon/ /sys/cray/pm_counters
ROCM Stack
RAPL MSRs
OTF2-to-Parquet
OTF-2
jupyter
Energy calculation for fine-grained regions (Jupyter Notebooks)
Parquet Trace
Score-P plugins
PAPI components
Extended PAPI Component For power and Energy measurements on Frontier
Custom Score-P Library Wrapper plugin for libsci, BLIS/Flame, rocBLAS
Extended OTF-2 Utility
Data Analytics for Power/Energy studies

Figure 1: Software stack diagram illustrating the integration of Score-P its plugins for PAPI, extended `cray_pm`, `hwmon` and `ROCM_SMI` PAPI component, the OTF2-to-Parquet conversion utility for fine-grained power measurements and energy estimation on Frontier

they are typically accessible only to system administrators through monitoring tools or databases, and often offer limited temporal resolution. These limitations have made it difficult to validate in-band readings or correlate them accurately with application performance events.

Younge e.t al [34] introduced a taxonomy that classifies power measurement tools into four levels. Level 1 includes job-wide aggregate data, Level 2 focuses on periodic sampling, Level 3 supports application-level instrumentation, and Level 4 enables multi-level correlation of data across in-band and out-of-band sources. This taxonomy has helped the community understand the landscape of available tools and their trade-offs between resolution, overhead, accessibility, integration across different information sources and tools, and their level of mapping to application performance events.

Level 1 and 2 monitoring tools analyze application workloads by treating applications, or groups of applications, as black boxes, reporting data primarily at the job or application level. This coarse-grained reporting lacks the detail needed for fine-grained performance analysis and may miss key application events due to undersampling or the quality of the power information sources, which can lead to overestimation or underestimation of energy measurements [33]. In contrast, performance tools that rely on instrumentation can map events to specific code regions at a higher cost, but they often lack access to sensors needed to capture accurate, instantaneous power or energy measurements. Instead, they rely on in-band model-based estimations or running power averages, which may not provide high-quality data or reliable estimates. Out-of-band approaches may offer better measurement accuracy but are more expensive, more difficult to access, and often lack the resolution needed to correlate their information with fine-grained application performance events at scale.

Recent advances in supercomputer design over the past decade have made data that was once available only through out-of-band channels directly accessible on compute nodes in-band via portable or standardized interfaces, for example, through PAPI or kernel-exposed sysfs filesystems such as the cray_pm counters [4, 11, 20], the generic hwmon class [23], and related mechanisms. This new capability eliminates the need for privileged access and enables applications to correlate energy and power usage with in-band methods, as well as to application performance events more directly through the use of performance tools. In addition, recent advances in plugin interfaces [26], combined with asynchronous sampling capabilities in direct measurement tools that integrate both sampling and instrumentation, enable the collection of power and energy metrics via independent threads. This approach reduces the overhead of in-band measurements performed by application threads, allowing high-frequency data collection without significantly perturbing execution, while still providing access to direct timing measurements within application performance events. These changes in the measurement system's capabilities enable the types of multi-level insight described by Younge to be captured within a single, unified toolchain and trace format, using a consistent clock and timestamping mechanism to avoid discrepancies that could arise from separately collected data with differing resolutions or timing sources. This also supports varying levels of granularity and information quality and allows measurements to be cross-referenced with the performance characteristics of applications.

In this paper, we present a novel toolkit that addresses these challenges and builds on the success of Score-P. It includes an improved ROCM_SMI PAPI component for GPU power metrics, an open-source cray_pm PAPI component that can be used alongside the RAPL component of PAPI for Trento CPUs, as well as Score-P and a set of custom plugins, enabling fine-grained millisecond-level power measurement for both CPUs and multi-GPU nodes. This fine-grained measurement is complemented by coarser-grain measurements using `cray_pm` counters, which provide power consumption data at a resolution of 10 Hz (100 milliseconds). This dual-level measurement approach allows us to capture detailed application-specific behavior while simultaneously comparing model-based in-band measurements with node-wide power and energy consumption data obtained from power sensors. By unifying these measurements and mapping them to application performance events, our toolset bridges the gap between localized performance bottlenecks, such as kernel-level inefficiencies, and broader system-level energy impacts. It enables developers to pinpoint potential energy bottlenecks, map high-resolution power measurements to application phases, and maintain scalability while accounting for variability in hardware performance and power consumption across GPUs and nodes.

The main contributions of this paper are as follows:

- Development of Extended PAPI Components: We extended PAPI to support ROCM_SMI, hwmon, and cray_pm counters for accessing fine-grained power and energy metrics, enabling detailed analysis of application behavior on Frontier's MI250X GPUs. Note: Cray PAPI provides access to cray_pm counters, but it is closed source, so we re-implemented it with the intent of integrating it with our ROCM_SMI and hwmon power components.
- We exposed GPU in-band energy information using `rsmi_dev_energy_count_get()` in our extended ROCM_SMI PAPI component, with the goal of providing access to energy data as well as enabling a more accurate estimation of instantaneous power on the MI-250X GPUs. This allows for precise mapping of power data to fine-grained GPU kernels. The current in-band method provided by ROCM_SMI, based on `rsmi_dev_power_ave_get()`, reports a running average of power rather than instantaneous values.
- Integration with Score-P: Our toolkit integrates our modified version of PAPI with Score-P plugins, allowing developers to trace power and energy consumption at different levels of resolutions for fine-grain regions of code that works with multiple GPUs (e.g. 4x AMD MI250X)
- We demonstrate the use of the toolset on several benchmarks to measure power and energy on both CPUs and GPUs, including SGEMM, DGEMM, and HPL.

We evaluated our toolkit across multiple workloads, showcasing its ability to resolve power and energy consumption at fine granularity, scale measurements across large systems, enhance kernel performance analysis, and provide insights that pave the way for application-level power and energy optimizations in the exascale era.

2 Performance toolkit for Power and Energy Studies on Frontier

Figure 1 provides an overview of the toolkit used for power and energy studies on Frontier. The toolkit builds upon established open-source tools, extending or complementing their capabilities to address the need for both fine-grained and coarse-grained power and energy measurements. The PAPI CoreTemp component, extended with `cray_pm` counters, is used for coarse-grained measurements of power and energy with out-of-bound board measurements. For fine-grained measurements, the `ROCM_SMI` component captures detailed power and energy information specific to GPU compute devices. Additionally, the Score-P x86 RAPL component enables precise energy measurements at the CPU core level. To support detailed data analysis, the toolkit includes an OTF2-to-Parquet utility, which exports trace data for processing with Python programs, Jupyter notebooks, and the Parquet trace format, allowing the calculation of fine-grained energy regions for instantaneous power estimations.

For the remainder of this section, we briefly introduce each tool, explain the development of the extensions, and demonstrate their use in fine-grained power measurements and overall energy consumption estimation.

2.1 Cray Power Management (PM) Counters

The Cray PM Counters[1] system provides telemetry for blade-level and node-level energy and power data in high-performance computing systems. Node-level data is accessible in-band through special sysfs files located at `/sys/cray/pm_counters`, offering detailed metrics such as point-in-time power (Watts), accumulated energy (Joules), and CPU and memory-specific measurements. Metrics include cpu_power, memory_power, and optional accelerator data like accel_power and accel_energy. Each metric is time-stamped in nanoseconds. The counters also provide information on power caps, system startup, scanning rates, and versioning, facilitating detailed power management and optimization for workloads. On frontier, the scanning rate is 10 hz.

2.2 PAPI

The Performance Application Programming Interface (PAPI) [14] is a widely used portable framework for accessing hardware performance counters on diverse CPU and GPU architectures. PAPI provides a `rocm_smi` component that supports monitoring power consumption, power capping, temperature, PCIe activity, GPU memory usage, and GPU utilization. Our work extends the `rocm_smi` component to expose GPU energy readings from `rocm_smi` at 1 millisecond resolution and updated this component to make it work with infinity interconnect with MI250X GPUs. Our work also extends PAPI's CoreTemp [31] component by integrating power and energy data from `cray_pm` counters, as well as power information from the Linux `hwmon` interface on Frontier. This PAPI extension enables interoperability with higher-level tracing and profiling tools such as CrayPat, TAU, and Score-P.

Score-P [16] is a scalable and open source performance measurement tool designed to provide profiling, event trace recording, and online analysis of HPC applications. It uses the *Open Trace Format 2* (OTF-2) [9] and supports multiple parallel programming paradigms, including MPI, OpenMP, CUDA, HIP, and SHMEM. Score-P can be used in conjunction with high-level visual tools like Vampir [32], CUBE [24], Scalasca [12], and TAU [28].

A key strength of Score-P is its plug-in infrastructure, which enables integration with external metric monitoring sources available via PAPI components or vendor-specific APIs. This functionality is leveraged in the `Async PAPI` (APAPI) plug-in, which we used to asynchronously sample hardware counters using an additional monitoring thread while minimizing application measurement overhead. The scope of the plugins can be configured to sample the metrics at

[1]https://cray-hpe.github.io/docs-csm/en-10/operations/power_management/user_access_to_compute_node_power_data/

Table 1: Comparison of ROCM_SMI and Cray PM for power and energy Monitoring on AMD MI250X

Feature	`rsmi_dev_energy_count_get()`	`rsmi_dev_power_ave_get()`	`cray_pm accelX_power/energy`
Scope	MI250X GPU	MI250X GPU	MI250X GPU (Blade Controller)
Sampling Interval	1 millisecond	1 millisecond	100 millisecond
Units	micro Joules	micro Watts	Watts / Joules
Granularity	High-resolution energy counter	Averaged power over time	Mid-resolution counters
Accessibility	`rocm_smi`	`rocm_smi`	/sys/cray/pm_counters
Sampling rate	Fixed Internal	Fixed interval	Fixed interval
Overhead	Sysfs access	Sysfs access	Sysfs access
Use Case	Fine-grained event correlation	Coarse-grained event correlation	Coarse-grained monitoring

the thread, process, node or application level. We developed a customized Score-P plugins to collect power and energy information via `cray_pm`, `rocm_smi`, and `RAPL` at the node level. This enables mapping these metrics to specific regions of the application code, as well as to different compute devices such as GPUs.

2.3 Estimating Instantaneous power utilization of AMD MI250x GPUs via energy readings

There are several ways to measure the power and energy of the MI250X GPUs using PM Counters and `rocm_smi` calls. Table 1 compares the different metrics, units, and resolutions for power and energy measurements. The goal is to understand how these power and energy metrics can be utilized for fine-grained regions of an application. The table shows that PM Counters provide coarse-grained measurements for both power and energy with a granularity of 100 milliseconds, whereas `rocm_smi` offers measurements at a granularity of 1 millisecond. However, the power metric provided by `rocm_smi` is based on an average, which may use a running average over 1-second time windows. This can affect the accuracy of power measurements for transient power spikes generated by fine-grained regions. As a result, we explored the idea of using the energy counter metric to estimate instantaneous power for fine-grained regions of code.

Existing tools for performance and power analysis, such as Omnitrace and rocProfiler, rely on `rocm_smi` GPU average power consumption data. Although this approach effectively smooths out short-term power fluctuations, it limits the ability to capture transient power peaks that are critical for understanding fine-grained power and energy behavior in application code regions. This limitation complicates the precise mapping of power usage and energy estimations to specific computational phases, potentially obscuring insights into energy efficiency and application-level optimizations.

Instantaneous power can be estimated with the accumulated energy readings measured from the `rsmi_dev_energy_count_get()` energy counter metric using the following formula:

$$\text{Power}_{\text{inst}}(i) \approx \frac{E(i) - E(i-1)}{\Delta t}$$

where $E(i)$ is the energy reading at time step i, and Δt is the time interval between successive readings (e.g., 1 millisecond). This estimation is aimed at capturing more transient power spikes that can be correlated with individual GPU kernel executions.

2.4 Running Average Power Limit (RAPL)

On Frontier, in-band energy measurements can be performed using RAPL (Running Average Power Limit), a model-based power estimation mechanism available on x86 Intel and AMD processors. For AMD architectures like Trento, which is used in Frontier, RAPL provides access to Model-Specific Registers (MSRs) that report CPU package-level power consumption and core-level metrics. This model estimates power based on internal performance counters and environment sensors such as critical path monitors, high-speed voltage droop detectors, and thermal diodes, rather than direct physical measurements [25].

RAPL supports high-resolution sampling (on the order of 1 ms) and allows researchers to map energy readings to fine-grained regions of code. However, these values are estimates, not direct measurements, and studies have shown that the model can miss DRAM power (only supported on Intel architectures), introduce data-dependent inaccuracies, and show inconsistent behavior under different core utilizations and frequencies, particularly when multiple cores are active or mixed-frequency configurations are used [21, 25]. Therefore, while useful for observing relative power trends and behavioral tracing, RAPL should not be relied upon for precise system-level energy accounting.

To complement RAPL, Frontier also provides the Cray Power Management (cray_pm) interface, which uses power sensors to measure the CPU socket and memory DRAM power. These sensor readings offer true physical values at a coarser granularity (100 ms), enabling cross-correlation with RAPL data to study power and energy fine-grain trends at the application level performance events. Previous studies have investigated the validation of RAPL measurements by comparing them with data from external power meters [21]. Additionally, advanced AMD CPUs such as Trento expose per-core energy estimates that can be accessed via the `hwmon` sysfs interface, but doing so requires elevated user privileges; therefore, we did not include them in our study.

We intentionally initialized the arrays with Thread 0 to observe CPU and memory power consumption during single-threaded execution. As a side effect, this caused poor memory placement due to the first-touch policy, resulting in the memory being bound to the NUMA domain of Thread 0. Consequently, other thread groups experience remote memory accesses, creating an imbalance where threads spend excessive time at the implicit barrier in a busy-wait state without performing useful computation.

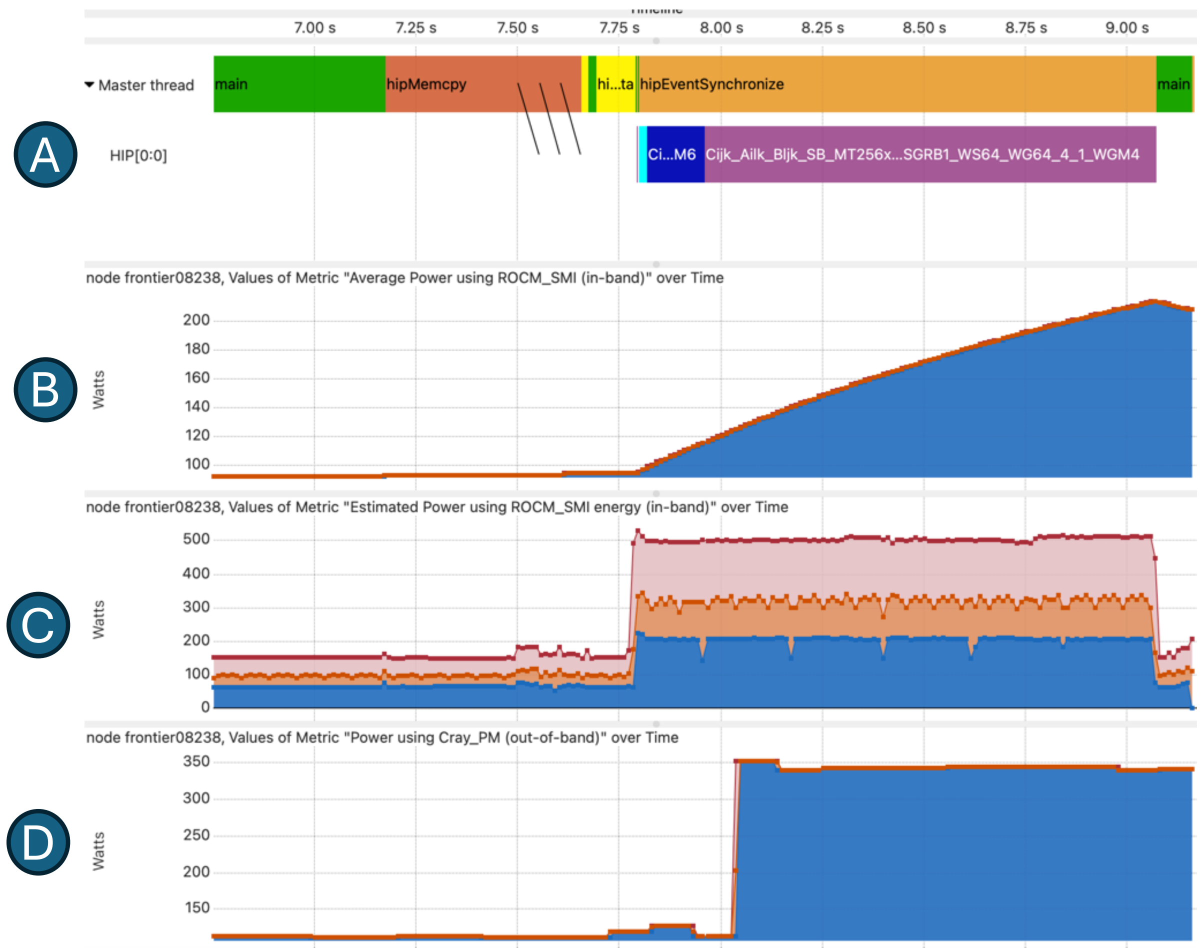


Figure 2: GPU power measurements of a `rocblas_sgemm` operation with matrix ranks ranging from 2 to 16,384, running 5 times each, were obtained using different types of GPU power measurement tools available on Frontier's MI250X via `rocm_smi` (Trace B) and `cray_pm` (Trace D) counters. Trace A shows the benchmark's regions of operation, including the initialization of matrices on the CPU, their transfer to GPU memory, and the GPU kernels executing the `rocblas_sgemm`. B: Displays the instantaneous power estimation calculated from the energy counter metric by dividing it over time.

Trace B presents the call stack view of the program's execution, highlighting the distinct regions traversed during runtime. Traces C and D display the power values reported by `cray_pm` for CPU and DRAM memory, respectively. When Thread 0 initializes the array, variability in memory power is observed. As the DGEMM operations begin across all threads, both CPU and memory power consumption increase. This provides an estimate of the average CPU socket power utilization at approximately 150 Watts.

Trace E shows the power estimation based on RAPL `Package0` energy values. While RAPL significantly overestimates power consumption, it accurately captures the same power trends observed with `cray_pm` CPU power. Trace F presents the power values from RAPL `PP0`, which reflect energy consumed by the CPU cores, excluding uncore components and memory controller power, which are measured in the RAPL `Package0` power counter. This view is particularly insightful, showing that power consumption aligns with core activity during DGEMM computation and drops when threads are stalled at the implicit barrier. This drop in power consumption by the cores during the barrier cannot be clearly distinguished at the package (CPU + memory) or socket (CPU) level, as seen in Traces C to E.

3 Evaluation

3.1 Target System

The toolkit was configured and adapted to work on the Frontier supercomputer at the Oak Ridge Leadership Computing Facility (OLCF) [22]. Frontier is composed of 9,856 HPE Cray EX nodes, each with one AMD EPYC 7A53 "Trento" CPU (64 cores) and four AMD MI250X GPUs. The MI250X GPUs are connected through AMD's Infinity Fabric, and each GPU is partitioned into two Graphic Compute Dies (GCDs), with each GCD treated as a stand-alone device with 64 GB of HBM2e memory. The node interconnect is based on HPE's Slingshot-11 network. The system runs the HPE

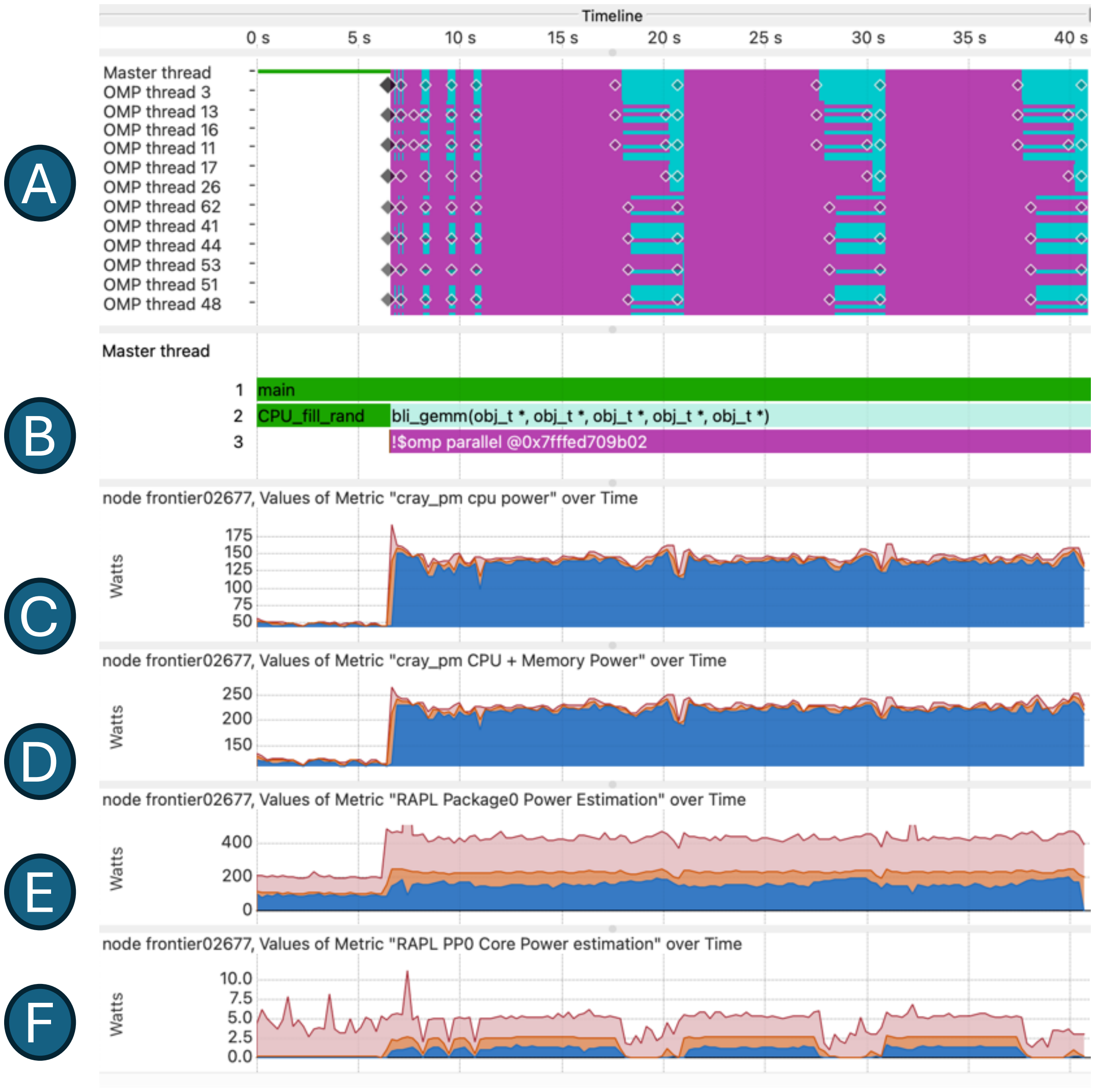


Figure 3: The `blis_dgemm` operation was executed with matrix ranks ranging from 2 to 16,384, each run three times, with power measurements obtained using both RAPL and the cray_pm CPU and memory power interfaces available on Frontier's nodes. Trace A illustrates the benchmark timeline across 64 threads, highlighting the function in which the threads spend most of their time. The green segment corresponds to a function where a single thread initializes and allocates arrays for the DGEMM operation. The dark blue represents an OpenMP parallel region, while the cyan indicates an implicit barrier within that region. Trace B presents the call stack view of the program's execution. Traces C and D display the power values reported by cray_pm for the CPU and CPU plus DRAM memory, respectively. Trace E shows the power estimation based on RAPL Package0 energy values, and Trace F presents the power values from RAPL PP0, which reflect the energy consumed by the CPU cores(s).

Cray Programming Environment, including the Cray MPICH MPI stack and ROCm software stack for GPU acceleration.

3.2 Instantaneous Power estimation

Figure 2 presents GPU power measurements for a `rocblas_sgemm` operation executed with matrix ranks ranging from 2 to 16,384, each run five times. Power data was collected using multiple sources on Frontier's MI250X: `rocm_smi` average power (Trace B), `cray_pm`

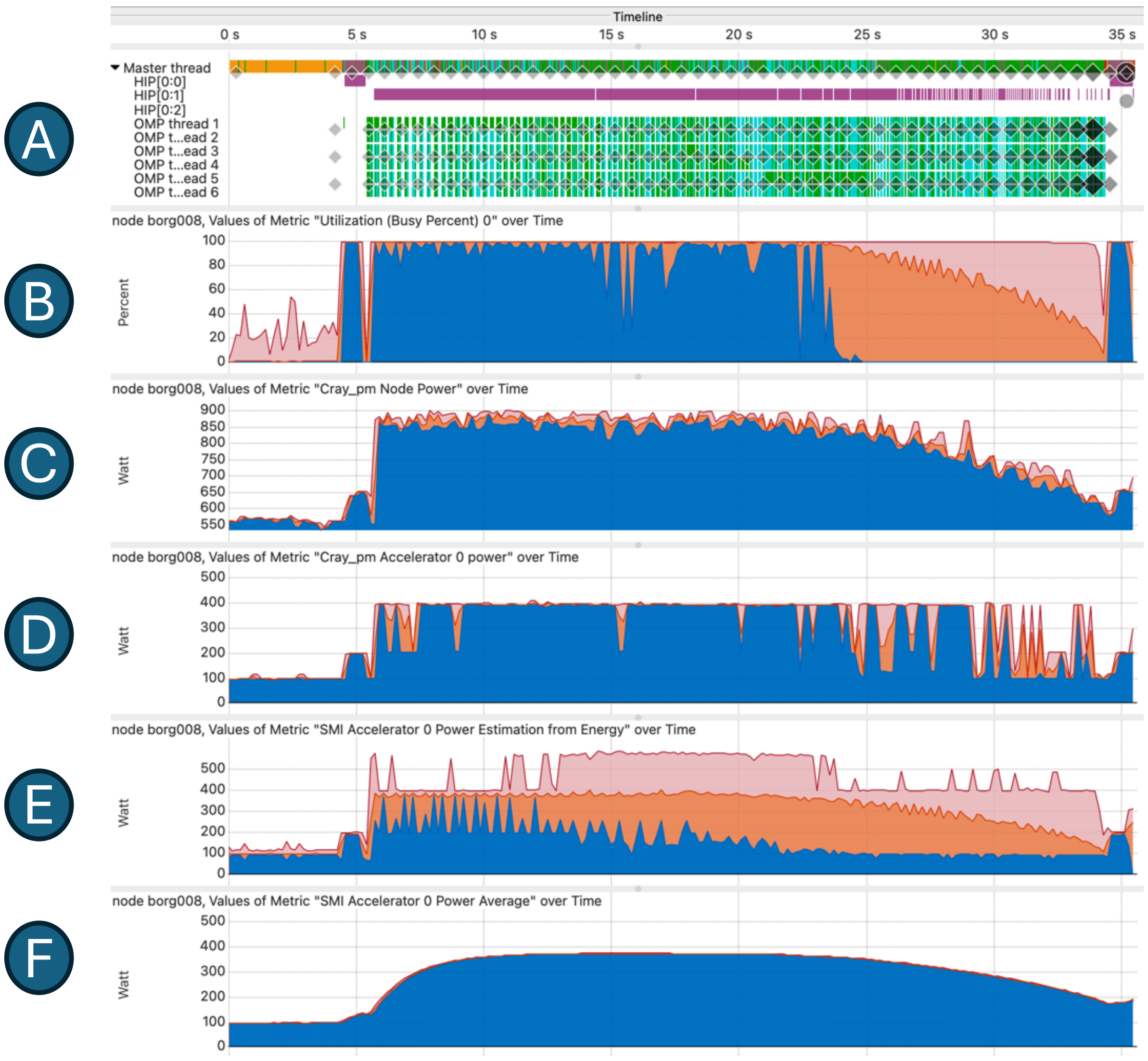


Figure 4: Power traces of the ROCm-enabled HPL benchmark running with one MPI rank, 56 OpenMP threads, and one GCD, showing the execution timeline and power measurements from both CPU and GPU components. Trace A shows the program execution timeline, including an initialization phase and a compute-intensive phase involving CPU threads and GPU kernel launches (HIP[0:0–2]). Trace B presents the GPU utilization as a percentage over time. Trace C shows the total node-level power from `cray_pm`, combining CPU and GPU power. Trace D displays GPU socket (Accelerator 0) power from `cray_pm`. Trace E shows instantaneous GPU power estimated from `rocm_smi` energy counters. Trace F presents the average GPU power as reported by `rocm_smi`. Color in Traces C–F indicates power range: blue for minimum, dark red for average, and light red for maximum power values per pixel.

counters (Trace D), and our custom instantaneous power approximation (Trace C), which we derived from GPU energy values returned by `smi_dev_energy_count_get()`. In Traces B, C, and D, color is used to indicate the power range: blue represents the minimum observed power, dark red indicates the average, and light red corresponds to the maximum power value. These colorings reflect sample summarization at the pixel level in the Vampir trace visualization tool. For the application context, Trace A highlights key application phases, including matrix initialization on the CPU, memory copies and data transfers to GPU memory, and the execution of GPU kernels performing the `rocblas_sgemm` computations.

The `rocm_smi` average power measurements rely on a running average that gradually increases over time, as observed between seconds 7.75 and 9 of the timeline, before stabilizing when executing the matrix of size 16,384. This averaging method fails to capture the power consumption of kernels processing smaller matrix sizes, impacting the ability to map power consumption to fine-grained GPU kernel regions. In contrast, the `cray_pm` power measurement in Trace D shows a power spike when GPU kernel execution begins; however, its sampling resolution is insufficient to capture kernels with durations below 0.1 seconds.

Trace C shows the instantaneous power estimate calculated using the method described in Section2.3. As seen in this trace, we observe the detection of transient power spikes at a finer granularity. However, this method introduces higher apparent spikes, as evidenced by the dark red regions in the trace, which display the maximum values per pixel.

3.3 RAPL evaluation

Figure 3 shows the execution of the `blis_dgemm` operation with matrix ranks ranging from 2 to 16,384, each run three times, with power measurements obtained using both RAPL and the `cray_pm` CPU and memory power interfaces, accessed via our extended PAPI implementation available on Frontier's nodes. In Traces C-F, color is used to indicate the power range: blue represents the minimum observed power, dark red indicates the average, and light red corresponds to the maximum power value. These colorings reflect sample summarization at the pixel level in the Vampir trace visualization tool.

Trace A illustrates the benchmark timeline across 64 threads. Initially, a single thread allocates and initializes the arrays, then populates them with random values using `cpu_fill_rand()`, followed by the execution of DGEMM operations of varying sizes using all threads. The green segment corresponds to the function where the arrays are allocated and initialized. The dark blue represents the OpenMP parallel region, while the cyan indicates an implicit barrier within that region.

3.4 ROCm HPL Evaluation

For our evaluation, we used the open-source ROCm-enabled HPL benchmark[2], an implementation of the High-Performance Linpack (HPL) algorithm which is optimized for AMD Instinct GPUs. The HPL experiments were carried out on borg, an internal OLCF cluster that has a configuration identical node and network configuration to Frontier. Our setup allows performance and power measurements to be taken using both in-band model-based and power sensors methods, as described in previous sections. Through this configuration, we evaluate the effectiveness of our fine-grained power instrumentation in capturing both GPU and CPU power trends during different phases of the HPL run, and compare the different power metrics across multiple GPUs.

Figure 4 shows the Vampir trace of the ROCm-enabled HPL benchmark running with one MPI rank, 7 OpenMP threads, and one GCD (Graphics Compute Die). This figure presents both the execution timeline and power measurements from various sources, providing a comprehensive view of CPU and GPU activity.

Trace A displays the execution timeline of the application, beginning with an initialization phase followed by a compute-intensive phase. This is evidenced by the sustained activity of the master and OpenMP team threads and the HIP[0:0–2] traces, which represent GPU kernel launches on the GCD. Trace B shows the GPU utilization as a percentage over time, where a sharp rise in usage aligns with the start of the compute phase, reflecting intensive DGEMM operations offloaded to the GPU.

Traces C through F present power measurements from both power sensors and in-band models, each using color to represent minimum (blue), average (dark red), and maximum (light red) power values at each pixel resolution in the Vampir tool. Trace C shows the total node-level power reported by the cray_pm interface, including both CPU and GPU components. This trace demonstrates a strong correlation with the overall compute phase, peaking around 850 watts. Trace D focuses specifically on the GPU socket (Accelerator 0) power from cray_pm, showing stable readings near 400 watts during active GPU execution and greater variability as the GPU becomes underutilized.

Trace E presents an instantaneous GPU power estimation computed from rocm_smi energy counters, using the formula described in Section 2.3. This trace captures fine-grained power fluctuations, showing consistent power levels during kernel activity and increasing variability, ranging from 100 to 450 watts, as the GPU transitions to lower utilization. Trace F shows the smoothed average GPU power as reported by rocm_smi, providing a more stable but lower resolution view of power behavior.

These traces highlight the importance of using multiple power monitoring sources for comprehensive analysis. Although averaged metrics like those in Trace F offer stability, they can obscure important short-term dynamics that are better captured by the high-resolution instantaneous estimation in Trace E. Meanwhile, cray_pm measurements (Traces C and D) serve as reliable references of actual power measurements, offering validated, coarser-grain power values that help contextualize and confirm trends observed in in-band readings. Notably, the average power reported by rocm_smi, our instantanous power estimation and the GPU socket power sensor from cray_pm both center around 400 watts during the active computation phase, reinforcing their mutual consistency.

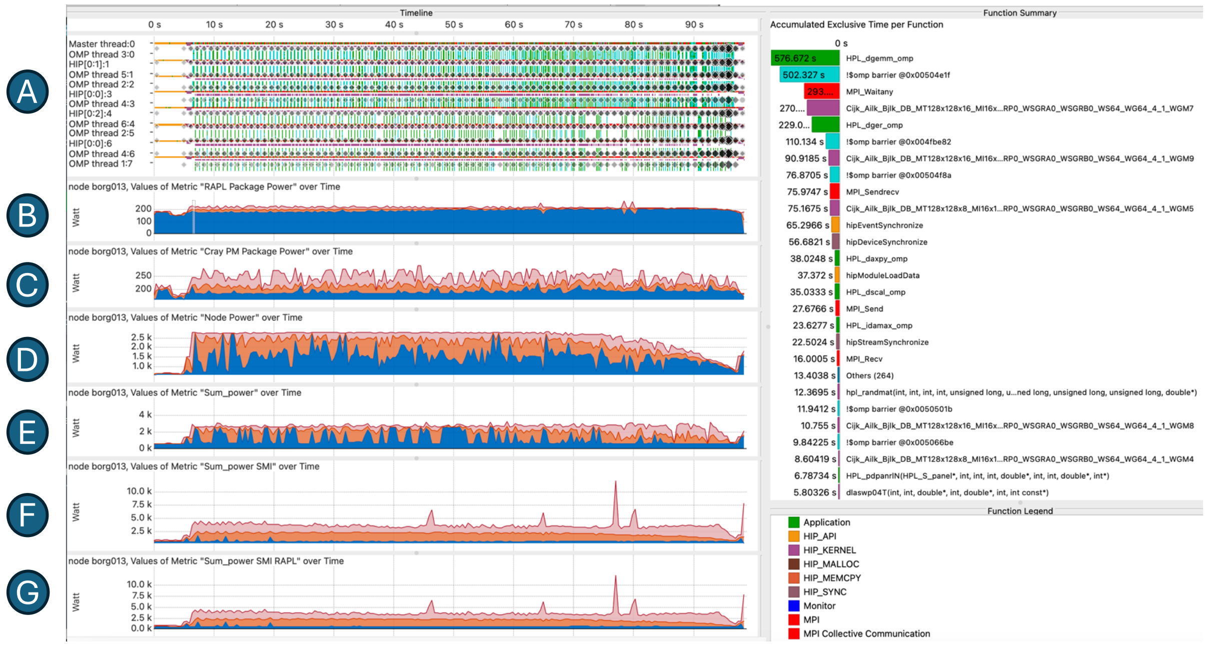
Overall, the correlation between GPU utilization (Trace B) and power consumption (Traces D–F) demonstrates the effectiveness of using fine-grained power data to track and analyze the phases of high-performance computing workloads and how these fine grain measurements can be used to attribute power consumption to different kernels in the code. This information is valuable for tuning applications and informing energy-aware runtime strategies.

Figure 5 extends our HPL evaluation to a larger-scale configuration of the ROCm-enabled HPL benchmark running with eight MPI ranks and 56 OpenMP threads across eight GCDs, distributed on four MI250X GPUs. The trace captures synchronized CPU and GPU activity as well as comprehensive power metrics collected from both in-band and power sensors sources.

Trace A shows the application timeline where each MPI rank offloads compute-intensive regions to the GPUs using HIP. Distinct HIP regions (e.g., HIP[0:*]) indicate parallel GPU kernel launches, revealing high concurrency and sustained activity across all GPUs during the DGEMM phases.

The function summary on the right confirms that the majority of the exclusive execution time is spent in dense linear algebra routines, consistent with the expected behavior of HPL. The color-coded legend maps each region to either CPU, GPU, or communication categories, helping identify how execution phases align with power usage patterns.

[2]https://github.com/ROCm/rocHPL


A
B
C
D
E
F
G
Timeline
0 s
10 s
20 s
30 s
40 s
50 s
60 s
70 s
80 s
90 s
Master thread:0
OMP thread 3:0
HIP[0:1]:1
OMP thread 5:1
OMP thread 2:2
HIP[0:0]:3
OMP thread 4:3
HIP[0:2]:4
OMP thread 6:4
OMP thread 2:5
HIP[0:0]:6
OMP thread 4:6
OMP thread 1:7
node borg013, Values of Metric "RAPL Package Power" over Time
node borg013, Values of Metric "Cray PM Package Power" over Time
node borg013, Values of Metric "Node Power" over Time
node borg013, Values of Metric "Sum_power" over Time
node borg013, Values of Metric "Sum_power SMI" over Time
node borg013, Values of Metric "Sum_power SMI RAPL" over Time
Watt
200
100
0
250
200
2.5 k
2.0 k
1.5 k
1.0 k
4 k
2 k
0 k
10.0 k
7.5 k
5.0 k
2.5 k
0.0 k
Function Summary
Accumulated Exclusive Time per Function
0 s
Function Legend

576.672 s	HPL_dgemm_omp
502.327 s	!$omp barrier @0x00504e1f
293....	MPI_Waitany
270....	Cijk_Ailk_Bjlk_DB_MT128x128x16_MI16x...RP0_WSGRA0_WSGRB0_WS64_WG64_4_1_WGM7
229.0...	HPL_dger_omp
110.134 s	!$omp barrier @0x004fbe82
90.9185 s	Cijk_Ailk_Bjlk_DB_MT128x128x16_MI16x...RP0_WSGRA0_WSGRB0_WS64_WG64_4_1_WGM9
76.8705 s	!$omp barrier @0x00504f8a
75.9747 s	MPI_Sendrecv
75.1675 s	Cijk_Ailk_Bjlk_DB_MT128x128x8_MI16x1...RP0_WSGRA0_WSGRB0_WS64_WG64_4_1_WGM5
65.2966 s	hipEventSynchronize
56.6821 s	hipDeviceSynchronize
38.0248 s	HPL_daxpy_omp
37.372 s	hipModuleLoadData
35.0333 s	HPL_dscal_omp
27.6766 s	MPI_Send
23.6277 s	HPL_idamax_omp
22.5024 s	hipStreamSynchronize
16.0005 s	MPI_Recv
13.4038 s	Others (264)
12.3695 s	hpl_randmat(int, int, int, int, unsigned long, u...ned long, unsigned long, unsigned long, double*)
11.9412 s	!$omp barrier @0x0050501b
10.755 s	Cijk_Ailk_Bjlk_DB_MT128x128x16_MI16x...RP0_WSGRA0_WSGRB0_WS64_WG64_4_1_WGM8
9.84225 s	!$omp barrier @0x005066be
8.60419 s	Cijk_Ailk_Bjlk_DB_MT128x128x8_MI16x1...RP0_WSGRA0_WSGRB0_WS64_WG64_4_1_WGM4
6.78734 s	HPL_pdpanrlN(HPL_S_panel*, int, int, int, double*, int, int, double*, int*)
5.80326 s	dlaswp04T(int, int, double*, int, double*, int, int const*)

🟩 Application
🟧 HIP_API
🟪 HIP_KERNEL
🟫 HIP_MALLOC
🟧 HIP_MEMCPY
🟪 HIP_SYNC
🟦 Monitor
🟥 MPI
🟥 MPI Collective Communication

Figure 5: Execution and power traces of the ROCm-enabled HPL benchmark running with eight MPI ranks, each using seven OpenMP threads (56 threads total), across eight GCDs on four MI250X GPUs. Trace A shows the program execution timeline, highlighting an initialization phase followed by a compute-intensive phase involving both CPU threads and GPU kernel activity (indicated by HIP[0:*]:* regions). The function summary on the right displays the accumulated exclusive time per function and provides a legend of application components. Trace B displays CPU power measured via the RAPL Package 0 counter. Trace C shows CPU package power from `cray_pm`, which includes both CPU and memory components. Trace D reports total node power from `cray_pm`. Trace E presents the combined power from four GPU accelerators, CPU, and memory using `cray_pm` (power sensors measurements). Trace F displays the sum of GPU power (estimated instantaneously from `rocm_smi` energy counters) along with CPU and memory power from `cray_pm` (mix of model-based in-band and power sensors measurements). Trace G shows a fully in-band measurement of combined GPU power (via `rocm_smi` energy counters) and CPU power (via RAPL). For Traces B–G, color represents power ranges at each pixel: blue for minimum, dark red for average, and light red for maximum power values.

Power measurements from Trace B (RAPL) and Trace C (`cray_pm` CPU package) provide a view of CPU-level consumption. Trace C in particular integrates memory and core power, showing more consistent power values across threads and MPI processes than RAPL alone, which has finer resolution but less variability.

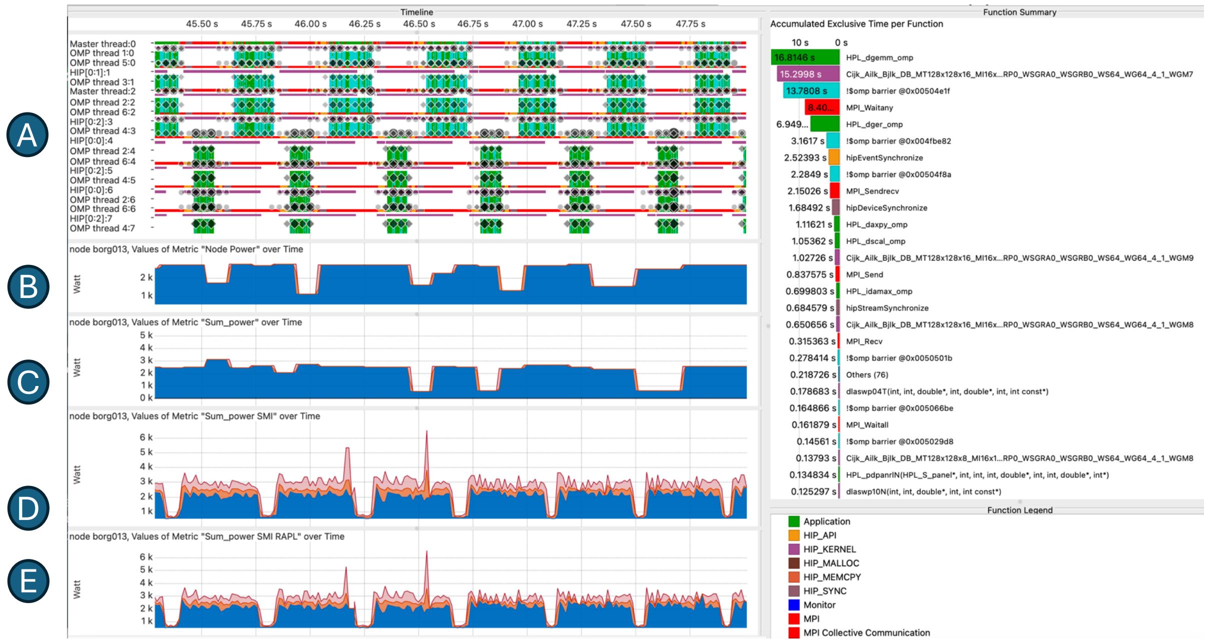
Trace D shows the total node power from `cray_pm`, integrating all components on the node (e.g., CPU, DRAM memory, network cards, NVMe, and GPUs). It peaks during compute-intensive phases and stabilizes during communication and idle periods. Trace E presents the sum of the power measured for the four GPUs, CPU, and DRAM memory using `cray_pm`, highlighting how the aggregated component-level power sensors compares to the total node power reported by `cray_pm`.

In contrast, Trace F uses high-resolution GPU power estimates derived from `rocm_smi` energy counters, combined with CPU and memory power from `cray_pm`. This hybrid method captures sharper transitions in power usage with some peaks in the estimation that overestimate power, better reflecting the run-time dynamics of GPU activity. Finally, Trace G limits the measurement exclusively to in-band methods—GPU instantaneous power from `rocm_smi` and CPU RAPL—offering a fully in-band view of power behavior.

The comparison across Traces E–G highlights the impact of measurement strategy on resolution and accuracy. Power sensors provide system-verified data useful for calibration and long-term averages coarser granularity, while in-band estimations allow fine-grained attribution of power to specific kernel regions. Together, they enable precise phase-level energy modeling.

When tracing multi-GPU workloads, hybrid metrics (like Trace F) are particularly valuable for identifying transient spikes or efficiency gaps across devices. Their values can be compared and estimated with the power sensor measurements for their accuracy. Moreover, the alignment between function execution (Trace A), kernel launches, and the resulting power fluctuations demonstrates how traces combined with diverse power sources can expose energy hotspots and load imbalances that can be mapped on a trace.

Figure 6 shows the same trace, but zoomed in to display approximately two seconds of execution in detail. Here, we can see that Traces D and E have sufficient fine-grain resolution to map power



16.8146 s	HPL_dgemm_omp
15.2998 s	Cijk_Ailk_Bjlk_DB_MT128x128x16_MI16x...RP0_WSGRA0_WSGRB0_WS64_WG64_4_1_WGM7
13.7808 s	!$omp barrier @0x00504e1f
8.40...	MPI_Waitany
6.949...	HPL_dger_omp
3.1617 s	!$omp barrier @0x004fbe82
2.52393 s	hipEventSynchronize
2.2849 s	!$omp barrier @0x00504f8a
2.15026 s	MPI_Sendrecv
1.68492 s	hipDeviceSynchronize
1.11621 s	HPL_daxpy_omp
1.05362 s	HPL_dscal_omp
1.02726 s	Cijk_Ailk_Bjlk_DB_MT128x128x16_MI16x...RP0_WSGRA0_WSGRB0_WS64_WG64_4_1_WGM9
0.837575 s	MPI_Send
0.699803 s	HPL_idamax_omp
0.684579 s	hipStreamSynchronize
0.650656 s	Cijk_Ailk_Bjlk_DB_MT128x128x16_MI16x...RP0_WSGRA0_WSGRB0_WS64_WG64_4_1_WGM8
0.315363 s	MPI_Recv
0.278414 s	!$omp barrier @0x0050501b
0.218726 s	Others (76)
0.178683 s	dlaswp04T(int, int, double*, int, double*, int, int const*)
0.164866 s	!$omp barrier @0x005066be
0.161879 s	MPI_Waitall
0.14561 s	!$omp barrier @0x005029d8
0.13793 s	Cijk_Ailk_Bjlk_DB_MT128x128x8_MI16x1...RP0_WSGRA0_WSGRB0_WS64_WG64_4_1_WGM8
0.134834 s	HPL_pdpanrlN(HPL_S_panel*, int, int, int, double*, int, int, double*, int*)
0.125297 s	dlaswp10N(int, int, double*, int, int const*)

🟩 Application
🟧 HIP_API
🟪 HIP_KERNEL
🟫 HIP_MALLOC
🟥 HIP_MEMCPY
🟫 HIP_SYNC
🟦 Monitor
🟥 MPI
🟥 MPI Collective Communication

Figure 6: Execution and power traces of the ROCm-enabled HPL benchmark running with eight MPI ranks, each using seven OpenMP threads (56 threads total), across eight GCDs on four MI250X GPUs. Trace A shows the program execution timeline, highlighting an initialization phase followed by a compute-intensive phase involving both CPU threads and GPU kernel activity (indicated by HIP[0:*]:* regions). The function summary on the right displays the accumulated exclusive time per function and provides a legend of application components. Trace B reports total node power from `cray_pm`. Trace C presents the combined power from four GPU accelerators, CPU, and memory using `cray_pm` (power sensors measurements). Trace D displays the sum of GPU power (estimated instantaneously from `rocm_smi` energy counters) along with CPU and memory power from `cray_pm` (mix of model-based in-band and power sensors measurements). Trace E shows a fully in-band measurement of combined GPU power (via `rocm_smi` energy counters) and CPU power (via RAPL). For Traces B–E, color represents power ranges at each pixel: blue for minimum, dark red for average, and light red for maximum power values.

activity to specific GPU kernels. However, some peaks may overestimate power. The `cray_pm` power sensor measurements lack the resolution to map power to individual kernels, but they are based on physical sensors and can be used to validate in-band power estimation methods.

Such fine-grain visibility is crucial for application tuning, exploring performance-energy trade-offs, and developing adaptive strategies to improve the energy efficiency of applications.

4 Related Work

Variorum [19] provides a portable and unified interface for monitoring and managing power usage across a wide range of hardware platforms, including x86, Arm, IBM Power, and NVIDIA GPUs. It supports multiple back-end APIs, such as RAPL [15], IBM Opal, `hwmon`, AMD's System Management Interface for EPYC processors, NVML, and the `rocm_smi` library. While PAPI provides a more general-purpose interface for accessing hardware counters, Variorum is focused on vendor-agnostic energy control, reflecting the increasing need for adaptable power tools in heterogeneous system environments. The Power Measurement Toolkit [5], developed for HPE Cray systems, utilizes the `pm_counters` interface to gather node-level power and energy data, including CPU and memory telemetry. This broader, system-level view complements application-centric tools by bridging low-level telemetry with performance metrics. LIKWID, a lightweight set of command-line utilities, provides access to performance and energy counters, and recent extensions have added GPU support [17].

Tools like TAU [27] and Score-P [16], together with its plug-in architecture [26], enable detailed runtime profiling and energy-aware tracing. The THAPI infrastructure [10] allows developers to intercept heterogeneous performance events and map them to specific program contexts, supporting optimization techniques that can leverage value-specific information. Vendor-supported solutions—including NVIDIA Nsight Systems [7], AMD Omnitrace [2], and Linaro MAP [1]—integrate closely with their respective hardware APIs (e.g., NVML, ROCM_SMI) to provide insights into both performance and power consumption, often down to the level of individual function calls or GPU kernel activity. CrayPat [20], built by HPE, integrates PAPI with system-specific telemetry to support balanced profiling and power analysis across applications.

Several frameworks focus on job-level and system-wide power tracking. For instance, Pika [8] connects with the SLURM job scheduler [29] to attribute energy use to workloads, enabling cluster-wide visibility into power consumption. NVIDIA's Data Center GPU

Manager (DCGM) [6] provides infrastructure for monitoring and managing multi-GPU and multinode deployments. MCBound [3] takes a step further by classifying workloads as memory-bound or compute-bound, guiding energy-efficient job placement and scheduling. These strategies contribute to the broader goal of intelligent, data-driven power management in large-scale HPC environments. Fine-grained optimization tools like the Dynamic Energy-Performance Optimizer (DEPO) [18] and READEX [13] dynamically adjust power limits using telemetry from RAPL and NVML, helping to strike a balance between energy savings and computational performance.

Our approach builds on top of PAPI and Score-P to provide a unified method for integrating the currently available sources of power and energy information on compute nodes in systems like Frontier. It ensures the necessary data quality and scalability to effectively relate fine-grained, region- or kernel-specific information for optimizing applications running at scale. Additionally, our contributions, built on PAPI, enable this information to be accessed by other tools that can read OTF2 traces, convert them into profiling data, or read PAPI counters directly.

5 Conclusions

This work presented a novel open-source toolkit that enables fine-grained, in-band power and energy measurements for AMD MI250X GPUs and Trento CPUs on the Frontier exascale system. By extending the PAPI framework to support ROCM_SMI energy counters, Cray PM interfaces, and integrating these with Score-P, we demonstrated the capability to capture millisecond-resolution power signatures that can be directly correlated with application behavior, including GPU kernels and CPU code regions. This toolkit supports both coarse-grained node-level telemetry and high-resolution in-band model based readings, while also allowing validation with power sensors.

Our experiments used BLIS DGEMM, rocBLAS SGEMM, and rocHPL to demonstrate the use of different metrics available on Frontier to estimate power utilization and related to the application timeline. First, estimating instantaneous GPU power from energy counters offers a significantly more accurate view of short-term power dynamics compared to the averaged power values typically provided by ROCM_SMI. Second, using multiple sources of power data, including RAPL, cray_pm, and ROCM_SMI, allows for a more comprehensive understanding of the power usage patterns across CPU and GPU components. Third, the memory placement policies in OpenMP applications that introduced issues with first-touch policy can introduce substantial energy overhead and imbalance, as demonstrated in our DGEMM experiments where the RAPL core counter was able to show the power inneficiencies of busy waiting but it was not shown in the other counters. These observations emphasize the importance of accurate and high-resolution power measurements for both understanding and optimizing application performance on heterogeneous systems.

The toolkit bridges the gap between application-level performance events and hardware-level power telemetry, enabling users to detect inefficiencies and energy hotspots. It is scalable and extensible, and its modular architecture enables integration with other tools that rely on OTF2 traces or direct counter readings.

As future work, although our tool supports the collection of power performance data across multiple GPUs and nodes, it remains challenging to effectively visualize and analyze such data at scale. We plan to enhance our analysis capabilities to better study multi-GPU behavior, fully exploit the AMD MI250X architecture, and validate the methodology at larger scales using real-world scientific applications running on Frontier.

Acknowledgments

This research used resources from the Oak Ridge Leadership Computing Facility, which is a US Department of Energy (DOE) Office of Science user facility supported under contract DE-AC05-00OR22725. This work is also supported by the DOE Office of Science, Advanced Scientific Computing Research, Express project "Leveraging Open-Source Simulators to Enable hw/sw Co-design of Next-generation HPC Systems" (DE-FOA-0002950).

References

[1] 2024. Linaro MAP: High-Performance Profiling and Analysis Tool. https://www.linaroforge.com/linaro-map/ Accessed: 2024-12-24.
[2] AMD. 2024. AMD Omnitrace Documentation. https://rocm.docs.amd.com/projects/omnitrace/en/latest/doxygen/html/index.html. https://rocm.docs.amd.com/projects/omnitrace/en/latest/doxygen/html/index.html Accessed: 2024-12-25.
[3] Francesco Antici, Andrea Bartolini, Zeynep Kiziltan, Ozalp Babaoglu, Yuetsu Kodama, et al. 2024. MCBound: An Online Framework to Characterize and Classify Memory/Compute-bound HPC Jobs. In *SC'24: Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis.* 1–15.
[4] Brian Austin and Nicholas J Wright. 2014. Measurement and interpretation of micro-benchmark and application energy use on the cray xc30. In *2014 Energy Efficient Supercomputing Workshop.* IEEE, 51–59.
[5] Stefano Corda, Bram Veenboer, and Emma Tolley. 2022. PMT: Power Measurement Toolkit. In *2022 IEEE/ACM International Workshop on HPC User Support Tools (HUST).* 44–47. doi:10.1109/HUST56722.2022.00011
[6] NVIDIA Corporation. 2024. NVIDIA Data Center GPU Manager (DCGM). https://developer.nvidia.com/dcgm. https://developer.nvidia.com/dcgm Accessed: 2024-12-25.
[7] NVIDIA Corporation. 2024. NVML Power and Temperature Metrics - Nsight Systems User Guide. https://docs.nvidia.com/nsight-systems/UserGuide/index.html#nvml-power-and-temperature-metrics-preview Accessed: 2024-12-21.
[8] Robert Dietrich, Frank Winkler, Andreas Knüpfer, and Wolfgang Nagel. 2020. Pika: Center-wide and job-aware cluster monitoring. In *2020 IEEE International Conference on Cluster Computing (CLUSTER).* IEEE, 424–432.
[9] Dominic Eschweiler, Michael Wagner, Markus Geimer, Andreas Knüpfer, Wolfgang E Nagel, and Felix Wolf. 2012. Open trace format 2: The next generation of scalable trace formats and support libraries. In *Applications, Tools and Techniques on the Road to Exascale Computing.* IOS Press, 481–490.
[10] Argonne Leadership Computing Facility. 2024. THAPI: Tracing Heterogeneous APIs. https://github.com/argonne-lcf/THAPI Accessed: 2024-12-24.
[11] Gilles Fourestey, Ben Cumming, Ladina Gilly, and Thomas C Schulthess. 2014. First experiences with validating and using the Cray power management database tool. *arXiv preprint arXiv:1408.2657* (2014).
[12] Markus Geimer, Felix Wolf, Brian JN Wylie, Erika Ábrahám, Daniel Becker, and Bernd Mohr. 2010. The Scalasca performance toolset architecture. *Concurrency and computation: Practice and experience* 22, 6 (2010), 702–719.
[13] Michael Gerndt. 2016. The READEX Project for Dynamic Energy Efficiency Tuning. In *Proceedings of the ACM Workshop on Software Engineering Methods for Parallel and High Performance Applications.* 11–12.
[14] Heike Jagode, Anthony Danalis, Giuseppe Congiu, Daniel Barry, Anthony Castaldo, and Jack Dongarra. 2024. Advancements of PAPI for the exascale generation. *The International Journal of High Performance Computing Applications* (2024), 10943420241303884.
[15] Kashif Nizam Khan, Mikael Hirki, Tapio Niemi, Jukka K Nurminen, and Zhonghong Ou. 2018. Rapl in action: Experiences in using rapl for power measurements. *ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS)* 3, 2 (2018), 1–26.
[16] Andreas Knüpfer, Christian Rössel, Dieter an Mey, Scott Biersdorff, Kai Diethelm, Dominic Eschweiler, Markus Geimer, Michael Gerndt, Daniel Lorenz, Allen Malony, Wolfgang E. Nagel, Yury Oleynik, Peter Philippen, Pavel Saviankou, Dirk Schmidl, Sameer Shende, Ronny Tschüter, Michael Wagner, Bert Wesarg, and

Felix Wolf. 2012. Score-P: A Joint Performance Measurement Run-Time Infrastructure for Periscope,Scalasca, TAU, and Vampir. In *Tools for High Performance Computing 2011*, Holger Brunst, Matthias S. Müller, Wolfgang E. Nagel, and Michael M. Resch (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 79–91.

[17] Karlo Kraljic, Daniel Kerger, and Martin Schulz. 2022. Energy Efficient Frequency Scaling onÂ GPUs inÂ Heterogeneous HPC Systems. In *Architecture of Computing Systems*, Martin Schulz, Carsten Trinitis, Nikela Papadopoulou, and Thilo Pionteck (Eds.). Springer International Publishing, Cham, 3–16.

[18] Adam Krzywaniak and Pawel Czarnul. 2023. Dynamic GPU Power Capping with Online Performance Tracing. *Future Generation Computer Systems* 145 (2023), 396–414.

[19] Sandia National Laboratories. 2024. *Variorum Documentation.* https://variorum.readthedocs.io/en/latest/ Accessed: 2024-12-24.

[20] Steven J Martin and Matthew Kappel. 2014. Cray XC30 power monitoring and management. In *Cray User Group Conference Proceedings*.

[21] Abdelhafid Mazouz, Benoît Pradelle, and William Jalby. 2014. Statistical validation methodology of CPU power probes. In *Euro-Par 2014: Parallel Processing Workshops: Euro-Par 2014 International Workshops, Porto, Portugal, August 25-26, 2014, Revised Selected Papers, Part I 20*. Springer, 487–498.

[22] Oak Ridge Leadership Computing Facility. 2025. Frontier User Guide. https://docs.olcf.ornl.gov/systems/frontier_user_guide.html Accessed: 2025-04-30.

[23] The Linux Kernel Organization. 2025. Naming and data format standards for sysfs files. https://www.kernel.org/doc/Documentation/hwmon/sysfs-interface Accessed: 2025-04-30.

[24] Pavel Saviankou, Michael Knobloch, Anke Visser, and Bernd Mohr. 2015. Cube v4: From performance report explorer to performance analysis tool. *Procedia Computer Science* 51 (2015), 1343–1352.

[25] Robert Schöne, Thomas Ilsche, Mario Bielert, Markus Velten, Markus Schmidl, and Daniel Hackenberg. 2021. Energy efficiency aspects of the AMD Zen 2 architecture. In *2021 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 562–571.

[26] Robert Schöne, Ronny Tschüter, Thomas Ilsche, Joseph Schuchart, Daniel Hackenberg, and Wolfgang E Nagel. 2017. Extending the functionality of score-p through plugins: Interfaces and use cases. In *Tools for High Performance Computing 2016: Proceedings of the 10th International Workshop on Parallel Tools for High Performance Computing, October 2016, Stuttgart, Germany*. Springer, 59–82.

[27] Sameer Shende and Allen D Malony. 2011. TAU: Performance Analysis and Tuning. *Innovative Parallel Computing (InPar)* (2011).

[28] Sameer S Shende and Allen D Malony. 2006. The TAU parallel performance system. *The International Journal of High Performance Computing Applications* 20, 2 (2006), 287–311.

[29] Osman Seckin Simsek, Jean-Guillaume Piccinali, and Florina M Ciorba. 2023. Accurate Measurement of Application-level Energy Consumption for Energy-Aware Large-Scale Simulations. In *Proceedings of the SC'23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis*. 1881–1884.

[30] Prasoon Sinha, Akhil Guliani, Rutwik Jain, Brandon Tran, Matthew D Sinclair, and Shivaram Venkataraman. 2022. Not all gpus are created equal: characterizing variability in large-scale, accelerator-rich systems. In *SC22: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 01–15.

[31] Vince Weaver. 2012. New features in the PAPI 5.0 release. *University of Tennessee, Tech. Rep* (2012).

[32] Matthias Weber, Ronny Brendel, Michael Wagner, Robert Dietrich, Ronny Tschüter, and Holger Brunst. 2019. Visual Comparison of Trace Files in Vampir. In *Programming and Performance Visualization Tools*, Abhinav Bhatele, David Boehme, Joshua A. Levine, Allen D. Malony, and Martin Schulz (Eds.). Springer International Publishing, Cham, 105–121.

[33] Zeyu Yang, Karel Adamek, and Wesley Armour. 2024. Accurate and Convenient Energy Measurements for GPUs: A Detailed Study of NVIDIA GPU's Built-In Power Sensor *(SC '24)*. IEEE Press, Article 22, 17 pages. doi:10.1109/SC41406.2024.00028

[34] Andrew J Younge, Ryan E Grant, James H Laros III, Michael Levenhagen, Stephen L Olivier, Kevin Pedretti, and Lee Ward. 2019. Small scale to extreme: Methods for characterizing energy efficiency in supercomputing applications. *Sustainable Computing: Informatics and Systems* 21 (2019), 90–102.