Characterizing the Impact of GPU Power Management on an Exascale System

Abstract

As GPU-accelerated high-performance computing (HPC) systems approach exascale performance, controlling energy consumption without compromising throughput is essential. Architectures such as the AMD MI250X-based Frontier supercomputer provide runtime mechanisms like frequency and power capping, enabling energy tuning without modifying application code. Although both target energy reduction, they operate via distinct hardware control paths and influence workloads differently. We present a comprehensive evaluation of these strategies on a leadership-class system using diverse HPC proxy applications representative of production workloads. Our study analyzes performance–energy trade-offs across multiple capping levels, node counts (1 and 32), and application profiles. Results show that frequency capping generally achieves higher energy efficiency and scalability, with gains of up to 13.2% without performance loss, while power capping is more effective for single-node runs or bursty GPU utilization. We also provide practical guidelines to help system administrators and users balance energy efficiency and performance in large-scale scientific workloads.

Keywords

Energy efficiency, GPU, Exascale Systems, Power Capping, Frequency Capping

1 Introduction

As high-performance computing (HPC) systems scale toward exascale and beyond, the ability to evaluate and model the performance-energy trade-offs of large-scale applications has become essential for system designers and practitioners. Modern supercomputers rely heavily on graphic processing unit (GPU) acceleration to deliver sustained floating-point throughput, yet this performance often comes at the cost of high power consumption and limited control over runtime efficiency. In this context, performance benchmarking and tuning are increasingly intertwined with energy-aware strategies that aim to maximize computational throughput while minimizing energy-to-solution. Hence, understanding and quantifying the impact of hardware-level runtime controls on performance is a key challenge in software/hardware co-design.

PMBS 2025, St. Louis, MO, USA

© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/XXXXXXX.XXXXXXX

Two controls that are widely supported in GPU platforms are power capping [11] and frequency capping, commonly implemented via dynamic voltage and frequency scaling (DVFS) [13]. Power capping imposes a real-time upper limit on device power draw, dynamically adjusting voltage and frequency via hardware-internal policies. On the other hand, frequency capping enforces a static limit on clock frequency, providing deterministic behavior but potentially limiting performance. While both mechanisms aim to improve energy efficiency or enforce power budgets, they influence application behavior through distinct control planes, i.e., reactive versus proactive, which can lead to markedly different outcomes in terms of runtime, energy usage, and hardware utilization.

The performance of these mechanisms is highly dependent on the application characteristics. Compute-bound applications may be sensitive to frequency constraints due to their reliance on high arithmetic throughput, whereas memory-bound workloads may tolerate aggressive frequency throttling with negligible performance degradation. Similarly, irregular workloads with bursty behavior may benefit from the adaptive nature of power management, while others may underutilize the available power envelope when constrained by static frequency limits. These divergent effects highlight the need for detailed benchmarking and comparative analysis, especially in heterogeneous, production-scale HPC environments.

In this paper, we present an extensive performance and energy benchmarking study of power and frequency management strategies on seven well-known GPU-accelerated scientific applications. Our objective is to characterize how these management techniques affect runtime, energy-to-solution, and energy efficiency metrics across diverse application profiles and system configurations. We evaluate 21 power cap levels and 31 frequency cap settings using seven representative GPU workloads on the Frontier supercomputer at the Oak Ridge Leadership Computing Facility (OLCF), exploring their scalability and sensitivity to hardware constraints. Rather than proposing a new power management technique, we evaluate how existing mechanisms behave under a range of configurations and application profiles. Through this extensive set of experiments, we show that:

- Runtime power management strategies can deliver energy efficiency improvements of up to 34.7% over the default execution of GPU applications, with frequency capping often outperforming power capping, especially at scale.
- The effectiveness of power management is highly dependent on the application and scale characteristics. Memory-bound workloads benefit from moderate frequency caps, while power capping was more effective for compute-intensive or bursty applications.
- While aggressive caps severely degrade performance, moderate configurations can provide favorable energy-performance trade-offs with <5% slowdown, making them practical for production HPC workloads.

The rest of this paper is structured as follows. We describe the GPU runtime power management strategies along with the related work in Section 2. Then, the methodology applied during the benchmarking is listed in Section 3. The results for performance and energy efficiency are discussed in Section 4, while we draw the final considerations in Section 5.

2 Background

In this section, we describe the two hardware-level techniques for controlling power in GPU-accelerated systems: power capping and frequency capping. Both are supported by modern GPUs but differ in implementation, control granularity, and performance impact. We first describe their operational mechanisms, then review prior studies on their effects on performance and energy consumption across various workloads.

2.1 GPU Frequency Capping

Often referred to as dynamic voltage and frequency scaling (DVFS), GPU frequency capping is a mechanism that constrains the operating clock frequencies of a GPU's core and memory subsystems by defining upper bounds on the clock domain, limiting the maximum performance capabilities of the device [13]. DVFS operates by selecting a specific frequency-voltage pair from a predefined set of performance states (P-states) exposed by the GPU firmware. Each P-state corresponds to a specific core frequency and associated voltage level that ensures stable operation. Lowering the operating frequency reduces both dynamic and static power consumption, due to decreased switching activity and the possibility of operating at reduced voltage levels. This relationship is governed by the CMOS power equation $P_{dynamic} = C \cdot V^2 \cdot f$, where C is the switching capacitance, V is the supply voltage and f is the frequency [13].

GPUs usually expose control interfaces for frequency capping via vendor-specific tools, such as `nvidia-smi` or `rocm-smi` on NVIDIA and AMD GPUs, respectively. These tools enable system administrators or users to restrict the maximum allowable frequency during application execution, either statically or dynamically, depending on support from the driver and firmware. The DVFS control can be applied at different granularity levels, including per-GPU, per-clock-domain (e.g., core, memory), or even per-workload via runtime libraries or job scheduler plugins. From a hardware perspective, DVFS is managed by a power management controller (e.g., NVIDIA's PMGR or AMD's SMU) that applies frequency scaling commands and coordinates voltage rails accordingly. In this scenario, frequency capping provides a proactive and deterministic control mechanism. Moreover, the latency of switching between P-states is typically in the microsecond range, making DVFS viable for fine-grained phase-based tuning. However, abrupt frequency reductions may also incur performance penalties, particularly in compute-bound kernels, as the number of operations executed per second is directly tied to the SM clock rate.

2.2 GPU Power Capping

GPU power capping is a runtime technique that constrains the maximum power consumption of a GPU by enforcing a power budget, typically expressed in watts. This mechanism is designed to limit dynamic power usage without modifying the application code or changing GPU scheduling policies [11]. Power capping is commonly used to reduce energy consumption, lower thermal output, and manage power-aware scheduling in large-scale systems, especially under facility-imposed power budgets or energy-aware operational policies. At the hardware level, power capping is implemented through a closed-loop feedback control mechanism integrated into the GPU's power management unit (PMU). Modern GPUs, such as from AMD and NVIDIA, expose interfaces that allow users or system software to set a target power limit via driver-level APIs (e.g., `nvidia-smi -power-limit` or `rocm-smi -setpoweroverdrive`). The PMU continuously monitors the power draw by sampling telemetry signals such as current and voltage levels across internal sensors (e.g., power rails for core, memory, and subsystems). When the observed power exceeds the configured cap, the GPU reduces its power consumption by dynamically adjusting performance-related knobs, including voltage levels, clock frequencies, and active compute resources.

Power capping often leads to frequency throttling, where core and memory clocks are scaled down temporarily to maintain power below the target limit. The frequency scaling response is determined by hardware firmware or driver heuristics, and it may vary depending on the workload's characteristics (e.g., memory-bound vs. compute-bound). Unlike frequency capping, power capping does not set a fixed frequency but rather enforces a power envelope, allowing the GPU to modulate its frequency and voltage adaptively in response to workload dynamics. This makes power capping a more flexible but also less predictable control mechanism. Moreover, the granularity and effectiveness of power capping depend on the GPU architecture, firmware support, and system-level thermal/power headroom. In HPC systems, power capping can be used in coordination with job-schedulers to enforce power ceilings, improve power predictability across jobs, and optimize the energy-performance trade-off under global constraints.

2.3 Related Work

Recent efforts to improve energy efficiency in GPU-accelerated HPC have explored power management strategies. In this subsection, we describe past research across these techniques and highlight the progression towards system-level optimization frameworks. Tapasya et al. conducted one of the first large-scale comparisons of power capping and frequency capping in GPU workloads using over 5,300 runs of the MUMMI molecular dynamics application on ddcMD [20]. Their results showed that reducing the power cap from 300 W to 170 W caused negligible performance degradation, while frequency capping introduced variability and instability. In contrast, Allen et al. identified inefficiencies in default hardware power caps on GPUs, finding up to 35% performance loss compared to a smarter, application-informed distribution of power between Streaming Multiprocessors (SMs) and memory subsystems [2].

Considering static capping policies, more recent studies have proposed dynamic and feedback-driven power capping frameworks. Kryzwaniak et al. introduced DEPO, a runtime system that adaptively adjusts NVIDIA power limits based on application behavior, achieving 25–30% energy savings with <5% performance overhead [12]. Similarly, Simmendinger et al. proposed a phase-aware dynamic capping mechanism that reacts to workload characteristics, reducing power usage by 20% without measurable runtime penalty [19]. Karimi et al. performed a telemetry-driven evaluation on Frontier using three months of operational logs [10]. By developing a decomposition model of GPU operational modes, the work projected energy savings of up to 8.5% with substantial real-world impact, estimating over 1,400 MWh of energy reduction. Application-specific studies have also informed the selection of optimal caps. Acun et al. analyzed the behavior of MILC on NERSC's Perlmutter system, showing that a 200W cap (50% of A100 TDP) reduced energy by 28% with <15% slowdown [1]. At the architectural level, Patrou et al. examined LSMS on NVIDIA's GH200 Grace Hopper platform, using multi-objective optimization (including energy-delay and Euclidean metrics) to guide dynamic GPU caps [17].

Recent research has moved toward intelligent, learning-based control systems. Yiming et al. (2024) proposed DRLCap, a deep reinforcement learning approach for dynamically adjusting frequency caps across different GPU architectures [21]. DRLCap used system-level profiling to detect program phase changes and optimized frequency settings, achieving average GPU energy savings of 22% on NVIDIA and 10% on AMD GPUs. User-aware and cooperative strategies have also emerged. Angelelli et al. (2024) introduced an Eco-Mode where users opt into power-capped execution voluntarily [3]. Simulations using real job traces from a Top500 system showed that once 30% of users adopt the mode, job termination rates under enforced caps drop, preserving system throughput and avoiding penalties.

Finally, predictive models based on job characteristics are gaining traction. Antici et al. developed a machine learning framework to predict power consumption for CPU-based jobs on Fugaku, achieving 90% accuracy [4]. Ding et al. proposed a unified model combining power variability, efficiency, and performance metrics to guide capping decisions in HPC workloads [6]. In the context of exascale GPU workloads, Lorenzon et al. introduced V-FORGE, a frequency-aware and variability-tolerant scheduling framework [14]. Their results on 400 AMD MI250X GPUs showed that selecting appropriate GPU frequency per application-node combination led to up to 41% EDP improvement, emphasizing the critical role of hardware–software co-optimization in modern HPC systems.

Our Contributions. Compared to previous studies that focus on proposing new runtime frameworks, application-specific tuning strategies, or predictive models for GPU power management, this work takes a comprehensive empirical approach aimed at providing practical guidance to HPC users and system administrators. Rather than introducing new mechanisms, our contribution lies in characterizing and comparing the behavior of existing power and frequency capping interfaces across a wide range of operational settings. While recent works such as DEPO [12], DRLCap [21], and V-FORGE [14] emphasize runtime adaptivity or learning-based optimization, our study complements these efforts by offering a systematic evaluation of 21 power cap levels and 31 frequency settings on seven production-grade, GPU-accelerated scientific applications, including compute- and memory-bound workloads. Unlike application-specific studies (e.g., MILC on A100 [1] or LSMS on GH200 [17], our evaluation considers multiple applications and scales, capturing both single-node and multi-node behavior on the Frontier supercomputer. Finally, we believe that the resulting guidelines bridge the gap between theoretical potential and operational applicability of power management strategies, offering a reference for energy-aware configuration policies in Exascale HPC systems.

3 Methodology

In this section, we present the benchmarks selected for our evaluation, provide details of the underlying infrastructure, and describe the performance and energy-related metrics used in the analysis.

3.1 Benchmarks

We consider a representative set of seven GPU-accelerated applications from distinct benchmark suites and scientific domains, covering a wide range of computational patterns, memory access behaviors, and arithmetic intensities. **Cholla**, a multi-dimensional hydrodynamics code based on the Piecewise Parabolic Method (PPM) for astrophysical simulations. It is a memory-intensive application with floating-point 64 arithmetic precision [1]. **HACC - Hydro**, a gas-dynamics version of the Hardware Accelerated Cosmology Code (HACC), focusing on hydrodynamics [8]. This application has communication overhead depending on the simulation scale, due to frequent data exchanges at domain boundaries. HACC is also sensitive to both memory bandwidth and compute throughput. **Kripke**, a performance proxy application that solves the 3D deterministic Sn (discrete ordinates) particle transport equations on structured grids. It is designed to evaluate and stress-test the performance of modern parallel architectures, focusing on memory access patterns, spatial decomposition strategies, and execution characteristics representative of production transport codes. **LAMMPS**, a molecular dynamics application widely used in materials science for simulating atomic and mesoscale systems. It supports a broad range of interatomic potentials and models, and its GPU-accelerated version offloads force computations and neighbor list construction. **Pennant**, a proxy application for unstructured mesh physics simulations involving Lagrangian hydrodynamics. Pennant demonstrates complex memory access patterns and sensitivity to both compute throughput and memory hierarchy performance [7]. **PortUrb**, a domain-specific urban flood simulation application based on finite-volume shallow water solvers [15]. The application is sensitive to memory bandwidth and mostly simple floating-point operations over large data arrays. **QuickSilver**, a Monte Carlo particle transport proxy application developed at LANL, is representative of high-performance radiation transport workloads. It exhibits irregular control flow and limited SIMD utilization [18]. Since Monte Carlo transport codes often rely on particle-based data structures, they result in high irregular and latency-bound memory access behavior.

We selected this set of applications because they exhibit different performance characteristics in terms of FLOPs/s, FLOPs/byte, and L2 cache hit rate, which are key indicators of computational intensity, memory bandwidth demand, and cache utilization, as shown in Table 1. For instance, *HACC* reaches a very high arithmetic intensity and high L2 cache efficiency, while *Kripke* presents a much lower compute intensity and lower L2 cache hit rate. *Cholla* and *Pennant* show similar compute intensity but differ in throughput and cache behavior. In addition to covering a broad spectrum of computational and memory-access patterns, these applications are

[1]https://github.com/cholla-hydro/cholla

Table 1: Characteristics of each application considering the average of all GPU kernels

	FLOPs/byte	FLOPs/s	L2Cache Hit (%)
Cholla	0.62	6.58E+11	37.51
HACC	215.04	4.67E+12	84.02
Kripke	0.10	3.90E+10	62.68
Lampss	3.41	5.71E+12	51.78
Pennant	0.67	5.57E+11	45.11
PortUrb	11.45	1.08E+12	61.32
QuickSilver	1.83	1.61E+10	74.72

widely executed in HPC centers, ensuring that the analysis captures representative workload behaviors observed in production environments.

3.2 Testbed System

The experiments were conducted on the Frontier supercomputer at the OLCF [5]. Each compute node is equipped with a 64-core AMD-optimized 3rd Gen EPYC 7763 CPU, supporting two hardware threads per core and 512 GB of DDR4 memory. The nodes feature four AMD MI250X accelerators, each composed of two graphics compute dies (GCDs), resulting in a total of eight GCDs per node. Each GCD includes 64 GB of high-bandwidth memory (HBM3E). For the purpose of this study, each GCD was treated as an independent GPU, yielding 8 GPUs per node. The observed behaviors discussed in the next sections reflect the interplay between application characteristics and GPU hardware design, including memory bandwidth constraints and the dynamic scaling behavior of this architecture. To assess the behavior of power and frequency capping techniques under different scalability conditions, all applications were executed in two configurations: a single-node run, utilizing all 8 GPUs, and a multi-node run using 32 nodes for a total of 256 GPUs.

The MI250X GPUs have a thermal design power (TDP) of 560W and support frequency scaling within a range of 500MHz to 1700MHz. In our frequency capping experiments, we evaluate 31 frequency configurations in steps of 40MHz (i.e., 500MHz, 540MHz, ..., 1700MHz). Similarly, for power capping, we explored 21 configurations ranging from 140W to 560W, in steps of 20W. All configurations were applied statically before execution using vendor-provided ROCm tools. Applications were compiled using the AMD ROCm 6.2.4 stack, with `hipcc` and the flags `-O3` and `-offload-arch=gfx90a` for performance optimization and target-specific code generation. For reproducibility, the experiments were executed with the environment modules `craype-accel-amd-gfx90a` and `rocm/6.2.4` located on Frontier.

3.3 Evaluated Metrics

We consider two metrics: performance (Perf) and performance per watt (Perf/Watt). Performance is defined as the inverse of the total execution time, representing the rate at which the application completes its workload. To assess energy efficiency, we calculate performance per watt as the ratio between performance and the average GPU power draw during execution. This metric quantifies the computational output delivered per unit of power consumed. To characterize power, energy, and other system-level metrics, we leveraged ***Omnistat***, an open-source, low-overhead monitoring tool [16]. Omnistat was originally developed to collect statistics from AMD Instinct™ GPUs[2], such as the Instinct MI250X models deployed on Frontier. For our application case studies on Frontier, Omnistat was configured using a runtime control file to enable the collection of core GPU metrics via the SMI interface. Additional optional data, including reliability, availability, and service (RAS) error counters, network traffic, and vendor-provided board-level power and energy measurements [9] were also gathered. Using Omnistat, we aggregated metrics on a per-SLURM job basis across all compute hosts assigned to each application run, under various frequency and power cap settings.

4 Evaluation

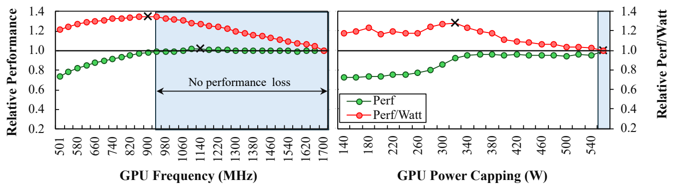
In this section, we first compare the raw performance and energy efficiency for the execution of each application with the evaluated frequency and power capping configurations on the single-node and 32-node scenarios. Second, we describe the observed behavior of selected applications through runtime traces. Then, we discuss the implications of these results, outlining guidelines for selecting the most appropriate power management strategy according to application characteristics and execution scale.

4.1 Performance and Energy Evaluation

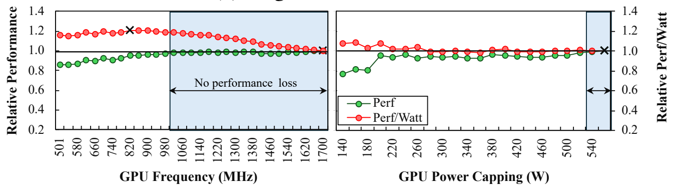
In this subsection, we present a comparative analysis of the GPU power management strategies based on normalized performance and performance-per-watt relative to the baseline configuration, which reflects default system behavior without power or frequency constraints. Figures 1 through 7 summarize results across applications and node counts, where normalized values above 1.0 indicate improvements over the baseline. We highlight with a blue rectangle the region corresponding to configurations that do not incur performance degradation compared to the baseline (e.g., No Performance Loss). This area represents the operating frequencies and power capping values where energy efficiency gains can be achieved without compromising application runtime. During the experiments, we got a standard deviation of 2% in the performance measurements. Therefore, results within this deviation range are considered statistically equivalent to the optimal configuration. Overall, power and frequency capping can enhance energy efficiency, though their effectiveness is application- and scale-dependent. Key factors include the compute and memory characteristics of each application and sensitivity to hardware throttling. For instance, moderate capping levels often preserve performance while improving efficiency, but aggressive frequency reductions typically incur significant performance penalties. The following analysis groups applications by their responsiveness to capping strategies.

Cholla, *Pennant*, and *HACC* (Figures 1,2, and 3) are examples of applications that show significant improvements in performance per watt when moderate power or frequency capping is applied, particularly at the single-node level. For *Cholla*, Perf/Watt improves by up to 34.7% at 900 MHz and 28% at 320 W, with performance remaining above 90% for frequency caps above 820 MHz. While these

[2]AMD, the AMD Arrow logo and combinations thereof are trademarks of Advanced Micro Devices, Inc.

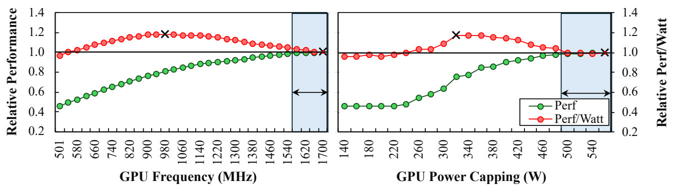


(a) Single-node execution

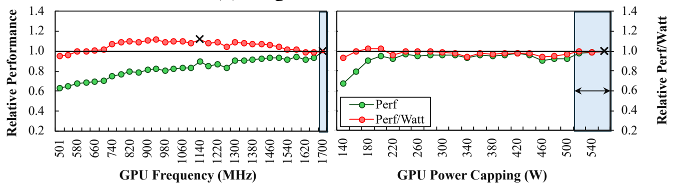


(b) 32-node execution

Figure 1: Cholla behavior

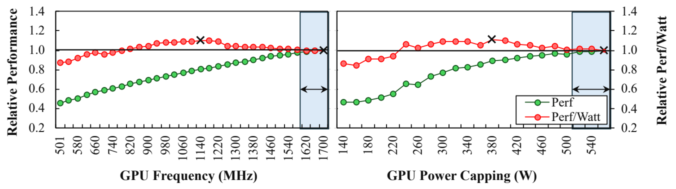


(a) Single-node execution

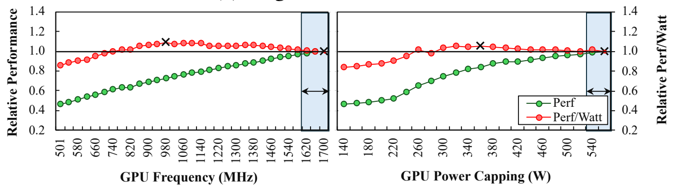


(b) 32-node execution

Figure 2: Pennant behavior

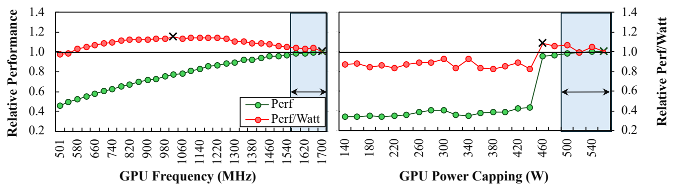


(a) Single-node execution

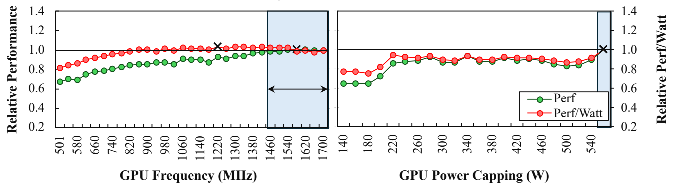


(b) 32-node execution

Figure 3: HACC behavior



(a) Single-node execution



(b) 32-node execution

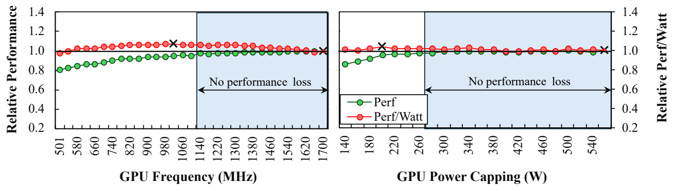
Figure 4: PortUrb behavior

gains are less pronounced at 32 nodes, improvements still reach 20% under frequency capping, and 8% with moderate power caps. Similarly, *Pennant* reaches up to 18% improvements in Perf/Watt at 360 W and 980-1020 MHz, with performance consistently above 80%. On 32 nodes, frequency capping between 740-1140 MHz yields 11% improvement with limited performance loss. *HACC* exhibits a particularly broad operating range. At 1 node, Perf/Watt peaks at 11% under both capping strategies with performance remaining above 80%. At 32 nodes, it consistently maintains over 80% performance and up to 6% energy efficiency improvements across 240-460 W and 940-1380 MHz. As observed, applications that exhibit this behavior can benefit from moderate frequency reductions, allowing for energy-aware tuning with minimal performance degradation.

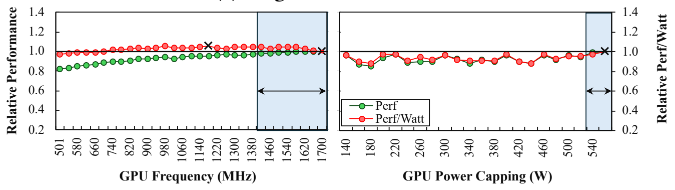
PortUrb (Figure 4) shows a clear trade-off between energy efficiency and execution time, especially in single-node runs. Frequency capping leads to up to 15% Perf/Watt improvement at 1020 MHz, but performance drops to 77% of the baseline. Power capping also helps when applied moderately (460-500 W), with 6% Perf/Watt improvement and little impact on performance, but aggressive power limits cause severe slowdowns without proportional energy gains. On 32 nodes, the application becomes less responsive to power capping, with only frequency capping maintaining a slight energy advantage, highlighting the limited benefit of capping at scale for workloads that are sensitive to core frequency.

Kripke and *LAMMPS* (Figures 5 and 6) demonstrate robust performance across a wide range of cap configurations, with small but consistent improvements in energy efficiency. For *Kripke*, Perf/Watt increases by up to 7% under frequency capping around 980-1020 MHz, with performance above 94%, and similar gains under moderate power caps (200-360 W). At 32 nodes, Perf/Watt remains 5% above baseline across 740-1540 MHz, confirming its low sensitivity to power management. *LAMMPS*, on the other hand, is largely unaffected by power capping, maintaining near-baseline performance and Perf/Watt across all configurations. However, frequency capping introduces more pronounced effects. While performance drops below 50% at low frequencies, energy efficiency peaks at 13% above baseline around 900-1220 MHz on 32 nodes. In this scenario, while *LAMMPS* resists power-related throttling, it can still benefit from selecting ideal frequency reductions, especially in large-scale runs.

QuickSilver (Figure 7) represents a class of applications for which power management strategies yields minimal benefit. On a single node, Perf/Watt remains effectively unchanged across all tested

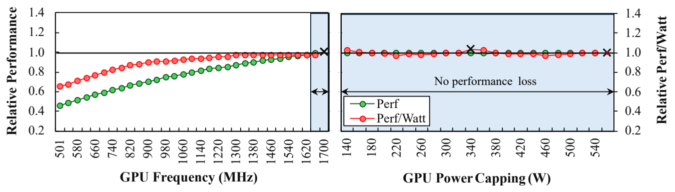


(a) Single-node execution

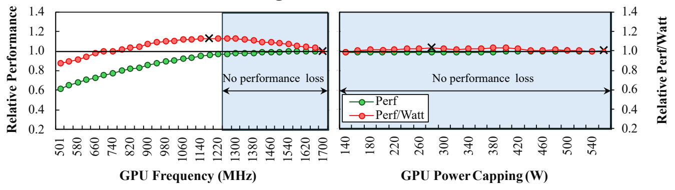


(b) 32-node execution

Figure 5: Kripke behavior



(a) Single-node execution



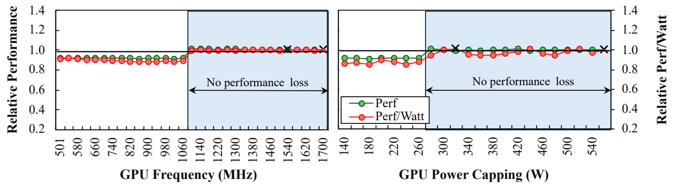
(b) 32-node execution

Figure 6: LAMMPS behavior

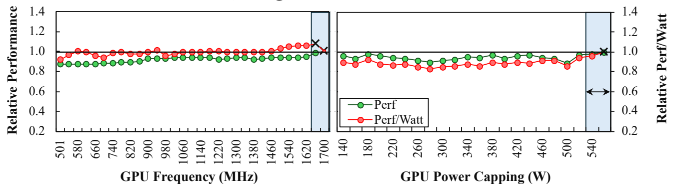
power and frequency settings. This suggests the application already operates near its optimal energy-performance point, possibly due to latency-bound behavior or limited GPU utilization. At 32 nodes, small improvements (up to 8%) in energy efficiency can be achieved with minor frequency reductions (e.g., at 1660 MHz) without sacrificing performance, though the gains are not significant enough to warrant aggressive tuning.

4.2 Application-Aware Evaluation of GPU Power Management Strategies

As we discussed in the previous section, the impact of GPU power management strategies on performance and energy efficiency varies significantly depending on the application characteristics and the system scale. While both techniques can lead to energy efficiency gains, their relative effectiveness is neither uniform nor interchangeable across all workloads. Figure 8 illustrates the energy efficiency gains over the baseline along with the impact on the application performance across all applications, when considering the configuration for power capping and frequency capping that delivers the best Perf/Watt outcome.



(a) Single-node execution



(b) 32-node execution

Figure 7: QuickSilver behavior

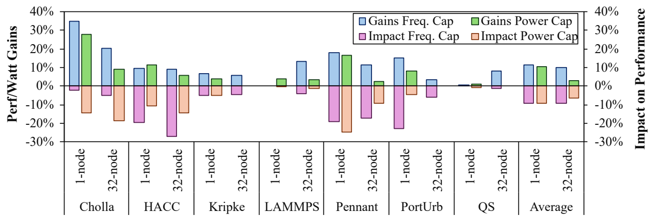


Figure 8: Energy efficiency gains and performance impact of the frequency and power capping configurations that yielded the best Perf/Watt results

At the single-node level, frequency capping often provides a better trade-off between energy efficiency and performance. For instance, *Cholla* achieves a 34.7% gain in Perf/Watt with only 2.3% performance degradation, while power capping results in a slightly lower energy efficiency gain (28.0%) and a significantly higher performance loss (14.4%). This trend is also observed in *Pennant*, where frequency and power capping yield similar energy benefits (18.1% for frequency and 16.8% for power capping), but frequency capping incurs a lower performance penalty (19.2% vs. 24.5%).

On the other hand, power capping was more effective than frequency capping for some applications, especially at the single-node level. In *HACC*, power capping yielded the highest energy efficiency gain (11.3%) with a performance impact of 10%, while frequency capping achieved slightly lower efficiency (9.6%) but with a significantly larger performance penalty (-19.6%). At scale, the performance impact of frequency capping worsened, reaching a 27.1% slowdown at 32 nodes, despite comparable energy gains (9.3%). Power capping, on the other hand, offered lower energy savings (5.8%) but preserved performance more effectively. To investigate the underlying behavior, we collected OmniStat traces for the configurations that achieved peak Perf/Watt for *HACC* (Figure 9). The traces provide a temporal view of GPU power draw and operating frequency throughout the application's execution. The values are the average of the eight GPU IDs (for frequency) and the four sockets (for power) along all 32 nodes. Under frequency capping (980 MHz), illustrated in Figures 9 (*a*) and (*c*), all GPUs operated at a fixed frequency with minimal variation, leading to stable power draw

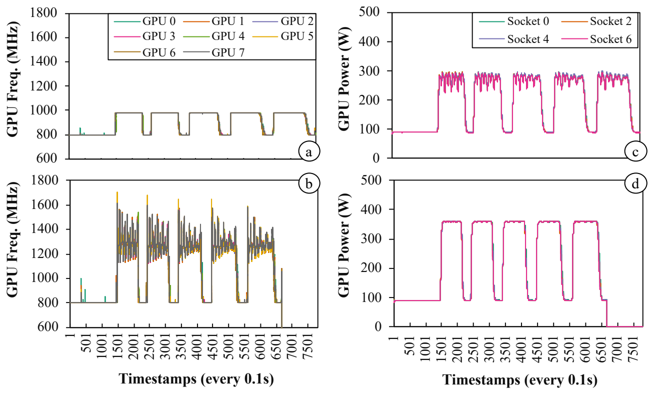


Figure 9: Execution traces for *HACC* on 32 nodes under the configurations that achieved the highest energy efficiency: frequency capping at 980 MHz and power capping at 360 W. Subfigures (a) and (c) depict GPU frequency and power draw over time for the frequency-capped execution, while (b) and (d) present the corresponding metrics for the power-capped execution.

around 300 W. While this configuration enforces consistent power use, it prevents dynamic frequency scaling, reducing performance potential. In contrast, under power capping (360 W), shown in Figures 9 (*b*) and (*d*), the GPU power draw remains tightly constrained at the cap, but frequency fluctuates dynamically. In particular, some GPUs reach up to 1700 MHz during phases with lower power demands, exploiting available headroom to accelerate execution. This adaptive behavior enabled better performance compared to the highest energy efficiency frequency capping configuration, highlighting the benefits of power-constrained frequency scaling in this type of application.

For *Kripke*, both capping strategies yield modest energy efficiency improvements at the single-node level (7.0% for frequency capping and 3.9% for power capping) with comparable performance overheads, making either option equally viable. At 32 nodes, however, power capping becomes less effective, providing no measurable improvement in Perf/Watt, whereas frequency capping maintains a 5.6% gain with a limited performance loss of 4.3%. A similar pattern emerges for *LAMMPS* at scale, frequency capping at 1120 MHz achieves a 13.2% increase in energy efficiency, while power capping at 280 W results in only a marginal 3.6% gain. In both cases, the performance degradation remains minimal ($\approx$ 4% under frequency capping and just 1% under power capping). To further investigate these differences, we collected runtime traces using OmniStat for the LAMMPS configurations that delivered the highest energy efficiency, following the same methodology used for *HACC*. As shown in Figure 10, frequency capping at 1120 MHz leads to uniformly reduced operating frequencies across all GPUs, resulting in lower power consumption without compromising performance. In contrast, under power capping at 280 W, GPU frequencies scale up to 1700 MHz, increasing power draw without yielding additional performance benefits. This behavior reflects the memory-bound nature of *LAMMPS*, where increases in core frequency do not translate into proportional performance gains due to limitations in memory throughput.

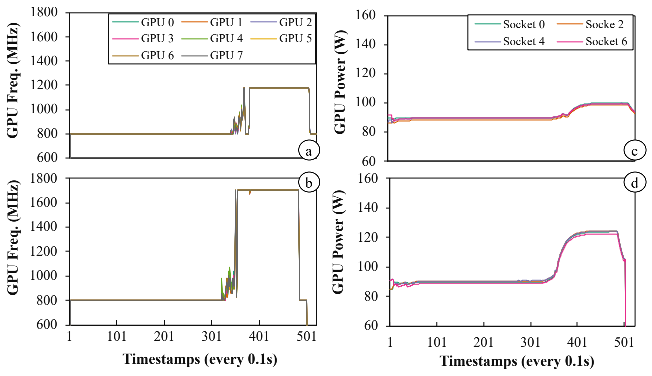


Figure 10: Execution traces for *LAMMPS* on 32 nodes under the configurations that achieved the highest energy efficiency: frequency capping at 1180 MHz and power capping at 280 W. Subfigures (a) and (c) depict GPU frequency and power draw over time for the frequency-capped execution, while (b) and (d) present the corresponding metrics for the power-capped execution.

4.3 Discussion

The results presented in the previous sections demonstrate that GPU power management can be a practical and effective mechanism to optimize energy consumption in HPC environments. Importantly, both techniques can be applied at runtime without requiring modifications to application source code, making them attractive options for users and system administrators aiming to balance energy efficiency and performance.

From a system-level perspective, power and frequency capping offer distinct trade-offs that can be tailored to specific application characteristics and operational goals. Frequency capping often delivers higher energy efficiency, especially at large scales, with relatively low performance penalties for applications that are compute-bound or resilient to moderate reductions in GPU clock speed. For instance, in applications such as *LAMMPS* and *Kripke*, frequency limits between 900-1200 MHz improved performance per watt without significant slowdown, highlighting a viable strategy for reducing energy consumption in production workflows. Conversely, power capping is more effective in single-node scenarios or workloads with high variability in GPU resource usage. By enforcing a fixed power budget, GPUs can opportunistically adjust their operating frequency depending on computational demand, allowing dynamic performance scaling within a constrained power envelope. This approach proved beneficial for applications like *HACC*, where power capping preserved higher frequency during compute-intensive phases without exceeding energy limits, yielding better overall performance compared to fixed frequency capping.

These findings suggest a guideline-based approach to runtime energy optimization. For memory-bound or latency-sensitive applications where increasing frequency yields minimal performance benefit, moderate frequency capping can reduce energy usage without affecting time-to-solution. For compute-intensive workloads with predictable GPU utilization, power capping can dynamically balance performance and energy, especially when configured near the GPU's energy-efficiency operating point. When operating at scale, frequency capping tends to scale better, as it reduces energy consumption while preserving synchronization and workload balance. Finally, aggressive capping (e.g., <700 MHz or <240 W) should be avoided unless energy savings are prioritized over performance, as such configurations often degrade efficiency and increase overall job time.

From an operational cost perspective, applying power management can significantly reduce both power consumption and cooling requirements, particularly for long-running jobs and large-scale simulations. In environments where energy usage is directly billed, such as cloud platforms or metered HPC centers, these reductions lead to measurable savings. Furthermore, as energy efficiency becomes a key criterion in system acquisition, procurement decisions, and institutional sustainability initiatives, the ability to reduce power draw without modifying application source code makes power management an attractive and non-invasive optimization technique. Moreover, GPU capping strategies provide a low-overhead and portable mechanism to tune energy usage based on workload behavior and execution scale. When appropriately configured, they enable users and administrators to achieve substantial improvements in energy efficiency with minimal performance loss.

5 Concluding Remarks

This paper presented a comprehensive evaluation of GPU power management techniques across a representative set of benchmarks. The analysis considered single-node and multi-node configurations, aiming to quantify trade-offs between performance and energy efficiency in HPC workloads. All evaluations were performed on the Frontier system, providing insights grounded in an exascale-class architecture.

Our results show that frequency capping often delivers higher energy efficiency with limited performance degradation. For example, *Cholla* and *Pennant* exhibited gains of up to 34.7% and 18.1% in performance-per-watt, respectively, under frequency capping, with performance losses contained to under 20%. Power capping, by contrast, proved more effective for bursty or irregular GPU usage patterns, such as in *HACC*, where it achieved an 11.3% improvement in energy efficiency with a lower performance penalty than frequency capping. For memory-bound applications like *LAMMPS*, moderate frequency limits (900–1200 MHz) yielded the best trade-off, reaching up to 13.2% improvement in energy efficiency with negligible impact on runtime. These results confirm that power management strategies can be tuned to application profiles to improve system-level energy proportionality.

While the sensitivity of each application to power and frequency capping is primarily determined by its computational and memory access characteristics, the resulting performance–energy trade-offs are also influenced by the architectural features of the hardware. In particular, the MI250 GPU exhibits specific behaviors in its DVFS mechanism and power limit enforcement, which affect the efficiency of runtime power management. Consequently, the effectiveness of a given power management strategy cannot be generalized across hardware generations. This highlights the need for continuous reassessment of energy optimization approaches as new GPU architectures are introduced, since static tuning methods may not transfer reliably across platforms.

The results presented in this study are relevant for system operators and users aiming to reduce energy consumption in GPU-accelerated high-performance computing environments without requiring modifications to application code. By identifying configurations that improve energy efficiency with minimal impact on performance, this work contributes to the development of practical and portable techniques for energy-aware execution. As a next step, we plan to explore runtime mechanisms that can adjust capping parameters automatically based on workload characteristics and system state, improving the adaptability and effectiveness of resource management in heterogeneous systems.

Acknowledgments

This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

References

[1] Fatih Acun, Zhengji Zhao, Brian Austin, Ayse K. Coskun, and Nicholas J. Wright. 2024. Analysis of Power Consumption and GPU Power Capping for MILC. In *SC24-W: Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 1856–1861. doi:10.1109/SCW63240.2024.00232

[2] Tyler Allen, Xizhou Feng, and Rong Ge. 2020. Performance Optimization in Power-Capped GPU Computing. Poster presented at the International Conference for High Performance Computing, Networking, Storage, and Analysis (SC20). https://sc20.supercomputing.org/proceedings/src_poster/poster_files/spostg111s2-file2.pdf Poster, evaluating miniFE on Titan XP, showing up to 35 % performance loss under default GPU power capping and proposing application-aware SM/memory power allocation.

[3] Luc Angelelli, Danilo Carastan-Santos, and Pierre-François Dutot. 2024. Run Your HPC Jobs innbsp;Eco-Mode: Revealing thenbsp;Potential ofnbsp;User-Assisted Power Capping innbsp;Supercomputing Systems. In *Job Scheduling Strategies for Parallel Processing: 27th International Workshop, JSSPP 2024, San Francisco, CA, USA, May 31, 2024, Revised Selected Papers* (San Francisco, USA). Springer-Verlag, Berlin, Heidelberg, 181–196. doi:10.1007/978-3-031-74430-3_10

[4] Francesco Antici, Andrea Borghesi, Jens Domke, and Zeynep Kiziltan. 2025. UoPC: A User-Based Online Framework to Predict Job Power Consumption in HPC Systems. In *ISC High Performance 2025 Research Paper Proceedings (40th International Conference)*. 1–12.

[5] Scott Atchley, Christopher Zimmer, John Lange, David Bernholdt, Veronica Melesse Vergara, Thomas Beck, Michael Brim, Reuben Budiardja, Sunita Chandrasekaran, Markus Eisenbach, Thomas Evans, Matthew Ezell, Nicholas Frontiere, Antigoni Georgiadou, Joe Glenski, Philipp Grete, Steven Hamilton, John Holmen, Axel Huebl, Daniel Jacobson, Wayne Joubert, Kim Mcmahon, Elia Merzari, Stan Moore, Andrew Myers, Stephen Nichols, Sarp Oral, Thomas Papatheodore, Danny Perez, David M. Rogers, Evan Schneider, Jean-Luc Vay, and P. K. Yeung. 2023. Frontier: Exploring Exascale. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (Denver, CO, USA) *(SC '23)*. Association for Computing Machinery, New York, NY, USA, Article 52, 16 pages. doi:10.1145/3581784.3607089

[6] Nan Ding, Oscar Antepara, Zhengji Zhao, Brian Austin, Leonid Oliker, Nicholas J. Wright, and Samuel Williams. 2025. Maximizing Power-Constrained Supercomputing Throughput. In *ISC High Performance 2025 Research Paper Proceedings (40th International Conference)*. 1–13.

[7] Charles R. Ferenbaugh. 2015. PENNANT: an unstructured mesh mini-app for advanced architecture research. *Concurr. Comput.: Pract. Exper.* 27, 17 (Dec. 2015), 4555–4572. doi:10.1002/cpe.3422

[8] Salman Habib, Vitali Morozov, Nicholas Frontiere, Hal Finkel, Adrian Pope, and Katrin Heitmann. 2013. HACC: Extreme scaling and performance across diverse architectures. In *SC '13: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. 1–10. doi:10.1145/2503210.2504566

[9] HPE Cray Supercomputing. 2025. PM Counters. https://support.hpe.com/hpesc/public/docDisplay?docId=dp00005587en_us&page=operations-hpcm/os/PM_Counters.html. Accessed: 2025-07-20.

[10] Ahmad Maroof Karimi, Matthias Maiterth, Woong Shin, Naw Safrin Sattar, Hao Lu, and Feiyi Wang. 2025. Exploring the Frontiers of Energy Efficiency using Power Management at System Scale. In *Proceedings of the SC '24 Workshops of the International Conference on High Performance Computing, Network, Storage, and Analysis* (Atlanta, GA, USA) *(SC-W '24)*. IEEE Press, 1835–1844. doi:10.1109/SCW63240.2024.00230

[11] Toshiya Komoda, Shingo Hayashi, Takashi Nakada, Shinobu Miwa, and Hiroshi Nakamura. 2013. Power capping of CPU-GPU heterogeneous systems through coordinating DVFS and task mapping. In *2013 IEEE 31st International Conference on Computer Design (ICCD)*. 349–356. doi:10.1109/ICCD.2013.6657064

[12] Adam Krzywaniak, Pawel Czarnul, and Jerzy Proficz. 2023. Dynamic GPU power capping with online performance tracing for energy efficient GPU computing using DEPO tool. *Future Generation Computer Systems* 145 (03 2023). doi:10.1016/j.future.2023.03.041

[13] Etienne Le Sueur and Gernot Heiser. 2010. Dynamic voltage and frequency scaling: The laws of diminishing returns. In *Proceedings of the 2010 international conference on Power aware computing and systems*. 1–8.

[14] Arthur F. Lorenzon, Antonio Carlos S. Beck, Philippe O. A. Navaux, and Bronson Messer. 2025. Energy-Efficient GPU Allocation and Frequency Management in Exascale Computing Systems. In *ISC High Performance 2025 Research Paper Proceedings (40th International Conference)*. 1–11.

[15] M. Norman, M. Gopalakrishnan Meena, K. Gottiparthi, N. Koukpaizan, and S. Nichols. 2025. PortUrb: A Performance Portable, High-Order, Moist Atmospheric Large Eddy Simulation Model with Variable-Friction Immersed Boundaries. *EGUsphere* 2025 (2025), 1–36. doi:10.5194/egusphere-2025-1135

[16] Omnistat. 2025. Omnistat: Scale-out cluster telemetry. https://github.com/ROCm/omnistat Accessed: 2025-07-20.

[17] Maria Patrou, Thomas Wang, Wael Elwasif, Markus Eisenbach, Ross Miller, William Godoy, and Oscar Hernandez. 2025. Power-Capping Metric Evaluation for Improving Energy Efficiency in HPC Applications. arXiv:2505.21758 [cs.DC] https://arxiv.org/abs/2505.21758

[18] David F. Richards, Ryan C. Bleile, Patrick S. Brantley, Shawn A. Dawson, Michael Scott McKinley, and Matthew J. O'Brien. 2017. Quicksilver: A Proxy App for the Monte Carlo Transport Code Mercury. In *2017 IEEE International Conference on Cluster Computing (CLUSTER)*. 866–873. doi:10.1109/CLUSTER.2017.121

[19] Christian Simmendinger, Marcel Marquardt, Jan Mäder, and Ralf Schneider. 2024. PowerSched - Managing Power Consumption in Overprovisioned Systems. In *2024 IEEE International Conference on Cluster Computing Workshops (CLUSTER Workshops)*. 1–8. doi:10.1109/CLUSTERWorkshops61563.2024.00012

[20] Patki Tapasya, Zachary Frye, Harsh Bhatia, Francesco Natale, James Glosli, Helgi Ingólfsson, and Barry Rountree. 2019. Comparing GPU Power and Frequency Capping: A Case Study with the MuMMI Workflow. 31–39. doi:10.1109/WORKS49585.2019.00009

[21] Wang Yiming, Meng Hao, Hui He, Weizhe Zhang, Qiuyuan Tang, Xiaoyang Sun, and Zheng Wang. 2024. DRLCap: Runtime GPU Frequency Capping with Deep Reinforcement Learning. *IEEE Transactions on Sustainable Computing* PP (09 2024), 1–15. doi:10.1109/TSUSC.2024.3362697