EVeREST: An Effective and Versatile Runtime Energy Saving Tool for GPUs

Anna Yue
Department of Computer Science
University of Minnesota
USA
yue00082@umn.edu

Pen-Chung Yew
Department of Computer Science
University of Minnesota
USA
yew@umn.edu

Sanyam Mehta
Hardware-software Codesign
Hewlett-Packard Labs
USA
sanyam.mehta@gmail.com

Abstract

Amid conflicting demands for ever-improving performance and maximizing energy savings, it is important to have a tool that automatically identifies opportunities to save power/energy at runtime without compromising performance. GPUs in particular present challenges due to (1) reduced savings available from memory bound applications, and (2) limited availability of low overhead performance counters. Thus, a successful tool must address these issues while still tackling the challenges of dynamic application characterization, versatility across processors from different vendors, and effectiveness at making the right power-performance tradeoffs for desired energy savings.

We propose Everest, a tool that automatically finds energy saving opportunities across GPUs at runtime. Specifically, Everest finds two unique avenues for saving energy using DVFS in GPUs in addition to the traditional method of lowering core clock for memory bound phases. Everest has very low overhead and works across different GPUs given its reliance on the minimum possible performance events for the needed characterization. Everest works at a finer granularity of individual application phases and utilizes built-in performance estimation to provide desired performance guarantees for an effective solution that outperforms existing solutions on the latest NVIDIA and AMD GPUs.

CCS Concepts: • **Software and its engineering** → *Power management*; • **Hardware** → **Chip-level power issues**; • **Computer systems organization** → *Single instruction, multiple data.*

Keywords: GPU, Power, Energy-efficiency, DVFS

PPoPP '25, March 1–5, 2025, Las Vegas, NV, USA
© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-1443-6/25/03
https://doi.org/10.1145/3710848.3710875

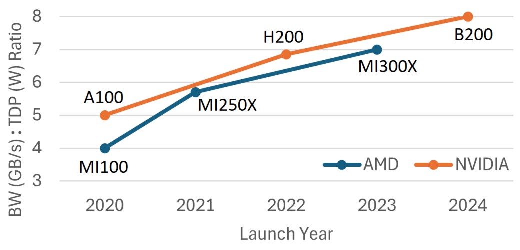


Figure 1. Bandwidth to TDP ratio across GPU generations

1 Introduction

As the world embraces newer and aggressive opportunities with generative AI, exascale computing, and beyond, it is imperative to develop effective solutions to combat the consequent rise in power and energy spending. This is especially important now in times of rising energy costs, higher power budgets that limit the ability to maintain and cool these servers, and governmental regulations that enforce sustainability requirements such as reduced carbon emissions [46]. While power and energy are critical, there is simultaneously an ever-increasing push for more performance, leading to conflicting demands for hardware and software. Considerable power and energy savings could be made while ensuring desired performance loss, which is the focus of this work.

A well researched approach that is proven to be effective for the purpose of power and energy savings especially in CPUs [29, 37, 38, 43, 45] is to selectively lower core clocks for memory bound phases/applications, which are much less sensitive to clock rates than the compute bound phases. Applying this same old formula on GPUs presents multiple challenges. These are as follows.

GPUs, the power-versus-memory wall and reduced savings. In addition to GPUs having a different microarchitecture than CPUs, the latest GPUs especially set themselves apart from CPUs in terms of the offered memory bandwidth thanks to the arrival and subsequent rapid advancements of High-Bandwidth Memory (HBM) technology. However, the available power (Thermal Design Power, or TDP) for these GPUs has not increased at the same pace. This is because of both the end of Dennard scaling and also cooling limitations in servers. Since GPU compute capability scales with available power, GPU performance is bottlenecked by the

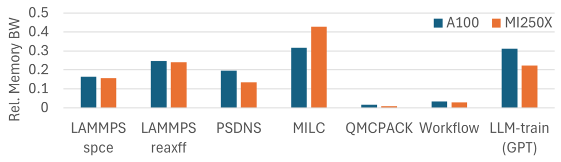


Figure 2. Average memory bandwidth relative to peak bandwidth on NVIDIA A100 and AMD MI250X

power wall sooner than the memory (bandwidth) wall. Figure 1 shows this trend in action, where memory bandwidth to TDP ratio is shown to continually increase across GPU generations from both AMD and NVIDIA.

With this increased bandwidth allocation for available compute/power, many GPU applications are less memory (bandwidth) bound. Figure 2 shows that average memory bandwidth (measured as memory utilization) consumed by various popular HPC/AI applications (defined in Table 1) on both NVIDIA A100 and AMD MI250X GPU does not exceed 50% of peak bandwidth even for AI/ML applications, and investigating further, only MILC and LLM-train (GPT) spike to at most 70-80% peak bandwidth. The latest GPUs have even higher bandwidth as shown in Figure 1, with the trend expected to continue with rapidly evolving high bandwidth memory (HBM) in the coming years. Consequently, the traditional approach of finding memory bound phases and saving power by lowering clock does not generally yield the desired benefit anymore. *Alternative approaches must be sought.*

GPU Performance counters and their problems. In order to optimize for energy, it is important to not only measure power but also performance at different clock rates. While measuring performance is effectively supported through low-overhead performance counters easily exposed to the user through various APIs *on CPUs*, collecting most of the performance counters is either associated with prohibitive overhead *on GPUs* (details in Section 3.4), or possible through intrusive application instrumentation that is often not viable.

In addition to overhead, another significant challenge is the consistency in terms of offered performance events by different GPUs. In this regard, there is already significant disparity among CPU vendors [36] such as Intel and AMD, and there is similar (or worse) disparity among GPU vendors (such as AMD and NVIDIA) in terms of performance events that could be collected. As a result, if an approach relies on a specific set of events on AMD GPUs, the approach is often not transferable to NVIDIA GPUs, and vice-versa.

GPUs and the lack of a practical energy saving tool. While prior art proposes solutions pertaining to GPUs, they rely on (1) hardware changes (such as collecting fine-grained information about performance sensitivity to frequency in a low-overhead manner [11]), or (2) some form of initial profiling of applications on the target system, or (3) offline training (e.g., a machine learning model) to predict power and performance based on observed performance events. In the first scenario, a solution depends on the corresponding GPU vendor to actually implement the proposed changes in hardware (which often does not materialize), and still may not be applicable for other vendors. In the second scenario, a solution requires an application to be run apriori on the system for it to be characterized for subsequent runs [7] - this is often not possible or not acceptable on real systems running many applications from many users. Also, an application's behavior could also change across runs owing to change in input, available power, host hardware [25, 42], etc. In the third scenario, a solution requires extensive offline training of a model on relevant applications using select performance counters to predict power and performance of test applications [8]. Given the disparity in events across processors, this approach is not portable. Moreover, it still requires an apriori run of the application to collect the relevant events for the model to predict accurately for the subsequent runs. In a nutshell, many of these approaches face challenges in practical adoption on GPUs and GPU-based systems that often have many users and many applications.

In this work, we propose Everest, an effective yet practical tool for dynamic energy savings in GPU-based systems. We make the following contributions that address the above-mentioned challenges:

1. Everest finds two novel energy saving opportunities in addition to the traditional approach of targeting memory bound phases, resulting in an effective solution with much improved energy savings on GPUs. These opportunities include finding optimal points of operation on the voltage-frequency (or power-frequency) curve of a GPU to achieve target performance (based on wall clock time), and adjusting GPU clocks based on observed utilization metrics of applications.
2. Everest is versatile. We demonstrate that memory utilization, which is easily available and accurate across GPU vendors and generations, is an effective metric in predicting an application's performance at different frequencies. Everest thus works on GPUs from different vendors that each have widely varying support in terms of available performance events.
3. Everest is a fully online/runtime solution that does not rely on apriori application profiling or model training. Everest accurately predicts performance of application phases with low overhead, and exploits the above-mentioned opportunities with DVFS for power/energy savings. This is achieved through employing a low-overhead process on each system node that dynamically collects select performance events and attributes them to application phases. Given Everest's ability to predict performance, Everest provides desired performance guarantees while saving energy.
4. We demonstrate the effectiveness of Everest on popular HPC and AI/ML applications on NVIDIA A100

and AMD MI250X. We find that Everest outperforms existing approaches and saves on average 20% more power and 16% more energy on A100 and 15% more power and 8% more energy on MI250X when meeting a 90% performance target, while also addressing the above-mentioned challenges.

2 Related Work

Our work builds on the extensive literature on Dynamic Voltage and Frequency Scaling (DVFS) for CPUs. We describe the limitations of these approaches when applied to GPUs, which contribute to the shortcomings of recent GPU works.

DVFS Background. Many studies [22, 23, 28, 45] demonstrate the significant potential for power and energy savings in CPUs and GPUs through DVFS. Effectively achieving these savings requires modeling power and performance at different voltage and frequency settings. Early works in CPUs [12, 19] developed highly accurate power models by identifying performance counters with the highest correlation with power consumption, with similar recent works for GPUs [7, 8]. However, modeling performance is far more complex, where a multitude of factors such as memory stalls, communication, and load imbalance can affect performance.

Many CPU works focus on identifying compute and memory phases [17, 20, 29, 37, 43], given that performance of compute phases scales proportionally with frequency while memory phases do not. These phases are identified using various performance events and most often include IPC (instructions per cycle) or on-chip/off-chip memory accesses [20, 21, 24, 26, 27], although some works measure metrics such as leading loads [17, 43] and critical path load latency [37] to account for memory-level parallelism and the nature of the memory subsystem. Other CPU works address the savings opportunities more commonly available in parallel workloads [16], including communication idle time [49] and load imbalance [10, 44, 52].

Limitations of CPU Solutions. Many works extensively analyze all available performance counters on various benchmarks to find the highest-correlating counters for power and performance estimation [8, 12, 47]. Since many of these counters are specific to a given system, these analyses are often not relevant on a different system and must be performed again. Once the counters are selected, either through these correlation studies or by first deriving the prediction model [20, 51, 53], they are most often used in either offline training or online analysis. Offline training of an analytical or ML model involves profiling with the selected metrics, typically at each available DVFS setting [8, 51]. These offline methods can be useful for a target system with known applications, but they otherwise assume an accurate system characterization which requires a comprehensive set of benchmarks that can account for the range in application behavior.

Other approaches monitor these counters at runtime, addressing some training/portability issues from offline analysis. Since these runtime approaches must perform online characterization, they may measure performance impact from frequency changes at the application-granularity [24] or at the phase/function-granularity [6]. Other approaches instead directly model performance to predict an appropriate frequency/powercap using feedback control for the next time interval [20, 21, 52].

In practice, modern operating systems have not adopted any above methods of application characterization for saving power/energy; the Linux ondemand governor for instance uses the simplest characterization, decreasing frequency when CPU utilization is low (i.e., fully idle) for no performance loss, but otherwise running at the highest clock.

GPU Solutions. Although numerous performance models have been proposed for CPUs, both offline [4, 50] and online [11, 38] models for GPUs are still emerging. In particular, the state-of-the-art online performance models from CRISP and PCSTALL build on the work from previous CPU approaches and monitor load critical path and load/store stalls [38] or IPC [11], but propose *hardware-based* solutions. We find this is because very few GPU performance counters allow for the same level of easy access and low overhead that CPU performance counters benefit from.

Software-based solutions do exist but must make use of the few performance metrics available at low overhead. Ali et al. derive offline power and performance models using both analytical [7] and ML [8] approaches, relying on the FP_Active and DRAM_Active metrics available on NVIDIA GPUs. However, they are unable to accurately predict performance for applications which are not memory bound whose performance is not affected by DVFS. Identifying this class of applications is crucial to make significant power/energy savings for no performance loss. Moreover, as offline models, they require an apriori run of the desired application to collect the necessary metrics, and their reliance on FP_Active prevents portability to AMD GPUs, where this same metric is not available.

EAR [15] and GEOPM [13] are two industry-usable approaches which employ software-based runtime solutions and primarily rely on GPU Utilization. In fact, EAR is currently in use in SuperMUC-NG, a supercomputer of the Leibniz Supercomputing Centre (LRZ) which sits at number 64 in the TOP500. In both EAR and GEOPM, GPU utilization is used similarly to the simple Linux CPU characterization, where the frequency/powercap of a GPU is scaled by its utilization. Specifically in EAR, powercaps are also redistributed among GPUs in the same node, so that the highest-utilized GPUs retain the most power. As a result, savings are primarily made when idling, and both approaches are unable to detect opportunities for power/energy savings during high GPU utilization. GEOPM also makes use of SM_Active, an NVIDIA-specific metric, to better reflect active cores.

3 Key Insights and Design Choices

We discuss key insights that motivate both the energy saving strategies and the choice of characterization strategy in Everest.

3.1 Voltage-Frequency Profile

While frequency control is easily accessible by the user, voltage control is typically controlled by the hardware to guarantee correct and stable operation. Figure 3 depicts the default voltage-frequency profiles on A100 and MI250X.

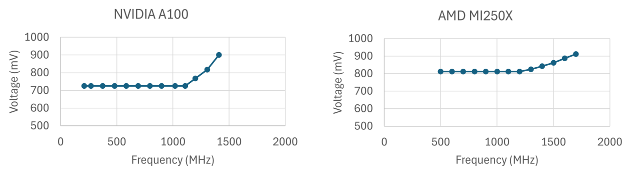


Figure 3. Default voltage-frequency curves for NVIDIA A100 and AMD MI250X.

Dynamic power is directly proportional to frequency and the voltage squared, so DVFS should allow for significant savings as both voltage and frequency decrease. However, as the figure indicates, voltage only decreases up to a certain point, at 1110MHz and 1200MHz for A100 and MI250X, respectively (and correspond to 79% and 70% of the maximum clock, respectively). Furthermore, the rate at which voltage increases with frequency (indicated by the slopes in the figure) is different across the two GPUs - A100 has a steeper slope than MI250X.

Implications: Decreasing frequency beyond this point will provide diminishing returns in power savings in exchange for performance loss and as a result will likely increase energy expenditure. Moreover, there can be considerable difference in how GPU vendors choose to vary voltage with frequency, which ultimately has a significant bearing on power and energy savings possible on respective GPUs.

3.2 Frequency-Power Profile

Figure 4 shows frequency-power profiles from popular HPC and AI/ML applications (descriptions in Section 6) on both A100 and MI250X. We observe for A100 that at clocks where voltage increases (i.e., beyond 1110MHz), the power begins increasing super-linearly with frequency in all applications. We also note that the absolute power consumption is significantly higher for the ML application owing to heavy computations involved. On the other hand, for MI250X we observe that power increases linearly or even sub-linearly with frequency due to the much smaller frequency-voltage slope. Therefore, it is especially important to accurately predict performance to make energy savings for MI250X.

Implications: A100 energy-inefficiency at the top end of the frequency spectrum is especially significant for AI/ML

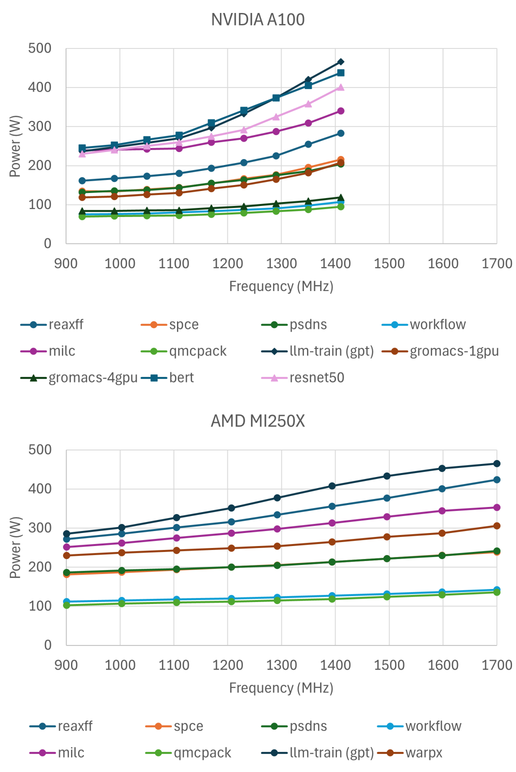


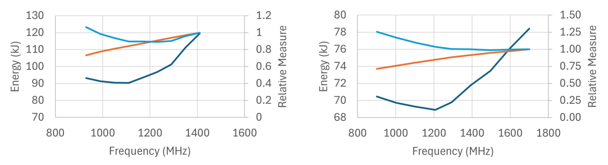
Figure 4. Frequency-power profile of GPU applications on NVIDIA A100 (top) and AMD MI250X (bottom)

applications that burn a lot of power. Even for general purpose HPC applications, this implies enhanced energy saving opportunities. MI250X saves less power comparably as frequency decreases, but can still save energy depending on performance.

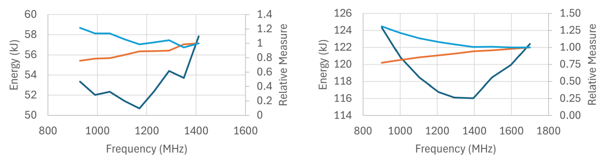
3.3 Frequency-Performance-Energy Profile

Figure 5 shows how performance, energy, and relative EDP (Energy Delay Product) change with frequency for select applications. Performance is calculated using wall clock time. We find that performance in some applications (LAMMPS reaxff) varies almost proportionally with frequency. Other applications' performance (Workflow) changes at a slower rate than frequency, and a few (WarpX/GROMACS) change very minimally with frequency. Interestingly, these differences in frequency-sensitivity occur despite applications no longer being bound by memory bandwidth, as noted earlier. We elaborate on this in next subsection.

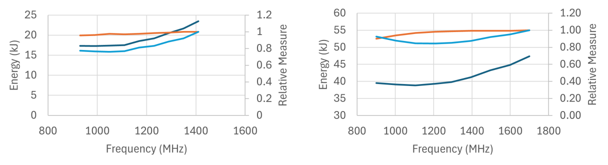
Additionally, many previous works optimize for energy or EDP. These figures show how on GPUs this often comes with 20% performance loss, if no performance threshold is defined, which is not always acceptable to users. Fortunately, significant power and energy savings can still be made even at low acceptable performance loss, especially for A100 but even on MI250X.



(a) LAMMPS reaxff.



(b) Workflow



(c) GROMACS-4gpu (left, A100), WarpX (right, MI250X)

Figure 5. Relative performance, energy and EDP as frequency changes on A100 (left), MI250X (right)

Moreover, these profiles illustrate that minimum energy and EDP are unlikely to be found at frequencies below a certain threshold, due to the voltage plateau and heavy performance loss beyond this point. As a result, we limit our analyses to frequencies above 900MHz. This also allows for a linear approximation of performance change with frequency (with different slopes for different applications), giving an average Pearson's correlation coefficient (R^2 value) of 0.97 for A100 and 0.94 for MI250X.

Implications. For low (<20%) performance loss thresholds, we can simply optimize for minimum power (i.e., choose the lowest frequency to meet the performance target) to achieve maximum energy savings. We can also model performance as linear with frequency change, as long as the slope can be determined.

3.4 Choice of Characterization Strategy

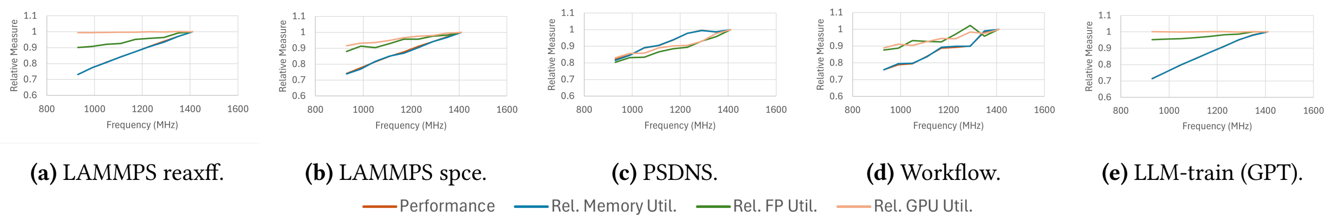
In addition to finding optimal points of operation on the voltage-frequency curve, there are other application-dependent scenarios where energy savings become possible from lowered application clocks. These scenarios are listed below. The chosen characterization strategy must be able to identify all of these scenarios across applications.

1. Many GPU applications utilize the asynchronous kernel launch and memory transfer features on modern GPUs to overlap the GPU kernel execution with CPU code and/or GPU kernel execution with memory transfers between device and host. As a result, in certain scenarios, application performance may not be limited by kernel execution time but by either the CPU or the memory transfer time. In such cases, application performance is less sensitive to GPU core clocks.
2. In GPU applications, there is also often a non-negligible overhead associated with the many GPU kernel launches. This overhead is independent of clock rate.
3. The traditional scenario of memory bound application phases that are less sensitive to GPU clocks.
4. In real applications, overall performance could be less sensitive to GPU core clocks due to a complex combination of the above factors.

The simplest and complete characterization strategy, of course, is to directly measure sensitivity of application performance to frequency, where performance is given by instructions executed per second as done in prior work on CPUs [21, 24, 26]. However, this is not possible on GPUs and alternatives are essential.

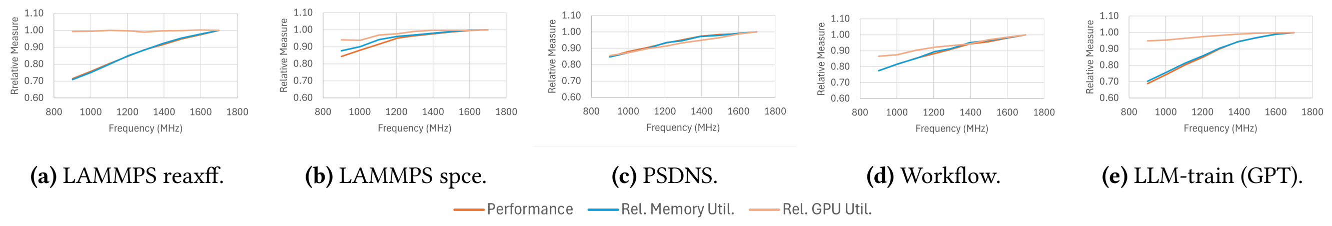
Available Performance Counters on GPUs. Profiling on GPUs is associated with significant overhead for two reasons. First, construction of calling contexts involves stack walking which can be very expensive especially for complex applications with deep call paths. For instance, we observe an overhead of 1.5x with NVIDIA Nsight Systems Profiler [3] on `gromacs`. Second, for obtaining fine-grained performance metrics that are useful for kernel characterization such as IPC, achieved occupancy, etc., GPU kernel execution needs to be serialized for correct attribution of instruction samples. This introduces prohibitive overhead and significantly alters the original application behavior itself. We observe NVIDIA Nsight Compute Profiler [2] to introduce >3x overhead even when choosing to profile select instances of the most important kernel in `gromacs`. Clearly, this is not useful for production runs.

Both NVIDIA and AMD, however, do provide a limited set of performance events in sampling mode where performance counters are not associated with kernels executed and thus do not require kernel serialization. NVIDIA and AMD vary considerably in their offerings of these low-overhead events. For instance, NVIDIA provides FP64/32/16 utilization events (i.e., FP_Active, which measures floating point activity, specifically the fraction of cycles the FP64/FP32/FP16 pipes are active), whereas these are absent for AMD. AMD, on the other hand, offers various core and cache-specific events that are absent for NVIDIA. The only common events offered by both are GPU Utilization and Memory Bandwidth Utilization (*MBU*). Since GPU utilization only captures the percentage of time the GPU is busy/active, it cannot, for instance, differentiate between compute bound and memory bound phases. We next show that the latter of these two events is a direct indicator of performance and is therefore appropriate for finding energy saving opportunities. Memory bandwidth utilization is calculated as follows.



(a) LAMMPS reaxff. **(b)** LAMMPS spce. **(c)** PSDNS. **(d)** Workflow. **(e)** LLM-train (GPT).

Figure 6. NVIDIA A100. Relative utilization vs. performance.



(a) LAMMPS reaxff. **(b)** LAMMPS spce. **(c)** PSDNS. **(d)** Workflow. **(e)** LLM-train (GPT).

Figure 7. AMD MI250X. Relative utilization vs. performance.

$$MBU = \frac{Data\ Transferred}{WCT} \qquad (1)$$

The numerator in the equation is the amount of data transferred to and from the GPU memory. The denominator is the Wall Clock Time (WCT), or the inverse of performance, which could be a function of GPU kernel time, and in some cases the CPU-GPU transfer time and CPU time as noted above. Since the data transferred over time is independent of GPU core frequency (stays constant as long as memory frequency is constant), Everest can achieve its goal of learning sensitivity of performance to frequency by observing MBU's sensitivity to frequency.

3.5 Characterization Study

We study the same applications to validate our choice of characterization strategy, measuring GPU Utilization and Memory Utilization at each available frequency. Additionally on A100, FP Utilization is available and was used for power/performance prediction in [7, 8], so we measure it as well.

To visualize how these metrics correlate with performance, we calculate each utilization relative to its value at maximum frequency, and plot with performance. Also, since GPU and FP Utilization increase as frequency decreases, we take the inverse. Figures 6 and 7 validate that Memory Utilization changes *almost exactly proportionally* to performance on both NVIDIA A100 and AMD MI250X, even on the applications not shown. As such, Memory Utilization has an average correlation coefficient of 0.997 across all applications on both GPUs, with a minimum of 0.97 for MILC on A100. GPU Utilization and FP Utilization on the other hand change at a much slower and varying rate with frequency. GPU Utilization achieves an average correlation coefficient of 0.80, with a minimum of 0.03 for LLM-train on A100 and 0.54 for reaxff on MI250X. FP Utilization achieves an average correlation coefficient of 0.93, with a minimum of 0.76 on Workflow.

Application phases. It is important to note that these measurements were collected and averaged over the entire application. But, the utilization metrics reflect phases too, as Figure 8 demonstrates. Thus, we can operate at a finer granularity and still make use of Memory Utilization.

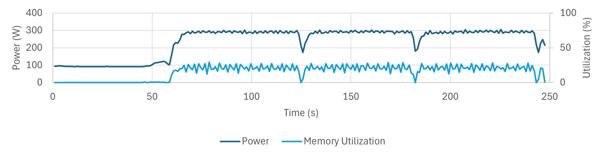


Figure 8. LAMMPS spce on MI250X at 1700MHz (max frequency). Memory Utilization indicates phases.

4 Our Approach

We discuss how Everest makes use of the design choices discussed above.

4.1 A Versatile Framework

As discussed in the previous section, Memory Utilization provides a low overhead avenue to accurately inform about performance at different frequencies across both NVIDIA and AMD GPUs. However, it would be ideal to characterize the application at runtime for a useful and practical solution. Knowing that performance can be approximated to vary linearly with frequency as discussed in Section 3.3, we find it sufficient to simply measure memory utilization (for a phase) at the maximum frequency and a lower frequency. This essentially allows us to derive the slope of the performance curve. We thus calculate frequency-sensitivity (FS) using the Memory Utilization measured at two specific frequencies, high and low, as follows:

$$\%FS = 100\% * \frac{\frac{Mem_{high}}{Mem_{low}} - 1}{\frac{Freq_{high}}{Freq_{low}} - 1}, 0\% \leq \%FS \leq 100\% \quad (2)$$

For example, if the frequency ratio is 1.5 (i.e., 1.5 GHz / 1GHz), but the Mem ratio is 1.25 (i.e., performance at 1.5GHz is only 25% faster than at 1GHz), then we consider that to be 50% frequency-sensitive. Note that the 'low' frequency should still be fairly high (e.g., 70% of the maximum) to minimize characterization overhead incurred from running the application at a low frequency.

4.2 Enhancing Energy Saving Opportunities

As noted earlier, considerable energy savings could be made if some performance could be sacrificed. However, it is crucial for both the user to define and for the tool to accommodate an acceptable performance loss in such a scenario. Everest, therefore, incorporates a performance degradation variable (PD) that provides a target performance to meet. PD is a percentage relative to performance at peak clock frequency. For example, 5% PD means that Everest should provide at least 95% relative performance. Since Mem Util represents performance, we define PD in terms of Mem Util as follows.

$$PD = 1 - \frac{Mem_{high}}{Mem_{low}} \quad (3)$$

Lastly, in order for Everest to guarantee the indicated level of performance while maximizing power and energy savings, it needs to calculate the ideal frequency for the phase. We combine equations 2 and 3 as follows.

$$\frac{\frac{1}{1-PD} - 1}{\%FS} + 1 = \frac{Freq_{high}}{Freq_{low}}$$

$$Freq_{ideal} = \frac{Freq_{high}}{1 + \frac{PD}{\%FS(1-PD)}} \quad (4)$$

Therefore, the ideal frequency is a function of %FS and the acceptable performance degradation, scaled by the maximum possible frequency.

5 Implementation

Using the design outlined above, we propose Everest, a cross-platform runtime tool that saves power and energy while meeting a desired performance target. One instance of Everest is launched per node per job, with each instance independently characterizing the application and making DVFS decisions (i.e., without global synchronization). This launching procedure is done by integrating with the job scheduler, which also directly provides the needed permissions for each launched Everest process to change clock frequencies.

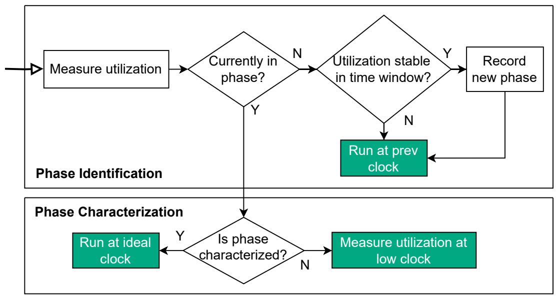


Figure 9. Everest design overview.

Upon launching, Everest monitors a single application process on the same node and periodically loops through three stages: phase identification, phase characterization, and frequency scaling. Figure 9 presents these stages through a flowchart, which we expand on here.

5.1 Phase Identification

Everest first needs to identify the current application phase. The granularity of the phase must be sufficiently large to measure the desired performance metric and capture the application phase as well as to perform the actual frequency scaling.

We define phases using a phase signature of GPU and memory utilization levels, where maintaining a stable average GPU/memory utilization for a window of time (e.g., 5s) reveals a phase. A significant change in utilization (e.g., 10%) indicates a phase change. While GPU Utilization is not used for performance prediction, we use it in our phase signatures to distinguish between phases where the GPU is being utilized but with very low/zero memory utilization and initialization phases where both GPU and memory utilization are very low/zero.

5.2 Phase Characterization

Secondly, Everest needs to characterize the phase to determine its frequency sensitivity. As Figure 8 suggests, we find that application phases tend to repeat (i.e., from a loop) which means we can benefit from storing this characterization information and thus reduce overhead from repeated characterization.

If in a phase that has not been characterized yet, Everest runs at a lower (e.g., 30% lower than max) frequency for the same window of time to measure (average) memory utilization in the desired phase.

GPU and memory utilization is obtained using DCGM (Data Center GPU Manager [39]) for NVIDIA GPUs (GR_ENGINE_ACTIVE and DRAM_ACTIVE) and ROCm SMI [9] for AMD GPUs (average_gfx_activity and average_umc_activity). Everest samples utilization values at the fastest rate of 1ms (NVIDIA A100) and 10ms (AMD MI250X) and takes the average every 1s for decision-making.

5.3 Frequency Scaling

Finally, Everest performs DVFS for the current phase, represented by the the green boxes in Figure 9.

If we are in a characterized phase, the ideal core frequency is calculated using Equation 4. We limit lowering GPU frequency to about 55% of the maximum for reasons discussed in Section 3.3. Frequency scaling commands are issued using nvidia-smi (NVML [1]) for NVIDIA and ROCm SMI for AMD. GPU frequencies are available at a fine granularity (e.g., 15MHz and 5MHz steps, respectively), so we simply round up to the next frequency step.

5.4 Phase Detection Sensitivity Analysis

Everest's ability to identify phases at runtime is crucial for accurate and effective operation, and is largely influenced by the chosen window time length. Figure 10 provides a subset of our sensitivity analysis of the window parameter for two applications with differing workload characteristics in terms of the length and repetition of application phases. LAMMPS spce repeats the same phase, each lasting 10s on A100 and 50s on MI250X. MILC has more unpredictable behavior with sometimes no discernible phase and sometimes 10s-90s phases.

As seen in Figure 8, individual samples of utilization can have high variance despite giving a consistent average. Since we are also unable to associate individual samples of GPU/memory utilization to a specific kernel or loop (without instrumentation), using a window (i.e., taking the average) allows Everest to account for the highly erratic behavior of individual samples. From a source code perspective, this also means Everest's phases often contain multiple GPU kernels. To mitigate errors, if application phases are too short and/or erratic, Everest will not detect any phases and remain at default (maximum) frequency, preventing a misprediction (of %CB), as observed for smaller window sizes. On the other hand, if the window size becomes larger than the application phase, then Everest may miss finer granularity phases. As a middle ground for the best phase detection and performance prediction on both NVIDIA A100 and AMD MI250X, we chose a 5s window length, but any window length that is long enough (>1s) is sufficient without much variability in results, as shown by Figure 10.

5.5 Overhead Analysis

Given the online approach adopted by Everest, it is important to consider potential overhead. First, since we characterize at runtime which involves lowering frequency to obtain a proper measurement, we could generate additional performance loss and thus extra energy expenditure. This overhead would be most detrimental for applications whose performance are extremely sensitive to frequency. We measure this overhead by running a version of Everest that only performs the frequency changes involved in characterization

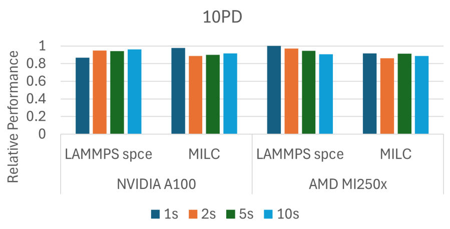


Figure 10. Relative performance of Everest with varying window lengths

but does not subsequently run at the calculated ideal frequency. We find that at most, in LAMMPS reaxff (which is the most frequency sensitive) on MI250X (which has the least aggressive voltage-frequency curve) this overhead generates an additional 0.5% performance loss and actually less energy expenditure due to the decrease in power from lowering frequency during characterization.

Secondly, since we characterize at runtime, we are losing potential power/energy savings (i.e., by not knowing the phase characterization ahead of time). These lost savings are demonstrated in Section 7.2, where Everest is compared to a static oracle which immediately runs at the ideal frequency for a given PD. Specifically, when Everest achieves the same performance prediction accuracy as the oracle, Everest saves at most 1.6% less energy on A100 (LAMMPS reaxff) and 2.1% less energy on MI250X (LLM-train).

6 Experimental Setup

Everest is evaluated on Cray EX systems containing the latest NVIDIA and AMD GPUs.

The NVIDIA system consists of nodes containing a single socket EPYC 7763 (Milan) 64-core CPU host and 4x NVIDIA A100 SXM 80GB GPUs. The AMD system consists of nodes containing a single socket EPYC 7A53 (Trento) 64-core CPU host and 4x AMD MI250X GPUs (8x GCDs). Both systems use Slingshot 11 high-speed network and run on the SLES15 SP4 operating system. A100 has a TDP of 500W with a frequency range of 210 MHz to 1410 MHz, while MI250X has a TDP of 560W with a frequency range of 500 MHz to 1700 MHz. Power measurements are obtained using PM counters available on Cray EX [34] and collect real power and energy data.

Everest is integrated with the Slurm job scheduler as a plugin [33] that accepts two additional (optional) parameters in addition to their normal job script: whether to enable Everest and the desired PD. For example, a user desiring to use Everest with 10% PD would submit:

```
srun -n ranks -use-everest -everest-pd=10
<other-standard-slurm-options> <executable>
```

We evaluate Everest with PD set to 5% and 10%. We additionally compare to the GPU approaches discussed in Section

2 from Energy Aware Runtime (EAR) [15] and Ali et al. [7], as well as a static oracle which chooses the ideal static frequency to meet the performance target.

We evaluate Everest using popular HPC and AI/ML benchmarks, summarized in Table 1.

Table 1. GPU Applications

Benchmark	NVIDIA/AMD	Domain	Input/Size
LAMMPS [48]	Both	HPC	SPC/E, 5.308 Million atoms
LAMMPS [48]	Both	HPC	ReaxFF, 5.308 Million atoms
MILC [14]	Both	HPC	nx=ny=nz=64; nt=96
PSDNS [31]	Both	HPC	Single precision; Problem size = 2048^3
Workflow [41]	Both	HPC/ML	Coupled compute/data; Size = 8^3
QMCPACK [30]	Both	HPC	NiO workflow; walkers=16; steps=10
LLM-train (GPT)[40, 54]	Both	AI	13B parameters, with DeepSpeed
GROMACS [5]	NVIDIA	HPC	lysozyme in water [32](1 & 4 GPUs)
BERT [35]	NVIDIA	ML	train_batch_size=56, learning_rate=0.000425
ResNet-50 [35]	NVIDIA	ML	train, batch-size=408, lr=10.5
WarpX [18]	AMD	HPC	max_steps=1000; nx = 1024; epsilon=0.01

7 Results and Discussion

Our evaluation of Everest involves assessing its ability to meet the indicated performance threshold while also being able to identify and ultimately exploit opportunities for power/energy savings, even across GPU vendors. We first examine the performance of Everest compared to the other approaches before presenting results optimizing for minimum power at each performance target and the subsequent energy savings achieved. All results are relative to running at maximum frequency.

7.1 Performance Evaluation

Figure 11 compares the relative performance of EAR, Ali et al., Everest, and the static oracle, at 5% and 10% PD. EAR-100% and EAR-75% indicate the % TDP at which the GPU power cap is set (e.g., 2000W and 1500W for 4x A100).

We first observe that EAR-100% achieves the same performance as running at the maximum frequency. Given that EAR attempts to maximize performance under a power cap, there is almost no benefit when provided a power cap equal to the TDP. Additionally, since EAR redistributes power among GPUs in a single node based only on utilization, we see instances of unintended performance loss when utilization is constantly changing and/or is imbalanced among GPUs, such as in Workflow. EAR-75% begins to show the limitations of the overall approach. Once the power cap is low enough, applications receive varying amounts of performance degradation. For instance, LAMMPS reaxff, LLM-train, and MILC on MI250X lose 7%, 11%, and 27% performance, respectively. These applications make full use of all 8 GPUs on their individual nodes, so the internal power redistribution sees no benefit. Even if power could be distributed between nodes, GPU utilization is not sufficient for determining which nodes demand the most power. For instance, QMCPACK on MI250X has 100% GPU utilization with only 136W average power consumption, while LLM-train consumes 480W on average with 85% GPU utilization. Thus, EAR crucially lacks an accurate performance prediction mechanism, limiting its opportunities for energy savings, as seen in the next subsection.

Next, we similarly observe that Ali-FP runs at the maximum frequency in all instances, except for QMCPACK. We find that this is a result of insufficient training of the analytical model on only DGEMM and STREAM benchmarks. While these benchmarks do represent the extremes of compute-boundedness and memory-boundedness, with DGEMM and STREAM achieving close to 100% and 0% FP activity respectively, they are not sufficient by themselves to capture the range of FP behavior exhibited by real-world applications. For instance, most of these applications at maximum frequency demonstrate an average FP activity between 3-38%. Only QMCPACK is (incorrectly) predicted to have low performance loss when running at 700MHz, due to its low FP activity (0.4%) coinciding with STREAM's FP activity. Notably, Ali-FP is not portable to AMD and thus is excluded from AMD evaluation.

Everest delivers the closest performance to the static oracle, with an average performance of 96% and 93% at 5PD and 10PD respectively on A100 and 98% and 93% at 5PD and 10PD respectively on MI250X. With the inclusion of measuring utilization at a *single* lower frequency, Everest improves its ability to predict performance, ultimately allowing for better energy savings.

7.2 Power and Energy Savings

Figures 12 and 13 compare the relative power and energy of all approaches, at 5% and 10% PD. [We include standard deviation error bars for the energy results, to indicate the observed low measurement variability.]

We first observe minimal savings from EAR-100%. While EAR can make power/energy savings when the GPU is idling (and thus guarantee no performance loss), these opportunities typically only appear during initialization. For instance, this allows for 16% power/energy savings for QMCPACK on A100. Even when savings can be achieved at no performance loss, such as in WarpX, EAR is unable to identify these opportunities. While EAR-75% does save more power/energy, it comes at the cost of unacceptable performance loss which disproportionately affect the most power-hungry applications. Similarly, Ali-FP makes limited savings, but only at the cost of 21% performance loss in QMCPACK.

Owing to its accurate performance prediction, Everest is able to make significant power and energy savings while meeting the performance guarantee. On A100, Everest achieves on average 16% power and 13% energy savings with 5PD and 23% power and 17% energy savings with 10PD. Due to the less favorable voltage-frequency curve on MI250X, Everest achieves somewhat less but still significant savings, with on average 9% power and 7% energy savings with 5PD and 15% power and 8% energy savings with 10PD.

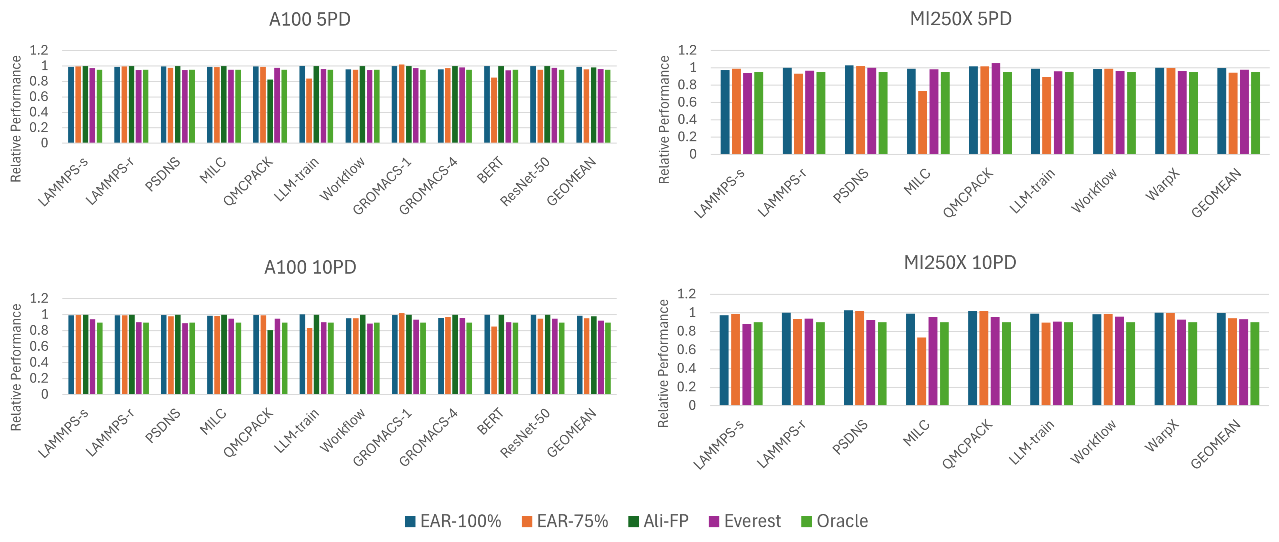


Figure 11. Relative performance of approaches relative to maximum frequency.

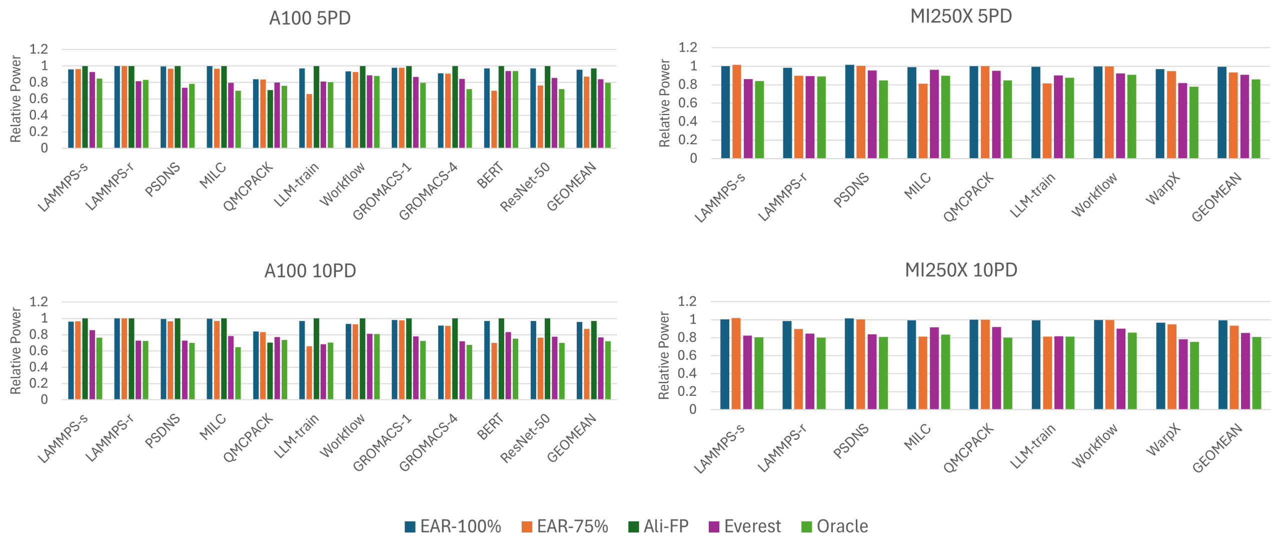


Figure 12. Relative power of approaches relative to maximum frequency.

For instance, Everest detects that savings are available at minimal performance loss in GROMACS-4 on A100 and WarpX on MI250X despite not being memory bound – even Everest (5PD) gets 16% power savings and 14% energy savings in GROMACS-4 and 18% power savings and 15% energy savings in WarpX. The key differentiation in Everest is its ability to detect when application performance is not bottlenecked by kernel execution time in these scenarios, based on observed memory utilization at different frequencies. For the other applications, Everest still makes significant savings by incorporating PD and accurately trading for performance. For example, Everest (10PD) in PSDNS achieves 27% power savings and 19% energy savings for 10.7% performance loss on A100 and 16% power savings and 10% energy savings for 7.6% performance loss on MI250X. These available savings become even greater for the AI/ML applications, especially on A100 – Everest (10%) in LLM-train achieves 32% power savings and 25% energy savings in exchange for 9.5% performance loss, and in ResNet-50 achieves 22% power savings and 19% energy savings for 5% performance loss. We believe this is a significant and promising result given the recent explosion of generative AI and could lead to energy-efficient use of the technology.

Lastly, while Ali et. al's analytical approach is less efficient than Everest, their ML performance model retains the same dependence on FP_Active, preventing evaluation on AMD

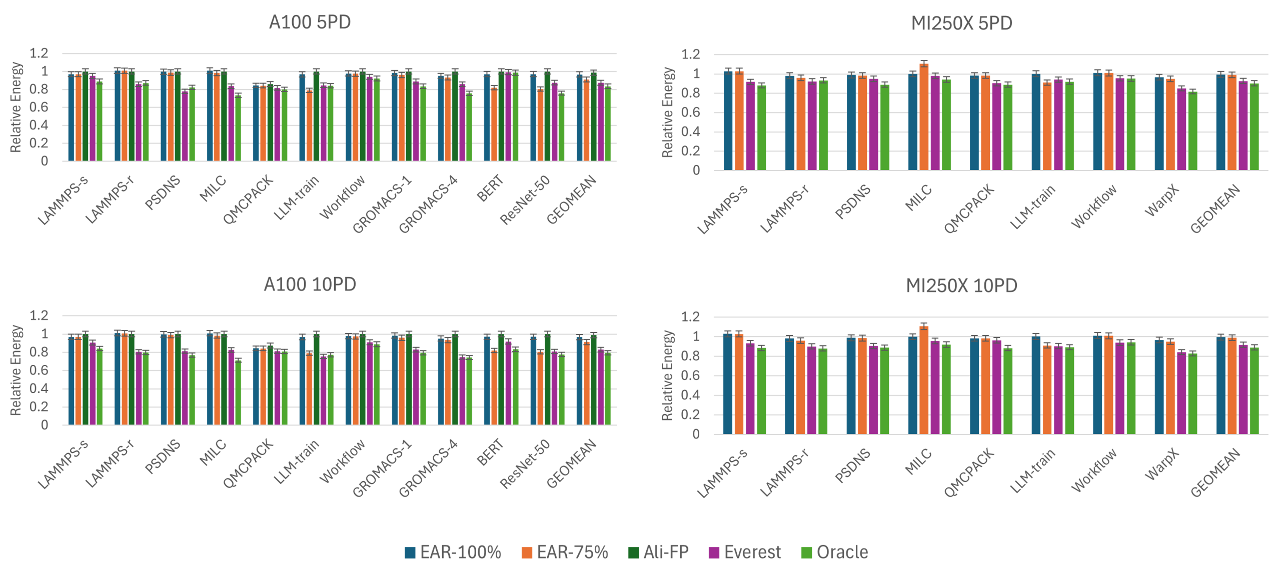


Figure 13. Relative energy of approaches relative to maximum frequency.

GPUs, in addition to the required apriori profiling of the desired application. Moreover, Everest on average achieves within 2% performance of the static oracle, only losing out on 4% power and energy savings on average. Thus, we demonstrate how Everest benefits from its novel identification of energy-saving opportunities across GPU vendors, while also avoiding the practical complications in an offline approach.

7.3 Future Work

Co-located applications. Our immediate future work includes extending Everest for co-located applications (done either through space or time sharing) due to their prevalence in data centers/cloud.

When time-sharing, we expect Everest to naturally adapt to applications that are context switching, where the arrival of a new application is similar to the onset of a new phase. The space-sharing scenario (i.e., applications share compute/memory resources) is more complex since the available runtime metrics are measured for the entire GPU rather than individual applications.

Assuming two co-located (space-sharing) applications, we arrive at three possible cases: (1) two frequency-sensitive (compute-bound) applications, (2) two frequency-insensitive (memory-bound, etc.) applications, (3) one frequency-sensitive and one frequency-insensitive application. In the first two cases, Everest can more easily ensure desired application performance for individual applications given their similar natures. In the third case, while Everest cannot guarantee precisely desired performance for individual applications (in this case, the chosen frequency affects the performance of the frequency-sensitive application more than desired) due to GPU-wide characterization, Everest could still ensure the desired GPU throughput, which is a more suitable metric in this scenario.

I/O operations. We also investigated NVLink and PCIe utilization for NVIDIA GPUs. For instance, our evaluated applications use both direct GPU-to-GPU communication (e.g., PSDNS, MILC, LLM-train), and via the CPU (e.g., GROMACS). We observe that although memory utilization and NVLink utilization are correlated (indicating communication overlapped with computation), there are certain sampling intervals (e.g., in MILC) where high NVLink bandwidth is seen with 0% memory utilization, implying that GPU-to-GPU communication (although involving reads from global memory) does not show as memory utilization due to the two traffics using separate interfaces. The occurrence of such intervals, however, indicates purely communication bound phases and could be utilized for additional energy savings, though memory utilization still captures the significant majority of energy-saving scenarios.

8 Conclusion

We identified three challenges towards a runtime energy saving solution in GPUs which Everest addresses - effectiveness in finding energy saving opportunities, versatility to operate across processor vendors, and independence from apriori profiling and training. Everest achieves versatility by relying on a single easily available and accurate performance metric for finding energy saving opportunities. Owing to its unique frequency sensitivity characterization and accurate performance loss prediction, Everest finds enhanced energy saving opportunities fully at runtime and thus saves on average 16% more energy than existing solutions on NVIDIA A100 and 8% more energy on AMD MI250X.

References

[1] 2023. NVIDIA Management Library (NVML). https://docs.nvidia.com/deploy/nvml-api/

[2] 2023. NVIDIA Nsight Compute Profiler. https://developer.nvidia.com/nsight-compute

[3] 2023. NVIDIA Nsight Systems Profiler. https://developer.nvidia.com/nsight-systems

[4] Y. Abe, H. Sasaki, S. Kato, K. Inoue, M. Edahiro, and M. Peres. 2014. Power and Performance Characterization and Modeling of GPU-Accelerated Systems. In *2014 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE Computer Society, Los Alamitos, CA, USA, 113–122. https://doi.org/10.1109/IPDPS.2014.23

[5] Mark James Abraham, Teemu Murtola, Roland Schulz, Szilárd Páll, Jeremy C. Smith, Berk Hess, and Erik Lindahl. 2015. GROMACS: High performance molecular simulations through multi-level parallelism from laptops to supercomputers. *SoftwareX* 1-2 (2015), 19–25. https://doi.org/10.1016/j.softx.2015.06.001

[6] Bilge Acun, Kavitha Chandrasekar, and Laxmikant V. Kale. 2019. Fine-Grained Energy Efficiency Using Per-Core DVFS with an Adaptive Runtime System. In *2019 Tenth International Green and Sustainable Computing Conference (IGSC)*. 1–8. https://doi.org/10.1109/IGSC48788.2019.8957174

[7] Ghazanfar Ali, Sridutt Bhalachandra, Nicholas J. Wright, Mert Side, and Yong Chen. 2022. Optimal GPU Frequency Selection using Multi-Objective Approaches for HPC Systems. In *2022 IEEE High Performance Extreme Computing Conference (HPEC)*. 1–7. https://doi.org/10.1109/HPEC55821.2022.9926317

[8] Ghazanfar Ali, Mert Side, Sridutt Bhalachandra, Nicholas J. Wright, and Yong Chen. 2023. Performance-Aware Energy-Efficient GPU Frequency Selection using DNN-based Models. In *Proceedings of the 52nd International Conference on Parallel Processing* (Salt Lake City, UT, USA) *(ICPP '23)*. Association for Computing Machinery, New York, NY, USA, 433–442. https://doi.org/10.1145/3605573.3605600

[9] AMD. [n. d.]. ROCm System Management Interface (ROCm SMI) Library. https://rocm.docs.amd.com/projects/rocm_smi_lib/en/docs-5.2.0/index.html. (accessed Nov 18, 2023).

[10] Sridutt Bhalachandra, Allan Porterfield, Stephen L. Olivier, and Jan F. Prins. 2017. An Adaptive Core-Specific Runtime for Energy Efficiency. In *2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. 947–956. https://doi.org/10.1109/IPDPS.2017.114

[11] Srikant Bharadwaj, Shomit Das, Kaushik Mazumdar, Bradford M. Beckmann, and Stephen Kosonocky. 2024. Predict; Don't React for Enabling Efficient Fine-Grain DVFS in GPUs. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 4* (Vancouver, BC, Canada) *(ASPLOS '23)*. Association for Computing Machinery, New York, NY, USA, 253–267. https://doi.org/10.1145/3623278.3624756

[12] W.L. Bircher, M. Valluri, J. Law, and L.K. John. 2005. Runtime identification of microprocessor energy saving opportunities. In *ISLPED '05. Proceedings of the 2005 International Symposium on Low Power Electronics and Design, 2005.* 275–280. https://doi.org/10.1145/1077603.1077668

[13] Christopher Cantalupo and Brad Geltz. [n. d.]. GEOPM - Global Extensible Open Power Manager. https://github.com/geopm/geopm. (accessed May 25, 2023).

[14] MIMD Lattice Computation (MILC) collaboration. [n. d.]. MILC collaboration code for lattice QCD calculations. https://github.com/milc-qcd/milc_qcd. (accessed May 25, 2024).

[15] Julita Corbalan, Lluis Alonso, Jordi Aneas, and Luigi Brochard. [n. d.]. Energy Aware Runtime (EAR). https://github.com/eas4dc/EAR. (accessed May 25, 2023).

[16] Julita Corbalan, Lluis Alonso, Jordi Aneas, and Luigi Brochard. 2020. Energy Optimization and Analysis with EAR. In *2020 IEEE International Conference on Cluster Computing (CLUSTER)*. 464–472. https://doi.org/10.1109/CLUSTER49012.2020.00067

[17] Stijn Eyerman and Lieven Eeckhout. 2010. A Counter Architecture for Online DVFS Profitability Estimation. *IEEE Trans. Comput.* 59, 11 (2010), 1576–1583. https://doi.org/10.1109/TC.2010.65

[18] Luca Fedeli, Axel Huebl, France Boillod-Cerneux, Thomas Clark, Kevin Gott, Conrad Hillairet, Stephan Jaure, Adrien Leblanc, Rémi Lehe, Andrew Myers, Christelle Piechurski, Mitsuhisa Sato, Neïl Zaim, Weiqun Zhang, Jean-Luc Vay, and Henri Vincenti. 2022. Pushing the Frontier in the Design of Laser-Based Electron Accelerators with Groundbreaking Mesh-Refined Particle-In-Cell Simulations on Exascale-Class Supercomputers. In *SC22: International Conference for High Performance Computing, Networking, Storage and Analysis*. 1–12. https://doi.org/10.1109/SC41404.2022.00008

[19] Guillaume Fieni, Romain Rouvoy, and Lionel Seinturier. 2020. SmartWatts: Self-Calibrating Software-Defined Power Meter for Containers. arXiv:2001.02505 [cs.DC]

[20] Rong Ge, Xizhou Feng, Wu-chun Feng, and Kirk W. Cameron. 2007. CPU MISER: A Performance-Directed, Run-Time System for Power-Aware Clusters. In *Proceedings of the 2007 International Conference on Parallel Processing (ICPP '07)*. IEEE Computer Society, USA, 18. https://doi.org/10.1109/ICPP.2007.29

[21] Neha Gholkar, Frank Mueller, and Barry Rountree. 2019. Uncore Power Scavenger: A Runtime for Uncore Power Conservation on HPC Systems. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (Denver, Colorado) *(SC '19)*. Association for Computing Machinery, New York, NY, USA, Article 27, 23 pages. https://doi.org/10.1145/3295500.3356150

[22] Anish Govind, Sridutt Bhalachandra, Zhengji Zhao, Ermal Rrapaj, Brian Austin, and Hai Ah Nam. 2023. Comparing Power Signatures of HPC Workloads: Machine Learning vs Simulation. In *Proceedings of the SC '23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis* (Denver, CO, USA) *(SC-W '23)*. Association for Computing Machinery, New York, NY, USA, 1890–1893. https://doi.org/10.1145/3624062.3624274

[23] Gabriel Hautreux and Etienne Malaboeuf. 2023. Reducing HPC Energy Footprint for Large Scale GPU Accelerated Workloads. In *Proceedings of the SC '23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis* (Denver, CO, USA) *(SC-W '23)*. Association for Computing Machinery, New York, NY, USA, 1860–1865. https://doi.org/10.1145/3624062.3624268

[24] Chung hsing Hsu and Wu chun Feng. 2005. A Power-Aware Run-Time System for High-Performance Computing. In *SC '05: Proceedings of the 2005 ACM/IEEE Conference on Supercomputing*. 1–1. https://doi.org/10.1109/SC.2005.3

[25] Yuichi Inadomi, Tapasya Patki, Koji Inoue, Mutsumi Aoyagi, Barry Rountree, Martin Schulz, David Lowenthal, Yasutaka Wada, Keiichiro Fukazawa, Masatsugu Ueda, Masaaki Kondo, and Ikuo Miyoshi. 2015. Analyzing and mitigating the impact of manufacturing variability in power-constrained supercomputing. In *SC '15: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 1–12. https://doi.org/10.1145/2807591.2807638

[26] Canturk Isci, Alper Buyuktosunoglu, Chen-yong Cher, Pradip Bose, and Margaret Martonosi. 2006. An Analysis of Efficient Multi-Core Global Power Management Policies: Maximizing Performance for a Given Power Budget. In *2006 39th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'06)*. 347–358. https://doi.org/10.1109/MICRO.2006.8

[27] Canturk Isci, Gilberto Contreras, and Margaret Martonosi. 2006. Live, Runtime Phase Monitoring and Prediction on Real Systems with Application to Dynamic Power Management. In *2006 39th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'06)*. 359–370. https://doi.org/10.1109/MICRO.2006.30

[28] Adrian Jackson, Alan Simpson, and Andrew Turner. 2023. Emissions and Energy Efficiency on Large-Scale High Performance Computing Facilities: ARCHER2 UK National Supercomputing Service Case Study.

In *Proceedings of the SC '23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis* (Denver, CO, USA) *(SC-W '23)*. Association for Computing Machinery, New York, NY, USA, 1866–1870. https://doi.org/10.1145/3624062.3624269

[29] Georgios Keramidas, Vasileios Spiliopoulos, and Stefanos Kaxiras. 2010. Interval-based models for run-time DVFS orchestration in superscalar processors. In *Proceedings of the 7th ACM International Conference on Computing Frontiers* (Bertinoro, Italy) *(CF '10)*. Association for Computing Machinery, New York, NY, USA, 287–296. https://doi.org/10.1145/1787275.1787338

[30] Jeongnim Kim, Andrew D Baczewski, Todd D Beaudet, Anouar Benali, M Chandler Bennett, Mark A Berrill, Nick S Blunt, Edgar Josué Landinez Borda, Michele Casula, David M Ceperley, et al. 2018. QMCPACK: an open source ab initio quantum Monte Carlo package for the electronic structure of atoms, molecules and solids. *Journal of Physics: Condensed Matter* 30, 19 (2018), 195901.

[31] Los Alamos National Laboratory (LANL). [n. d.]. Psuedo-spectral direct numerical simulation in Python/MPI. https://github.com/lanl/PsDNS. (accessed May 25, 2024).

[32] Justin A. Lemkul. 2018. From Proteins to Perturbed Hamiltonians: A Suite of Tutorials for the GROMACS-2018 Molecular Simulation Package. *Living Journal of Computational Molecular Science* 1, 1 (Oct. 2018), 5068. https://doi.org/10.33011/livecoms.1.1.5068

[33] Slurm Workload Manager. [n. d.]. SPANK. https://slurm.schedmd.com/spank.html. (accessed February 28, 2024).

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

[35] Peter Mattson, Christine Cheng, Cody Coleman, Greg Diamos, Paulius Micikevicius, David Patterson, Hanlin Tang, Gu-Yeon Wei, Peter Bailis, Victor Bittorf, David Brooks, Dehao Chen, Debojyoti Dutta, Udit Gupta, Kim Hazelwood, Andrew Hock, Xinyuan Huang, Atsushi Ike, Bill Jia, Daniel Kang, David Kanter, Naveen Kumar, Jeffery Liao, Guokai Ma, Deepak Narayanan, Tayo Oguntebi, Gennady Pekhimenko, Lillian Pentecost, Vijay Janapa Reddi, Taylor Robie, Tom St. John, Tsuguchika Tabaru, Carole-Jean Wu, Lingjie Xu, Masafumi Yamazaki, Cliff Young, and Matei Zaharia. 2019. MLPerf Training Benchmark. arXiv:1910.01500 [cs.LG]

[36] Sanyam Mehta. 2022. Performance Analysis and Optimization with Little's Law. In *2022 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. 12–23. https://doi.org/10.1109/ISPASS55109.2022.00002

[37] Rustam Miftakhutdinov, Eiman Ebrahimi, and Yale N. Patt. 2012. Predicting Performance Impact of DVFS for Realistic Memory Systems. In *2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*. 155–165. https://doi.org/10.1109/MICRO.2012.23

[38] Rajib Nath and Dean Tullsen. 2015. The CRISP performance model for dynamic voltage and frequency scaling in a GPGPU. In *2015 48th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 281–293. https://doi.org/10.1145/2830772.2830826

[39] Nvidia. [n. d.]. NVIDIA Data Center GPU Manager. https://github.com/NVIDIA/DCGM. (accessed Nov 3, 2023).

[40] Microsoft NVIDIA. [n. d.]. Megatron-DeepSpeed. https://github.com/bigscience-workshop/Megatron-DeepSpeed. (accessed Nov 3, 2023).

[41] Oak Ridge National Laboratory (ORNL). [n. d.]. OLCF-6 Workflow Benchmark. https://www.olcf.ornl.gov/wp-content/uploads/OLCF-6_Workflow_description-7.5.24.pdf. (accessed May 25, 2024).

[42] Barry Rountree, Dong H. Ahn, Bronis R. de Supinski, David K. Lowenthal, and Martin Schulz. 2012. Beyond DVFS: A First Look at Performance under a Hardware-Enforced Power Bound. In *2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum*. 947–953. https://doi.org/10.1109/IPDPSW.2012.116

[43] Barry Rountree, David K. Lowenthal, Martin Schulz, and Bronis R. de Supinski. 2011. Practical performance prediction under Dynamic Voltage Frequency Scaling. In *2011 International Green Computing Conference and Workshops*. 1–8. https://doi.org/10.1109/IGCC.2011.6008553

[44] Barry Rountree, David K. Lownenthal, Bronis R. de Supinski, Martin Schulz, Vincent W. Freeh, and Tyler Bletsch. 2009. Adagio: Making DVS Practical for Complex HPC Applications. In *Proceedings of the 23rd International Conference on Supercomputing* (Yorktown Heights, NY, USA) *(ICS '09)*. Association for Computing Machinery, New York, NY, USA, 460–469. https://doi.org/10.1145/1542275.1542340

[45] Robert Schöne, Daniel Hackenberg, and Daniel Molka. 2012. Memory Performance at Reduced CPU Clock Speeds: An Analysis of Current X86_64 Processors. In *Proceedings of the 2012 USENIX Conference on Power-Aware Computing and Systems* (Hollywood, CA) *(HotPower'12)*. USENIX Association, USA, 9.

[46] Woong Shin, Vladyslav Oles, Ahmad Maroof Karimi, J. Austin Ellis, and Feiyi Wang. 2021. Revealing power, energy and thermal dynamics of a 200PF pre-exascale supercomputer. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (St. Louis, Missouri) *(SC '21)*. Association for Computing Machinery, New York, NY, USA, Article 12, 14 pages. https://doi.org/10.1145/3458817.3476188

[47] David C. Snowdon, Etienne Le Sueur, Stefan M. Petters, and Gernot Heiser. 2009. Koala: A Platform for OS-Level Power Management. In *Proceedings of the 4th ACM European Conference on Computer Systems* (Nuremberg, Germany) *(EuroSys '09)*. Association for Computing Machinery, New York, NY, USA, 289–302. https://doi.org/10.1145/1519065.1519097

[48] A. P. Thompson, H. M. Aktulga, R. Berger, D. S. Bolintineanu, W. M. Brown, P. S. Crozier, P. J. in 't Veld, A. Kohlmeyer, S. G. Moore, T. D. Nguyen, R. Shan, M. J. Stevens, J. Tranchida, C. Trott, and S. J. Plimpton. 2022. LAMMPS - a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales. *Comp. Phys. Comm.* 271 (2022), 108171. https://doi.org/10.1016/j.cpc.2021.108171

[49] Akshay Venkatesh, Abhinav Vishnu, Khaled Hamidouche, Nathan Tallent, Dhabaleswar Panda, Darren Kerbyson, and Adolfy Hoisie. 2015. A case for application-oblivious energy-efficient MPI runtime. In *SC '15: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 1–12. https://doi.org/10.1145/2807591.2807658

[50] Qiang Wang and Xiaowen Chu. 2020. GPGPU Performance Estimation With Core and Memory Frequency Scaling. *IEEE Transactions on Parallel and Distributed Systems* 31, 12 (2020), 2865–2881. https://doi.org/10.1109/TPDS.2020.3004623

[51] Andreas Weissel and Frank Bellosa. 2002. Process Cruise Control: Event-Driven Clock Scaling for Dynamic Power Management. In *Proceedings of the 2002 International Conference on Compilers, Architecture, and Synthesis for Embedded Systems* (Grenoble, France) *(CASES '02)*. Association for Computing Machinery, New York, NY, USA, 238–246. https://doi.org/10.1145/581630.581668

[52] Daniel C. Wilson, Siddhartha Jana, Aniruddha Marathe, Stephanie Brink, Christopher M. Cantalupo, Diana R. Guttman, Brad Geltz, Lowren H. Lawson, Asma H. Al-rawi, Ali Mohammad, Fuat Keceli, Federico Ardanaz, Jonathan M. Eastep, and Ayse K. Coskun. 2021. Introducing Application Awareness Into a Unified Power Management Stack. In *2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. 320–329. https://doi.org/10.1109/IPDPS49936.2021.00040

[53] Qiang Wu, V.J. Reddi, Youfeng Wu, Jin Lee, D. Connors, D. Brooks, M. Martonosi, and D.W. Clark. 2005. A dynamic compilation framework for controlling microprocessor energy and performance. In *38th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'05)*. 12 pp.–282. https://doi.org/10.1109/MICRO.2005.7

[54] Junqi Yin, Sajal Dash, Feiyi Wang, and Mallikarjun Arjun Shankar. 2023. FORGE: Pre-Training Open Foundation Models for Science. In *SC23: International Conference for High Performance Computing, Networking, Storage and Analysis*. 1–13. https://doi.org/10.1145/3581784.3613215