Co-designing Power Management with Job Scheduling for Efficient Exascale Computing
Matthew D. Sinclair and Shivaram Venkataraman
University of Wisconsin-Madison
{sinclair,shivaram}@cs.wisc.edu

Topics: Architectures, Applications, Co-Design
Methodologies

1 Motivation

In recent years, supercomputers have increasingly targeted petascale and exascale levels of computing. To reach this lofty goal, many of these systems have turned to using large numbers of compute accelerators, which offer greater power efficiency and thus enable exascale computing for emerging AI and HPC workloads within a constrained power budget. For example, nearly all of the top 10 supercomputers in the Top-500 today utilize GPUs and the top ranked supercomputer, Oak Ridge National Labs’ Summit, has approximately 24000 NVIDIA Volta V100 GPUs. Moreover, the recently announced Aurora, El Capitan, and Frontier supercomputers (as part of the DOE’s CORAL-2 program) are expected to have even more GPUs. Further, with the development of new accelerators and customized chips (e.g., TPUs, GraphCore), future supercomputers will likely be comprised of a large variety of compute devices. However, using accelerators increases heterogeneity at multiple levels, including the architecture, resource allocation, competing user needs, and manufacturing variability. Accordingly, exascale-class and beyond systems need to efficiently handle many simultaneous jobs while balancing PM and multiple levels of heterogeneity.

Recent studies on supercomputers have shown that PM can impact application performance by up to 20% on CPUs in supercomputing systems, even when the CPUs have the same architecture and vendor SKU [1, 2, 4, 6]. This variation occurs due to the manufacturing process and the chip’s power constraints [2]. However, despite their increasingly widespread use in modern HPC systems and supercomputers, little work exists that examines how PM and manufacturing variability in other accelerators (e.g., GPUs) affect application performance. Further, since there is no standardized way for accelerators to expose PM information, system management software (e.g., operating systems or job schedulers) struggle to control performance and predictability. Given their importance in driving modern HPC systems and supercomputers, it is imperative to understand how PM affects application performance. We next present some initial results on how PM can impact performance, which highlights the need for co-designing power-management policies alongside job scheduling.

2 Challenges

To observe the impact of PM on GPU performance, we chose GPU applications that stress the steady state power consumption of the GPU. Specifically, we use a SGEMM kernel from NVIDIA’s cuBLAS library, which stresses the compute, and STREAM, which stresses the memory subsystem. These applications are representative of key kernels from next-generation workloads including machine learning. We use them to study the behavior of GPUs across a number of clusters ranging from Cloudlab1, a small GPU cluster with 12 NVIDIA Volta V100 GPUs, to Oak Ridge (Summit supercomputer), Sandia (Vortex), and TACC (Frontera and Longhorn) which contain between 216 and 24000 V100 GPUs, and which use a variety of cooling methods (air, mineral oil, and water).

To study the effects on performance, we size the benchmarks to fully occupy all of the GPU’s streaming multiprocessors (SMs). Additionally, as the GPU’s PM controller relies on dynamic voltage and frequency scaling (DVFS) to maintain the per-GPU power limit for safe operation [5], it is important that the kernels run long enough for the DVFS controller to reach a stable state (i.e., a constant SM frequency) [3]. We define 1 run of our experiment as 100 repetitions of each benchmark’s kernel. The repetitions help avoid statistical bias and any other additional transient effects. In all our experiments we use V100 GPUs in the SXM2 configuration (max frequency: 1530 MHz, TDP: 300W). Moreover, to avoid transient effects, we collected data for multiple runs on the same machine over multiple weeks.

In total, we ran over 100000 experiments and our preliminary study has identified several key insights.2 Regardless of cooling approach or cluster size, we observed 5%-10% performance variability caused by the lack of global PM (similar to prior work for CPUs [2]). Some outliers are severe: for example, in TACC, PM throttled 300W TDP GPUs to as low as 250W, causing a 20% slowdown. However, while performance variations are directly correlated with frequency changes, temperature and power consumption are in general not directly correlated and are poor indicators of GPU performance. Finally, we also observed that PM can be influenced by spatial (i.e., neighboring GPUs) and temporal (i.e., kernels running previously on same GPU) effects. Although mineral oil and water cooling reduce the relative temperature variation, we still observed performance and power vari-

1https://cloudlab.us

2Due to space constraints we do not show figures of these results.
As multi-GPU experiments become increasingly common, having such a variation across GPUs due to PM can significantly affect applications that coordinate across GPUs. Further we find that the performance variation also depends on the application (STREAM vs. SGEMM). We also discussed our preliminary findings with researchers at AMD and NVIDIA to validate that this behavior was beyond that expected from process variation.

Unfortunately, it is difficult to address these challenges directly, because modern GPUs make only local PM decisions where each node merely allows its GPUs (and other processors) to optimize for its power consumption. This means that we cannot perform application-aware global PM, where scheduling frameworks or administrative tools can effectively place workloads to avoid slowdowns. This motivates our proposed co-design that we describe next.

3 Opportunity

To overcome the observed performance variability in modern systems, we propose working with system vendors, administrators, and integrators, as well as end users to create a new ecosystem that transforms the efficiency of PM in existing systems and creates best-in-class methodologies that can be adopted to improve both current and future systems. Figure 1 shows our overall approach.

Thus far we focused on GEMM and STREAM, which are representative of workloads such as machine learning on modern GPUs. However, further experiments with additional applications that stress different system components are needed to design application-specific PM policies. We will further enhance these application-specific PM policies by making job schedulers variability-aware. Variability-aware schedulers will utilize software-runtime co-design to identify and harness the performance variation across GPUs in existing systems. As a result, schedulers can optimize for each application’s power needs. Moreover, grouping GPUs with similar performance variability together and scheduling jobs across those GPUs, we can reduce time spent waiting for stragglers and ensure consistent behavior. Finally, in addition to identifying performance variation amongst accelerators, we will extend our experiments into a benchmark suite (and corresponding user-facing interface) that is run periodically to provide better tools for system administrators to identify slow accelerators and mark them for further investigation, reboot, or potential replacement.

However, scheduling jobs more efficiently at the software and runtime layers is limited in its ability to quickly, dynamically change policies as cluster conditions evolve. A major limiter to further improving efficiency is the lack of standards for exposing power information in modern accelerators. Thus, for future systems we will build on the insights generated by our optimizations for current systems, and apply co-design that makes the hardware, software, and runtime layers aware of the variance in the systems. To do this, we will design a standard for accelerators to expose PM information from the hardware to the software and runtime. Using this information, instead of performing PM locally, we plan to develop a global power management scheme to enable optimal PM decisions across accelerators and further reduce performance variability.

4 Timeliness

As modern HPC systems are designed to use increasing number of accelerators such as GPUs, it is imperative to understand how power management affects the behavior of applications. Our preliminary results show that significant performance variation already exists on modern systems. Thus, given the US government’s planned investment in additional exascale-class and beyond machines with even more accelerators (which will further increase both heterogeneity and variability), our proposed work is both timely and important to improving the efficacy and utility of these machines. By making power management a first-class citizen in these systems, our co-design will ensure more consistent performance between runs, help administrators better spot and investigate bad GPUs, and reduce thermal throttling, thereby improving energy efficiency.

References