Performance Optimization in Power Capped GPU Computing

Tyler Allen, Xizhou Feng, and Rong Ge
{tnallen,xizhouf,rge}@clemson.edu
Clemson University

1 Introduction

Power has been consistently considered as a top challenge in HPC. To address the power challenge, power efficient GPU accelerators become a standard component in mainstream systems and contribute the majority of compute capacity. Nevertheless, GPUs are still constrained by limited permissible power and required to sustain performance growth. In many situations, there is a need to set a power cap on GPUs. Nevertheless, in our research we find the default hardware power capping on GPUs loses performance by up to 35%, in comparison to the maximum achievable performance.

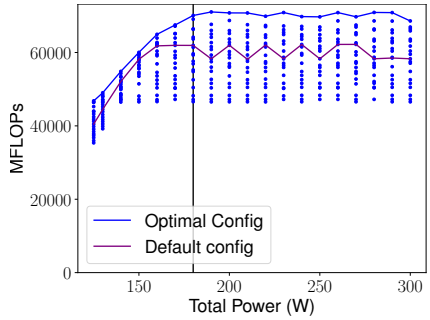


Figure 1. Performance of miniFE vs total power cap.

We conduct a set of power capping experiments on a system accelerated with an Nvidia Titan XP card, and compare to the default hardware power capping as shown in Figure 1. We make the following observations:

- Default hardware power capping underperforms significantly compared to maximum achievable, by up to 35%.
- Power allocation across GPU SMs and global memory determines the resulting performance. The optimal power allocation outperforms a poor one by over 40%.
- There exists a certain power P^{tot}_{max}, beyond which the maximum acheivable performance stops growing. For example, P^{tot}_{max} = 190 Watts for miniFE with the studied data size.

Detailed analysis indicates that the default power capping uses the same strategy for power allocation between GPU SMs and global memory. Specifically, the allocated memory power allows memory to run at the highest speed, no matter what is the imposed total power cap or what is the running application or kernel. Such obliviousness results in inferior performance and yet full power utilization.

Closing the performance gap and determining P^{tot}_{max} are critical to addressing the power challenge in GPU computing. Simply adopting strategies from CPUs is problematic due to different architectures and performance and power dynamics. In this work, we take a data-driven approach and perform in-depth analysis of the impacts of power allocations and present an application-aware power allocation strategy to quickly identify optimal allocations in power capped GPU computing. Our main contribution includes:

- We reveal that the default hardware power capping on GPUs loses significant performance.
- We identify the power and performance dynamics under different SM-memory power allocations.
- We explore an application-aware heuristic power capping algorithm that delivers near-optimal performance with lightweight profiling.

2 Dynamics of SM-Memory Power Allocations

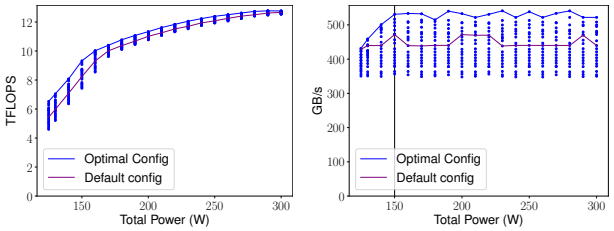
The miniFE example not only highlights the power cap has a decisive impact on achievable performance, but also reveals the importance of cross-component power allocation. Understanding the impacts of power caps and identifying the patterns help us better understand the power problems and quickly identify the optimal power allocations in HPC.

2.1 The Effects of Power Caps

Figure 2 shows the overall results for DGEMM and STREAM. The curve is $\texttt{perf}_{max}$ vs. P_{cap}, and the data points with the same total power budget are performance from different processor-memory power allocations.

For DGEMM, as the total power cap P_{cap} increases, the achievable max performance $\texttt{perf}_{max}$ increases monotonically. The $\texttt{perf}_{max} \sim P_{cap}$ curve consists of several segments, each approximately in a piece-wise linear relation. The growth rate gradually decreases, indicating diminishing return from increased power cap. It is expected that performance would further increase if the architecture included more hardware and consumed more power.

For STREAM, the $\texttt{perf}_{max} \sim P_{cap}$ curve is different. While performance increases with small power caps, it flattens once P_{cap} exceeds P^{tot}_{max}. This indicates that STREAM is unable to use more power than DGEMM. Such observations comply with the fact that DGEMM is compute intensive and has a larger power demand and benefits significantly from the range of SM performance states.



(a) DGEMM in CPU computing (b) STREAM in GPU computing

Figure 2. Upper performance bound vs. with power cap.

2.2 Categorization of Power Allocation Scenarios

As performance differs significantly, depending on how the power cap is distributed across processors and memory, it is necessary to identify the optimal allocation. To gain insights, we conduct extensive experiments where we vary the processor-memory distribution under various power caps. From experimental results we find that the impacts of power allocations fall into three categories. To be consistent with the aforementioned simplified cross-component power coordination problem, we explicitly note P_{SM}, P_{mem}, $P_{cap} = P_{SM} + P_{mem}$.

For a given total power cap P_{cap}, when more power is allocated to the global memory, applications present three trends:

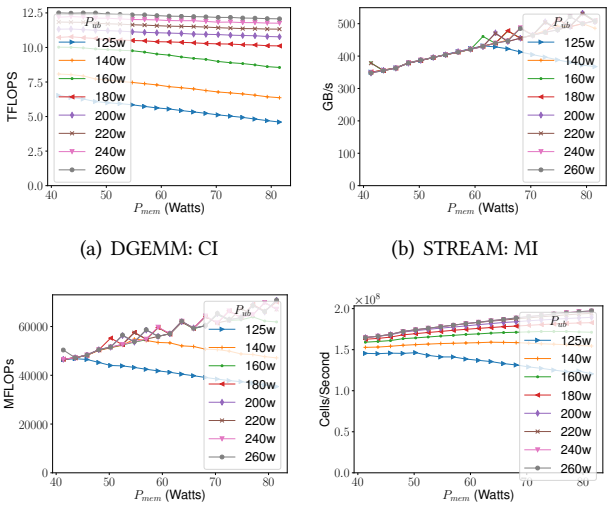
1. Compute intensive (CI): performance is largely constant or decreases, regardless of the value of P_{cap}. The performance curves are dispersed and diverge as memory power allocation increases.
2. Memory intensive (CI): performance increases monotonically at a large rate, and the performance curves under different P_{cap} overlap when memory power allocation is small. Performance continues to increase at the same rate if P_{cap} is large, but begin to decreases if P_{cap} is small.
3. Applications in between (BT): performance increases monotonically but at a small rate if P_{cap} is large, increases then decreases otherwise. The performance curves are clustered at small memory power caps and diverge as memory power allocation increases.

3 An Application-Aware Heuristic Method

Identifying max achievable performance for a given power cap involves extensive profiling, not applicable for online use. We design a heuristic application-aware algorithm, which only needs two runs to obtain two parameters. Figure 4 shows the evaluation results. Our heuristic method can achieve a performance which is 97% of the max value obtained through brute force profiling.

4 Conclusion

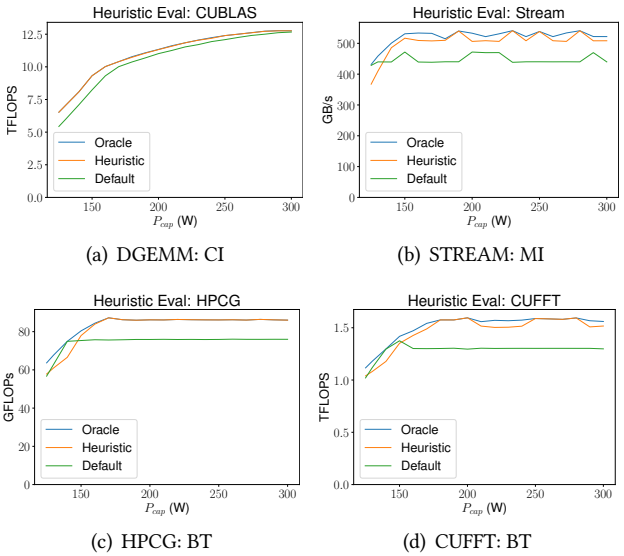
We find that the hardware power capping loses significant performance, and explore application-aware power capping to balance power across SM and memory to meet demands



(a) DGEMM: CI (b) STREAM: MI

(c) miniFE: MI (d) Cloverleaf: BT

Figure 3. Performance trends as memory power allocation increases under various total power caps.



(a) DGEMM: CI (b) STREAM: MI

(c) HPCG: BT (d) CUFFT: BT

Figure 4. Evaluation of our heuristic method.

from application. Our proposed heuristics can deliver near-optimal performance with lightweight profiling. In the future, we will evaluate the findings on multiple generations of GPUs, and develop methodology to more accurately pinpoint P^{tot}_{max} and optimal allocations.

References

[1] R. Ge, X. Feng, Y. He, and P. Zou. 2016. The Case for Cross-Component Power Coordination on Power Bounded Systems. In *2016 45th International Conference on Parallel Processing (ICPP)*. 516–525.

[2] T. Patki, Z. Frye, H. Bhatia, F. Di Natale, J. Glosli, H. Ingolfsson, and B. Rountree. 2019. Comparing GPU Power and Frequency Capping: A Case Study with the MuMMI Workflow. In *2019 IEEE/ACM Workflows in Support of Large-Scale Science (WORKS)*. 31–39.