Optimizing Performance in Power Bounded GPU Computing

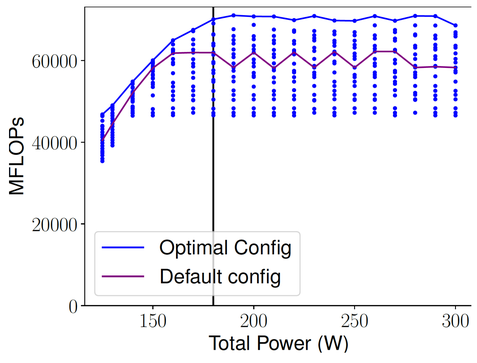
Tyler Allen, Xizhou Feng (faculty advisor), Rong Ge (faculty advisor)
Clemson University
{tnallen, xizhouf, rge}@clemson.edu

The Problem: Power Challenge in GPU Computing

Power is 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.

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.

Performance Gap and Causes



MiniFE on an Nvidia Titan XP

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. We make the following **observation:**

(1) Default hardware power capping underperforms significantly compared to maximum achievable, by up to 35%.

(2) There exists a certain power P_{max}^{tot}, beyond which application performance stops growing.

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. Such obliviousness results in inferior performance and yet power budget fully consumed .

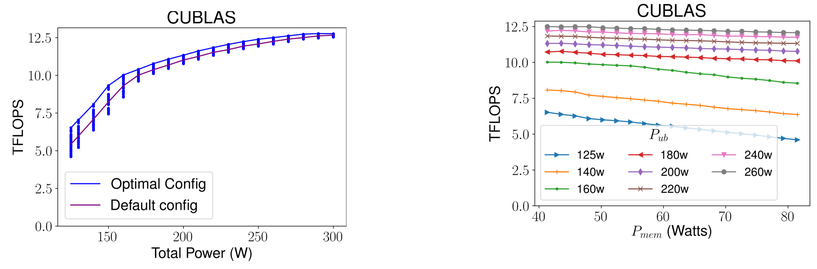
Closing the performance gap and determining P_{max}^{tot} are critical to addressing power challenge in GPU computing. Simply adopting strategies from research on CPUs is problematic due to different architectures and performance and power dynamics. In this work, we present an application-aware power allocation strategy for power capped GPU computing.

Findings and Contributions of This Work

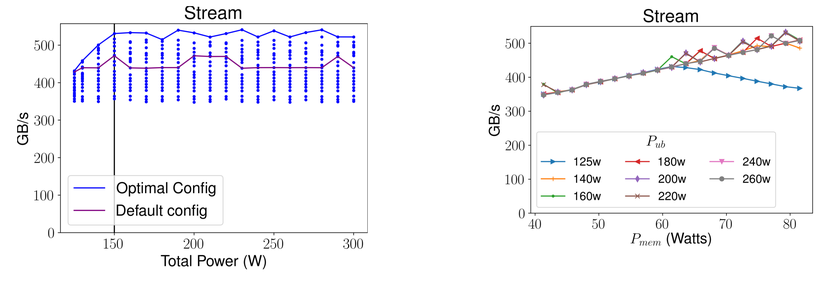
- We reveal that the default hardware power capping on GPUs loses significant performance, due to the obliviousness to applications and imposed power cap.
- We identify application performance patterns under different SM-memory power allocations.
- We present an application-aware heuristic power capping algorithm that delivers near-optimal performance with lightweight profiling.

Dynamics of SM-Memory Power Allocations

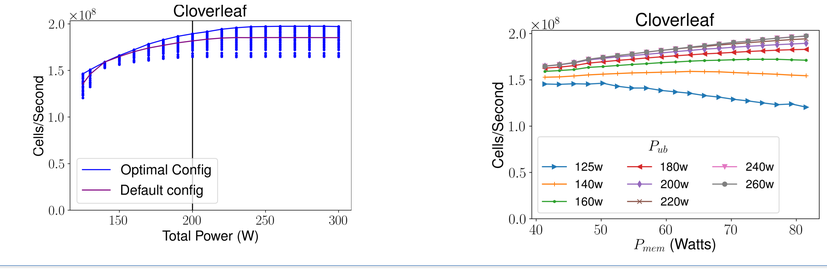
CI: Compute intensive app: (1) max achievable performance keeps increasing with the total power cap, and (2) performance is largely constant or decreases with memory power allocation.



MI: Memory intensive app: (1) max achievable performance quickly flattens, and (2) performance greatly increases with memory power given sufficient total power.



BT: apps in between: (1) max performance increases until total power cap is relatively large, and (2) performance noticeably increases with memory and SM power if given sufficient total power, increases then decreases if given small total power.



Application-Aware Heuristic Alg and Evaluation

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 for profiling for each kernel:

- P_{max}^{tot}: total power when no (default) cap imposed
- P_{ref}^{tot}: total power when SM runs at 785 MHz (min freq pairing with max mem freq) and mem at max freq.
- P_{min}^{mem}, P_{max}^{mem}, P_{min}^{tot}, a: pre-obtained for the GPU card

Heuristic algorithm: given power cap P,

if P > P_{max}^{tot}:

log "more power budget than needed"

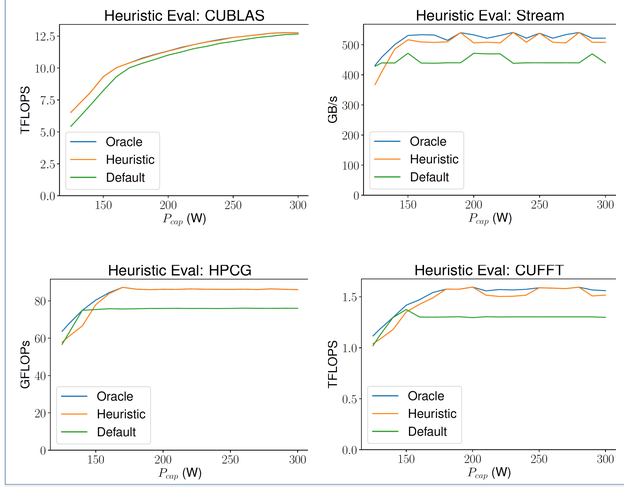
if CI: $P^{mem} \leftarrow P_{min}^{mem}$, $P^{SM} \leftarrow$ P - P^{mem}

else:

if P >= P_{ref}^{tot}: $P^{mem} \leftarrow P_{max}^{mem}$, $P^{SM} \leftarrow$ P - P^{mem}

else: $P^{mem} \leftarrow P_{min}^{mem}$ + a(P − P_{min}^{tot}), $P^{SM} \leftarrow$ P - P^{mem}

Eval results: 97% on average in comparison to the max achievable (Oracle)



Conclusions

- The hardware power capping loses significant performance
- Optimal power capping must be application-aware and balance power across SM and mem
- Heuristics can deliver near-optimal performance with lightweight profiling
- In the future, we will evaluate the findings on multiple generations of GPUs, and design runtime power capping algorithms