



GPGPU Performance Estimation with Core and Memory Frequency Scaling



Qiang Wang, Xiaowen Chu

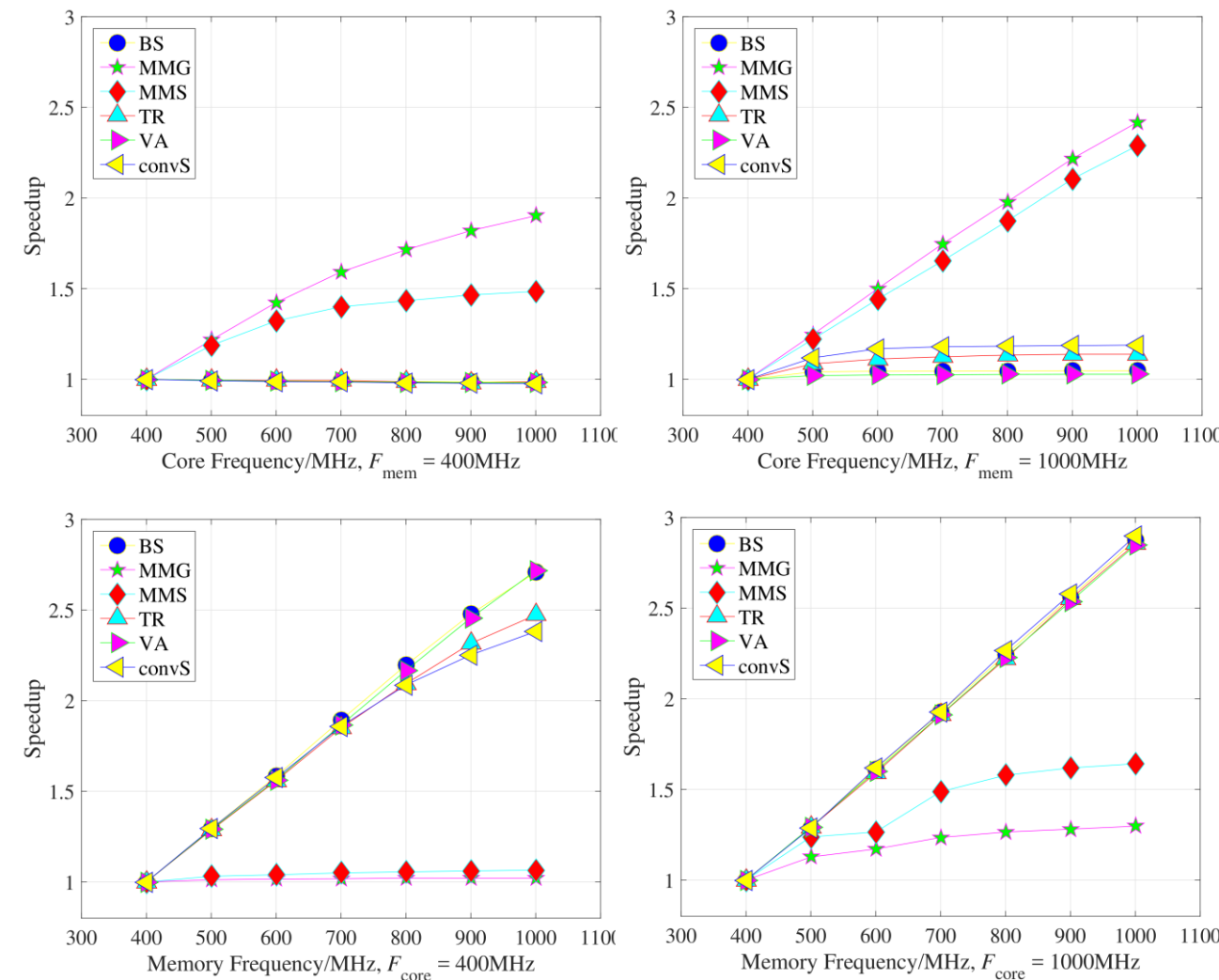
Department of Computer Science, Hong Kong Baptist University

ABSTRACT

Graphics processing units (GPUs) support dynamic voltage and frequency scaling to balance computational performance and energy consumption. However, simple and accurate performance estimation for a given GPU kernel under different frequency settings is still lacking for real hardware, which is important to decide the best frequency configuration for energy saving. We reveal a fine-grained analytical model to estimate the execution time of GPU kernels with both core and memory frequency scaling. Over a wide scaling range of both core and memory frequencies among 20 GPU kernels, our model achieves accurate results (4.83% error on average) with real hardware. Compared to the cycle-level simulators, our model only needs simple micro-benchmarks to extract a set of hardware parameters and kernel performance counters to produce such high accuracy.

OBJECTIVES

- Different GPU applications may have various utilizations of different hardware components, changing the frequencies may lead to diverse performance scaling behaviors for them.
- Different GPU applications have different sensitivities to f^{core} and f^{mem} respectively. Even the sensitivity to f^{core} may change under different fixed f^{mem} , and vice versa.
- Our goal is to establish an estimation model that can predict the execution time of a given GPU kernel under different core and memory frequency settings. This can help determine which frequency configuration save most energy for a specific GPU application.



METHODOLOGY

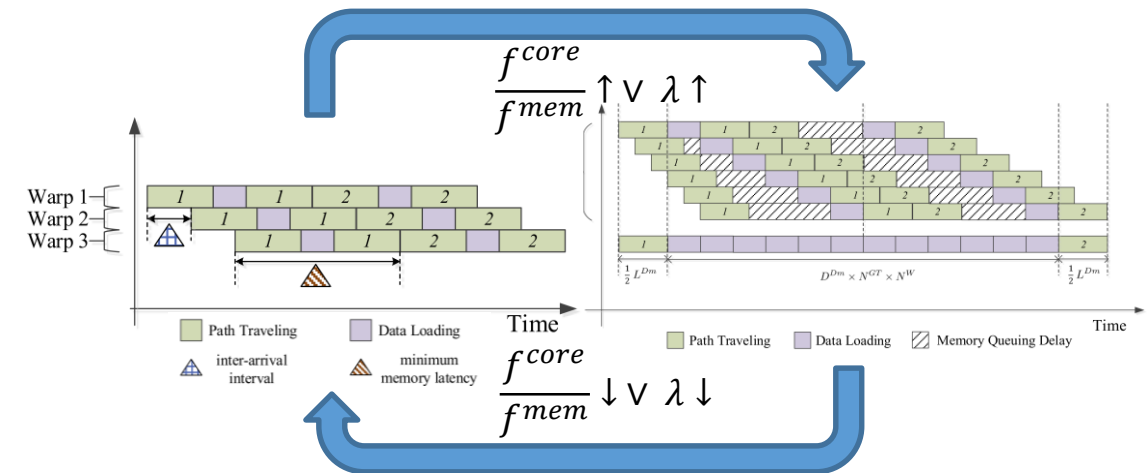
□ DRAM Performance Modeling with Frequency Scaling

- When the memory system is idle and only contains the overhead of path traveling between SMs and DRAM, and data access in DRAM (refer to the left figure).

$$T^{Gm} = \frac{1}{\lambda} \times (N^W - 1) + L^{Dm} \times N^{GT}$$

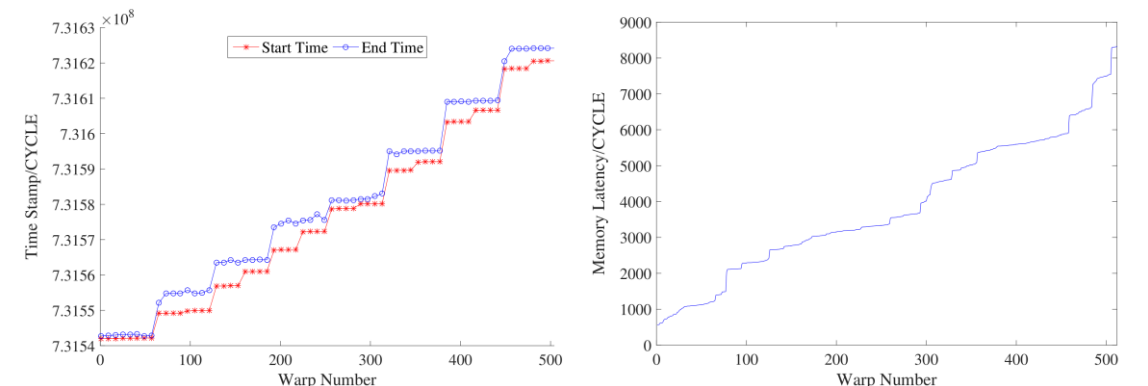
- If the memory system is saturated due to intensive memory requests, most of the requests must wait in the queue until the previous ones have been finished (refer to the right figure).

$$T^{Gm} = L^{Dm} + D^{Dm} \times N^{GT} \times N^W$$



□ Quantitative Analysis on GTX 980

- Minimum DRAM latency: $L^{Dm} = 277.32 + 222.78 \times \frac{f^{SM}}{f^{MEM}}$
- DRAM queuing delay: $D^{Dm} = 8.1762 + \frac{805.03}{f^{MEM}}$



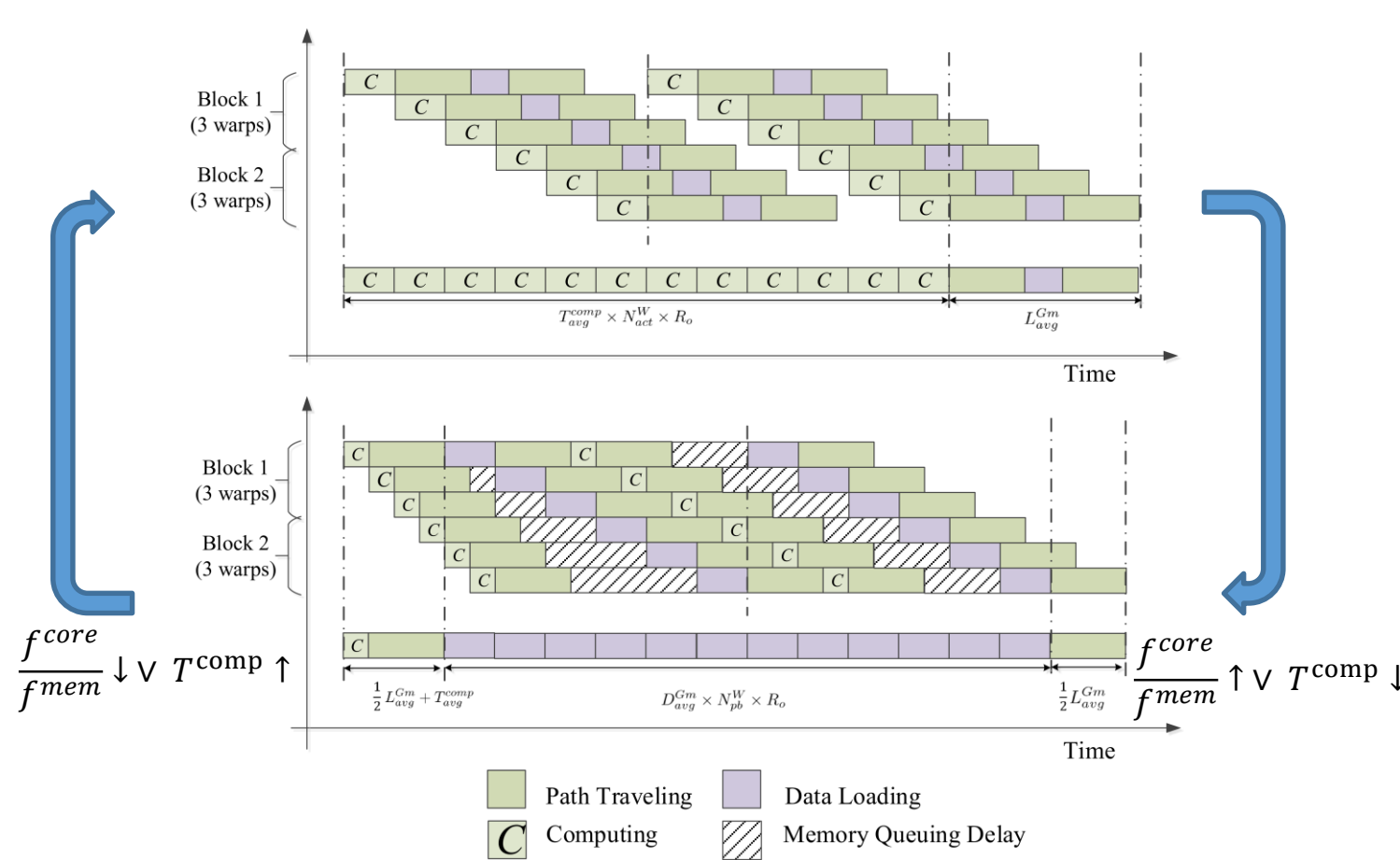
□ GPGPU Performance Modeling with Frequency Scaling

- When there are enough compute instructions to be issued and the memory requests are not too intensive due to long computation periods, the global memory latency can be hidden (refer to the upper figure).

$$T^{act} = T_{avg}^{comp} \times N_{act}^W \times R_o + L^{Gm}$$

- When the memory bandwidth is saturated or there are not enough warps to issue compute instructions, one memory request must wait until all of the outstanding requests have been finished. We can regard the compute cycles as the inter-arrival time of two consequent memory requests (refer to the bottom figure.)

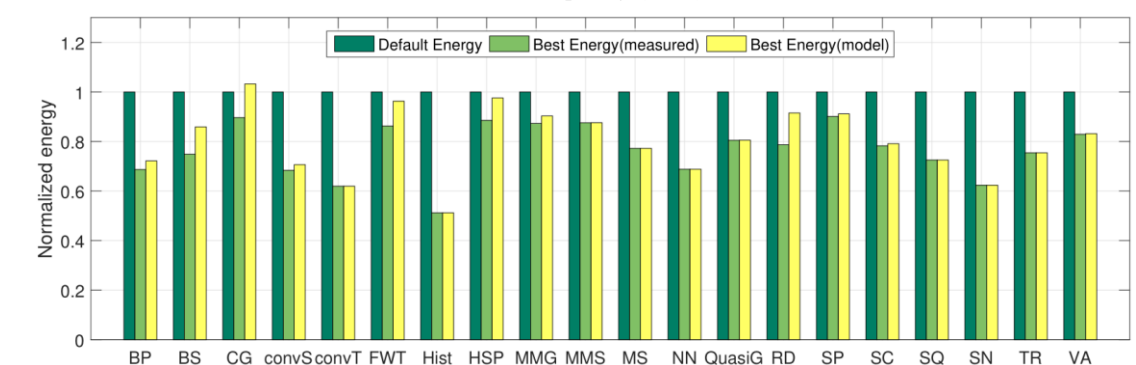
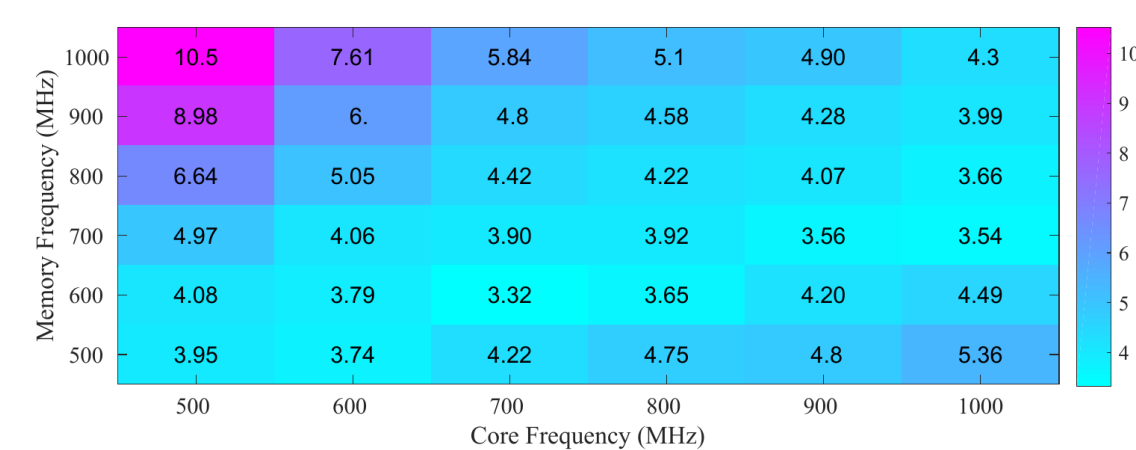
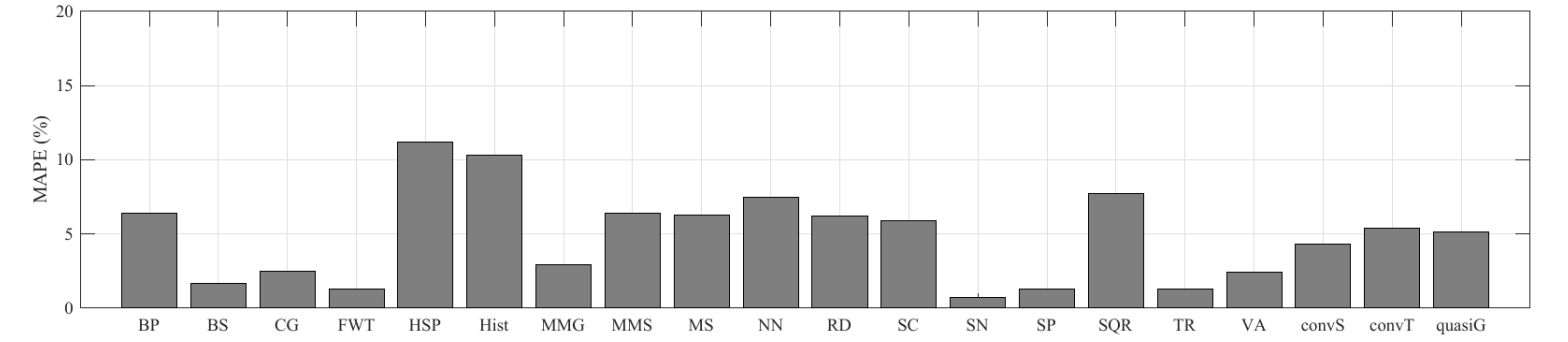
$$T^{act} = L^{Gm} + T_{avg}^{comp} + D^{Gm} \times N_{pb}^W \times R_o$$



RESULTS

- **Model Accuracy:** The left-bottom figure shows the mean absolute precision error (MAPE) of each tested GPU kernel across 36 frequency configurations. Our performance model estimates achieve high accuracy with an average MAPE of 4.83%. The right-upper figure shows the average error of each frequency setting.

- **Energy Conservation:** The power estimation model described in [7] adopts Support Vector Regression (SVR) to build correlations between performance counters of GPU kernels and GPU runtime power consumption. We combine it with our performance model to predict the energy consumption of each kernel under different frequency settings. The right-bottom figure illustrates the results. With the predictions of our model, we finally achieve up to 48.8% energy saving and an average of 20% among 20 tested GPU kernels.



CONCLUSIONS

We demonstrate a GPGPU performance predictor for a wide range of core and memory frequencies. Our model can quickly and accurately predict the execution time of a GPU kernel on real hardware, which is important to deriving real-time energy conservation suggestions with DVFS techniques. We show that our performance estimation method can achieve a MAPE of 4.83% across up to 2× both core and memory frequencies scaling among 20 realistic GPU kernels.

Potentially our performance model can be integrated into a real-time power management system for GPUs based on DVFS techniques. Using the best frequency configuration predicted by the models, we can save an considerable amount of energy consumption compared to adopting the default setting.

REFERENCES

- [1] Robert A Bridges, Neena Imam, and Tiffany M Mintz. 2016. Understanding GPU Power: A Survey of Profiling, Modeling, and Simulation Methods. ACM CSUR 49, 3 (2016), 41.
- [2] Xiaoming Chen et al . 2014. Run-time technique for simultaneous aging and power optimization in GPGPUs. In Design Automation Conference (DAC), 2014 51st ACM/EDAC/IEEE. IEEE, 1–6.
- [3] Sunpyo Hong and Hyesoon Kim. 2009. An Analytical Model for a GPU Architecture with Memory-level and Thread-level Parallelism Awareness. In Proceedings of the 36th ISCA (ISCA '09).
- [4] Xinxin Mei, Qiang Wang, and Xiaowen Chu. 2017. A survey and measurement study of GPU DVFS on energy conservation. Digital Communications and Networks 3, 2 (2017), 89 – 100.
- [5] Rustam Miftakhutdinov, Eiman Ebrahimi, and Yale N Patt. 2012. Predicting performance impact of DVFS for realistic memory systems. In Proceedings of the 45th MICRO, 2012.
- [6] Rajib Nath and Dean Tullsen. 2015. The CRISP performance model for dynamic voltage and frequency scaling in a GPGPU.
- [7] Qiang Wang and Xiaowen Chu. 2017. GPGPU Power Estimation with Core and Memory Frequency Scaling. SIGMETRICS Perform. Eval. Rev. 45, 2 (Oct. 2017), 73–78.

CONTACTS

E-mail: {qiangwang, [chxw](mailto:chxw@comp.hkbu.edu.hk)}@comp.hkbu.edu.hk

Source Code: <https://github.com/blackjack201312/NV-DVFS-Benchmark.git>