Micro-benchmarking GPU micro-architectures: A review

Suhas Thejaswi Muniyappa
Department of Information and Computer Science
Aalto University, Espoo - 02150, Finland.
email: suhas.muniyappa@aalto.fi

This report is the result of coursework for LC-1310 – Academic Communication for MSc Students at Aalto university and my research with Prof.Petteri Kaski, Aalto university.

1 Introduction

The per-core CPU performance has remained almost same over the last ten years. Even tough recent Haswell and Broadwell micro-architectures benefit from advanced vector extensions (AVX), the speedup achieved is rather modest [4, 6]. A better way to achieve the speedup is to run the programs in parallel. Clearly, graphical processing units (GPUs) with thousands of cores provide a cost-effective solution for executing algorithms which can scale. However, GPUs are built specifically for the purpose of processing graphics and do not have hardware support for advanced instructions like most CPUs do. Due to lack of hardware support, using GPUs for general-purpose computation requires special modification in the code and a complete knowledge of the GPU architecture [5]. However, engineering documentation available for GPUs is very limited; they often entail unknown hardware behavior, which is not disclosed by the hardware manufacturers. In this situation, micro-benchmarking the hardware along with additional fine-grain benchmarks provide a way to determine the hardware characteristics and identify the potential performance penalties. In this article, we review the techniques used for micro-benchmarking GPUs and discuss experimental setup for determining the hardware characteristics.

2 Micro-benchmarking

Micro-benchmarking is a process of determining the hardware characteristics by creating artificial workloads, using a very small and specific piece of code. From over two decades, scientists have been trying to devise better approaches for micro-benchmarking the computer hardware. Among these, clearly, pointer chasing is one of the most successful and widely used approaches to benchmark the computer hardware [1, 2, 7, 8]. In this section, we discuss about micro-benchmarking, using pointer chasing and fine-grain pointer chasing approaches.

2.1 Pointer chasing

Pointer chasing is a systematic micro-benchmarking approach for obtaining the hardware characteristics. In this approach, the array elements are initialised with the index of the next memory access and the distance between two consecutive memory accesses is called *stride size*. The latency of memory access is the time difference in clock-cycles, between the memory access issue and the availability of data in the processor register. In pointer chasing experiment, the complete array is traversed sequentially to record the average memory access latency. The latency of memory access mainly depends on the stride size, which varies across experiments [8]. Furthermore, the characteristics of the hardware can be deduced from the access latency data and it is discussed in Section 3.

Micro-benchmarking using pointer chasing was introduced by Saavendra *et al.* [7, 8] for benchmarking CPUs. Furthermore, pointer chasing method was successfully used for benchmarking GPUs

by Wong *et al.* [9] and Meltzer *et al.* [3]. Micro-benchmark experiments using pointer chasing are based on the following three assumptions:

1. The cache replacement policy is least recently used (LRU).
2. All cache-sets have same size.
3. In memory address, the bits that identify the cache set are immediately followed by the bits that identify the offset.

However, these assumptions do not hold for most recent GPU micro-architectures. Some GPUs indeed have different cache-set size; in addition, cache replacement policy is not LRU [1, 2]. To overcome the limitations of pointer chasing, Mei *et al.* [1] have introduced the fine-grain pointer chase approach to benchmark GPUs, which does not consider the assumptions made by Saavedra *et al.* [7, 8].

2.2 Fine-grain pointer chasing

Fine-grain pointer chasing is a more advanced micro-benchmarking approach, in which every memory access latency is recorded and analysed, this approach is feasible in most modern GPUs because of the presence of relatively larger shared-memory, which can be used to store the sequence of memory access latencies, without interfering the normal data access operations [1]. Furthermore, the hardware characteristics can be deduced from the sequence of memory access latencies and the latency pattern, and it is discussed in Section 3. It is important to note that, the fine-grain pointer chasing stores every memory access latency, whereas pointer chasing only stores the average memory access latency of the complete array.

The shared-memory of GPUs is not sufficient to perform the fine-grain pointer chase for arrays larger than the shared-memory size [1]. However, fine-grain pointer chase approach can be extended by piecewise linear construction to record memory access latency for array size larger than shared-memory size. In this approach, the memory access latency of array elements with-in the range of the shared-memory is recorded in each iteration. Furthermore, experiment is repeated piecewise in a sliding-window model, to get the comprehensive memory access latency data of complete array.

3 Determining hardware characteristics

In this section, we discuss the experimental setup introduced by Mei *et al.* [1], to determine hardware characteristics of the GPUs using fine-grain pointer chasing approach. It has four steps:

1. Determine the *cache-size* C. Set *stride-size* $s = 1$, initialize *array size* $N = 1$ and increase N gradually at the granularity of 1, until the memory access latency increase. Cache-size is the maximum value of N before memory access latency increase.
2. Determine the *cache-line-size* b. Set $s = 1$, begin with $N = C + 1$ and increase N gradually at the granularity of 1. When $N < C + b + 1$, the number of cache misses are close. However, if N is increased to $C + b + 1$, there is a sudden increase in the number of cache-misses, despite the fact that N is increased by 1.
3. Determine the number of *cache-sets* T. Set $s = b$, $N = C$ and increase N at the granularity of b. Every increment of N causes cache-miss of a new cache-set. For $N > C + (T - 1)b$, all cache sets are missed. Finally, number of cache-sets T can be deduced from cache miss pattern.
4. Determine the cache replacement policy from the cache miss pattern obtained from step 2 and step 3.

4 Conclusion

The rapid increase in the computational capability of GPU hardware, along with recent improvements in its programmability, have made GPU hardware a promising platform for computationally demanding tasks in a wide variety of application domains. Recent GPU architectures provide

tremendous memory bandwidth and computational horsepower. However, bandwidth comes at the cost of latency and a sophisticated memory system. Understanding the hardware characteristics and sophisticated memory system of GPUs is necessary when one seeks to overcome the performance bottlenecks and optimize an algorithm for near-peak-bandwidth performance. For GPUs, the engineering documentation available on memory hierarchy is very limited. In this situation, micro-benchmarking the hardware together with additional fine-grain benchmarks provide an approach to optimization. Furthermore, fine-grain benchmark helps us to understand the memory system and memory access patterns of the GPUs.

References

[1] Mei, X., and Chu, X. Dissecting GPU memory hierarchy through microbenchmarking. *IEEE Transactions on Parallel and Distributed Systems Preprint*, 99 (2016), 1.

[2] Mei, X., Zhao, K., Liu, C., and Chu, X. Benchmarking the memory hierarchy of modern GPUs. *Network and Parallel Computing: 11th IFIP WG 10.3 International Conference Proceedings (NPC)* (2014), 144–156.

[3] Meltzer, R., Zeng, C., and Cecka, C. Micro-benchmarking the C2070. In *GPU Technology Conference, category: parallel programming, languages and compilers. poster PP13* (2013), GTC.

[4] Moore, C. Data processing in exascale-class computer systems. In *The Salishan Conference on High Speed Computing* (2011), HSC.

[5] Owens, J. D., Luebke, D., Govindaraju, N., Harris, M., Krüger, J., Lefohn, A. E., and Purcell, T. J. A survey of general-purpose computation on graphics hardware. In *Computer graphics forum* (2007), vol. 26, Wiley Online Library, pp. 80–113.

[6] Rupp, K. 40 years of microprocessor trend data. https://www.karlrupp.net/2015/06/40-years-of-microprocessor-trend-data, Accessed 8.12.2016.

[7] Saavedra, R. H., and Smith, A. J. Measuring cache and TLB performance and their effect on benchmark runtimes. *IEEE Transactions on Computers 44*, 10 (1995), 1223–1235.

[8] Saavedra-Barrera, R. H. *CPU performance evaluation and execution time prediction using narrow spectrum benchmarking.* PhD thesis, University of California, Berkeley, 1992.

[9] Wong, H., Papadopoulou, M.-M., Sadooghi-Alvandi, M., and Moshovos, A. Demystifying GPU microarchitecture through microbenchmarking. In *IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)* (2010), IEEE, pp. 235–246.