Characterizing Memory Bottlenecks in GPGPU Workloads

Saumay Dublish, Vijay Nagarajan, Nigel Topham
University of Edinburgh
{saumay.dublish, vijay.nagarajan, nigel.topham}@ed.ac.uk

***Abstract*—GPUs are often limited by the off-chip memory bandwidth. With the advent of general-purpose computing on GPUs, cache hierarchy has been introduced to filter the bandwidth demand to the off-chip memory. However, the cache hierarchy presents its own bandwidth limitations in sustaining such high levels of memory traffic. In this work, we characterize the bandwidth bottleneck in GPUs present across the memory hierarchy for general-purpose applications. We show that the improvement in performance achieved by mitigating the bandwidth bottleneck in the cache hierarchy can exceed the speedup obtained by a memory system with a baseline cache hierarchy and high bandwidth off-chip memory. We also show that addressing the bandwidth bottleneck in isolation at specific levels can be sub-optimal and can even be counter-productive. Therefore, we show that it is imperative to resolve the bandwidth bottleneck synergistically across different levels of the memory hierarchy.**

I. Introduction

Since the introduction of memory hierarchies in GPUs, scientific and enterprise workloads have increasingly used GPUs to address their massive computational demands. Such workloads present a high demand on the off-chip memory bandwidth and therefore, the cache hierarchy helps in filtering the bandwidth demand to the off-chip memory. However, due to high cache miss rates and cache thrashing, the off-chip bandwidth bottleneck is only partly mitigated. In addition, the cache hierarchy exposes its own bandwidth limitations in sustaining such high levels of memory traffic [1], thereby resulting in high congestion in the memory system. Therefore, scattered nature of the bandwidth bottlenecks in GPUs motivate us to analyze the bandwidth implications of the memory hierarchy as a whole.

To this end, we aim to characterize and understand the severity of the bandwidth problem posed by the three levels of the memory hierarchy, *viz.*, private L1s, shared L2 and off-chip memory, and also characterize the role of their peripheral network elements such as interconnects and request buffers. We show that due to bandwidth limitation, there is severe congestion between the L1 and L2 as well as between the L2 and off-chip memory. Such high levels of congestion lead to increased memory latencies which has three major implications. ❶ In memory-intensive applications, due to insufficient computation to mask such high memory latencies, such latencies appear in the critical path of system performance. ❷ High latencies of outstanding miss requests lead to prolonged contention of cache resources such as Miss Status Holding Registers (MSHRs) and replaceable cache lines. This effect increases the memory latencies even further as succeeding requests get serialized and have to wait for outstanding misses to complete and relinquish the resources. ❸ Back pressure from a congested lower level further throttles the cache pipeline and prevents it from operating at peak throughput, exacerbating the bandwidth limitation in the cache hierarchy. A combination of the above factors force the cores to stall, leading to performance degradation.

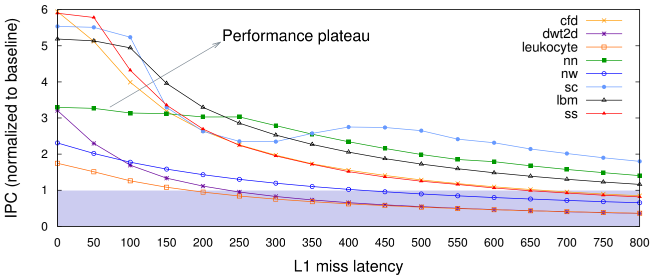

Fig. 1. Performance variation with increasing L1 miss latency.

II. Latency Tolerance Profile

Memory-intensive applications often run into memory misses causing all of the warps to stall due to pending memory instructions. In such a case, miss latencies get exposed due to lack of sufficient overlapping computation and therefore lie in the critical path, directly impacting performance. Fig.1 shows the impact of memory latencies on performance using a representative set of benchmarks. Our baseline architecture is modeled upon Nvidia's GTX480 Fermi GPU and is simulated using GPGPU-Sim. In this study, we modify the memory hierarchy of the baseline architecture so that all the L1 miss responses are returned with a fixed and pre-determined latency that is varied in the simulator and is represented on the *x*-axis. The resultant performance is plotted on the *y*-axis which is normalized to the performance of the baseline architecture.

We make two major observations about the *baseline memory latencies*, *i.e.*, the point on the *x*-axis where the performance curve intercepts the IPC of 1× (shaded region), and therefore, matches the average memory latency of the baseline architecture. ❶ For most benchmarks, the baseline memory latencies are significantly higher than the latencies of *performance plateau* (or peak performance). Therefore, the baseline performance is far from saturation with respect to memory latencies. ❷ The baseline memory latencies are also critically higher than the ideal access latencies of L2 (120 cycles) and DRAM (additional 100 cycles via L2). This suggests that there is considerable congestion in the memory system since traversing the memory system takes significantly higher latencies than the minimum memory access latencies of L2 and DRAM. In summary, the above results indicate that there lies a significant opportunity to improve performance by reducing the memory latencies incurred due to congestion in the memory hierarchy.

III. Measuring the Bandwidth Bottleneck

We quantify the congestion between L1 and L2 by measuring the occupancy of the L2 access queues. We observe that on average, the L2 access queues are full for 46% of their usage lifetime. Similarly, we quantify congestion at the DRAM and observe that the DRAM access queues are full for 39% of their usage lifetime. High occupancy of queues across the memory hierarchy suggests high levels of congestion. Furthermore, we note that as congestion propagates higher in the memory hierarchy, the resultant back pressure throttles the caches (and cores) and prevents them from operating at peak throughput. However, even sufficiently provisioning the memory resources to allow the memory system to operate at peak throughput may not completely resolve the bandwidth bottleneck *as the peak throughput itself can be a limiting factor*. Therefore, in the next section, we discuss the design parameters that either allow the caches (and cores) to achieve peak throughput or increases the peak throughput itself, thus alleviating congestion in the memory hierarchy.

IV. Design-Space Exploration and Results

We summarize our architectural design space in Table I. We categorize each parameter based on whether it increases the peak throughput (shown as '$+$' type) or enables the corresponding memory level in achieving the existing peak throughput (shown as '$=$' type). To evaluate the design-space, we increase the listed architectural parameters to up to 4× their initial values. We make an exception when such scaling reaches a saturation. Note that we choose the scaling coefficient as 4× just to demonstrate the potential of resolving congestion at each level of the memory hierarchy. The actual scaling would also take into account the cost overheads incurred in scaling each parameter.

TABLE I
Consolidated design space to mitigate congestion.

Design Parameter	Type	Baseline value	Scaled value (∼4×)
(a) DRAM			
Scheduler queue	=	16 entries	64 entries
DRAM Banks	=	16 banks/chip	64 banks/chip
Bus width	+	32-bits/chip	64-bits/chip
(b) L2 Cache			
L2 miss queue	=	8 entries	32 entries
L2 response queue	=	8 entries	32 entries
MSHR	=	32 entries	128 entries
L2 access queue	=	8 entries	32 entries
L2 data port	+	32 bytes	128 bytes
Flit size (crossbar)	+	4 bytes	16 bytes
L2 banks	+	2 banks/partition	8 banks/partition
(c) L1 Cache			
L1 miss queue	=	8 entries	32 entries
MSHR (L1D)	=	32 entries	128 entries
Memory pipeline width	=	10	40

Upon scaling the architectural parameters at the respective levels of the memory hierarchy, we observe an average speedup of 4%, 59% and 11% on increasing the bandwidth of L1, L2 and DRAM alone. We further observe an average speedup of 69% and 76% on increasing the combined bandwidth of L1-L2 and L2-DRAM respectively, which is greater than the respective sum of the individual gains. Therefore, we demonstrate that synergistic scaling yields better results than increasing the bandwidth at the memory levels independently. This is because solving the problem in isolation can lead to even more congestion elsewhere in the memory system. For instance, we observe that to prevent throttling of L1 cache, increasing the L1 bandwidth by increasing the MSHRs to handle more outstanding misses can lead to performance degradation due to an even higher congestion between L1 and L2. However, matching the increased bandwidth demand of L1 at L2 significantly improves performance. The average performance improvements also suggest that mitigating congestion in the cache hierarchy exceeds the benefit obtained by a memory system with baseline cache hierarchy and high bandwidth off-chip memory.

V. Conclusion and Future Work

In this work, we demonstrate the bandwidth limitations posed by the memory system in GPUs for general-purpose workloads. We show that the bandwidth bottlenecks lead to high congestion in the memory system, in turn leading to high latencies that appear in the critical path. We perform a design-space exploration and show that increasing the bandwidth in isolation at specific levels of the memory hierarchy can be sub-optimal, and can even lead to performance degradation. We also show that the performance improvement obtained by synergistically improving the bandwidth of the cache hierarchy surpasses the speedup achieved by a memory system with baseline cache hierarchy and high bandwidth DRAM. Therefore, we demonstrate the criticality of the bandwidth bottleneck in the cache hierarchy. In future, we plan to assess the complexity and cost of the various design configurations in order to evaluate most cost-effective ways to mitigate the bandwidth bottleneck.

References

[1] A. Sethia, D. Jamshidi, and S. Mahlke, "Mascar: Speeding up GPU warps by reducing memory pitstops," in *High Performance Computer Architecture (HPCA), 2015 IEEE 21st International Symposium on*, pp. 174–185, Feb 2015.

[2] K. Kim, S. Lee, M. K. Yoon, G. Koo, W. W. Ro, and M. Annavaram, "Warped-preexecution: A GPU pre-execution approach for improving latency hiding," in *2016 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 163–175, March 2016.

[3] G. Sun, C. Hughes, C. Kim, J. Zhao, C. Xu, Y. Xie, and Y. K. Chen, "Moguls: A model to explore the memory hierarchy for bandwidth improvements," in *38th Annual International Symposium on Computer Architecture (ISCA), 2011*, pp. 377–388, June 2011.