Evaluating Warp Scheduling Algorithms for CPU-GPU

Priya Sundaresan
Dept. of Computer Science, Rutgers University
priya.sun@rutgers.edu

Abstract

Heterogeneous System Architecture (HSA) has changed the way CPU-GPU interacts with each other. Unified address space obviates the need for the expensive memory copies between CPU and GPU and also provides programmability benefits. Processor vendors have begun to include HSA support specifically for CPU-GPU. Recently many research works are being carried out in redesigning GPU architecture to obtain benefits of unified address space. In this work, we evaluate the wavefront scheduling algorithms on recently developed TLB-enabled GPUs [1]. We evaluate the performance of such GPU for workloads from Rodinia benchmark [2]. We also analyse the impact of schedulers on cache and TLB behaviour and conclude that cache and TLB-conscious scheduling schemes introduced in [1] performs better than traditional schedulers.

1. Introduction

AMD's Heterogeneous System Architecture (HSA) provides Heterogenous Uniform Memory Access (hUMA) to CPUs and GPUs. It radically changes the way CPU-GPU interact with each other. Unified address space makes data structure and pointers globally visible among compute units. This eliminates the need to copy data from CPU to GPU or burdening CPU with pin-down memory.

Today's GPUs and CPUs typically use separate virtual and physical address spaces. Main memory may be physically shared, but is usually partitioned, or allows unidirectional coherence (e.g., ARM allows accelerators to snoop CPU memory partitions but not the other way around). GPU address translation has traditionally been performed by Input/ Output Memory Management Unit (IOMMU) TLBs in the memory controller, leaving caches virtually-addressed. [1] was first to study address translation on GPUs. TLB were designed per core to provide translation benefits. Further, TLB conscious warp scheduling (TCWS) schemes are proposed and performance is evaluated.

In this work we evaluate the TLB-enabled GPU for workloads from Rodinia benchmark. We analyze the performance, TLB and cache behavior for various scheduling schemes like GTO, Two-level warp scheduling [3] and TCWS. Our results shows that TLB-enabled GPU with TCWS scheduler achieves performance close to GPUs without TLB and also increases cache and TLB hit rates significantly.

2. Methodology

We use TLB-enabled GPGPU-Sim for our experiment. The GPU modeled by GPGPU-Sim is composed of SIMT cores connected via interconnection network to memory partitions. Each SIMT core models highly multithreaded SIMT processors. The TLB is placed in each SIMT core prior to L1 cache and is accompanied by a Page Table. The overall hardware architecture is shown in Figure 1. Figure 2 shows zooms SIMT core with translation hardware. We have, per core, 1024 threads with 32KB L1 data cache (128 byte cache line and LRU policy), 128-entry 4-port

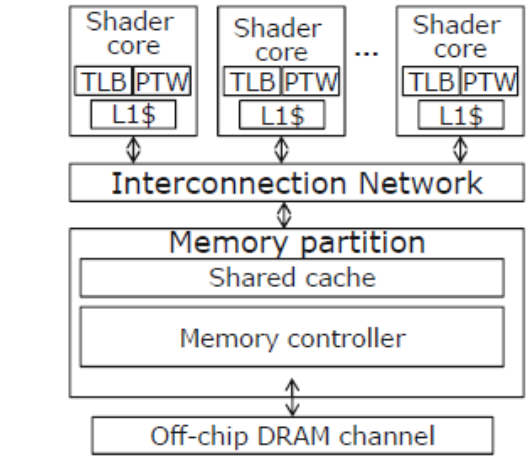

Figure 1 GPU hardware with TLB and PTW per SIMT core

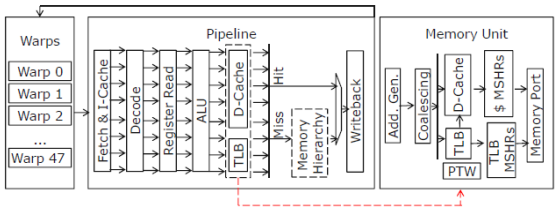

Figure 2 SIMT core pipeline with address translation

TLB,16KB shared memory, and 128KB unified L2 cache shared by all SIMT cores. The results focus on 4KB page size. We use workloads from Rodinia benchmark to analyze the performance of TLB-enabled GPUs for different scheduling schemes and optimizations.

3. Evaluation of warp scheduling algorithms

3.1 Analysis of Augmented TLB for GPUs

We analyze the performance of per-core TLB shared among SIMD lanes. The TLB is placed before L1 cache making cache physically-tagged and virtually-indexed so that TLB access can overlap cache access. Since GPUs are designed to perform SIMD operations on threads within a warp, a warp execution can potentially prompt multiple TLB misses. In worst case, all TLB misses can be to different virtual pages. Figure 3 shows the mean and maximum translations requested by warp. The average page divergence for HotSpot is relatively high. NW has maximum page divergence for a warp but mean divergence is low when compared to others. This is because NW has large memory footprint (512 MB for 8K X 8k input) but works with smaller number of threads per warp (16 threads). There is a significant usage of cache memory among warps which leads to decrease in mean translations requested per warp. The results indicate that the importance of inter-warp locality for cache and TLB reuse. Figure 4 shows the miss penalty of TLB and cache incurred for each benchmark.

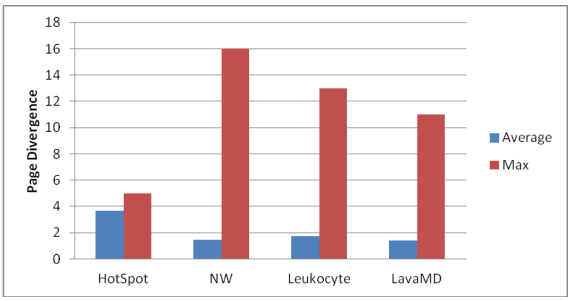

Figure 3 Average number of distinct translations requested per warp and maximum translation requested by any warp

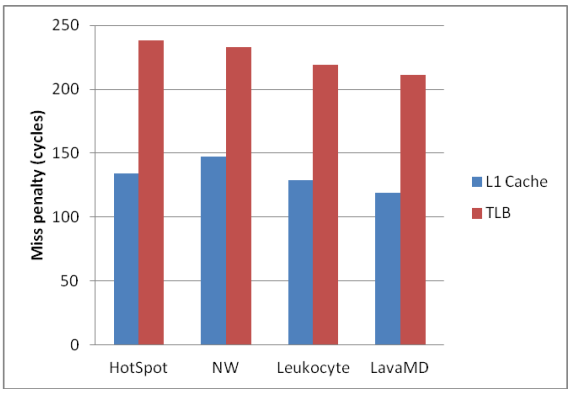

Figure 4 Average miss penalty for TLB and cache

TLB miss penalty is much more than cache and can potentially increase idle cycles for the core.

[1] proposes optimizations to improve performance of TLB through non-blocking TLBs. Two mechanisms are proposed in this paper. One is to swap-in warp whose memory references are all TLB hit. But this may limit schedulable warps and results in idle cycles if there are no such warps. The other idea is to overlap TLB miss with cache access of TLB-hit threads within same warp. This will improve cache hit rate (provided there is no cache thrashing) and facilitates early cache miss detection. The page table walk (PTW) scheduling performance is also improved by coalescing memory access to page table walker and interleaving memory references from different page table walks. Overall, these optimizations aimed to reduce TLB miss latency and overlap with useful work.

Figure 5 shows performance improvement for non-blocking version with PTW scheduling optimizations when compared to blocking TLB. The greedy wavefront scheduler is used. The speedup is still less. The greedy or round-robin scheduler tends to make all warps arrive at same time after long latency memory operation, thus increasing idle cycles.

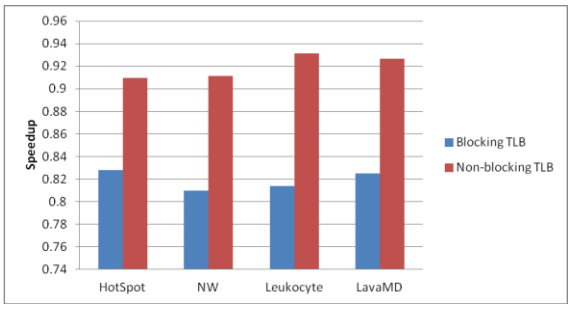

Figure 5 Speedup comparison of Blocking TLB and Non-blocking TLB with optimizations and PTW scheduling for GTO

3.2 Two-level warp scheduling

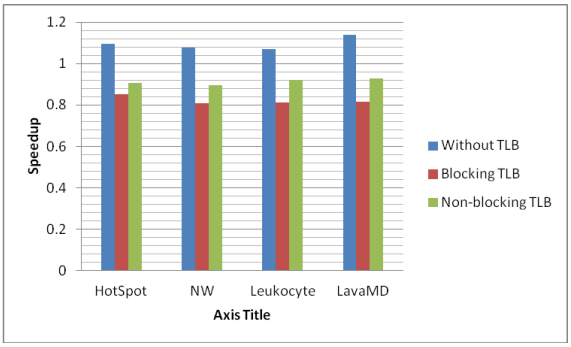

Figure 6 Speedup comparison of Blocking TLB and Non-blocking TLB with optimizations and PTW scheduling for Two-level warp scheduler

[3] proposed two-level scheduler that has active and pending pools to group warps and schedule based on whether they are waiting for short or long latency memory operations. It also de-

creases the scheduler effort to identify ready warp. Figure 6 shows the speedup achieved for two-level scheduler with 8 active warp pool. The results are nearly same as with other schedulers. In all these cases, non-blocking version of TLB performs well than blocking version. So far we have discussed optimizations for TLB-miss handling. The next section evaluates scheduling schemes introduced to improve TLB-hit rate.

3.3 Cache Conscious and TLB aware warp scheduling

As seen in previous section, non-blocking version of TLB achieves much more speedup than blocking version. Let us compare TLB and cache behavior for the same. Figure 7 shows TLB hit-miss and cache hit-miss rate and for blocking and non-blocking TLB. TLB hit rates are not improved significantly for non-blocking version. This is expected behavior since the architecture doesn't support more than one TLB miss. Cache hit rate is improved by less than 10% for HotSpot and NW but remains almost same for Leukocyte and LavaMD. This indicates that speedup doesn't come from overlapping TLB miss with cache access within same warp. There are two possibilities for this behavior. One possibility is cache must have been thrashed by swapped-in warps due to poor scheduling. The other possibility is there may be more threads in warp with TLB miss but cache hit.

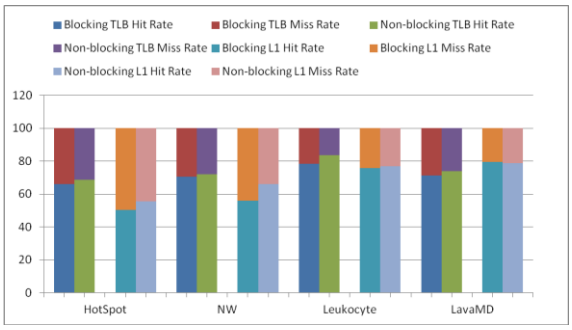

Figure 7 TLB and cache hit-miss rate for blocking and non-blocking TLB

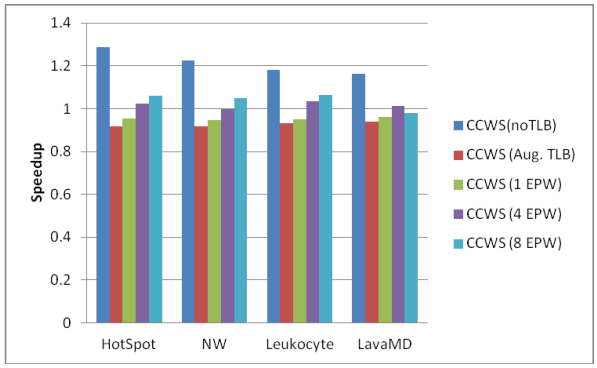

Figure 8 TLB conscious warp scheduling with different number of entries per warp in VTA

The pages may be referenced long back in the past. By the time TLB miss is serviced, the cache line may be evicted. These observations emphasis the need to take into account the inter-warp locality w.r.t TLB and cache and intra-warp TLB-cache co-relation. Also, from the observation of cache-hit rate and speedup mismatch for Leukocyte and LavaMD non-blocking version, we can conclude that the PTW scheduling optimizations is effective in decreasing idle cycle and hence speedup.

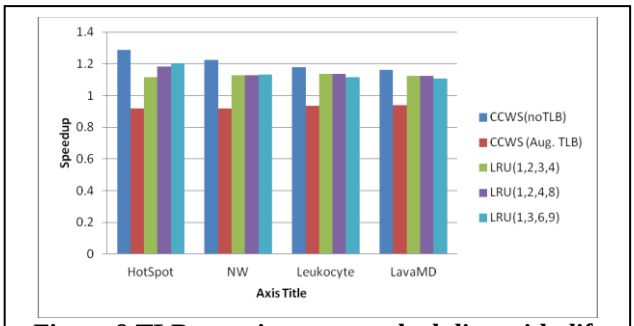

Figure 9 TLB conscious warp scheduling with different LRU depth weights

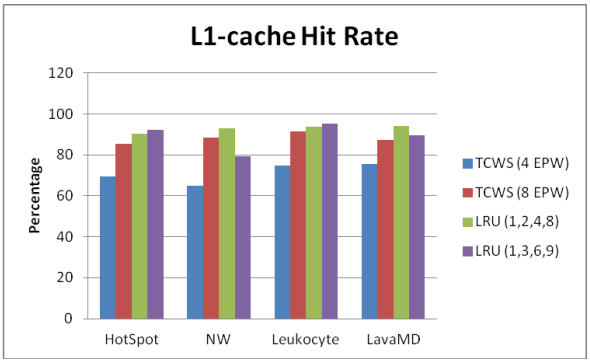

Figure 10 TLB conscious warp scheduling with different LRU depth weights

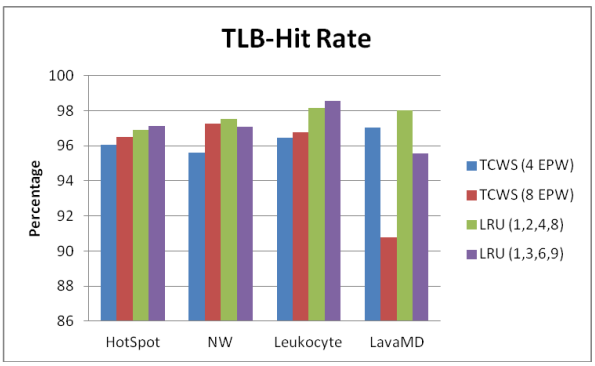

Figure 11 TLB conscious warp scheduling with different number of entries per warp in VTA

Cache Conscious Wavefront Scheduling (CCWS) proposed in [4] improves performance by 20%. [1] proposes schemes to take into account TLB information along with cache for scheduling. Here we will look into TLB Conscious Warp Scheduling (TCWS) technique. TCWS maintains Victim Tag Array (VTA) containing virtual addressed tags and looked upon TLB miss. The Lost Locality Score (LLS) is updated to identify warp that cause thrashing

and share same working set. Please refer to the paper [1] for hardware architecture of scheduler.

Figure 8 shows speedup for various number of entries per warp (EPW) in VTA. 8 EPW performs well as expected. Updating only for TLB miss makes scheduler less adaptive. Now, LSS score is updated for TLB hit also. LRU stack of page table entry is maintained. The scheme weights hits with different score based on how recent they are. Figure 9 shows speedup for LRU stack addition with different scores. We can see that TCWS with LRU performs best and speedup is close to CCWS without TLBs.

Figure 10 and 11 graph TLB-Hit and cache-Hit rate for TCWS with and without LRU stack. The hit-rate of both cache and TLB has improved significantly when compared to non-blocking TLB scheme (Figure). The hit-rate for TLB increased by nearly 35% and for cache increased by atleast 10% to a maximum of 50% (for HotSpot). LRU with (1,2,4,8) scoring performs well with significant TLB and cache hit-rate.

4. Conclusion

We evaluated the performance and characteristics of cache and TLB for various scheduling algorithms. The results shows promising future for TLB-enabled GPUs to support HSA. The benefit from address translation doesn't come for free. Infact they underperform GPUs. Warp scheduler must be redesigned to utilize TLB and cache. TCWS is found to perform well for all workloads in Rodinia benchmark. Many research is carried out in line of warp scheduling. Rather than changing the replacement policy for cache, optimizing the warp formation and scheduling techniques yield better results. Further evaluation is need for CPU-GPU architectures by simulating both CPU and GPU real-time workload.

References

[1] B. Pichai, L. Hsu, A. Bhattacharjee, "Architectural Support for Address Translation on GPUs"

[2] S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer, S.-H.Lee, and K. Skadron. Rodinia: A benchmark suite for heterogeneous computing. In Proceedings of the IEEE Workload Characterization Symposium, pages 44–54, 2009.

[3] M. Gebhart et al., ''Energy-Efficient Mechanisms for Managing Thread Context in Throughput Processors,'' Proc. ACM/IEEE Int'l Symp. Computer Architecture, ACM Press, 2011, pp. 235-246.

[4] T. Rogers, M. O'Connor, and T. Aamodt, "Cache Conscious Wavefront Scheduling,"MICRO, 2012.

[5] A. Bakhoda, G. Yuan, W. Fung, H. Wong, and T. Aamodt, "Analyzing CUDA Workloads Using a Detailed GPU Simulator," ISPASS, 2009.

[6] V. Narasiman, C. J. Lee, M. Shebanow, R. Miftakhutdinov, O. Mutlu, and Y. N. Patt. Improving GPU performance via large warps and two-level warp scheduling. In MICRO 44: Proceedings of the 44th annual IEEE/ACM International Symposium on Microarchitecture, December 2011..