Memory Compression Coordinated and Optimized Prefetching in GPU Architectures

Nandita Vijaykumar
nandita@cmu.edu

1. Introduction

Traditionally, GPU architectures have been primarily focused on throughput and latency hiding. However, as the computational power of GPUs continues to scale with Moore's law, an increasing number of applications are becoming limited by memory bandwidth [1]. Also, data locality and reuse are becoming increasingly important with power-limited technology scaling. The energy spent on off-chip memory transfers are orders of magnitude greater than the energy spent on on-chip accesses and actual computation [2]. Main memory compression in CPU/GPUs using techniques like Linearly Compressed Pages [3] could provide reduced memory bandwidth as well as increased memory capacity and energy reduction. GPUs could potentially benefit more from main memory compression as the data is usually more regular and the DRAM structure is more flexible.

GPU architectures typically tolerate long memory access latencies by concurrently executing many threads which are grouped into batches known as warps. When one warp is stalled by a long latency memory access, another warp is swapped in thereby overlapping the latencies of multiple accesses. However, if all the warps are simultaneously stalled on a long latency memory access, it could lead to significant underutilization of compute resources. This leads to the problem of how to improve tolerance towards long memory access latencies in a massively threaded GPU system when thread level parallelism is not sufficient. In conventional CPUs, prefetching is a commonly used technique to mitigate high memory access latencies. Directly applying prefetching to a bandwidth limited GPU system, however, could in fact hurt performance [4]. Prefetch requests, whose accuracy cannot be guaranteed, increase the number of memory requests which could delay other demand requests. This problem is more severe in GPUs as the number of memory requests pending at any given is much higher than in a CPU. In addition to this, prefetching useless cachelines to the very limited GPU last level cache could cause significant cache pollution further degrading performance. Prefetching could however be very helpful if the memory system can accommodate the additional requests without delaying the demand requests.

2. Problem Statement

Prefetching is a technique that has been used effectively in CPUs to reduce high latency memory access related stalls [5, 6, 7, 8, 9, 10]. But integrating prefetching techniques into severely bandwidth limited GPU architectures is an interesting challenge. The goals for this project are as follows:

- To study the positive and possible detrimental impacts of different conventional prefetching mechanisms on the last level cache of a GPU.
- To evaluate the sensitivity and performance of different workloads with different prefetchers tuned to different levels of aggressiveness.
- To identify an effective prefetcher based mechanism to mitigate high off-chip memory access latencies without degradation in performance due to excessive bandwidth consumption or cache pollution.
- To evaluate the interaction between prefetching and main memory compression in GPUs. For example, with Linearly Compressed Pages [3], it is possible to retrieve more than one compressed cache block in a single access.
- Coordination between the prefetcher and the compression engine could lead to more efficient, accurate data transfers and better utilization of the last level cache. The aim is to evaluate and implement synergistic request filtering, prefetcher throttling and memory compression mechanisms to effectively integrate prefetching and DRAM compression in a GPU architecture.

3. Related Work

Prefetching in GPUs. Existing warp scheduling policies do not enable very effective integration of data prefetchers. Jog et al. [11, 12] propose prefetch-aware thread scheduling policies that enable L1 cache prefetchers to be effective in tolerating memory latencies. Lee et al. [4] propose inter-thread prefetching mechanisms tailored to GPGPUs along with an adaptive throttling mechanism to reduce the negative impact of prefetching. Arnau et al. [13] show the ineffectiveness of standard prefetching techniques for graphics applications and use a decoupled access/execute mechanism for graphics processing. Sethia et al. [14] aim to reduce the number of thread contexts required to hide long memory access latencies and propose an adaptive prefetcher by identifying fixed-offset address and thread invariant access patterns between threads. Meng et al. [15] proposed Dynamic Warp Subdivision where performance improves due to the advantages of prefetching effects by early execution of some threads in a warp.
Main Memory Compression. Satish et al. [16] investigate a mechanism to improve the performance of memory bound

workloads using a hardware based memory I/O link compression technique where data is stored and retrieved from main memory in a compressed form to reduce bandwidth requirements. Tremaine et al. [17] propose a memory controller design, Pinnacle, based on IBM Memory Extension Technology [18] to manage main memory in CPUs where mappings between the compressed pages and virtual memory is handled using an uncompressed real address space. Ekman and Stenstorm [19] propose a main memory compression design using a variant of Frequent Pattern Compression and the operating system maps the uncompressed virtual address space directly to the compressed physical address space. The Linearly Compressed Pages framework [3] aims to compress main memory in CPUs in order to increase the effective capacity. This mechanism also has a prefetching effect by bringing in multiple blocks in a single transfer.

Adaptive Prefetching. Some adaptive prefetching mechanisms that have been proposed for CPUs include [20] where a feedback mechanism is used to adjust prefetcher aggressiveness based on prefetcher timeliness, accuracy and cache pollution. Ebrahimi et al. [21] coordinate multiple prefetchers in a multicore architecture to address prefetcher caused inter-core interference. Lee et al. [22] evaluate the benefits and limitations of different hardware and software prefetching techniques. Dahlgren et al. [7] measured the prefetch accuracy and adjusted the prefetch distance based on the accuracy. Zhuang and Lee [23] proposed hardware based pollution filters for processors employing aggressive prefetching. Some other prefetching and prefetch control techniques that have been proposed in the CPU context include [24, 25, 26].

4. Methodology And Plan

We intend to conduct these experiments and evaluations using GPGPU-Sim [27]. GPGPU-Sim provides a detailed simulation model of a contemporary GPU (such as NVIDIA's Fermi and GT200 architectures) running CUDA workloads and includes an integrated and validated energy model.

Milestone 1 – 14th October:
- Review existing literature on prefetching in GPUs and CPUs and main memory compression mechanisms.
- Get familiar with GPGPU-Sim.

Milestone 2 – 28th October:
- Incorporate prefetchers into the simulator cache model.
- Sensitivity studies with different prefetchers.
- Study the data patterns in GPGPU workloads.

Milestone 3 – 4th November:
- Evaluate and improve effectiveness of prefetching on improving performance and reducing bandwidth usage.
- Evaluate the impact of cache pollution, prefetch accuracy and timeliness on performance.
- Study the interaction between scheduling and prefetcher performance.
- Evaluate the interaction between different prefetchers.

Milestone 4 – 18th November:
- Study the interaction between main memory compression schemes and different prefetching mechanisms.
- Implement and evaluate an integrated mechanism to effectively coordinate the prefetcher and compression engine.
- Evaluate and reduce cache pollution caused by prefetching and accesses to compressed blocks.
- Study the impact of data re-alignment in main memory to improve the coordinated performance impact of the prefetching and memory compression.

References

[1] Khailany Bauer, Cook. Cudadma: optimizing gpu memory bandwidth via warp specialization. In *SC '11 Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, 2011.

[2] Mark Gebhart, Stephen W. Keckler, Brucek Khailany, Ronny Krashinsky, and William J. Dally. Unifying primary cache, scratch, and register file memories in a throughput processor. In *Proceedings of the 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*, 2012.

[3] Gennady Pekhimenko, Vivek Seshadri, Yoongu Kim, Hongyi Xin, Onur Mutlu, Michael A. Kozuch, Phillip B. Gibbons, and Todd C. Mowry. Linearly compressed pages: A main memory compression framework with low complexity and low latency. In *MICRO-46*, 2013.

[4] Jaekyu Lee, Nagesh B. Lakshminarayana, Hyesoon Kim, and Richard Vuduc. Many-thread aware prefetching mechanisms for gpgpu applications. In *Proceedings of the 2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture*, 2010.

[5] J. Fu and J. Patel. Data prefetching in multiprocessor vector cache memories. In *ISCA-18*, 1991.

[6] W. C. Fu, J. H. Patel, and B. L. Janssens. Stride directed prefetching in scalar processors. In *MICRO-25*, 1992.

[7] F. Dahlgren, M. Dubois, and P. Stenström. Sequential hardware prefetching in shared-memory multiprocessors. In *IEEE Transactions on Parallel and Distributed Systems, 6(7):733–746*, 1995.

[8] Iacobovic, Sorin, et al. Effective stream-based and execution-based data prefetching. In *ICS*, 2004.

[9] K. J. Nesbit and J. E. Smith. Ac/dc: An adaptive data cache prefetcher. In *PACT-13*, 2004.

[10] N. P. Jouppi. Improving direct-mapped cache performance by the addition of a small fully-associative cache and prefetch buffers. In *ISCA-17*, 1990.

[11] Adwait Jog, Onur Kayiran, Asit K. Mishra, Mahmut T. Kandemir, Onur Mutlu, Ravishankar Iyer, and Chita R. Das. Orchestrated scheduling and prefetching for gpgpus. In *Proceedings of the 40th Annual International Symposium on Computer Architecture*, 2013.

[12] Adwait Jog, Onur Kayiran, Nachiappan Chidambaram Nachiappan, Asit K. Mishra, Mahmut T. Kandemir, Onur Mutlu, Ravishankar Iyer, and Chita R. Das. Owl: cooperative thread array aware scheduling techniques for improving gpgpu performance. In *ASPLOS '13*, 2013.

[13] José-María Arnau, Joan-Manuel Parcerisa, and Polychronis Xekalakis. Boosting mobile gpu performance with a decoupled access/execute fragment processor. In *Proc. of the 39th Annual International Symposium on Computer Architecture*, 2012.

[14] Ankit Sethia, Ganesh Dasika, Mehrzad Samadi, and Scott Mahlke. Apogee: adaptive prefetching on gpus for energy efficiency. In *Proceedings of the 22nd international conference on Parallel architectures and compilation techniques*, 2013.

[15] J. Meng, D. Tarjan, and K. Skadron. Dynamic warp subdivision for integrated branch and memory divergence tolerance. In *ISCA-37*, 2010.

[16] Vijay Sathish, Michael J. Schulte, and Nam Sung Kim. Lossless and Lossy Memory I/O Link Compression for Improving Performance of GPGPU Workloads. In *PACT*, 2012.

[17] R. Brett Tremaine et al. Pinnacle: IBM MXT in a Memory Controller Chip. 2001.

[18] B. Abali, H. Franke, D. E. Poff, R. A. Saccone, C. O. Schulz, L. M. Herger, and T. B. Smith. Memory expansion technology (mxt): software support and performance. 2001.

[19] Magnus Ekman and Per Stenstrom. A Robust Main-Memory Compression Scheme. In *ISCA-32*, 2005.

[20] Santhosh Srinath et al. Feedback Directed Prefetching: Improving the Performance and Bandwidth-Efficiency of Hardware Prefetchers. In *HPCA-13*, 2007.

[21] E. Ebrahimi, O. Mutlu, J. L. Chang, and Y. Patt. Coordinated control of multiple prefetchers in multi-core systems. In *Proc. of the 42nd Annual International Symposium on Microarchitecture*, 2009.

[22] Jaekyu Lee, Hyesoon Kim, and Richard Vuduc. When prefetching works, when it doesnt́, and why. 2012.

[23] X. Zhuang and H.-H. S. Lee. A hardware-based cache pollution filtering mechanism for aggressive prefetches. In *ICPP-32*, 2003.

[24] E. Ebrahimi, O. Mutlu, and Y. Patt. Coordinated control of multiple prefetchers in multi-core systems. In *MICRO-42*, 2009.

[25] K. J. Nesbit and J. E. Smith. Data cache prefetching using a global history buffer. In *HPCA-15*, 2004.

[26] Doug Joseph and Dirk Grunwald. Prefetching using markov predictors. In *Proceedings of the 24th annual international symposium on Computer architecture*, 1997.

[27] A. Bakhoda, G. Yuan, W. W. L. Fung, H. Wong, and T. M. Aamodt. Analyzing cuda workloads using a detailed gpu simulator. In *IEEE ISPASS*, 2009.