PISTONCache: Shared DRAMCache for Integrated Heterogeneous Systems

Abstract

This document is intended to serve as a sample for submissions to ISCA 2016. We provide some guidelines that authors should follow when submitting papers to the conference. In an effort to respect the efforts of reviewers and in the interest of fairness to all prospective authors, we request that all submissions follow the formatting and submission rules detailed below.

1. Introduction

The remarkable advances in computing power of the modern microprocessor over the last few decades can predominantly be attributed to Moore's Law and advances in manufacturing technology that has allowed shrinking of transistor sizes. Miniaturization of transistors has allowed addition of specialized on-chip hardware circuitry for acceleration units. A study in 2010 by Koomey et. al [14] found that the amount of computation that could be done per unit of energy doubled about every 18 months. However, to reach exascale and beyond requires a thousand fold decrease in energy consumed per flop computed. Graphics processing units (GPUs) have evolved from being fixed-function pipelines and are being used to accelerate data parallel code for general purpose (GP) applications. Compared with multi-core CPUs, GPGPUs offer the potential for better performance at lower energy. Traditionally these discrete processors have had their own independent memory systems. To take advantage of the discrete GPUs, the CPU must copy data to the GPUs memory and back. This data movement is wasteful and expends energy while also adding latency as the transfer happens over a slower PCIe bus. The separate address spaces and complex programming models that need to manage 2 sets of data further impede expansion of the workloads that benefit from GPGPU computing.

Modern processor chips have had lower capacity GPUs on-die allowing for graphics rendering only. However in view of the widespread use of the GPU for general purpose applications, processor manufacturers including AMD[2], Intel[4], and NVIDIA[1] are beginning to allow general purpose OpenCL/CUDA programs to run on their Integrated Heterogeneous System (IHS) platform which were so far restricted to graphics rendering. To this effect the HSA Foundation [3] was setup to develop and define cross-vendor hardware specifications and software development tools needed to allow application software to better use this architecture. This architecture provides a shared virtual address space making pointer sharing semantics possible on CPU and GPU which simplifies programming. Further a shared physical address space reduces GPU initialization time and enables several high level languages to also take advantage of the parallel processing synergistically with the CPU. Programmers can now write applications that seamlessly integrate CPUs with GPUs while benefiting from best attributes of each. Finer-grain data parallel sections in applications like garbage collection [6] of virtual machines and parallel stream processing like face detection, compression, encryption-decryption etc. can now use the integrated GPGPU to deliver better performance. This integrated architecture also has the added advantage of being able to run programs whose dataset sizes are not constrained by the size of memory on the GPU. The GPU can invoke traditional operating systems paging mechanisms and hardware MMUs to fault pages. As IHS platforms gain widespread adoption in HPC systems, improving the performance of these platforms will become of paramount importance in the near future. AMD proposed designs also hints towards this direction [18]

In contrast to the processing capabilities of the modern microprocessor, DRAM memory speeds have not kept pace commensurately to serve the increasing demands of the processors. This speed imbalance coupled with a limited growth in pin counts has led to the memory [19] and bandwidth [17] wall for off-chip DRAM systems which often becomes a performance limiting factor. The advent of die-stacking technology [8] provides a way to integrate disparate silicon die of NMOS DRAM chips and CMOS logic chips with better interconnects. The implementation is accomplished either by 3D vertical stacking of DRAM chips using through-silicon vias (TSV) interconnects or horizontally/2.5D stacking on a interposer chip as depicted in Figure 1. This allows the addition of a sizable DRAM chip close to processing cores. The onchip DRAM memory can provide anywhere from a couple of hundreds of megabytes to a few gigabytes of storage at high bandwidths of 400GB/s compared to the 90GB/s of DDR4 bandwidth. The better interconnect also lowers the latency of access by around 20-25% compared to off-chip memory. This on-chip DRAM capacity has been advocated to be used as a large last level cache which is transparent to software for improveming performance of multicore CMPs in several works in literature. Similarly in the context of IHS chips, the high bandwidth stacked DRAMCache can cater to large bandwidth requirements for the high MLP and throughput oriented GPUs while latency oriented CPU applications can benefit from reduced latency of data access from the stacked DRAMCache thus providing improved overall system performance. The stacked DRAM Cache also reduces energy consumed per access for the overall system in line with the goals of IHS.

As we race towards exascale compute facilities more of the die will be use for processing elements and accommodating

larger size SRAM based caches or even higher density STT-RAM based caches [20] of sizes which will be useful for these large number of computing elements will not be feasible. These on die caches also suffer from congestion on the NoC and require special designs to be efficient. An alternate to this is to use vertical die stacking to provide large capacity caches which have significant bandwidth and some latency benefits as well.

However, this introduces novel challenges in managing contention for shared memory resources where the DRAMCache is the first level of shared capacity in the memory hierarchy of the two heterogeneous processor architectures. When a GPU kernel is launched it creates large number of concurrently running threads in lock-step in a SIMD execution model and injects large number of requests to DRAMCache. The GPU can also switch execution between groups of threads to hide this memory access latency and exploit available parallelism which further exacerbates the problem. This causes bottlenecks in request queues and contention for sets in the DRAMCache thus severally impacting CPU performance. Although the GPU can sustain longer memory latencies, as capabilities of execution units in GPUs of IHS chips increase, threads will be able issue larger number of memory requests more rapidly and available parallelism to hide memory latency decreases, thus reducing throughput of the entire system. Since the GPU can execute in a burst mode reserving shared resources will lead to idling and under-utilization of resources.

The primary contributions of this work are summarized as follows:

- We propose the addition of a large capacity stacked DRAM for IHS processors. This first level shared memory capacity between CPU and GPU would be used a last-level cache to contain the majority of the large working set of GPU which otherwise will not fit in a on-chip SRAM cache. The considerable bandwidth provided by the DRAMCache should benefit GPU programs and the lower latency of access will assist latency sensitive CPU applications.
- We study the effects of sharing this resource between the processors and show that conventional DRAMCache architectures do not perform well when applied naively for IHS processors. This is the first of the kind evaluation for a IHS setup.
- We propose an intelligent IHS aware scheme for managing occupancy levels for a direct mapped DRAMCache called *PISTON*. *PISTON* is lightweight and at the same time avoids explicit cache partitioning which would result in unused capacity when either cores are idle. *PISTON* does not compromise on .
- The predictor is central to extracting performance from a DRAMCache by avoiding high latency tags-in-DRAM lookups by predicting a miss ahead of time and starting an early fill request to memory in parallel. We adapt the DRAMCache predictor to be IHS aware in concordance with our occupancy control mechanism. This allows each type of processor to maximize the use of bandwidth and latency benefits provided by the DRAMCache.
- Finally we enable *PISTON* to dynamically adjust occupancy at runtime for varying cache requirements of heterogeneous cores to maximize the overall system throughput that can be got by using this DRAMCache.

To the best of our knowledge, this is the first study on the interactions of IHS with a shared stacked DRAMCache. The rest of the paper is organized as follows. Section 2 demonstrates the performance that can be gained by adding a DRAMCache to a IHS chip and motivates the need for architecting an IHS aware approach to DRAMCache organization. We present the principles of PISTON Cache and its descibe its working in Section 3 . Next, in Section 4 we show the experimental setup and methodology followed by evaluation results in Section 5. Section 6 presents related work in this area and Section 7 concludes the paper.

2. Motivation

As seen in section 1i, incorporating GPU into the same die as CMPs provides for several advantages and the use of this as a general purpose computing resource is gaining traction in applications. However the directions for improving the performance of these integrated contrasting processors is not straight forward. In this work we propose the addition of a large capacity stacked DRAM between CPU and GPU. This capacity will be used as a last level cache before accessing the off-chip DRAM. Adding smaller capacity SRAM caches as shared levels does not contribute much to performance (why?) and DRAMs are better suited to handle larger number of requests and interference (why?). And hence the DRAMCache is the first level shared memory capacity in the hierarchy of the 2 processors. Due to the requirement of large storage, of order of the last level SRAM caches, prior works have placed metadata alongside the data in the DRAM devices and focused on improving access latencies for the accesses. Set associativity usually improves cache hit rates by reducing conflict misses. However in DRAMCaches associativity comes at a cost of increased hit latencies. Since each way comparison would require a tag to be burst out of the DRAM. Metadata is placed in DRAM

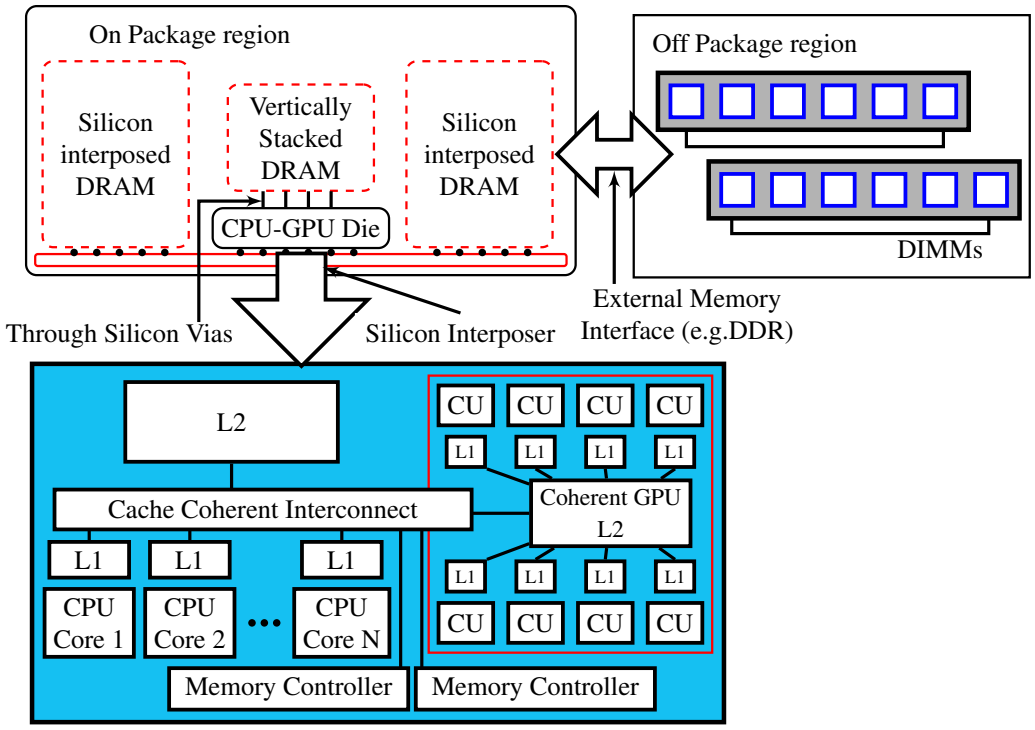
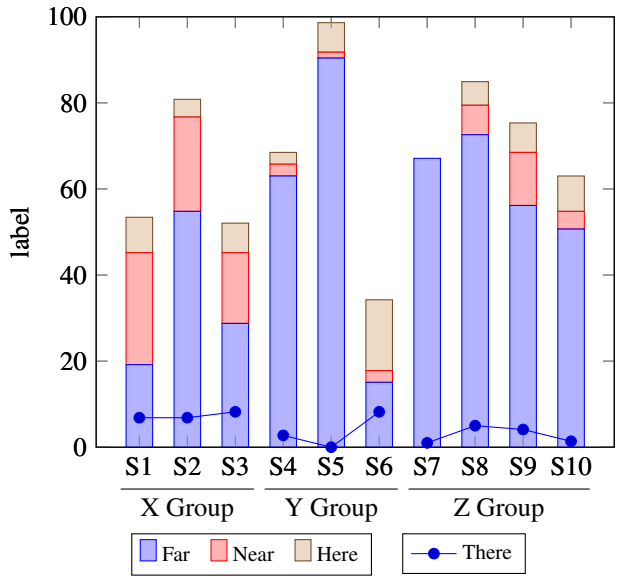


Figure 1: Architecture of a Integrated Heterogeneous System



3. Chaining Mechanism

3.1. Stacked DRAM Organization

As discussed in Section 2 we propose to use the stacked DRAM capacity as a large last level cache. PISONCache uses a direct mapped cache as proposed in [16] and commercially adopted in the stacked MCDRAM on the Knights Landing generation of the Intel XeonPhi processor [5]. In this organization the metadata is stored as a TAD (Tag-and-Data) unit alongside the data and is retrieved from the DRAM in multiple bursts. These stacked DRAM provide large amount of bandwidth using a large number of smaller capacity banks and TSV to transfer data. This exploit this abundant Bank Level Parallelism PISTONCache to modifies the address mapping scheme to map a single cache line across different banks within the same channel (vault). Contrary to popular belief this organization exploits the higher BLP as opposed to RBH. Since the data in the cache correspond to cache lines and even consecutive addresses are mapped to different cache sets stored in different rows, the DRAMCache inherently does not see high row buffer hit rates.

4. Experimental Setup & Methodology

We evaluate the performance of the DRAMCache over a Integrated Heterogeneous Processor using the a cycle accurate simulator gem5-gpu [15] which is configured to simulate cache coherent unified CPUs and GPUs. The cache hierarchy has per SM private GPU L1 that are non-inclusive of the shared GPU L2 cache and can hold stale data. However, GPU L2 cache is coherent with all levels of the CPU hierarchy. The CPU and GPU cache are kept coherent using the VI_Hammer protocol. Table 1 provides the setup details.

Workloads Applications having high and medium memory intensive behaviours from SPEC CPU2006 suite [12] were chosen to form a multi-programmed workload. We use the Rodinia benchmark suite [9] to represent applications that use the GPU offload for execution. These Rodinia applications are modified to elide the memcpy api calls so as to run with

unified virtual and physical address spaces. Combinations of quad-core and eight-core multi-programmed workloads were coupled with a full Rodinia benchmark to create a representative mix of applications that would run in a intergrated heterogenous system. These composite workloads embody different levels of total memory intensity produced by the CPU and GPU cores.

DRAMCache Design The DRAMCache we evaluate here is the first level shared cache between the 2 split cache hierarchies of CPUs and GPUs, while they already share a last level on-chip SRAM cache homogenously. Acess to memory is interleaved across across multiple memory controllers and a biased interleaving, due to uniformly strided address patters, is restricted by using a basic XOR-based hashing mechanism. Each memory controller manages a 4GB DDR3 memory device at 1600MHz DRAM and a 32MB HMC vault at 2500MHz. The stacked DRAM vault caches data from corresponding the 4GB memory device that the controller is responsible for. This setup ensures that there are no cross bus requests between controllers. Since our setup has the limitations of having a 1-to-1 channel mapping between a stacked DRAM vault and a off-chip DRAM channel, our model does not provide sufficient channel level parallelism as is quintessential in a stacked DRAM device, where the large number of vaults and TSVs allow the stacked DRAM to provide a larger bandwidth. To circumvent this limitation we increase the number of layers (ranks) to provide higher amount of parallelism in our stacked DRAM. However, the the bank capacity are retained as would be in a standard stacked DRAM. We faithfully model a Fill Queue [10] for fill requests to insert data into the cache on the return path from main memory. We also model MSHR and WriteBuffers with their associated latencies to realize the precise working of caches. Our DRAMCache also respects all significant timing and functional parameters using the DRAMCtrl [7] model. The DRAMCache is non-inclusive [13] of the L2 caches above it and uses a write-noallocate policy.

Simulation Methodology We fast-forward the initialization phase of each workloads up until just before the launch of the first kernel of the GPU program. We ensure that each core is fast-forwarded by atleast 2 Billion instructions and in total each quad-core workload by 20 billion instructions and each eight-core workload executes x Billion, on average in the fast-forward phase. This is accomplished by adding no-ops to the Rodinia benchmarks for the duration until the initialization quota of the SPEC programs is complete. We then warm the cache until the fastest core completes 250 million instructions. During the warmup phase the GPU program is not executed. Timing simulations were then run for 250 million and 100 million for each CPU core for quad-core workload and eight-core workloads respectively. As is the norm, when a core finishes its quota of instructions, it continues to execute until all the cores have completed.

The Rodinia application uses an extra CPU core and offloads to the integrated GPU. These applications are modified to execute in a *conditioned loop* such that there is no corruption of data structures in the program. The *conditioned loop* is run infinitely and represents the region of interest of the Rodinia benchmark. This region includes sections of CPU activity and GPU offloads in the execution, as is typical of a HSA program which will exhibit an interleave of offloaded regions and serial CPU sections. However, only the performance statistics for the first execution of the *conditioned loop* in the program are considered. In cases where the ROI is longer than the complete run of the CPU workloads, the statistics for the last completed kernel are used.

Performance Metric We report performance by adapting the ANTT [11] metric for Integrated Heterogeneous Systems and define it as follow

$$ANTT_{HSA} = \frac{1}{n_{cpu}} \sum_{i=1}^{n_{cpu}} \frac{CPI_i^{IHS}}{CPI_i^{SP}} + \frac{CPI^{GPU_{IHS}}}{CPI^{GPU}}$$

where CPI_i^{IHS} and $CPI^{GPU_{IHS}}$ denote the cycles per instruction achieved by the i^{th} CPU and GPU when running in a IHS setup respectively; whereas CPI_i^{SP} and CPI^{GPU} denote the same when i^{th} CPU and GPU are running standalone respectively.

CPU Core	GHz, OoO x86-64 ISA
Abstract font	10pt, italicized
as	12pt, bold
as	10pt, bold
Caption font	9pt, bold
References	8pt

Table 1: Configuration of the simulated system

Workload	Multi-program SPEC2006	Rodinia
Qne1(2)	cactusADM;gcc;bzip2;sphinx3	needle
Qne2(8)	astar;mcf;gcc;bzip2	needle
Qne3(16)	gcc;libquantum;leslie3d;bwaves	needle
Qkm1(10)	astar;soplex;cactusADM;libquantum	k-means
Qkm2(12)	milc;mcf;libquantum;bzip2	k-means
Qkm3(15)	bzip2;gobmk;hmmer;sphinx3	k-means
Qga1(14)	astar;mcf;soplex;cactusADM	gaussian
Qga2(23)	soplex;milc;cactusADM;libquantum	gaussian
Qga3(26)	milc;libquantum;gobmk;leslie3d	gaussian
Qhs1(4)	astar;milc;gcc;leslie3d	hotspot
Qhs2(32)	gcc;gobmk;leslie3d;sphinx3	hotspot
Qsr1(6)	astar;cactusADM;libquantum;sphinx3	srad
Qsc1(7)	astar;mcf;gobmk;sphinx3	streamcluster
Qlu1(21)	astar;cactusADM;libquantum;sphinx3	lud

Table 2: Workloads

5. Results

6. Related Work

7. Conclusion

References

[1] "Fastest processors, smartphones, and tablets - nvidia tegra," http://www.nvidia.com/object/tegra.html.

[2] "The future of the apu - braided parallelism," http://developer.amd.com/wordpress/media/2013/06/2901_final.pdf.

[3] "Hsa foundation standards," http://www.hsafoundation.com/standards/.

[4] "Intel graphics opencl," https://software.intel.com/en-us/node/540387.

[5] "Intel xeonphi processor datasheet," http://www.intel.com/content/dam/www/public/us/en/documents/datasheets/xeon-phi-processor-x200-product-family-datasheet.pdf.

[6] "Java sumatra," http://openjdk.java.net/projects/sumatra/.

[7] *2014 IEEE International Symposium on Performance Analysis of Systems and Software, ISPASS 2014, Monterey, CA, USA, March 23-25, 2014*. IEEE Computer Society, 2014. [Online]. Available: http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=6832911

[8] B. Black, M. Annavaram, N. Brekelbaum, J. DeVale, L. Jiang, G. H. Loh, D. McCaule, P. Morrow, D. W. Nelson, D. Pantuso, P. Reed, J. Rupley, S. Shankar, J. Shen, and C. Webb, "Die stacking (3d) microarchitecture," in *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO 39. Washington, DC, USA: IEEE Computer Society, 2006, pp. 469–479. [Online]. Available: http://dx.doi.org/10.1109/MICRO.2006.18

[9] 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 2009 IEEE International Symposium on Workload Characterization (IISWC)*, ser. IISWC '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 44–54. [Online]. Available: http://dx.doi.org/10.1109/IISWC.2009.5306797

[10] A. J. Cheng-Chieh Huang, Vijay Nagarajan, "Dca: a dram-cache-aware dram controller," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, November. 2016.

[11] S. Eyerman and L. Eeckhout, "System-level performance metrics for multiprogram workloads," *IEEE Micro*, vol. 28, no. 3, pp. 42–53, May 2008.

[12] J. L. Henning, "Spec cpu2006 benchmark descriptions," *SIGARCH Comput. Archit. News*, vol. 34, no. 4, pp. 1–17, Sep. 2006. [Online]. Available: http://doi.acm.org/10.1145/1186736.1186737

[13] C. C. Huang, R. Kumar, M. Elver, B. Grot, and V. Nagarajan, "C3d: Mitigating the numa bottleneck via coherent dram caches," in *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Oct 2016, pp. 1–12.

[14] J. Koomey, S. Berard, M. Sanchez, and H. Wong, "Implications of historical trends in the electrical efficiency of computing," *IEEE Annals of the History of Computing*, vol. 33, no. 3, pp. 46–54, March 2011.

[15] J. Power, J. Hestness, M. Orr, M. Hill, and D. Wood, "gem5-gpu: A heterogeneous cpu-gpu simulator," *Computer Architecture Letters*, vol. 13, no. 1, Jan 2014. [Online]. Available: http://gem5-gpu.cs.wisc.edu

[16] M. K. Qureshi and G. H. Loh, "Fundamental latency trade-off in architecting dram caches: Outperforming impractical sram-tags with a simple and practical design," in *Proceedings of the 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO-45. Washington, DC, USA: IEEE Computer Society, 2012, pp. 235–246. [Online]. Available: http://dx.doi.org/10.1109/MICRO.2012.30

[17] B. M. Rogers, A. Krishna, G. B. Bell, K. Vu, X. Jiang, and Y. Solihin, "Scaling the bandwidth wall: Challenges in and avenues for cmp scaling," in *Proceedings of the 36th Annual International Symposium on Computer Architecture*, ser. ISCA '09. New York, NY, USA: ACM, 2009, pp. 371–382. [Online]. Available: http://doi.acm.org/10.1145/1555754.1555801

[18] M. J. Schulte, M. Ignatowski, G. H. Loh, B. M. Beckmann, W. C. Brantley, S. Gurumurthi, N. Jayasena, I. Paul, S. K. Reinhardt, and G. Rodgers, "Achieving exascale capabilities through heterogeneous computing," *IEEE Micro*, vol. 35, no. 4, pp. 26–36, July 2015.

[19] W. A. Wulf and S. A. McKee, "Hitting the memory wall: Implications of the obvious," *SIGARCH Comput. Archit. News*, vol. 23, no. 1, pp. 20–24, Mar. 1995. [Online]. Available: http://doi.acm.org/10.1145/216585.216588

[20] J. Zhan, O. Kayıran, G. H. Loh, C. R. Das, and Y. Xie, "Oscar: Orchestrating stt-ram cache traffic for heterogeneous cpu-gpu architectures," in *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Oct 2016, pp. 1–13.