

HAShCache: Heterogeneity Aware Shared DRAMCache for Integrated Heterogeneous Systems

Adarsh Patil, Ramaswamy Govindarajan

Department of Computer Science and Automation
Indian Institute of Science

January 23, 2018

Appears in ACM Transactions on Architecture and Code Optimization (TACO) Dec 2017

1 Introduction

- Integrated Heterogeneous Systems (IHS)
- 3D Die-stacked DRAM

2 Motivation

3 The HASHCache proposal

- HASHCache Design
- HASHCache Mechanisms

4 Evaluation Methodology

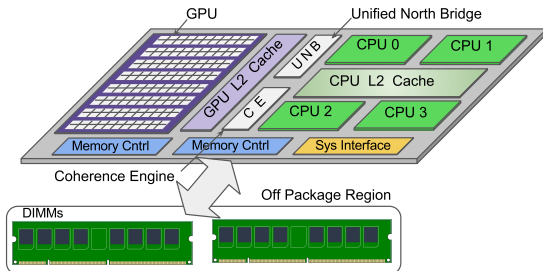
5 Results

6 Conclusion

Integrated Heterogenous Systems (IHS) Architecture

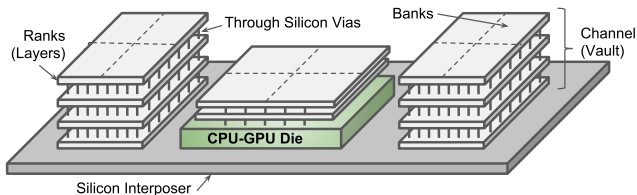
Latency-oriented CPU cores + Throughput-oriented GPGPU SMs **on-chip**

- Simplifies Programming - Shared Virtual Memory, pointer sharing
- Allows GPGPUs to operate on data sets larger than memory size
- Reduces overall energy consumption
- Share resources - NoC, caches, memory controllers, DRAMs
- e.g. AMD APUs, Intel Iris, NVIDIA Denver

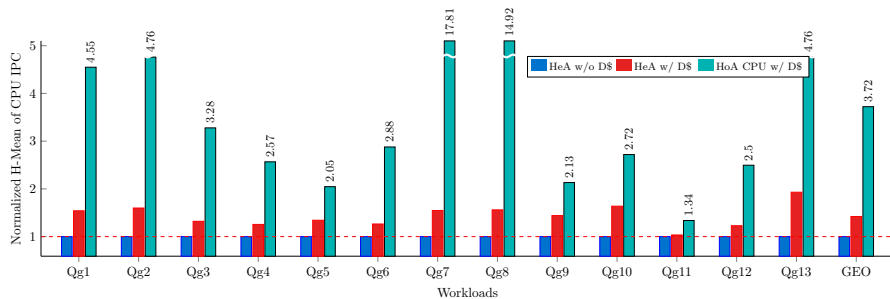


3D Die-stacked DRAM

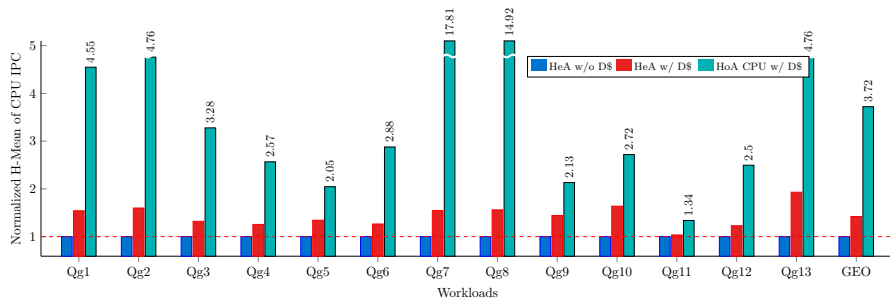
- DRAM layers placed closer to processing cores using 2.5D interposer and/or 3D TSV (through-silicon-vias)
- **Large capacity** - can contain IHS working set
- **Large Bandwidth** - beneficial for GPGPUs
- **Slightly lower access latency** - beneficial for CPUs
- Our proposal uses this capacity as a **hardware managed cache**
- e.g. HBM (AMD/Hynix), HMC (Intel/Micron)



(a) CPU Performance in IHS with Naive DRAMCache

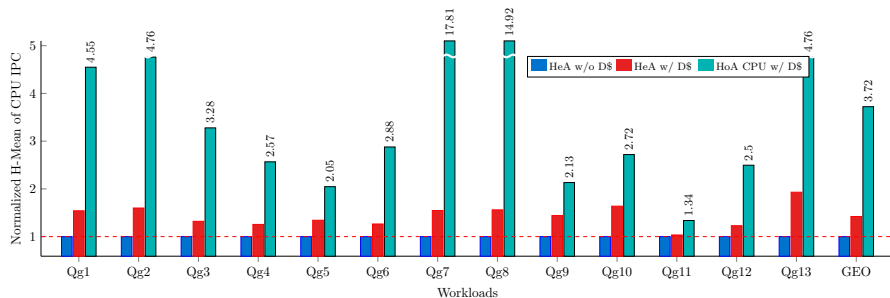


(a) CPU Performance in IHS with Naive DRAMCache



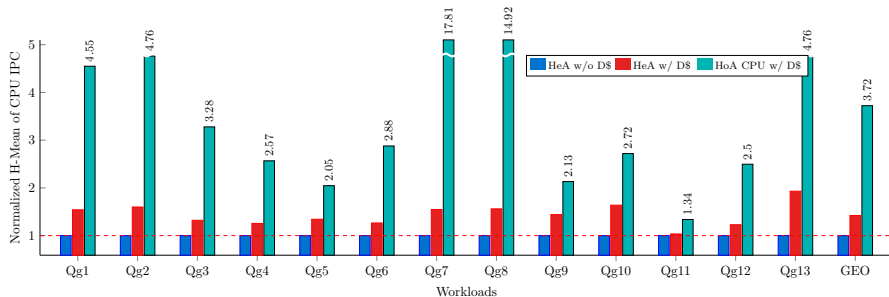
- CPU performance improves by 42% over baseline

(a) CPU Performance in IHS with Naive DRAMCache



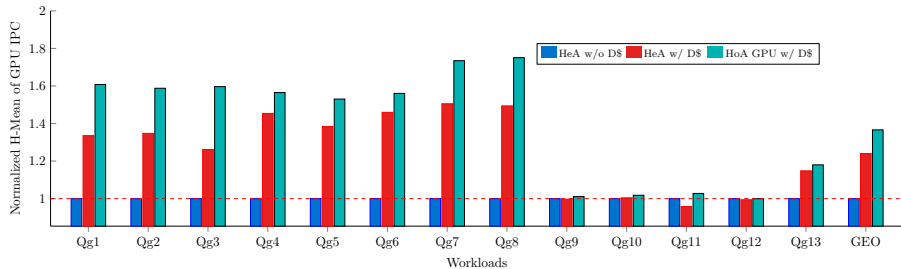
- CPU performance improves by 42% over baseline
- Ideal Homogeneous CPU achieves 372% improvement over baseline

(a) CPU Performance in IHS with Naive DRAMCache

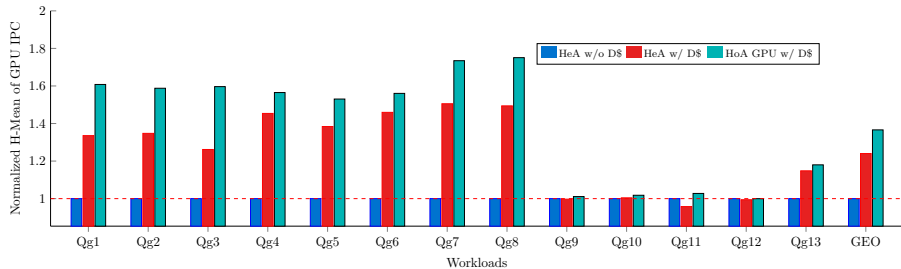


- CPU performance improves by 42% over baseline
- Ideal Homogeneous CPU achieves 372% improvement over baseline
- **Performance gap of almost 260% compared to the ideal**

(b) GPU Performance in IHS with Naive DRAMCache

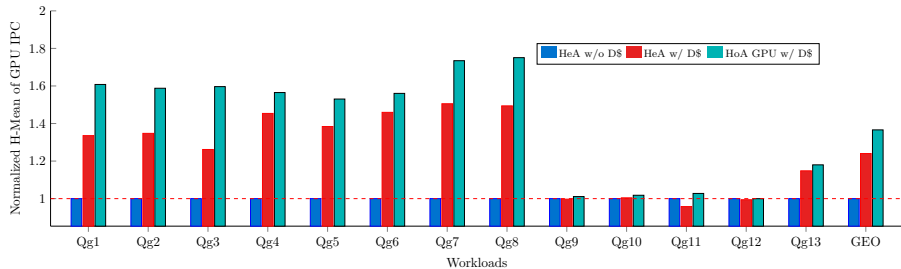


(b) GPU Performance in IHS with Naive DRAMCache



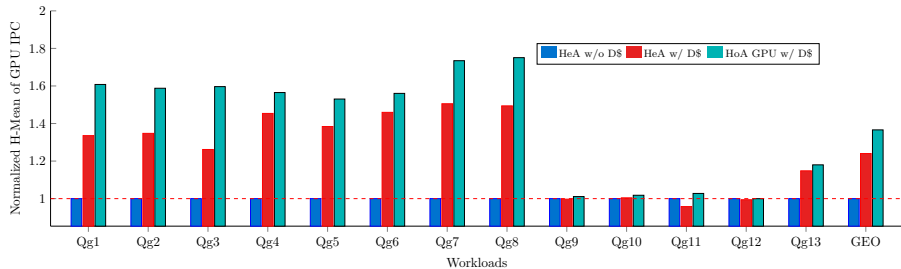
- GPU performance improves by 24% over baseline

(b) GPU Performance in IHS with Naive DRAMCache



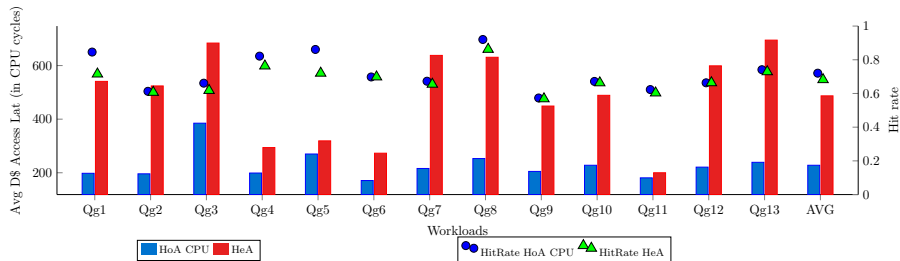
- GPU performance improves by 24% over baseline
- Ideal Homogeneous GPU achieves 35% improvement over baseline

(b) GPU Performance in IHS with Naive DRAMCache



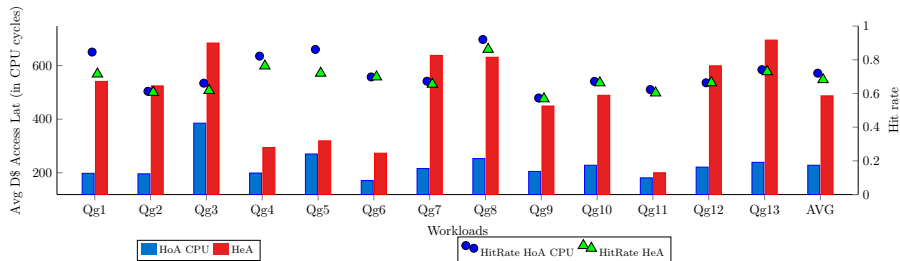
- GPU performance improves by 24% over baseline
- Ideal Homogeneous GPU achieves 35% improvement over baseline
- **Performance gap within 10% of its ideal**

Cause for CPU Performance gap



- CPU Hit rates marginally affected (about 4%)

Cause for CPU Performance gap



- CPU Hit rates marginally affected (about 4%)
- Increase in avg mem access latency by 213% due to co-running GPU

The HShCache Proposal

- An optimized DRAMCache solution for IHS processors
- Efficient DRAMCache design for heterogeneous architecture
 - Carefully architect the first order design constraints
 - Cache block size, metadata overheads, set associativity, miss penalty, addressing scheme
- Three Heterogeneity aware DRAMCache mechanisms
 - **Heterogeneity aware DRAMCache scheduler - PrIS**
Reduce large access latencies for CPU requests
 - **Heterogeneity aware Temporal Bypass - ByE**
Allow CPU req to utilize the idle off-chip DRAM Bandwidth
 - **Heterogeneity aware Spatial Occupancy Control - Chaining**
Allow GPU to better use DRAMCache Bandwidth

HAShCache Design

Metadata overhead

Caching at 128 byte granularity (as opposed to larger 1-2KB granularity)
Tags stored in DRAM alongside data and streamed out in multiple bursts (TADs)

Set Associativity

Direct mapped cache organization

Reducing Miss Penalty

Cache Hit/miss Predictor based on MAP-I prediction for CPU
No such mechanism for GPU

Addressing Scheme

Addressed with RBH addressing scheme (Row-Rank-Bank-Column-Channel)
(as opposed to BLP addressing scheme (Row-Column-Rank-Bank-Channel))

HAShCache Design

Metadata overhead

Caching at 128 byte granularity (as opposed to larger 1-2KB granularity)
Tags stored in DRAM alongside data and streamed out in multiple bursts (TADs)

Set Associativity

Direct mapped cache organization

Reducing Miss Penalty

Cache Hit/miss Predictor based on MAP-I prediction for CPU
No such mechanism for GPU

Addressing Scheme

Addressed with RBH addressing scheme (Row-Rank-Bank-Column-Channel)
(as opposed to BLP addressing scheme (Row-Column-Rank-Bank-Channel))

HAShCache Design

Metadata overhead

Caching at 128 byte granularity (as opposed to larger 1-2KB granularity)
Tags stored in DRAM alongside data and streamed out in multiple bursts (TADs)

Set Associativity

Direct mapped cache organization

Reducing Miss Penalty

Cache Hit/miss Predictor based on MAP-I prediction for CPU
No such mechanism for GPU

Addressing Scheme

Addressed with RBH addressing scheme (Row-Rank-Bank-Column-Channel)
(as opposed to BLP addressing scheme (Row-Column-Rank-Bank-Channel))

HAShCache Design

Metadata overhead

Caching at 128 byte granularity (as opposed to larger 1-2KB granularity)
Tags stored in DRAM alongside data and streamed out in multiple bursts (TADs)

Set Associativity

Direct mapped cache organization

Reducing Miss Penalty

Cache Hit/miss Predictor based on MAP-I prediction for CPU
No such mechanism for GPU

Addressing Scheme

Addressed with RBH addressing scheme (Row-Rank-Bank-Column-Channel)
(as opposed to BLP addressing scheme (Row-Column-Rank-Bank-Channel))

Heterogeneity aware DRAMCache scheduler - PrIS

OBJECTIVE: Reduce large access latencies for CPU req at DRAMCache

Heterogeneity aware DRAMCache scheduler - PrIS

OBJECTIVE: Reduce large access latencies for CPU req at DRAMCache

- DRAM Memory Controllers use a scheduler to reorder request from the queue to best suit DRAM characteristics (e.g. First Ready-FCFS)

Heterogeneity aware DRAMCache scheduler - PrIS

OBJECTIVE: Reduce large access latencies for CPU req at DRAMCache

- DRAM Memory Controllers use a scheduler to reorder request from the queue to best suit DRAM characteristics (e.g. First Ready-FCFS)

OBSERVATIONS:

- large number of GPU requests \implies exhaust the queue positions \implies DRAMCache blocked and CPU requests being rejected
- GPU requests have good row buffer locality \implies preferentially scheduled \implies large queueing latency for CPU requests

Heterogeneity aware DRAMCache scheduler - PrIS

OBJECTIVE: Reduce large access latencies for CPU req at DRAMCache

- DRAM Memory Controllers use a scheduler to reorder request from the queue to best suit DRAM characteristics (e.g. First Ready-FCFS)

OBSERVATIONS:

- large number of GPU requests \implies exhaust the queue positions \implies DRAMCache blocked and CPU requests being rejected
- GPU requests have good row buffer locality \implies preferentially scheduled \implies large queueing latency for CPU requests

CPU Prioritized IHS aware Scheduler

Queue entry reservation for CPU req when queues reach critical levels

CPU Request Prioritization for next request to be serviced

Binary selection for scheduling simplifies the scheduling algorithm to a single stage and incurs no additional hardware overhead

Heterogeneity aware Temporal Bypass - ByE

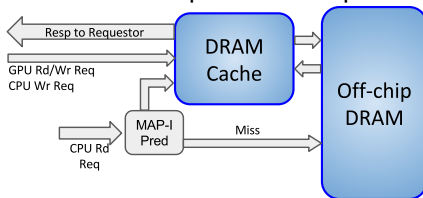
OBJECTIVE: Utilize the idle off-chip DRAM Bandwidth

Heterogeneity aware Temporal Bypass - ByE

OBJECTIVE: Utilize the idle off-chip DRAM Bandwidth

OBSERVATIONS:

- Large DRAMCache capacity \implies high hit rates and long block residency time before being evicted \implies under-utilized off-chip DRAM bandwidth
- Baseline incorporates a hit/miss predictor for CPU requests and starts early off-chip access for a request that is predicted to be a miss

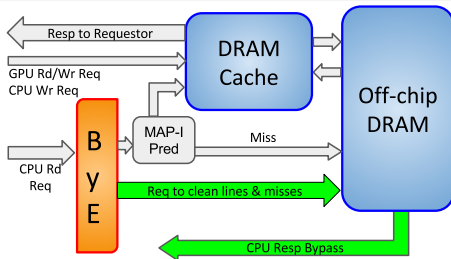


- However, we observe that when GPU is running, CPU requests returning from off-chip access wait at DRAMCache for verification of hit/miss via a tag match

Heterogeneity aware Temporal Bypass - ByE

Bypass Enabler (ByE)

Bypasses CPU req to clean lines & cache misses when GPU is running
Achieved using a **Counting Bloom Filter** that tracks dirty lines



- Bypassed CPU requests are not filled into DRAMCache
- Hardware Overhead: 256KB (0.4% of cache capacity)

Heterogeneity aware Spatial Occupancy Control - Chaining

OBJECTIVE: Allow GPU to better use DRAMCache Bandwidth

Heterogeneity aware Spatial Occupancy Control - Chaining

OBJECTIVE: Allow GPU to better use DRAMCache Bandwidth

OBSERVATIONS:

- GPU Hit rates in DRAMCache should be high enough that the GPU requests do not have to frequently use the relatively constricted off-chip DRAM bus
- GPU can trade off access latency for higher hit rates
- CPU applications have smaller working sets of a few MBs
- Small size GPU L2 caches provide limited filtering of traffic as compared to the CPU L2 cache

Heterogeneity aware Spatial Occupancy Control - Chaining

OBJECTIVE: Allow GPU to better use DRAMCache Bandwidth

OBSERVATIONS:

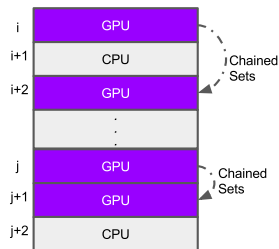
- GPU Hit rates in DRAMCache should be high enough that the GPU requests do not have to frequently use the relatively constricted off-chip DRAM bus
- GPU can trade off access latency for higher hit rates
- CPU applications have smaller working sets of a few MBs
- Small size GPU L2 caches provide limited filtering of traffic as compared to the CPU L2 cache

Chaining

Provides **pseudo associativity for GPU** lines inserted in DRAMCache
Ensures **minimum occupancy of CPU lines** in the DRAMCache

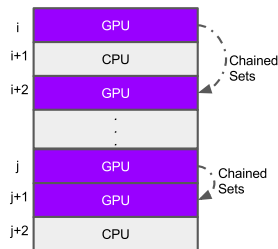
Heterogeneity aware Spatial Occupancy Control - Chaining

- Linear probing like scheme to resolve conflicts



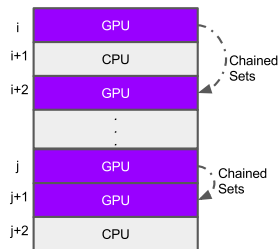
Heterogeneity aware Spatial Occupancy Control - Chaining

- Linear probing like scheme to resolve conflicts
- Maintains low-threshold value (I_{cpu}) occupancy for CPU lines in DRAMCache by ensuring GPU line does not replace CPU lines when this threshold is reached



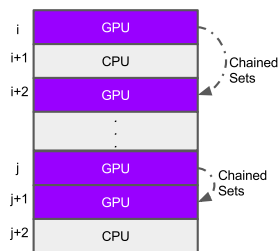
Heterogeneity aware Spatial Occupancy Control - Chaining

- Linear probing like scheme to resolve conflicts
- Maintains low-threshold value (I_{cpu}) occupancy for CPU lines in DRAMCache by ensuring GPU line does not replace CPU lines when this threshold is reached
- For a GPU line evicting a GPU line, chaining tries to replace a CPU line in any of the consecutive 3 locations



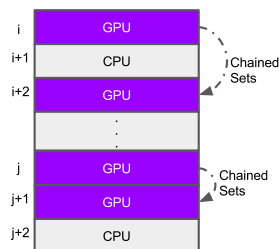
Heterogeneity aware Spatial Occupancy Control - Chaining

- Linear probing like scheme to resolve conflicts
- Maintains low-threshold value (I_{cpu}) occupancy for CPU lines in DRAMCache by ensuring GPU line does not replace CPU lines when this threshold is reached
- For a GPU line evicting a GPU line, chaining tries to replace a CPU line in any of the consecutive 3 locations
- For all CPU requests, data is always inserted into the original location



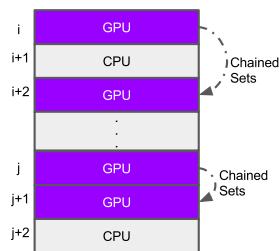
Heterogeneity aware Spatial Occupancy Control - Chaining

- Linear probing like scheme to resolve conflicts
- Maintains low-threshold value (I_{cpu}) occupancy for CPU lines in DRAMCache by ensuring GPU line does not replace CPU lines when this threshold is reached
- For a GPU line evicting a GPU line, chaining tries to replace a CPU line in any of the consecutive 3 locations
- For all CPU requests, data is always inserted into the original location
- Chained set location is represented by a 2 bit offset (used for lookup) + 2 bit reverse chain bits for the chained set (used for eviction)

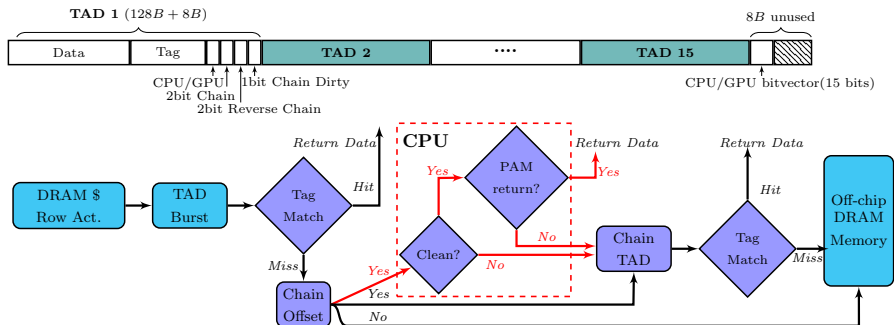


Heterogeneity aware Spatial Occupancy Control - Chaining

- Linear probing like scheme to resolve conflicts
- Maintains low-threshold value (I_{cpu}) occupancy for CPU lines in DRAMCache by ensuring GPU line does not replace CPU lines when this threshold is reached
- For a GPU line evicting a GPU line, chaining tries to replace a CPU line in any of the consecutive 3 locations
- For all CPU requests, data is always inserted into the original location
- Chained set location is represented by a 2 bit offset (used for lookup) + 2 bit reverse chain bits for the chained set (used for eviction)
- psuedo-associativity of atmost 1 and atmost 2 tag matches are needed



Heterogeneity aware Spatial Occupancy Control - Chaining



Chaining organization within the DRAMCache row and working

- CPU/GPU share address space, hence require both sets to be checked
- Chain dirty bits avoid second set lookup for CPU
- Chain offset, Reverse chain offset, Chain dirty bits, set ownership information uses the unused bits in the DRAMCache row

Experimental Setup

CPU Core	five 4-wide OoO x86 cores @2.5 GHz
CPU Caches	32KB 8-way split I/D private L1 Cache, 1MB 8-way shared split L2 Cache, 128B lines
GPU Core	eight Fermi SMs@700MHz , 2x2 GTO warp sched 64K registers, 96KB scratch memory
GPU Caches	64KB 4-way private L1 cache, 512KB, 16-way assoc L2 Coherent Cache
Stacked DRAM	2 vaults, HMC_2500_x64 , 2KB Page $t_{CL}-t_{RCD}-t_{RP}-t_{RAS}=9.9ns-10.2ns-7.7ns-21.6ns$ 8 layers/vault, 4 banks/layer 64 byte burst size, Peak bandwidth 40GB/s
Off-chip DRAM	2 channels, DDR3_1600_x64 , 1KB page $t_{CL}-t_{RCD}-t_{RP}-t_{RAS}=13.75ns-13.75ns-13.75ns-35ns$ 1 rank/channel, 8 banks/rank 64 byte burst size, Peak bandwidth 25GB/s

Simulator

- gem5-gpu
- SE mode
- Heterogenous Cache Coherence

DRAMCache

- Memory-side cache
- MSHR, WriteBuffers
- Fill Queue
- Non-inclusive

Table: Configuration of the simulated system

Medium to high memory intensive *Multi-programmed SPEC 2006* combinations on CPU coupled with a *Rodinia no-copy* GPU benchmark

Name	Multi-program SPEC2006	Rodinia
Qg1	cactusADM;gcc;bzip2;sphinx3	needle
Qg2	astar;mcf;gcc;bzip2	needle
Qg3	gcc;libquantum;leslie3d;bwaves	needle
Qg4	astar;soplex;cactusADM;libquantum	k-means
Qg5	milc;mcf;libquantum;bzip2	k-means
Qg6	bzip2;gobmk;hmmer;sphinx3	k-means
Qg7	soplex;milc;cactusADM;libquantum	gaussian
Qg8	milc;libquantum;gobmk;leslie3d	gaussian
Qg9	astar;milc;gcc;leslie3d	hotspot
Qg10	gcc;gobmk;leslie3d;sphinx3	hotspot
Qg11	astar;cactusADM;libquantum;sphinx3	srad
Qg12	astar;mcf;gobmk;sphinx3	streamcluster
Qg13	astar;cactusADM;libquantum;sphinx3	lud

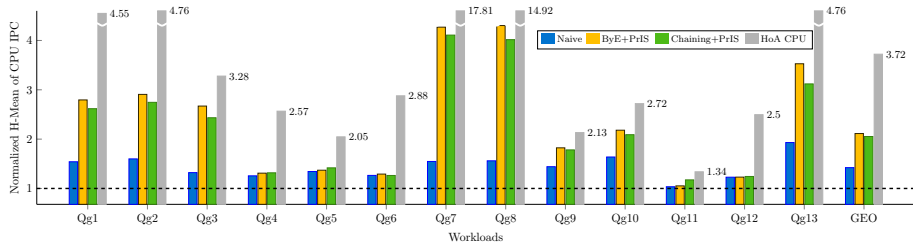
Table: Workloads

- Fast-forward init phase until just before launch of the first kernel of the GPU program
- Each CPU forwarded by atleast 2B instructions (avg 20B per mix)
- Warmup phase until the fastest core reaches 250M instructions
- Detailed Timing simulations - atleast 250M instructions on each core
- Rodinia benchmarks run in a *conditioned loop* represents the ROI
- Performance Metric - Harmonic mean, Weighted Speedup

$$H-MEAN_{CPU} = \frac{n_{cpu}}{\sum_{i=1}^{n_{cpu}} \frac{1}{IPC_i^{CPU}}}, \quad H-MEAN_{GPU} = \frac{n_{gpu}}{\sum_{i=1}^{n_{gpu}} \frac{1}{IPC_i^{GPU}}}$$

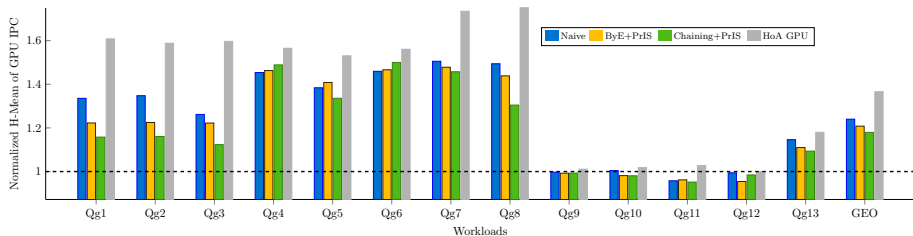
$$H-MEAN_{IHS} = \frac{n_{cpu} + n_{gpu}}{\sum_{i=1}^{n_{cpu}} \frac{1}{IPC_i^{CPU}} + \sum_{i=1}^{n_{gpu}} \frac{1}{IPC_i^{GPU}}}, \quad WS = \sum_{i=1}^{n_{cpu}} \frac{IPC_i^{CPU_{IHS}}}{IPC_i^{SP}} + \sum_{i=1}^{n_{gpu}} \frac{IPC_i^{GPU_{IHS}}}{IPC_i^{GPU}}$$

Results - Individual CPU and GPU performance

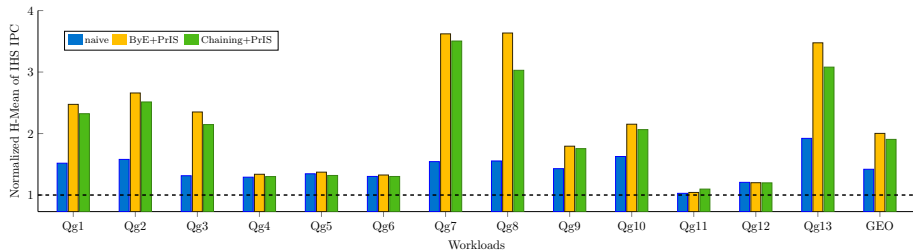


ByE+PrIS - CPU perf 49% ↑, GPU perf 3% ↓ over naive DRAMCache

Chaining+PrIS - CPU perf 46% ↑, GPU perf 6% ↓ over naive DRAMCache

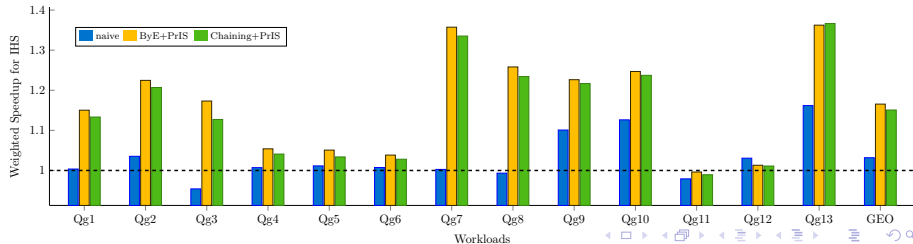


Results - System Performance



ByE+PrIS - H-Mean of IPC 207% ↑, WS of IPC 16% ↑

Chaining+PrIS - H-Mean of IPC 201% ↑, WS of IPC 15% ↑



Comparison with Related Work

- Staged Memory Scheduling (SMS) ¹
 - ① implements a 3 stage DRAM scheduler
 - ② requires large buffers per stage
 - ③ complex multi stage algorithm compared to PrIS
 - ④ idling of buffers when corresponding cores are idle.

¹Staged memory scheduling: achieving high performance and scalability in heterogeneous systems, Rachata Ausavarungnirun et. al., ISCA 2012

²A Mostly-Clean DRAM Cache for Effective Hit Speculation and Self-Balancing Dispatch, Jaewoong Sim et. al., MICRO 2012

Comparison with Related Work

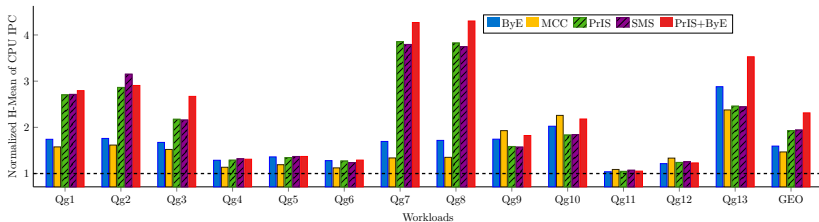
- Staged Memory Scheduling (SMS) ¹
 - ① implements a 3 stage DRAM scheduler
 - ② requires large buffers per stage
 - ③ complex multi stage algorithm compared to PrIS
 - ④ idling of buffers when corresponding cores are idle.

- Mostly Clean Cache (MCC) ²
 - ① dispatches some requests to off-chip DRAM
 - ② uses a hybrid write policy for the DRAMCache
 - ③ supports write-through and write-back policies for different rows
 - ④ limits number of rows in write-back mode which allows MCC limit the amount of dirty data in the DRAMCache.

¹Staged memory scheduling: achieving high performance and scalability in heterogeneous systems, Rachata Ausavarungnirun et. al., ISCA 2012

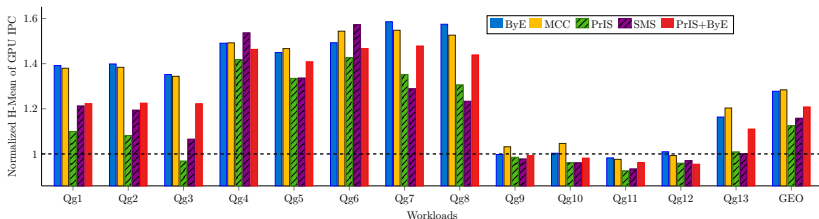
²A Mostly-Clean DRAM Cache for Effective Hit Speculation and Self-Balancing Dispatch, Jaewoong Sim et. al., MICRO 2012

Comparison with Related Work

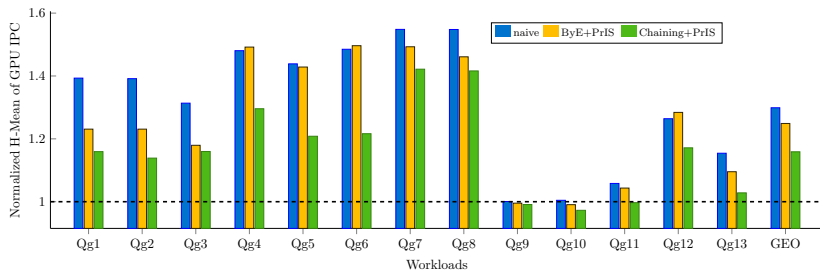
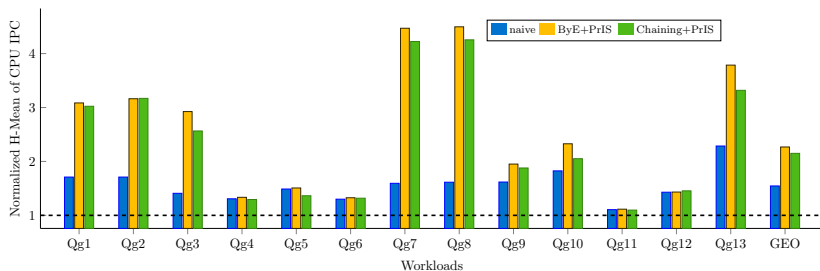


PrIS+ByE - CPU 19% ↑, GPU 5% ↑ compared to SMS

PrIS+ByE - CPU 57% ↑, GPU 6% ↓ compared to MCC



Sensitivity Study with 128MB DRAMCache

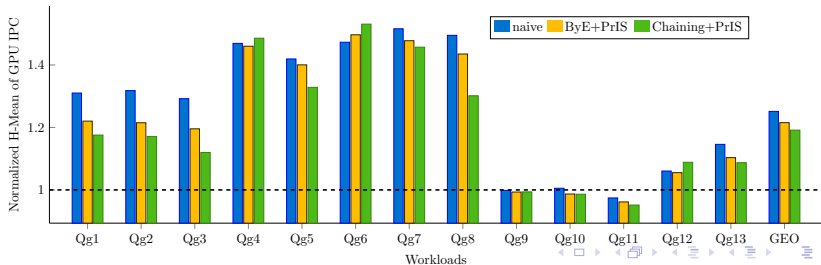
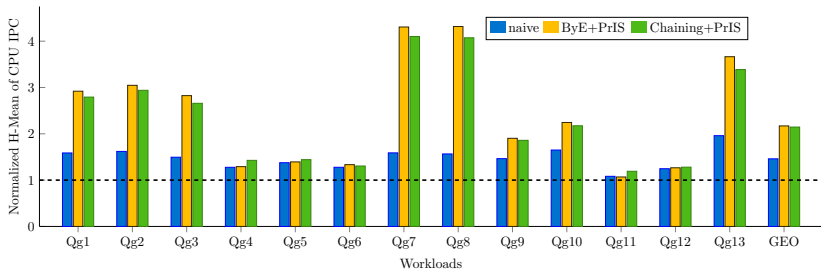


- We present a case for performance improvement of IHS with a stacked DRAMCache device
- Quantify the effects of co-running on each processor and show that heterogeneity adversely affects CPU performance
- We carefully design a DRAMCache organization for IHS processors
- We propose 3 heterogeneity-aware optimizations for DRAMCache - a scheduler, a spatial bypass and a temporal occupancy control scheme that improves performance of the overall system by over 200% over a system with no DRAMCache and 41% over a system with a naive DRAMCache
- Significant benefits of using a stacked DRAMCache for IHS processors, far exceeding the usefulness of the such devices in homogeneous GPUs and multi-core CPU systems.

BACKUP

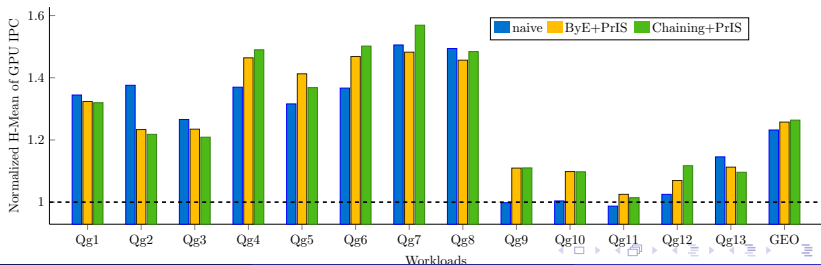
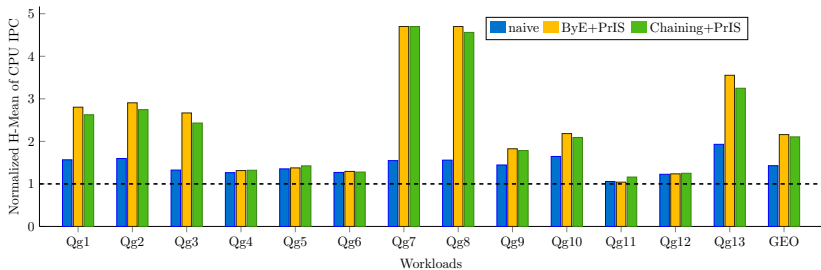
Off-chip DRAM with latency similar to stacked DRAM

DD3 2133 MHz like off-chip DRAM



Higher Bandwidth DRAMCache

Doubling the DRAMCache bandwidth 80 GB/s



Partitioned DRAMCache to alleviate interference

