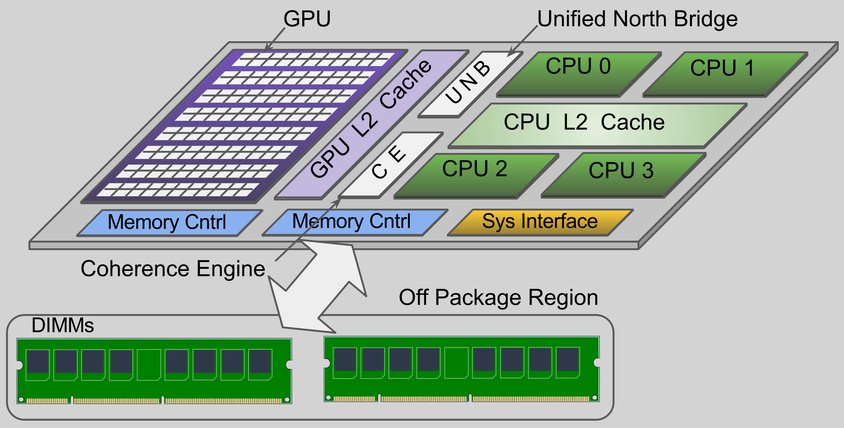
HAShCache: Heterogeneity Aware Shared DRAMCache For Integrated Heterogenous Architectures

Adarsh Patil, R Govindarajan

Department of CSA, Indian Institute of Science, Bangalore

Integrated Heterogenous Systems (IHS) Architecture

- *Throughput-oriented* GPGPU SMs + *Latency-oriented* CPU cores on-chip
- Shared Physical/Virtual Address Space and a Unified Memory Hierarchy
- Improved Programmability
- AMD APUs, Intel Iris, NVIDIA Denver



Vertically Stacked DRAM

DRAM Layers stacked using 2.5D interposer or 3D TSV

	Stacked DRAM	**Off-chip DRAM**
Capacity	$\sim$ 64MB - 4GB	$\sim$ 4GB - 128GB
Bandwidth	$\sim$ 500GB/s	$\sim$ 90 GB/s
Latency	$\sim$ 30ns - 35ns	$\sim$ 50ns
Interconnect	TSV (through-silicon-vias)	Memory Channels
Standards	$HBM_{(AMD/Hynix)}$, $HMC_{(Intel/Micron)}$	DDR4. GDDR5

Motivation and Design

Performance

- Naive addition of DRAM$ over IHS
  - CPU performs 42% better while Homogeneous CPU achieves 372% improvement
    2.6x performance gap
  - GPU performs 24% better while Homogeneous GPU achieves 26.4% improvement
    10% performance gap
- Un-managed interference and Heterogeneity in the DRAM$

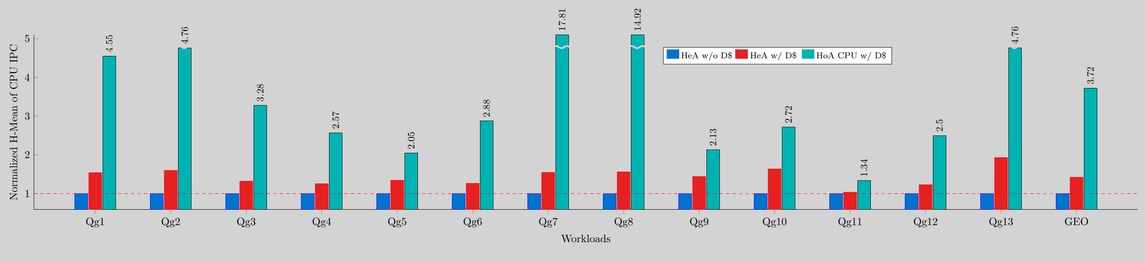


Figure: Performance comparison of CPU & GPU in IHS with D$ vs Homogeneous with D$

Causes for sub-optimality of DRAM$

- Increased DRAM$ access times for CPU despite comparable hit rates
- Allow GPU to occupy enough cache to benefit from the large DRAM$ bandwidth

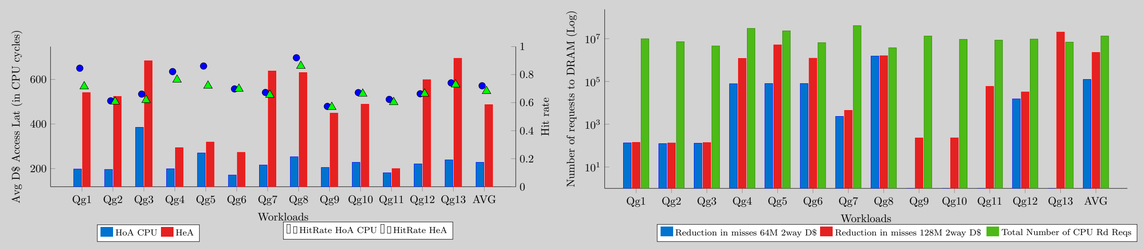


Figure: (a)CPU D$ Access Latency and Hit Rates (b)GPU Misses with 2-way assoc cache

Design Point	**Design Decision**
Metadata Overhead	Tags in DRAM, 128 Byte TAD (Tag-and-Data) Units
Set Associativity	Direct Mapped
Miss Penalty	Miss Predictor for CPU requests
Addressing Scheme	Row-Rank-Bank-Column-Channel (RoRaBaCoCh)

Table: HAShCache Design Decisions

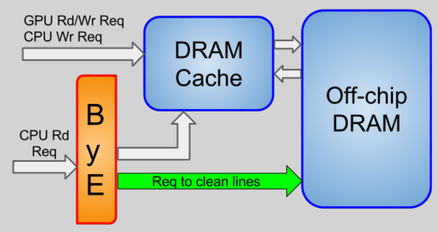
HAShCache = PrIS + ByE + Chaining

① Hetereogenity Aware DRAM$ Scheduling: PrIS

- *OBJECTIVE:* Reduce large access latencies for CPU requests at DRAM$
- Large number of GPU requests $\Longrightarrow$ queues fill up rapidly $\Longrightarrow$ CPU request rejected
- GPU requests have good row buffer locality $\Longrightarrow$ preferentially scheduled $\Longrightarrow$ large queuing latency for CPU requests
- *Achieved* using
  - Queue entry reservation for CPU requests when queues reach critical levels
  - CPU Prioritized FR-FCFS with IHS-aware scheduling algorithm

② Temporal Selective Bypass Enabler : ByE

- *OBJECTIVE:* Utilize the idle DRAM bandwidth
- Bypass CPU requests to clean cache lines and cache misses
- *Achieved* using a *Counting Bloom Filter* that tracks dirty lines in cache
- Overhead: 256KB (0.4% of cache capacity)



③ Spatial Occupancy Control : Chaining

- *OBJECTIVE:* Allow GPU to better use DRAM$ bandwidth
- *Achieved* by providing pseudo-associativity for GPU, thus improving GPU hit rate
- Provides guaranteed minimum occupancy for CPU lines in the cache
- GPU set conflicts resolved by evicting an adjoining "*chained*" set belonging to the CPU
- Overhead: NIL, uses unused bits in DRAM$ rows

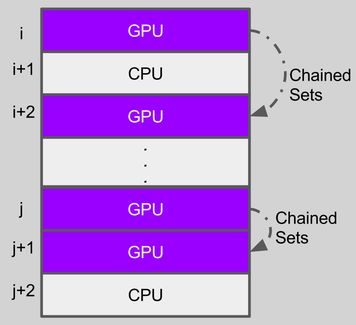


Figure: HAShCache Row Organization and Access Path of a request

Results

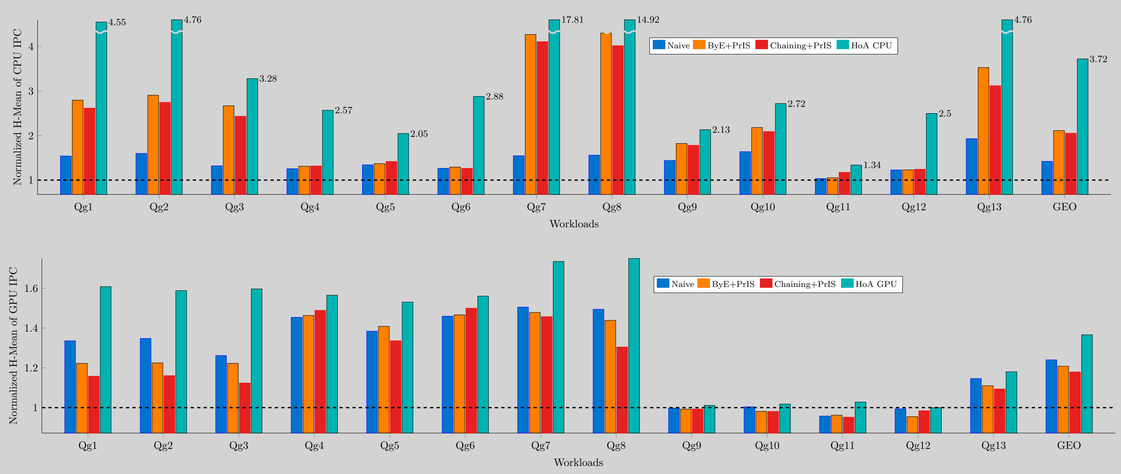


Figure: Speedup obtained by HAShCache mechanisms for (a)CPU (b)GPU

Conclusion

- HAShCache - Heterogeneity aware organization - improves IHS performance
  - achieves better resource utilization - reduces energy consumed
- Compared to a heterogeneity unaware DRAM$ (naive)
  - *Chaining* + *PrIS* improves perf of CPU by 44% by trading off just 6% of GPU perf
  - *ByE* + *PrIS* improves perf of CPU by 48% while sacrificing just 3% of GPU perf
- Overall, HAShCache improves system performance by
  - 41% over a naive DRAM$
  - 211% over the baseline system with no DRAM$

*This work has been submitted to the 50th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-50) and is currently under review

The authors can be contacted at adarsh.patil@csa.iisc.ernet.in / govind@csa.iisc.ernet.in