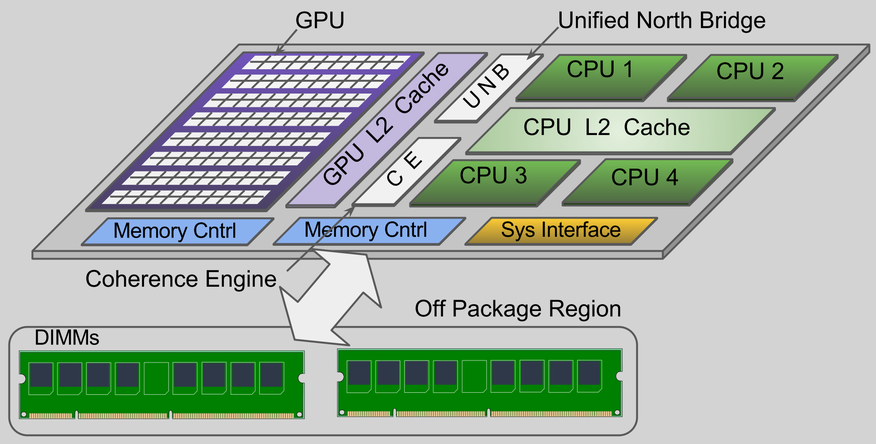
HAShCache: Heterogeneity Aware Shared DRAMCache For Integrated Heterogenous Architectures

Adarsh Patil, R Govindarajan

Department of CSA, Indian Institute of Science, Bangalore

Integrated Heterogenous Systems (IHS) Architecture

- *Throughput-oriented* GPGPU SMs placed alongside *Latency-oriented* CPU cores on-chip
- Shared Physical/Virtual Address Space and a Unified Memory Hierarchy
- Cache Coherent Interconnect
- Improved Programmability
  - i Avoids explicit management of data (memcpy, cudamalloc etc.)
  - ii Allows pointer sharing
  - iii GPU application data set sizes need not be constrained by memory size
  - iv Enables HLLs to exploit fine grained parallelism synergistically with CPUs
- AMD APUs, Intel Iris, NVIDIA Denver

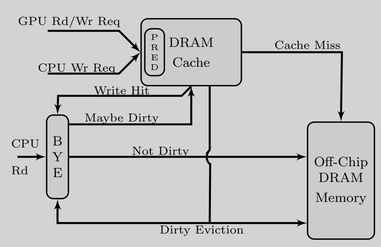


① Hetereogenity Aware DRAM$ Scheduling: PrIS

- *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)
- Fig shows working of HAShCache + ByE



③ Spatial Occupany 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 in same row
- Chaining done until minimum CPU occupancy threshold is reached
- Overhead: NIL, uses unused bits in DRAM$ rows

- CPU fill requests always inserted in original set
- GPU fill request inserted as per table below

Threshold Reached?	Original Set	Not Chained	Chained Set	
			CPU	GPU
No	CPU	replace original	replace chained	replace original
	GPU	Chain to nearest CPU set	replace chained	replace original
Yes	CPU	Chain to nearest GPU Set	do not insert	replace chained
	GPU	replace original	replace original	replace original

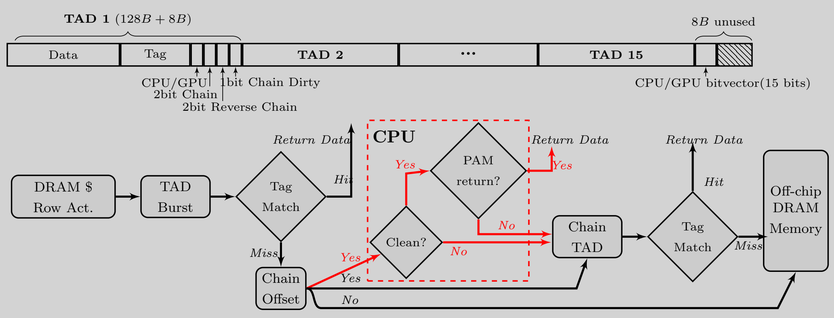


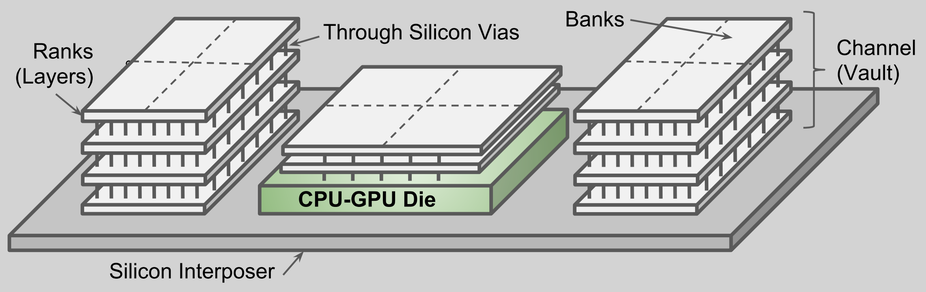
Figure : HAShCache Row Organization and Access Path of a request

Conclusion

- HAShCache is heterogeneity aware and improves system throughput and achieves better resource utilization
- Compared to a carefully designed by 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 IHS performance by 41% over a naive DRAM$ and by 211% over the baseline system with no stacked DRAM

Vertically Stacked DRAM

- DRAM Layers stacked using 2.5D interposer or 3D TSV
- Large capacity ~order of GBs
  - can help contain the working sets of IHS workloads
- Very High Bandwidth ~500 GB/s (90GB/s DDR4)
  - can assist throughput-oriented GPU
- 20 % reduction in access latency
  - can enhance CPU performance
- Samsung Wide IO, AMD/Hynix HBM, Intel/Micron HMC



Motivation

Performance

- When running homogeneously CPU CPU and GPU both gain from addition of DRAM$
- Effect of Co-running
  - CPU performance severely degraded, drops 320% from homogeneous CPU
  - GPU remain relatively unaffected, drops 7% from homogeneous GPU
- Naive addition of DRAM$ over IHS
  - CPU recovers only 42% of the lost performance over homogeneous CPU
  - GPU still performs 15% better than homogeneous GPU

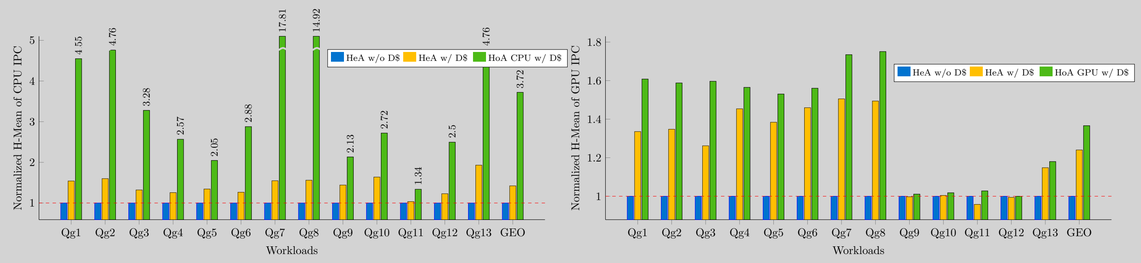


Figure : Performance comparison of CPU & GPU in IHS with D$ vs Homogeneous with D$

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

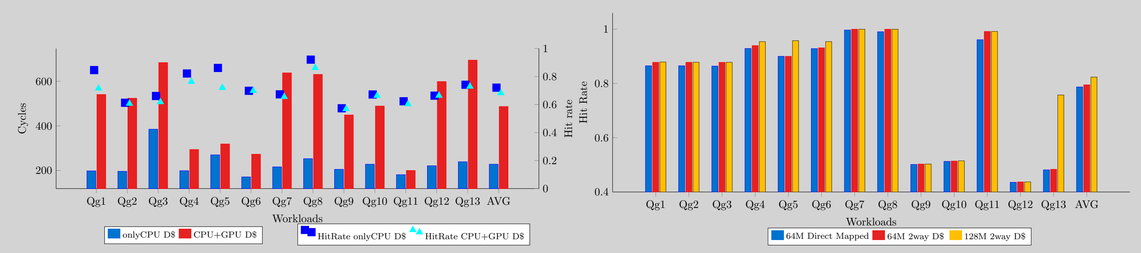


Figure : (a)CPU D$ Access Latency and Hit Rates (b)GPU Hit rate with 2-way assoc cache

HAShCache Design for IHS Processors

- Large Metadata overhead
  - Tags-in-DRAM, 128 Byte Tag-and-Data (TAD) Units
- Set Associativity
  - Direct Mapped - avoids added latency due to multiple tag lookups
- Miss Penalty
  - Initiate early access to memory for CPU using a miss predictor
- RBH vs BLP addressing scheme
  - Despite higher BLP, DRAM$ performs better using RBH addressing
- Effect of Interference
  - DRAM$ must be optimized to regain CPU performance lost due to interference

Results

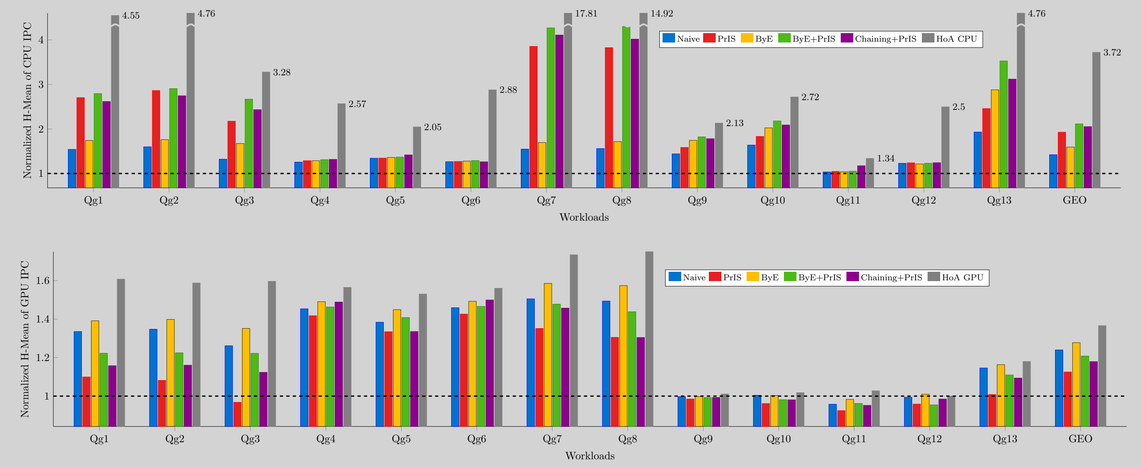


Figure : Speedup obtained by HAShCache mechanisms for (a)CPU (b)GPU