Heterogeneous Cache Hierarchy Management for Integrated CPU-GPU Architecture

Hao Wen*, Wei Zhang

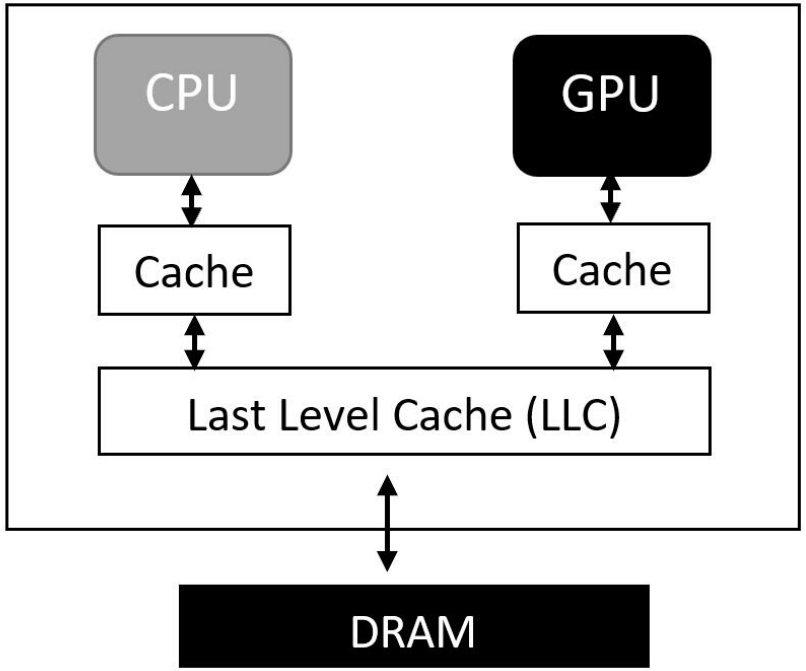
Department of Computer Engineering and Computer Science

University of Louisville

* Graduated from ECE, VCU

Introduction

- Current heterogeneous CPU-GPU architectures integrate general-purpose CPUs and highly thread-level parallelized GPUs in the same die and share the same DRAM, last-level cache (LLC), as well as the interconnection network.



Introduction

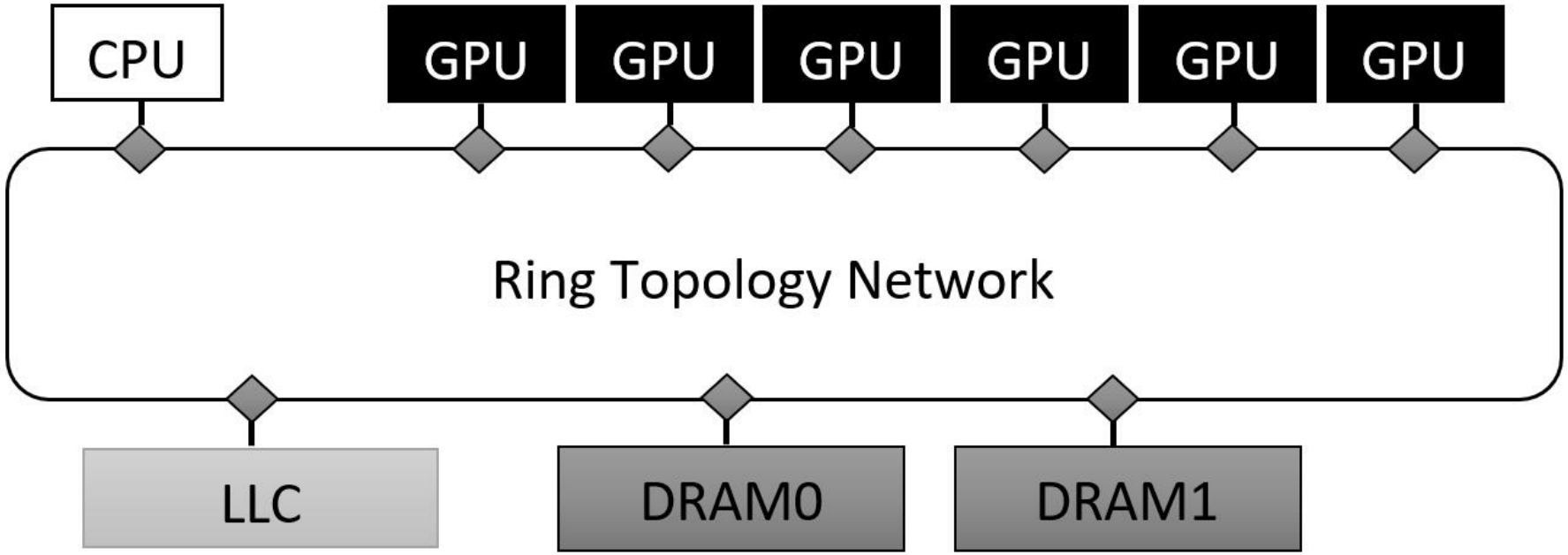
- For the two-level cache hierarchy, it can be designed to be inclusive or exclusive. The inclusion property dictates the contents of a lower level cache be a subset of the contents of a higher level cache, while the exclusive cache hierarchy may achieve a larger effective cache size since the contents

Prior Work

- Prior works focus on exploiting inclusive/exclusive cache properties to evaluate/improve performance on CPU cache hierarchy
- however, how the inclusive/exclusive properties affect the GPU performance and interact with the CPU in the LLC for the CPU-GPU heterogeneous architecture remains largely unknown

Background

- The integrated CPU-GPU heterogeneous architecture evaluated in this work

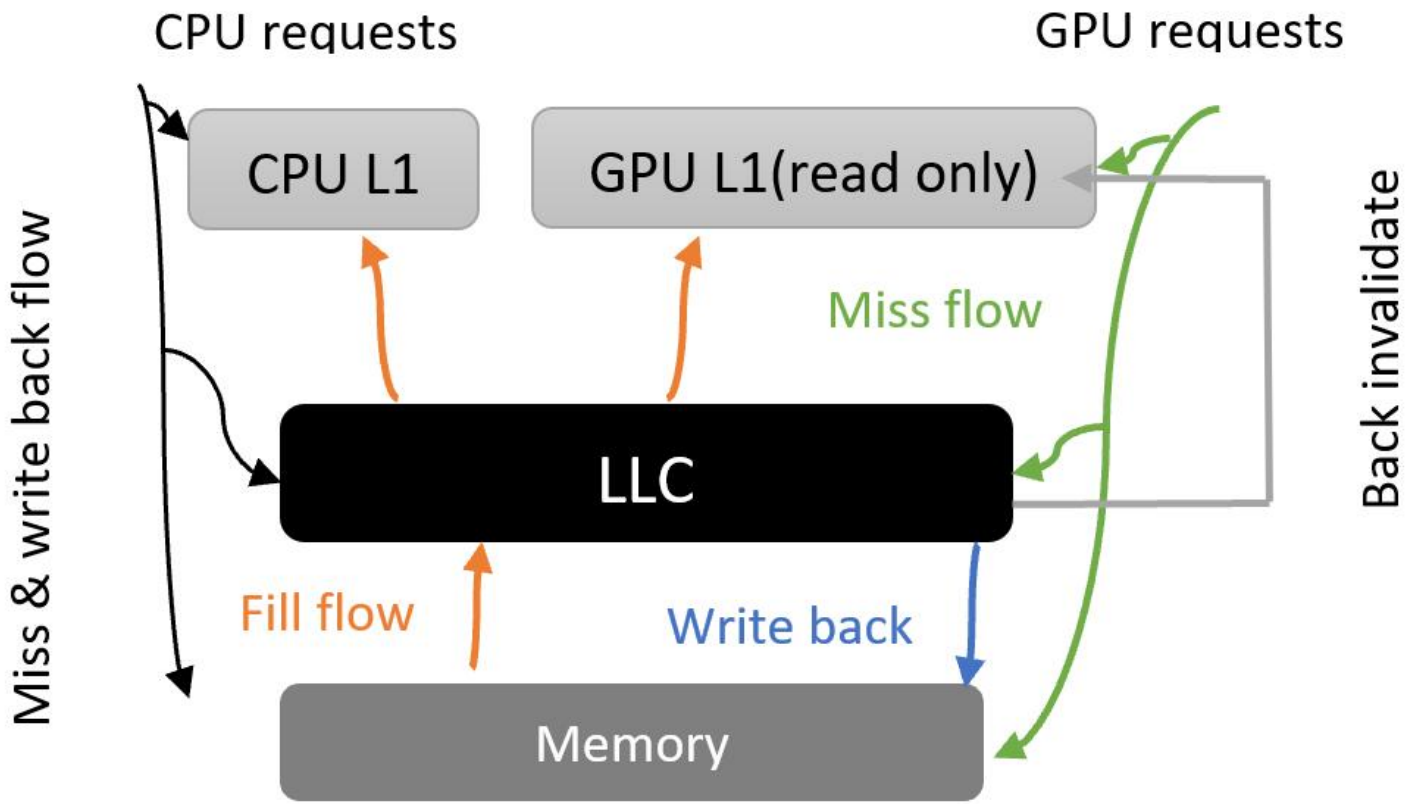


Background

- we propose and evaluate three different cache hierarchy management policies
  - Selective GPU LLC fill
  - Selective GPU L1 write back
  - A hybrid policy combining the first two, and selectively replacing CPU blocks in the LLC

Background

- Base line cache access flow



Selective GPU LLC fill

- Since GPU applications have much more memory requests than CPU and tend to monopolize the resources in the LLC, GPU applications tend to have more power to evict CPU cache lines.

#	Benchmark (CPU:GPU)	CPU	CPU-GPU	GPU	GPU-CPU
1	fmm:pathfinder	18.94%	73.29%	91.71%	92.41%
2	milc : b+tree	91.11%	99.68%	50.37%	50.76%
3	bzip2 : bfs	59.04%	65.32%	8.3%	10.21%
4	perlbench : backprop	14.59%	31.31%	17.31%	19.94%

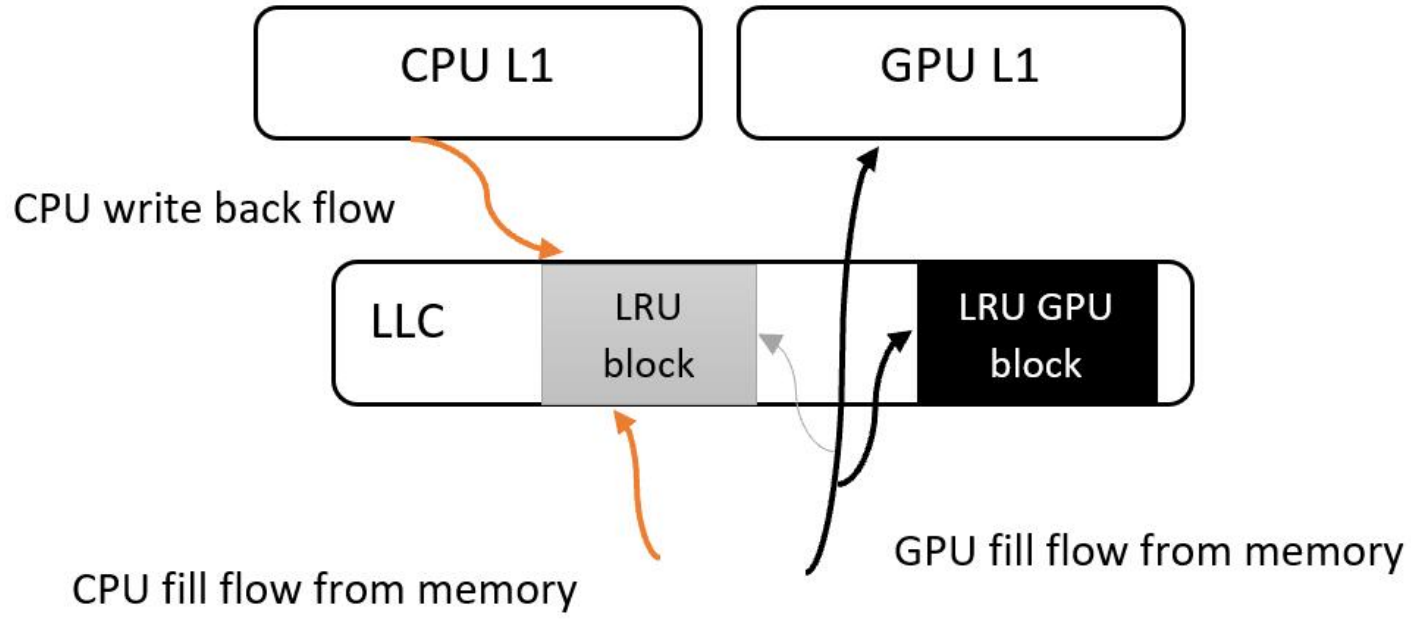
Selective GPU LLC fill

- To reduce the CPU-evicted-by-GPU conflict misses in the LLC, we selectively fill the LLC for GPU requests from the memory.

- For the GPU fill flow from memory, if the LRU block is a CPU block, instead of replacing the LRU CPU block, the GPU fill block will bypass the LLC and be directly filled to GPU L1 cache

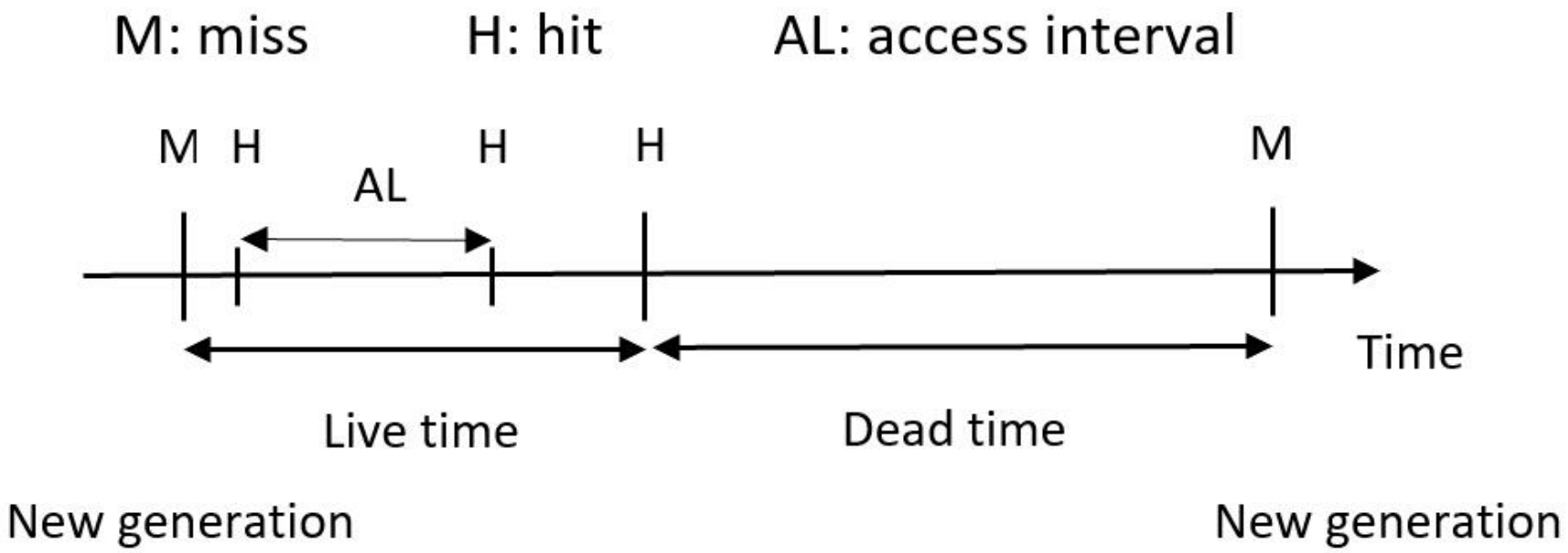
Selective GPU LLC fill

- If the LRU block is a GPU block, it works as a normal GPU fill in the LLC. This policy will give the CPU higher priority in the LLC utilization



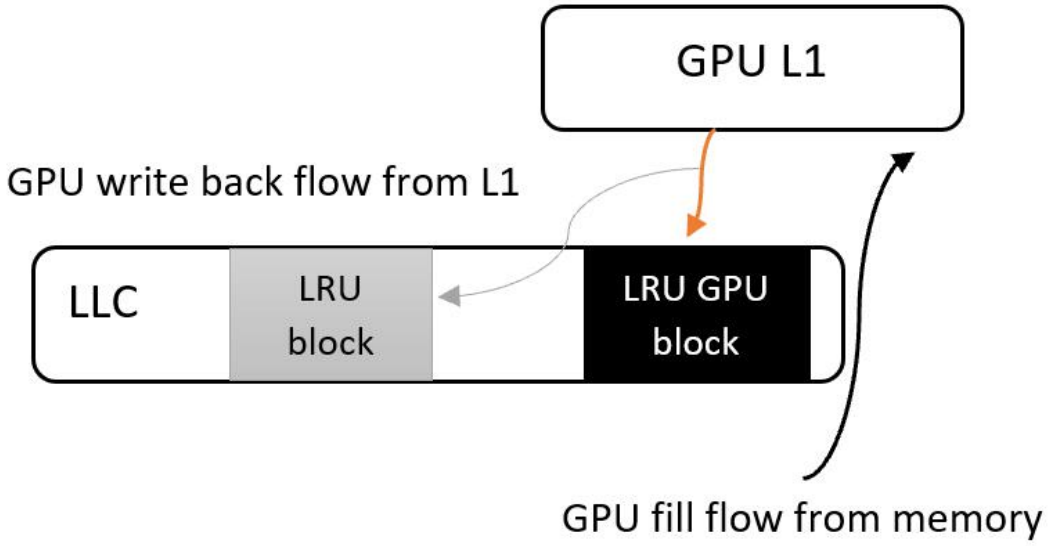
Selective GPU L1 Write Back

- Cache generation



Selective GPU L1 Write Back

- if the GPU block is not found in both L1 and L2, instead of fetching from the main memory and placing in both L1 and L2, the missing block will only be filled to L1.
- When blocks in the GPU L1 cache are replaced, they will be selectively written back to the LLC.



Selective GPU L1 Write Back

- Evicted blocks from the GPU L1 cache will be written back to LLC if they are predicted to be reused in the near future
- If the written back blocks cause eviction in the LLC, we give CPU blocks in the LLC higher priority and always replace the LRU GPU block, unless there are no GPU blocks can be found in a cache set.

Selective GPU L1 Write Back

- How to predict?
  - ➢In order to predict whether the evicted blocks in the GPU L1 cache will be reused or not in the near future, we take advantage of the access interval during the live time of a cache generation.
  - ➢ The idea is to predict the cache line is still alive if the distance between the last hit and the time being replaced is smaller than a pre-determined value

Selective GPU L1 Write Back

- Implementation
  - Every cache line is associated with a counter
  - Every time the cache line is accessed or brought to the cache for the first time, the counter is reset to zero.
  - The counter is incremented at fixed time intervals. If there is no access to the cache line and the counter reaches the pre-set value, it is predicted that this cache line is dead

Selectively Replacing CPU Blocks In the LLC

- For the selective GPU L1 write back, the written back GPU blocks in the LLC may not be highly reused (the prediction is not 100% accurate).
- In order to improve the LLC performance for GPU, the GPU blocks in the LLC not only get filled through the GPU L1 write backs, but also can be filled through the GPU fill flow from memory.
- Therefore, the selective GPU LLC fill and selective GPU L1 write back can be combined together.

Selectively Replacing CPU Blocks In the LLC

- Currently, no matter selective GPU LLC fill or selective GPUL1writeback,CPUblocksarenotallowedtobereplaced by GPU requests (both fill requests from memory and write backs from GPU L1).

- Due to the higher priority of CPU blocks in the LLC, the GPU LLC miss rate will increase, leading to possible GPU performance overhead.

Selectively Replacing CPU Blocks In the LLC

- The high CPU LLC priority rule may be relaxed to reduce GPU performance overhead while maintaining the maximum CPU performance improvement.

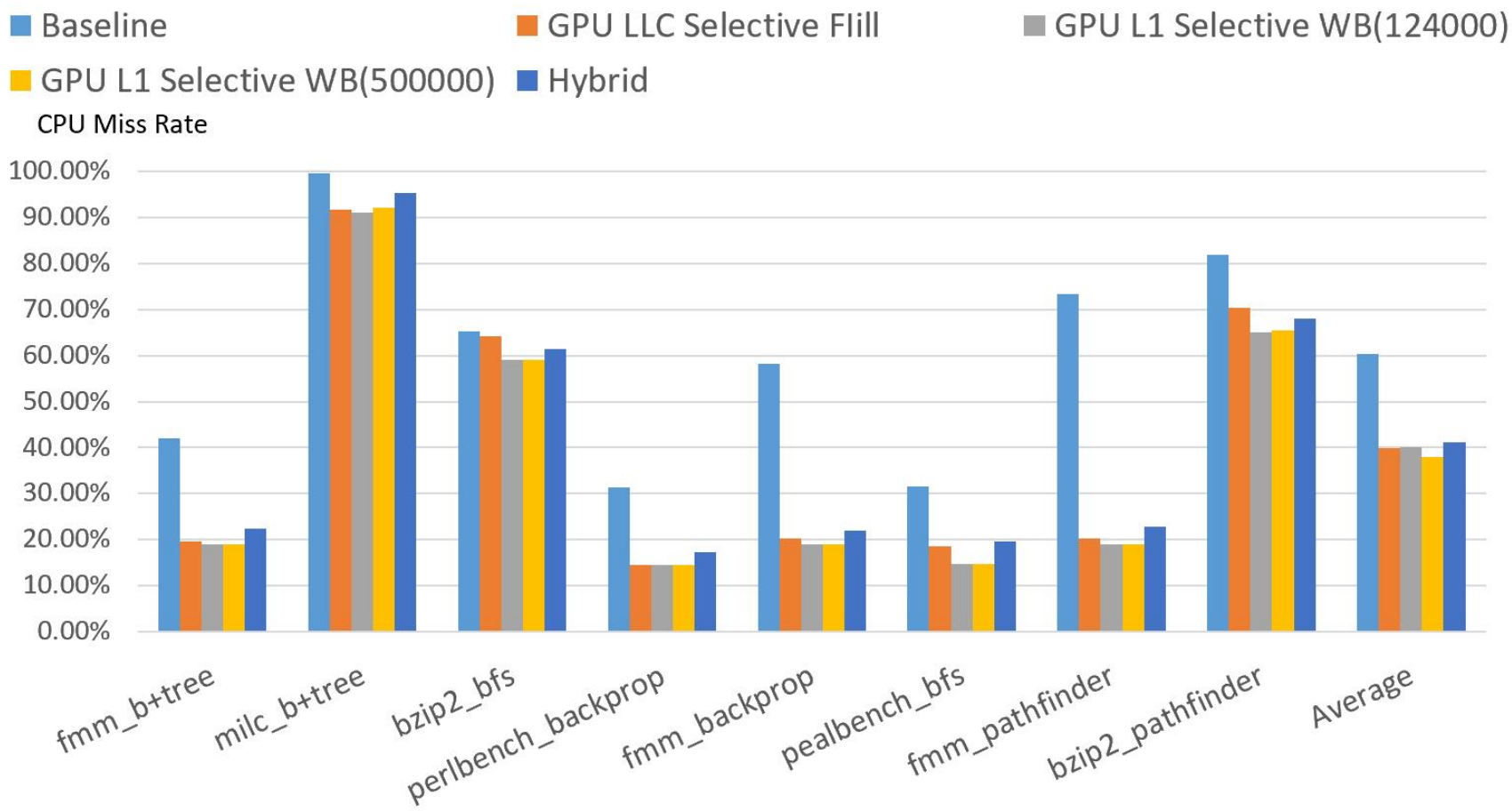
- The idea is still using the concept of the cache generation. When a CPU block in the LLC is about to be replaced by a GPU request, it is allowed if this CPU block is predicted to be dead. Again, the counter-based predicting method is used.

Experimental Environment

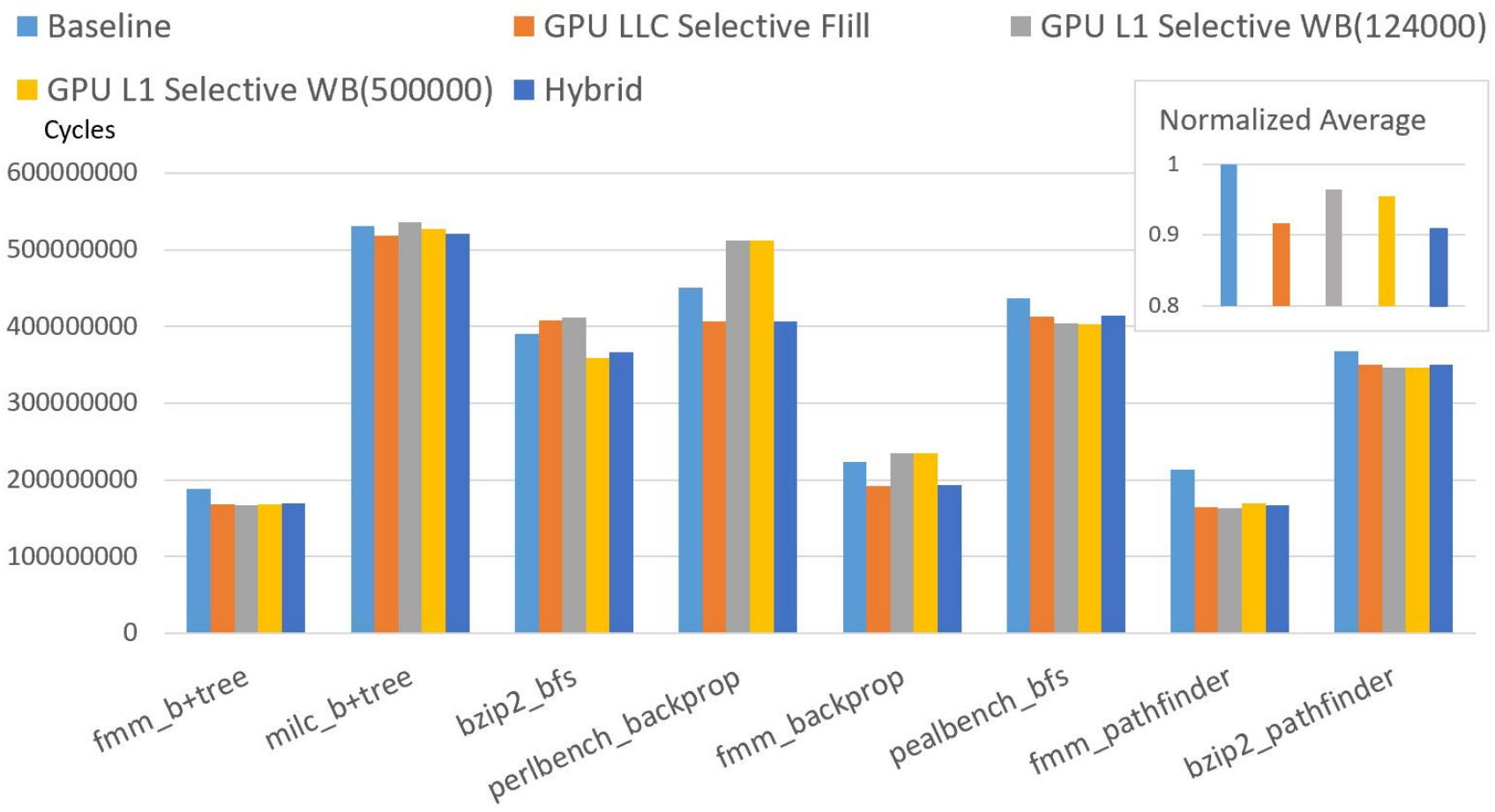
- We use MacSim , a CPU-GPU heterogeneous simulation framework to evaluate our work.

CPU	1 core, 3Ghz
GPU	6 cores, 1.5Ghz
CPU L1 icache per core	32KB, assoc 8
CPU L1 dcache per core	32KB, assoc 8
GPU L1 icache per core	4KB, assoc 8
GPU L1 dcache per core	32KB, assoc 8
LLC	2MB, assoc 16, 1Ghz
Network	ring topology, 1Ghz
DRAM Controller	2 channels, 1.6Ghz
DRAM scheduling policy	FRFCFS

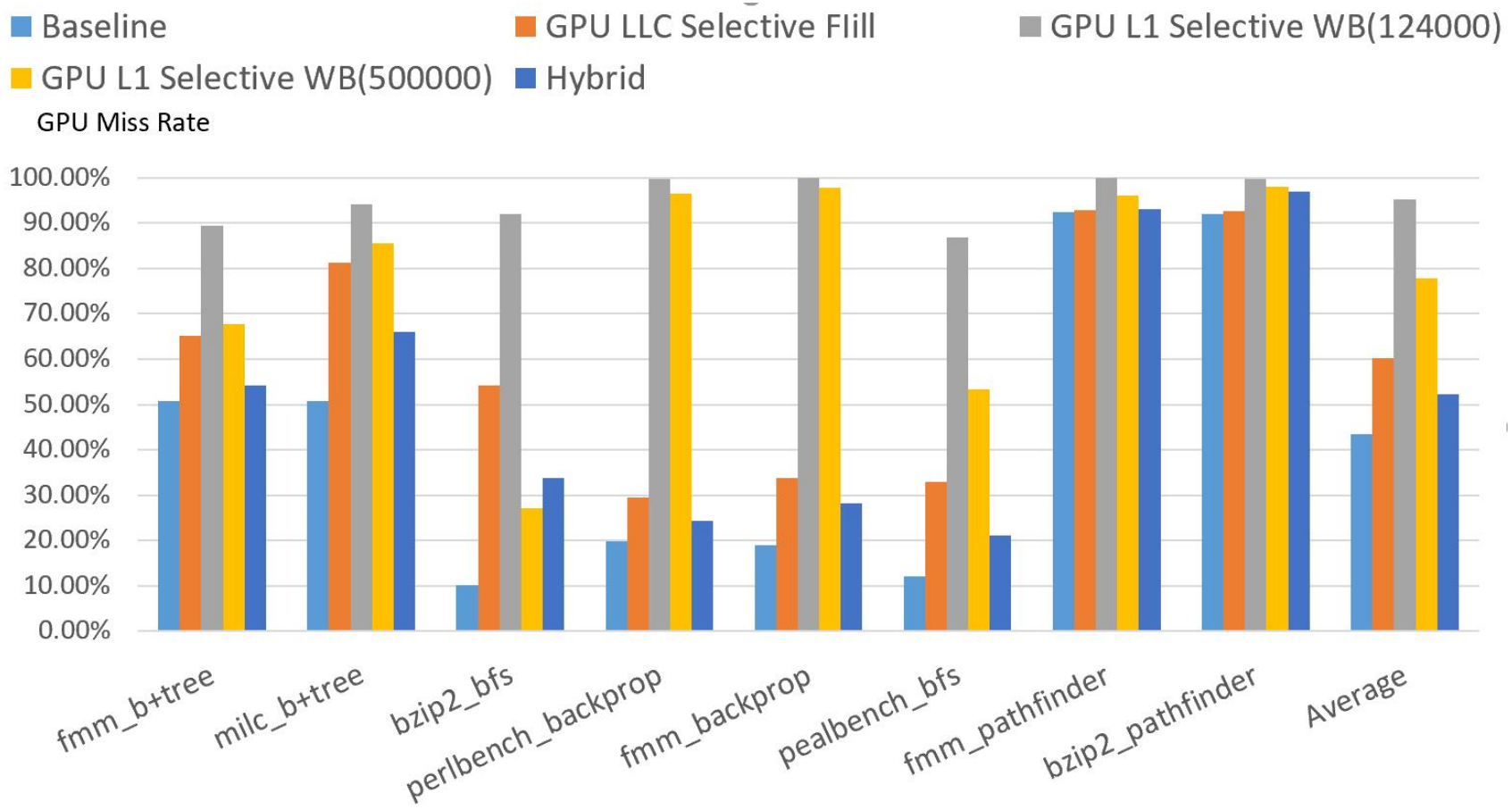
CPU LLC miss rate for different policies



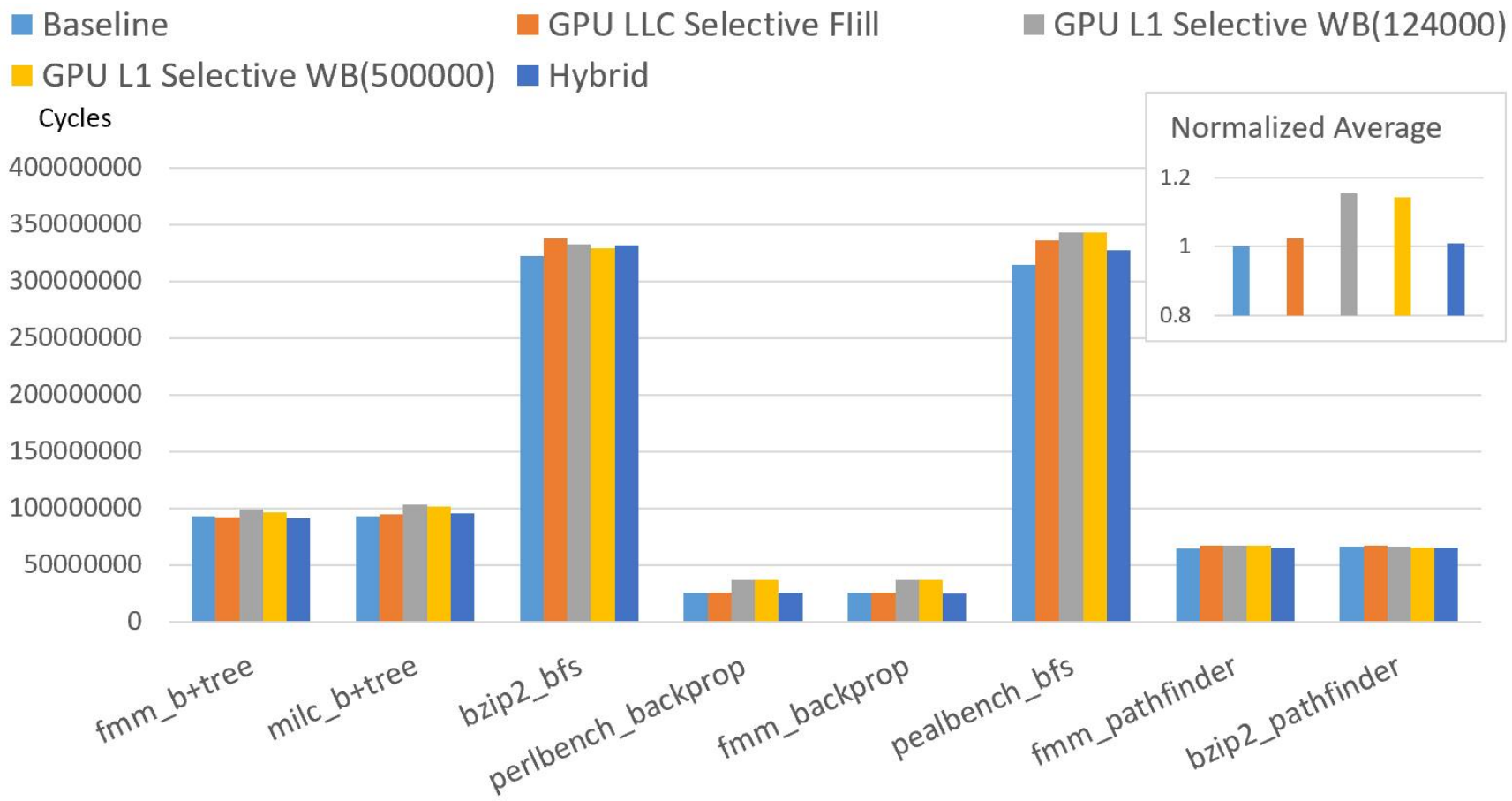
CPU performance for different policies



GPU LLC miss rate for different policies



CPU performance for different policies



Thank you!