CAMELab.org
Computer Architecture and Memory Systems Lab.

UT DALLAS

Adaptive Cache Management for Energy-Efficient GPU Computing

Xuhao Chen

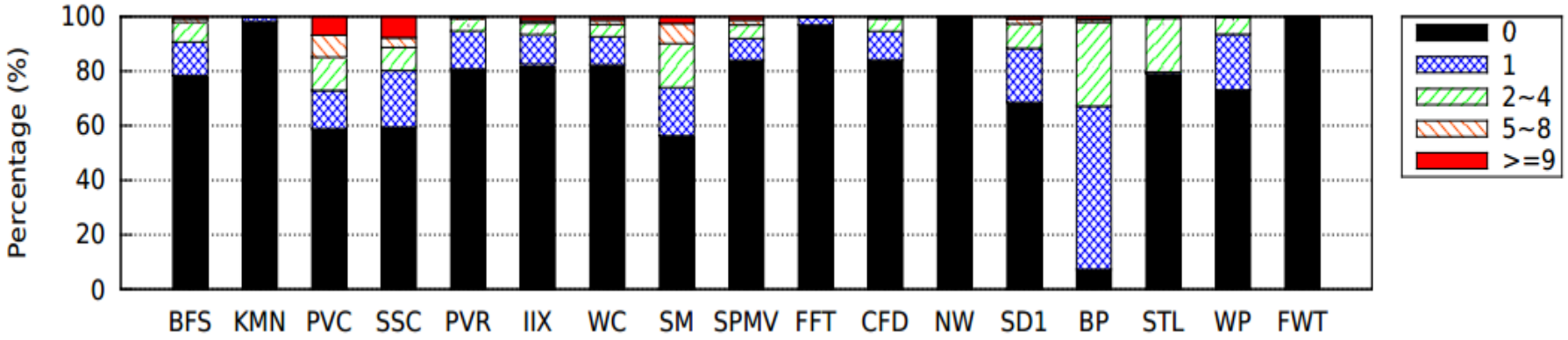
MICRO 47

Overview

CAMELab.org
Computer Architecture and Memory Systems Lab.
UT DALLAS

- Challenges: Throughput-oriented execution model, GPGPU, generates massive amount of memory requests which cause cache contention and resource congestion.
- Observation:
  - ***Cache contention***: The working sets of the massive threads in GPGPU is much larger than L1D cache size. This causes severe cache threshing.
  - ***Resource congestion:*** Cache efficiency doesn't directly correlate with system performance in GPUs. The GPU resource, i.e. NOC and DRAM bandwidth can be bottleneck to the performance.
- Target: Protect cache hierarchy from contention and reduce resource congestion through adaptive cache management.
- Approach: CBWT(coordinated bypassing and warp throttling)
- Performance: harmonic mean IPC improvement of 74% and 17% over baseline GPU architecture and optimal static warp throttling.

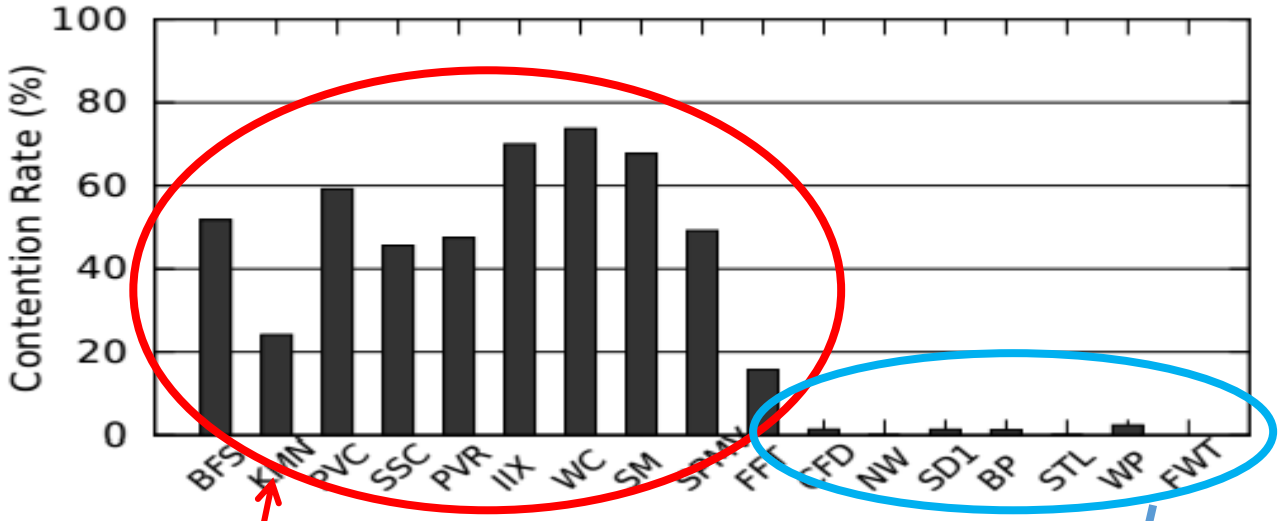
Observation 1



L1 cache reuse count distribution. It shows the number of repeated accesses to cache blocks in L1, after block fill in.

- Two potential explanations to this figure:
- 1. the lifetime of cache block is very short in GPGPU. It means this cache line only be used for once and never be touched again.
- 2. Because of massive memory requests from threads, working set size of threads is much larger than L1 cache size. Thus cache blocks are early evicted before the end of their lifetime.
- This paper consider reason 2 is the dominant one and focus on solving it.

Observation 2



The L1 cache contention ratio detected by the L2 cache. A repeated memory request from L1 cache to L2 cache is regarded as contention. The contention ratio is calculated by # of contentions and total number of requests to L2 cache.

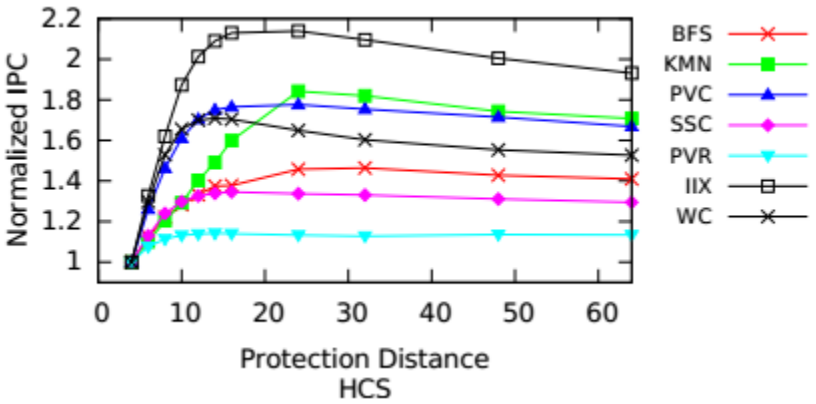
- Red circle shows, a lot of memory request come from L1 is because of cache threshing.
- This supports reason 2.

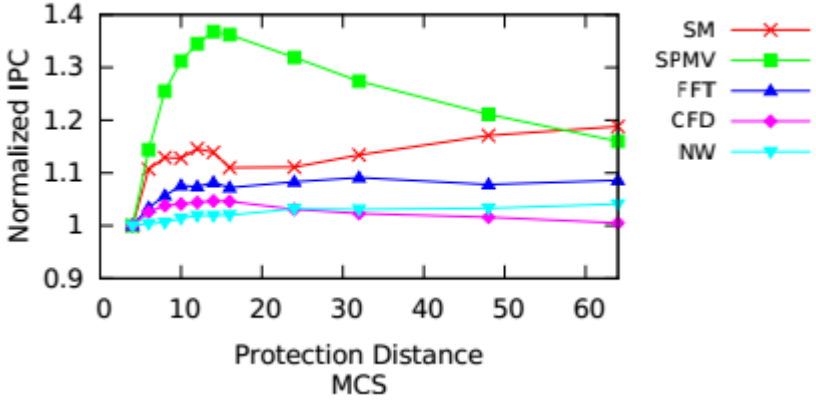
Two reasons:

- The cache lock is dead after first access. So there are not too much contention.
- L2 cache also experience cache threshing, which increase memory access times of L2 cache.

Cache Bypassing

- Purpose: enlarge the throughput while reducing cache contention.
- Approach: Protection Distance Prediction (PDP).
- Details:
  - Each line has a remaining prediction distance value (RPD).
  - Once filling, a cache line is set to RPD.
  - Each memory access to this line will decrement RPD by 1.
  - If RPD>0, this cache line cannot be evicted. Otherwise, it can be replaced by new data.
  - If RPD of all cache lines > 0, new data will be bypassed.
  - Reuse distance(RD) is predicted by hardware sampling.



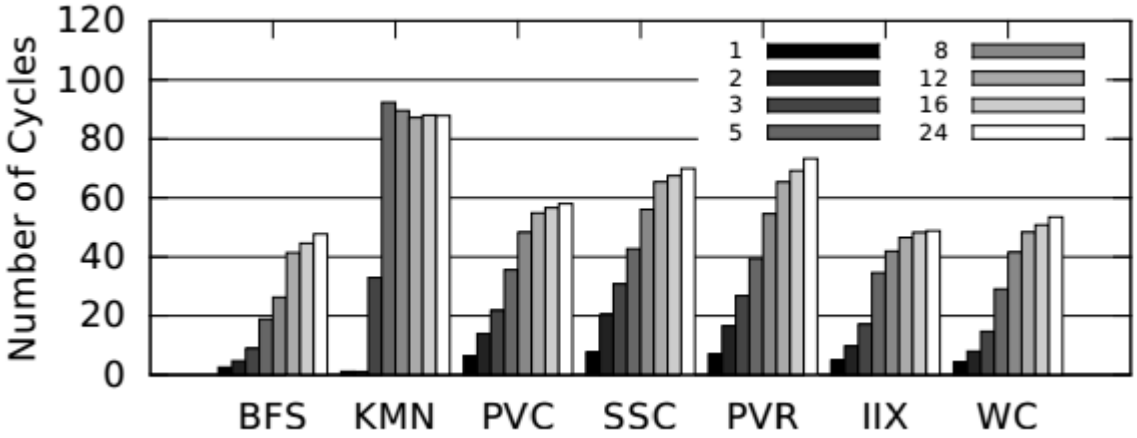


Contention Detection

- Purpose: To dynamically change the threshold of protection distance values.
- Approach: periodically detect the number of early evictions, called “lost locality score”(LLS).
- Details:
  - Add Victim Bits to L2 Cache tag array entries.
  - Repeated memory access will increment Victim Bits by 1.
  - Periodically detect the number of repeated memory access.
  - Decide threshold based on the lost locality score.

Figure 9. Hardware Extensions on the L2 Cache

Observation 3

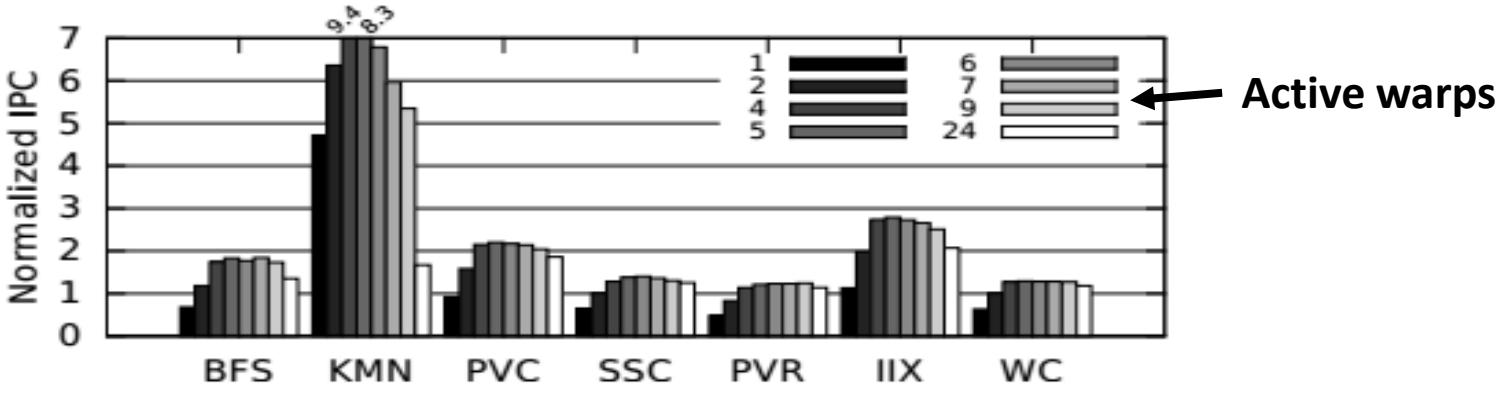


Average NOC latency changes as maximum active warps in a SM increases from 1 to 24.

- Two Hints from this figure:
- 1. Simply increasing the number of active warps cannot directly increase the performance. Because the mass traffic in NOC will degrade the performance.
- 2. Using cache bypass also may result in performance degradation because of resource congestion.

Warp Throttling

- Purpose: throttling active warp number to reduce traffic congestion of NOC.
- Problem: static warp throttling is difficult to get optimized.



- Solution: Adaptive warp throttling based on congestion detection.
- Details:
  - Profiling the behavior of NOC and get the threshold of packet latency.
  - Sampling packet randomly and detect the transfer latency.

Coordinated Bypassing and Warp Throttling(CBWT)