Perceptron Learning for Coherence-Aware Reuse Prediction

(Running Multiple Benchmarks at a time)

Advisor: Dr. Eun Jung (EJ) Kim
Presented by: Snehil Verma

TEXAS A&M UNIVERSITY

Overview

- Motivation
- Introduction
- Related Works
- Coherence-Aware Reuse Prediction (main idea)
- Methodology
- Results
- Conclusion

Motivation

Though the **LRU replacement policy** has been a de-facto policy for a long time, it is not robust for recency-unfriendly cache access patterns.

It is important to detect the cache blocks with high reuse to avoid unnecessary evictions and misses.

Introduction

- **Inclusive cache hierarchy** simplifies the coherence protocol.
- **Inclusion property** - The data cached in higher level caches should be a subset of lower level caches.
- Since the inclusive last level caches (LLCs) are often **unaware of the temporal locality** of the higher level caches, blocks with high temporal locality in higher level may be consequently evicted from the cache hierarchy.
- This may **limit the performance** for inclusive caches.
- Hence, it is essential to **keep the important blocks** (most reused) in the cache hierarchy.

Related Works

- **Reuse Prediction**: Sampling Dead Block Prediction (SDBP) & Perceptron.
- **Inclusive Cache Management**: Temporal-Based Multilevel Correlating (TMC) & Temporal Locality Aware (TLA).
- **Sharing Awareness Cache Management**

Coherence-Aware Reuse Prediction

- **Sharing Awareness Cache Management**
- Using **Perceptron** reuse prediction to learn the correlation among the features and reuse, and use it to guide replacement policy.

Five alternatives

- **Bias**: In Bias, we use the number of sharers of the requested line and use it as bias for the perceptron prediction.
- **NumSharers and NumSharersHash**: In these alternatives, we query the number of sharers and use it as another feature to index its weight table.
- **OneHot and OneHotHash**: OneHot represents the one hot encoding of the sharers of a cache block.

Simulation Configuration

- Execution-driven simulator Zsim.
- L2 is configured as inclusive due to the limits of ZSim

TABLE I
SYSTEM CONFIGURATION

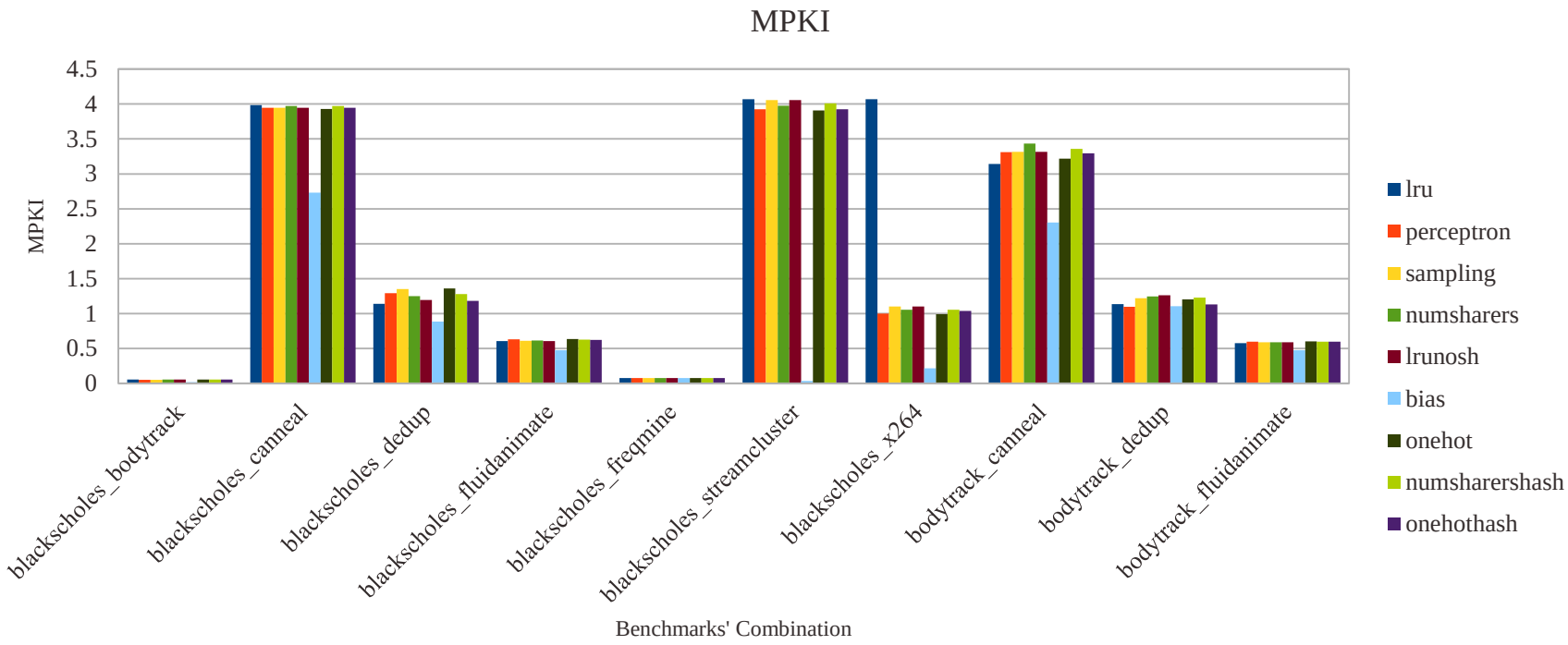
Cores	Westmere-like OOO at 2.4 GHz, 8 cores (MT)
L1 caches	32 KB, 4-way set-assoc, split D/I, 3 cycles
L2 caches	Private, 256 KB, 8-way set-assoc, inclusive, 7 cycles
L3 cache	Shared, 4, 8, 16 MB, inclusive, 8-way set-assoc, 27 cycles
Coherence	MESI, 64B lines
Memory	DDR3-1333 MHz, 4 ranks/channel, 4 channels

Workloads and Replacement Policies

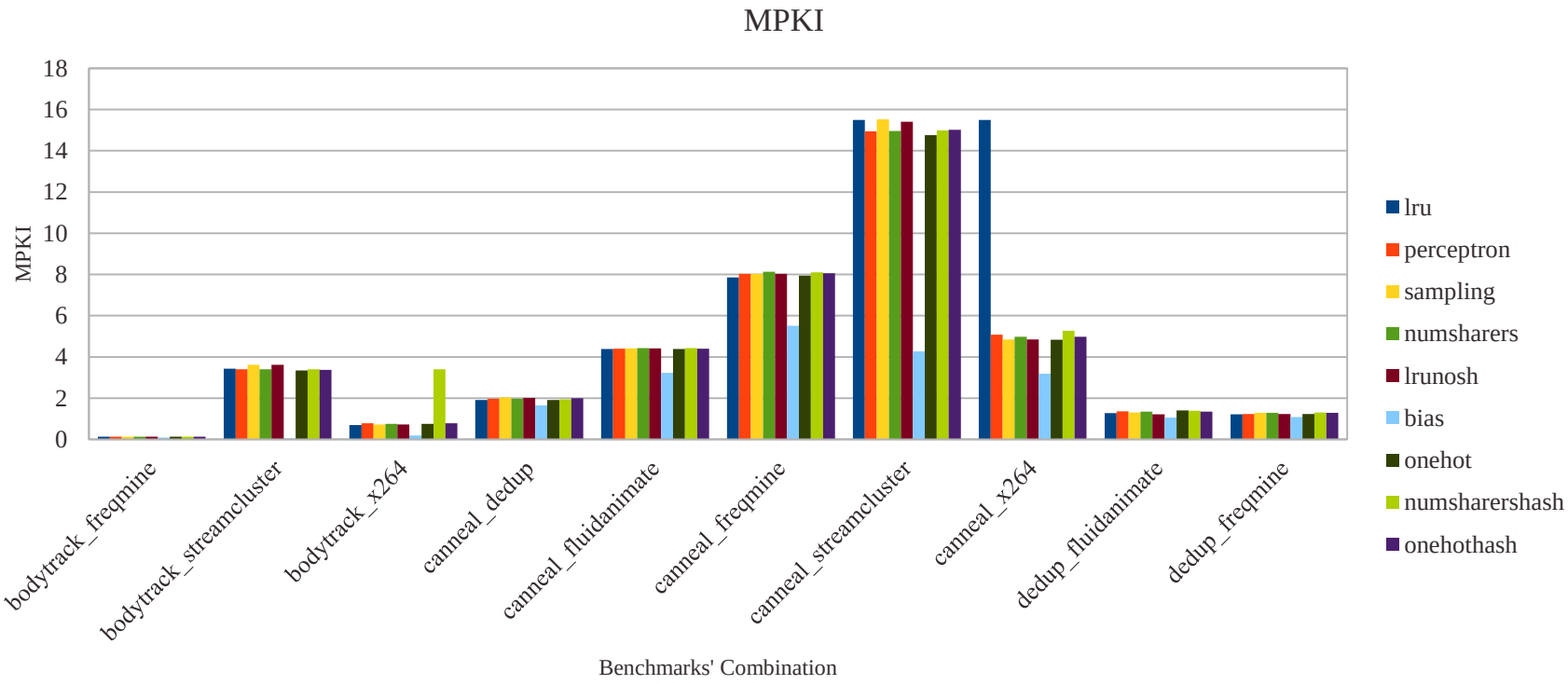
- 8 multi-threaded applications and kernels from the PARSEC benchmark suite.
- Excluded *ferret*, *raytrace* and *vips* due to compilation issues and *facesim* due to long simulation time.
- Run in a combination of two with 4 threads each, using large input data set.

- SDBP and Perceptron implemented using the same parameters from the paper.
- Variations in Perceptron: Bias, NumSharers, NumSharersHash, OneHot and OneHotHash.
- The baseline replacement policy is LRU (Sharers Aware).
- LRU (no sharers aware) replacement policy is also evaluated.

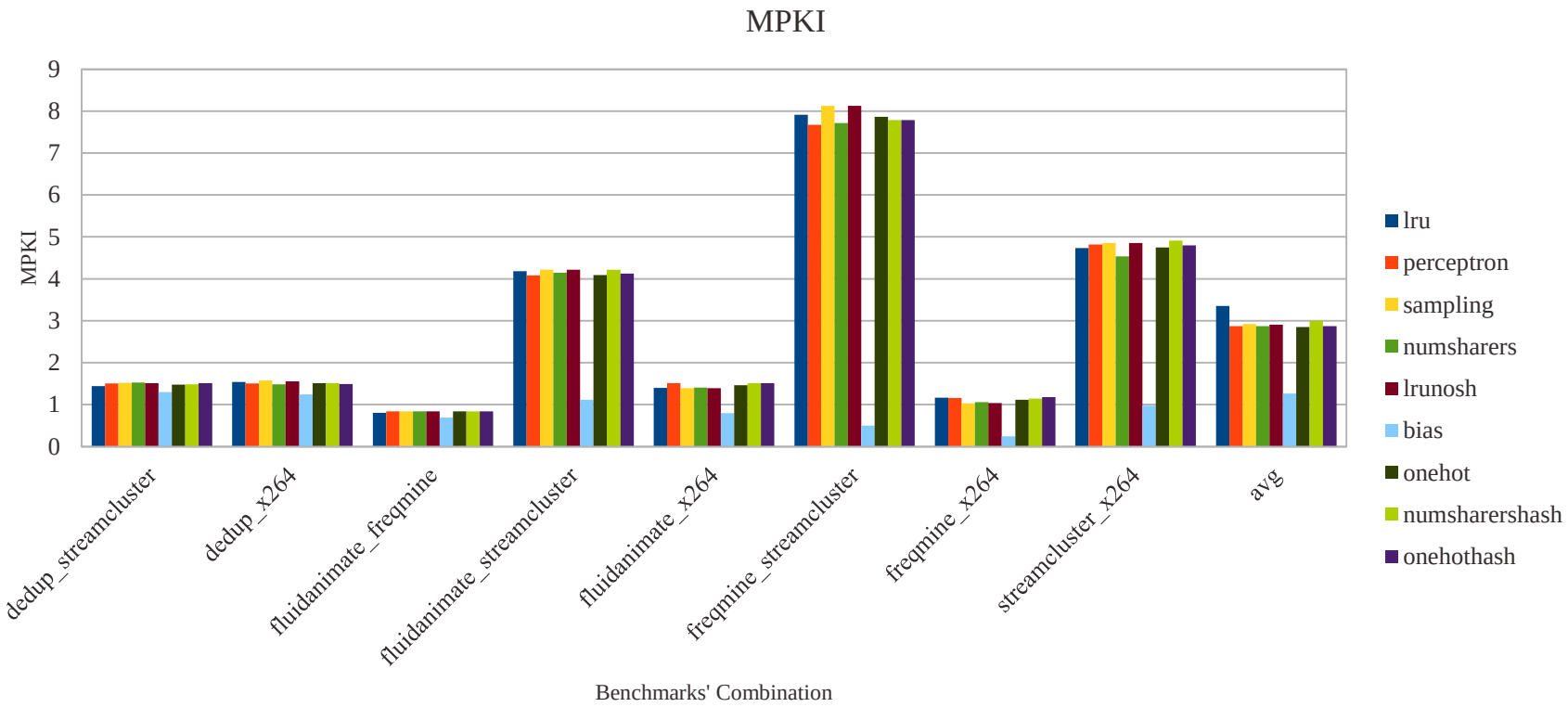
Results (4MB LLC) (MPKI)



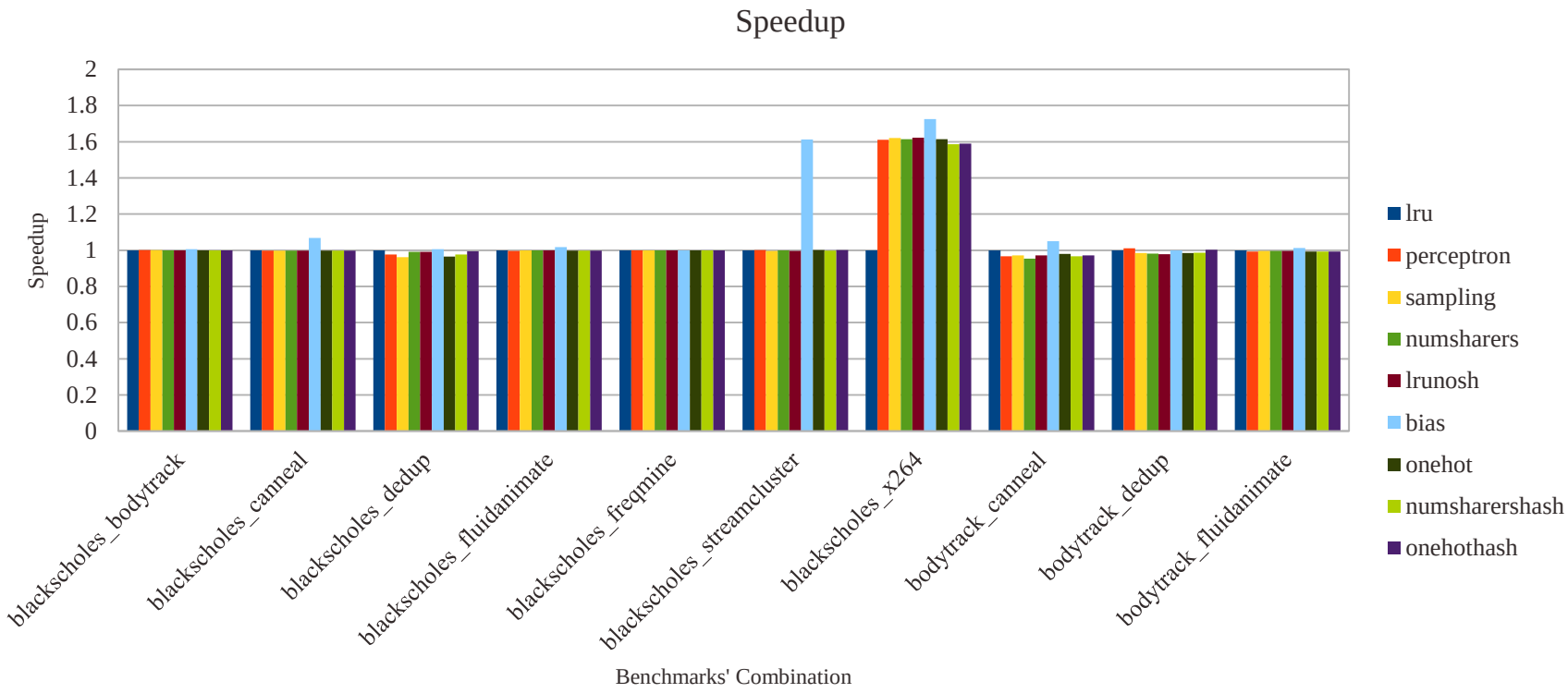
Results (4MB LLC) (MPKI)



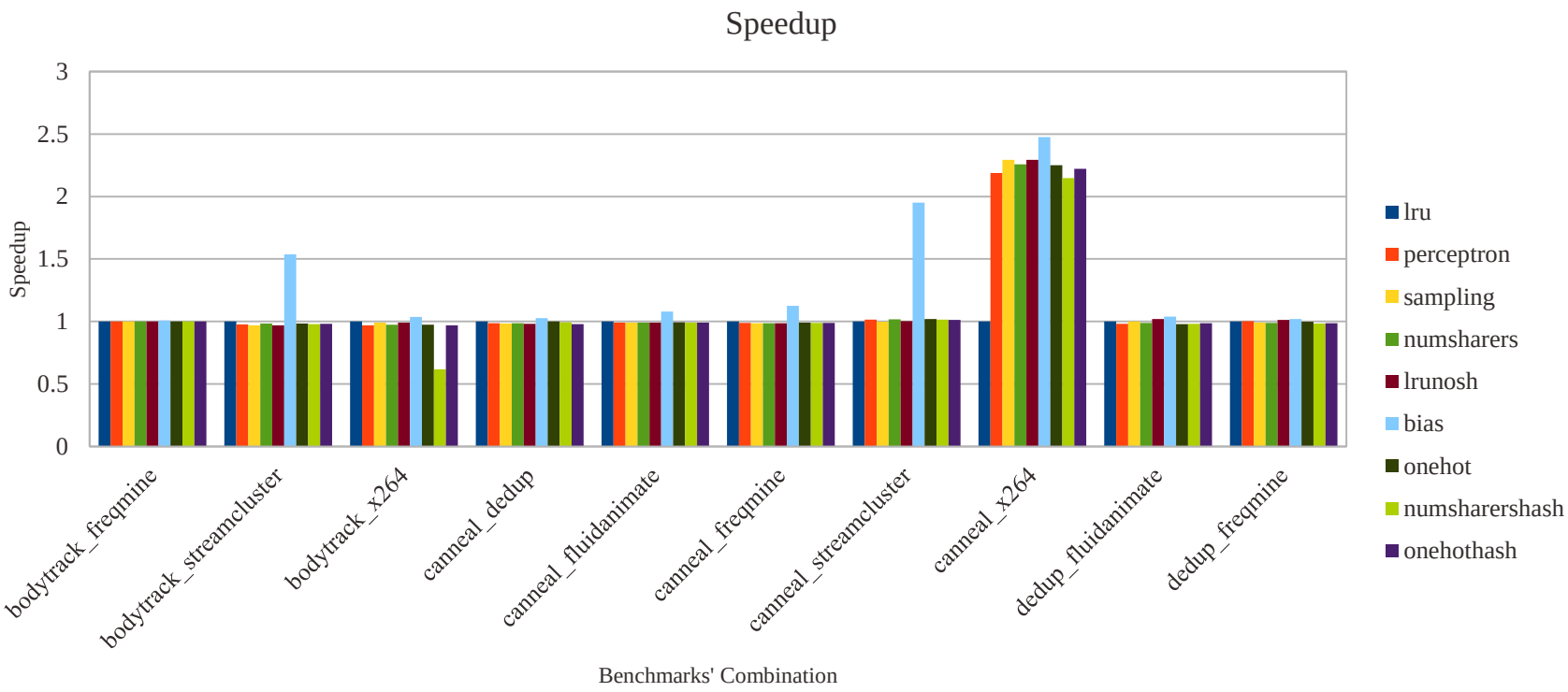
Results (4MB LLC) (MPKI)



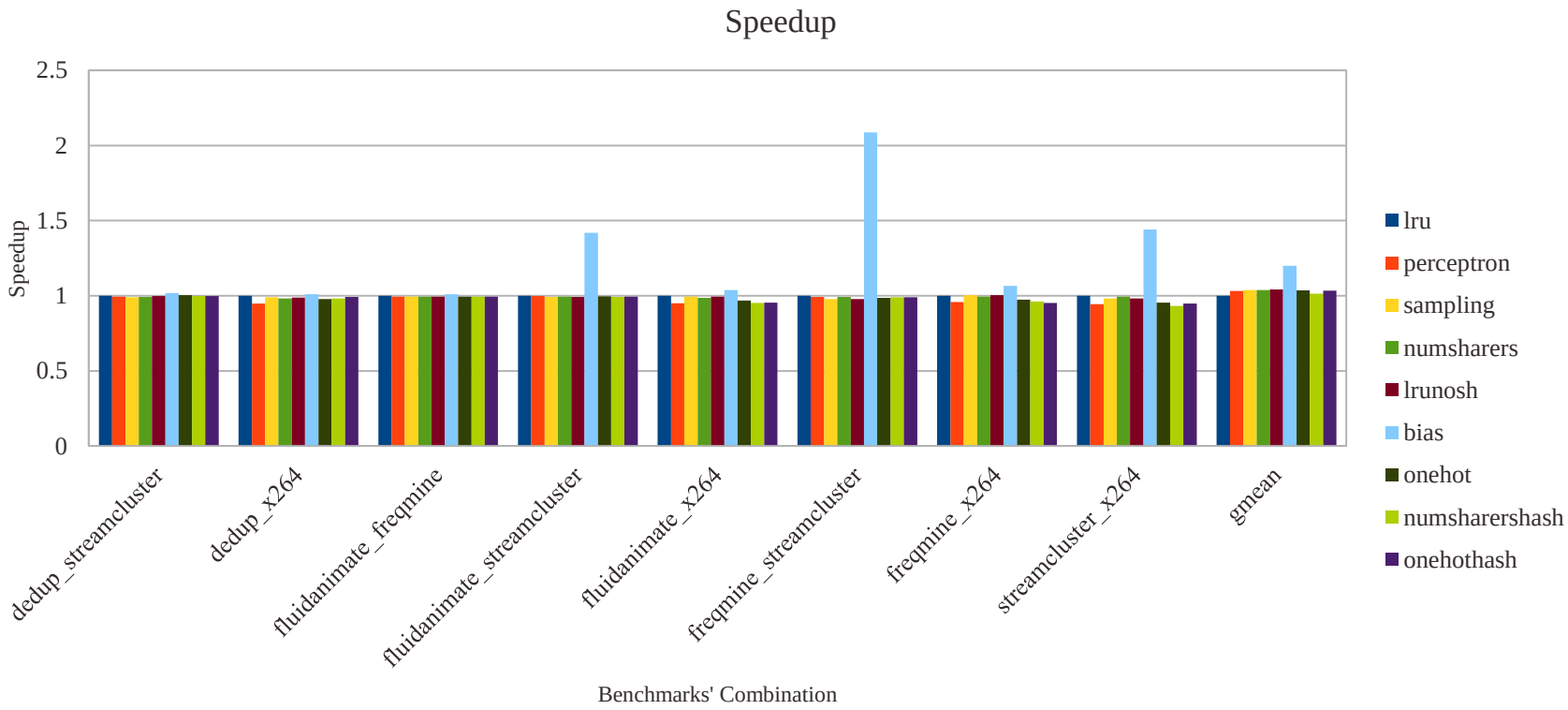
Results (4MB) (Speedup)



Results (4MB) (Speedup)



Results (4MB) (Speedup)



Results (4MB) (Observations)

- On an average, perceptron with bias as a feature, shows much less MPKI, dropped to 40%, w.r.t LRU.
- Even other variations of perceptron shows lower MPKI w.r.t LRU.
- For a combination of *streamcluster* with every other benchmark except *dedup*, with perceptron bias replacement policy, MPKI dropped to at least 25% w.r.t LRU.
- For a combination of *x264* & *blackscholes* and *x264* & *canneal*, MPKI respective to every replacement policy dropped to around 33% and 25% respectively, w.r.t LRU.

- Perceptron with bias as a feature, achieves geometric mean speedup of 20% over LRU.
- Even other variations of perceptron shows marginal improvement over LRU.
- Similar to MPKI, for a combination of *streamcluster* every other benchmark except *dedup*, bias achieves a speedup of at least 40% over LRU.
- For a combination of *x264* & *blackscholes* and *x264* & *canneal*, every replacement policy achieves a speedup of 60% and 120% respectively, over LRU.

Conclusion and Future Plans

- We derive five perceptron alternatives and for 4MB & 8MB LLC they all show improvement, on an average.
- Specially, perceptron implemented with bias as a feature outperforms every other replacement policy and shows a major improvement.

- In near future, we plan to analyze and explain the results regarding which we need to observe the number of blocks evicted for every number of sharers.

Thank You

TEXAS A&M UNIVERSITY