Enhanced Base-Delta Compression with Memory Pooling

Electrical & Computer ENGINEERING

Aditya Bhandaru, Gennady Pekhimenko, Onur Mutlu

Carnegie Mellon

Overview

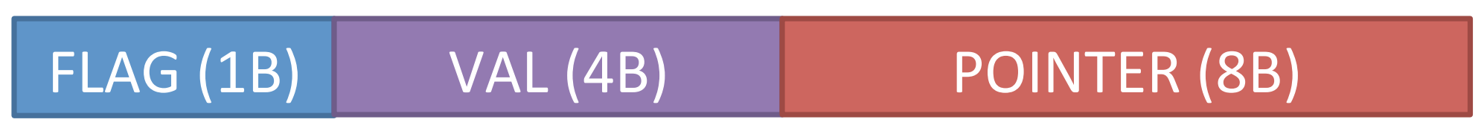
- Base-Delta Compression [Pekhimenko et. al., PACT'12] proposes a promising technique for increasing on chip cache capacity using **compression**.
- **B+Δ** offers good compression but incurs an **additional access latency**.
- **B+Δ** suffers poor compressibility when adjacent data in memory have large value ranges.

- **Observation**: Traditional compilers and memory-allocators are unaware of **B+Δ** cache compression in hardware.
- **Key Idea**: Arrange data in memory to optimize B+Δ compressibility.
- **Solution**: Recent literature on Memory Pooling and Data Splitting [Curial et. al., ISMM'08] and related work seem promising.

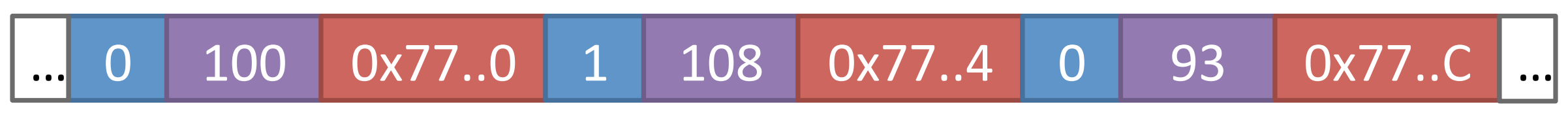
Mechanisms

Basic Splitting-Pooling Example (64-bit)

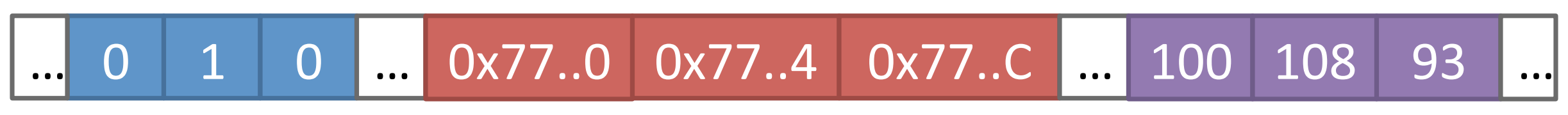
Simple struct (a node perhaps)



In memory layout (high range in adjacent values)



After split-pool allocation (much lower range)



Proof of Concept Methodology

- To test the affect of splitting and pooling on **B+Δ** compression, we manually restructured programs for optimal data layout.
- Ideally these pointer transformations will be implemented in the compiler.

Effect of Pooling on B+Δ Compression Block Types

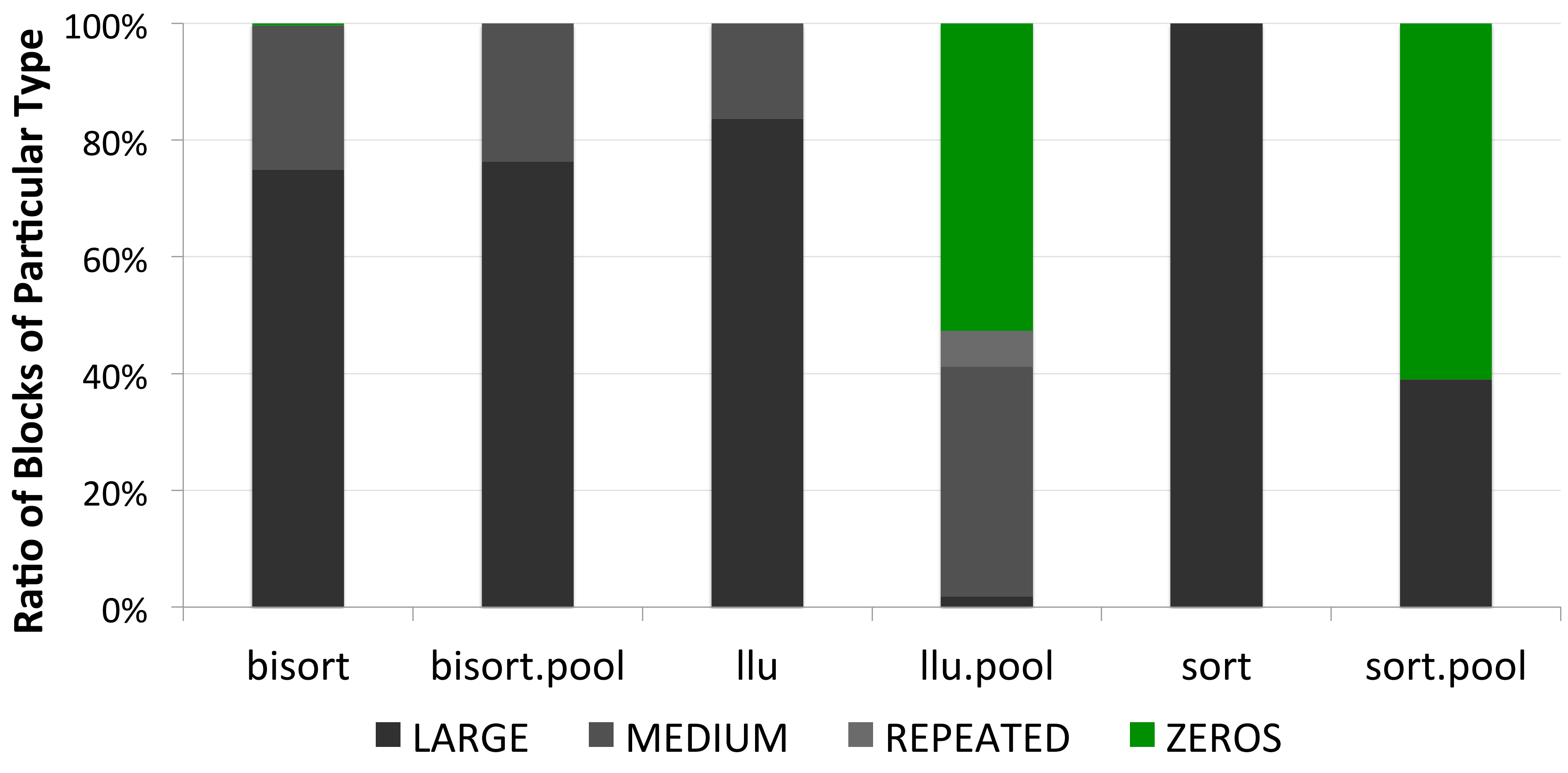


Figure 1. Each column shows the ratio of block-types for B+D compression with and without splitting and pooling. Notice the large increase in 1-byte all-zero blocks, and general decrease of large, uncompressed blocks.

***B+Δ** compression on a 2MB, 16-way, 32BiB cache.*

Motivation

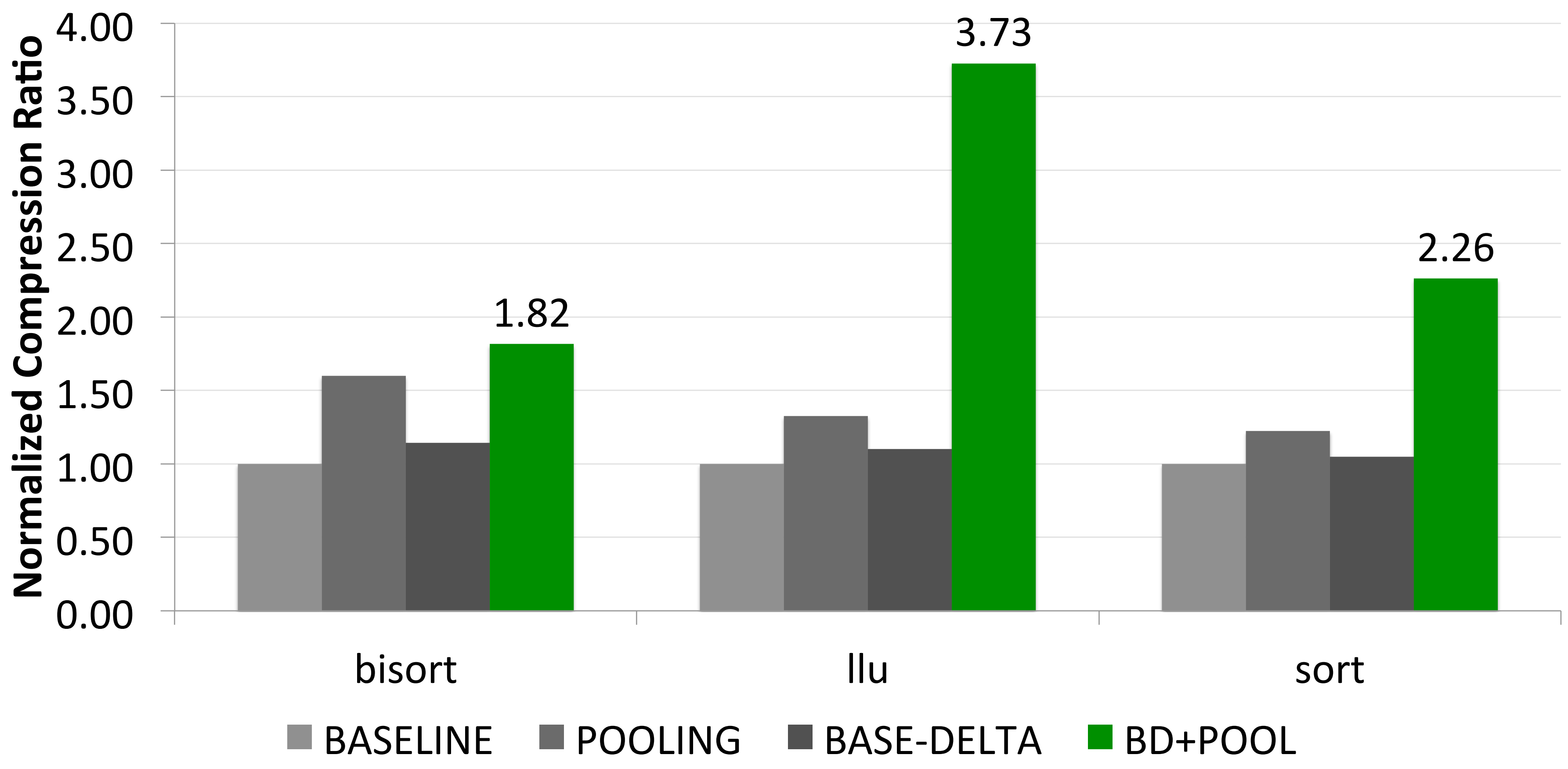
Problem: Can we mitigate low compressibility cases for **B+Δ** compression?

- Increase viability for B+Δ implementation in hardware, and justify the extra access latency.
- Proposals like Memory Pooling and Data Splitting **already improve locality and reduce value range** in adjacent data values.

- But they have not yet been applied to **B+Δ**!

Results

Normalized Compression Ratio for Benchmarks



Conclusions

Min-Eviction: a novel replacement policy for the compressed cache

- Outperforms current state-of-the-art replacement policies
- First to consider both compressed block size and probability of reuse
- Simple to implement

Further Work:

- **Global Min-Eviction**: a global replacement policy for the compressed decoupled variable way cache that applies similar insight as Min-Eviction
- **Fairness** in compressed cache replacement
- **Multi-core evaluation and analysis** (see paper): 4% increase in normalized weighted speedup over LRU in heterogeneous workloads