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.

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**!

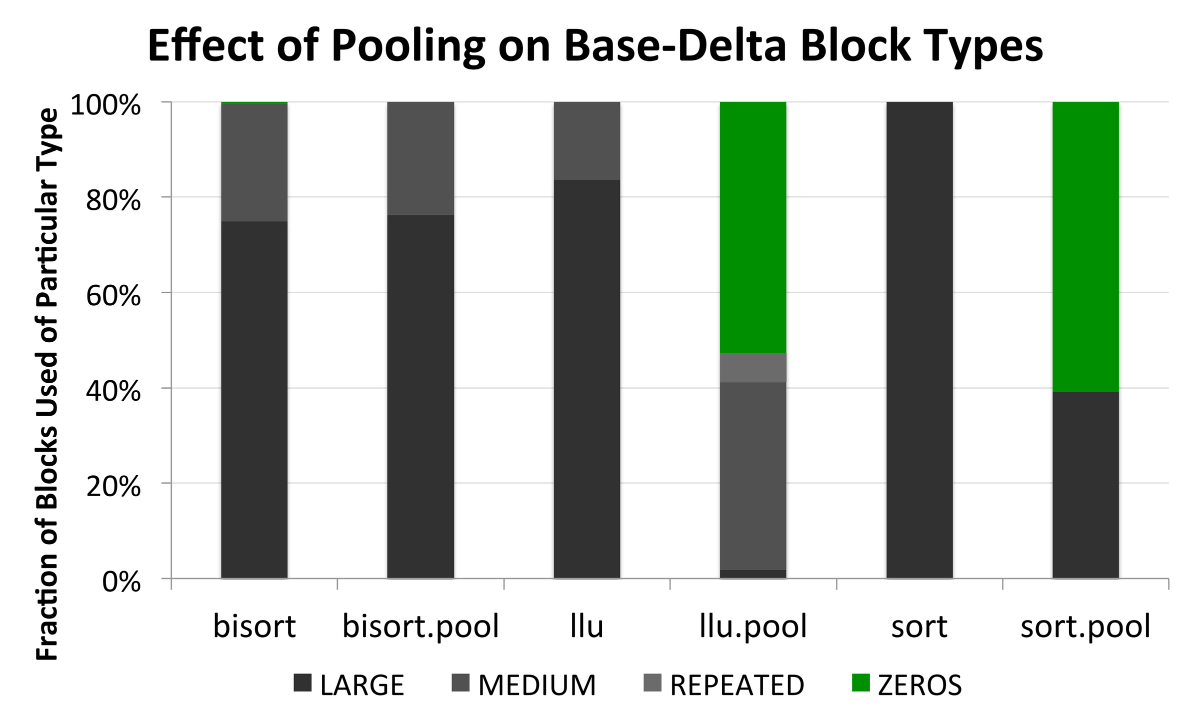
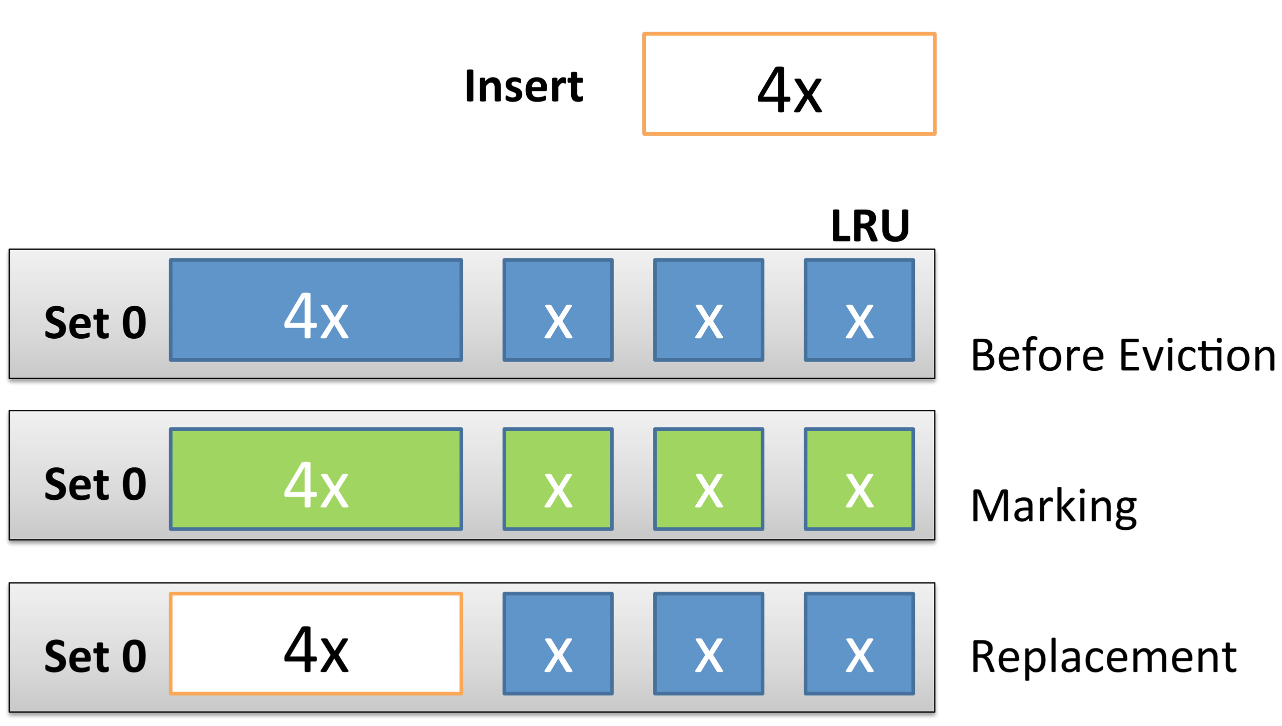


Figure 1. Each column shows the ratio block-types for B+D with and without splitting and pooling.

Notice the large increase in 1-byte all-zero blocks, and general decrease of large, uncompressed blocks.

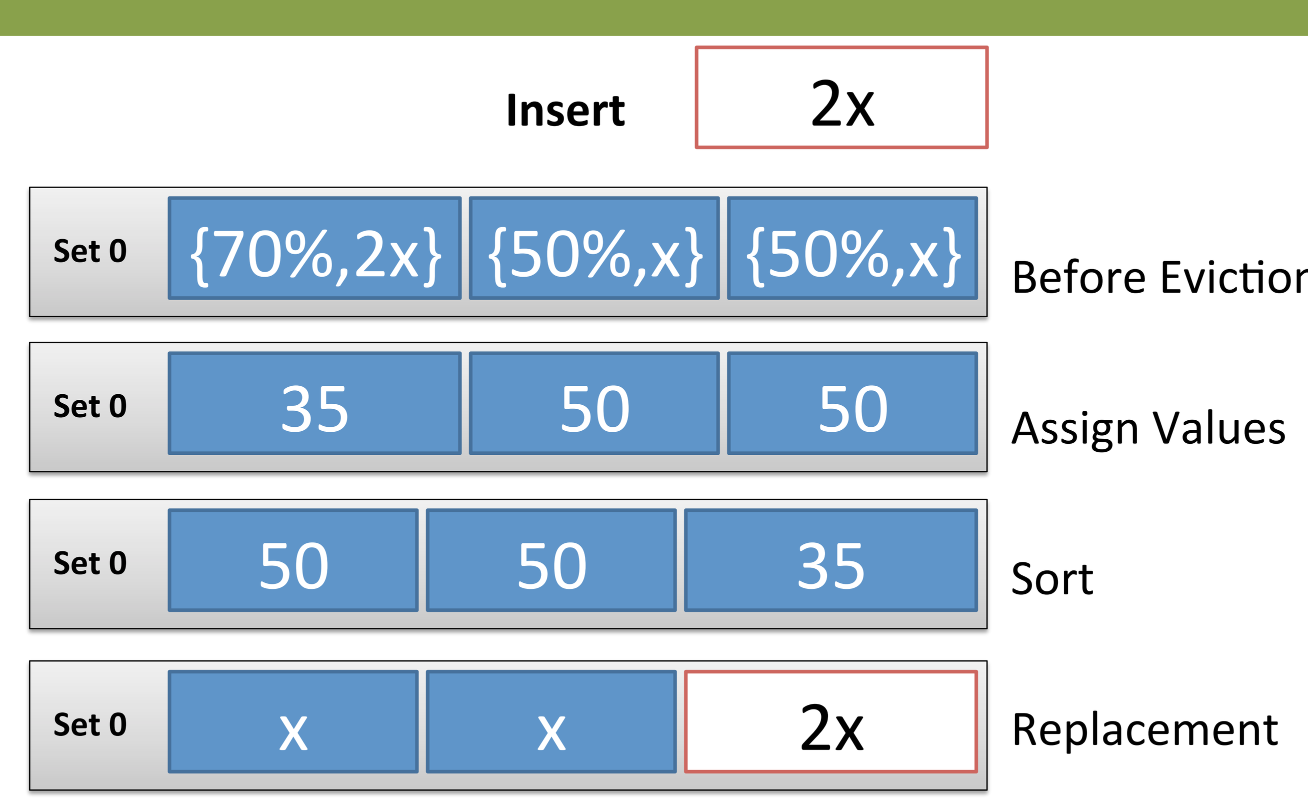
Mechanisms



•Policy 1: Min-LRU

Insight: LRU evicts more blocks than necessary

Key Idea: Evict only the minimum number of LRU blocks



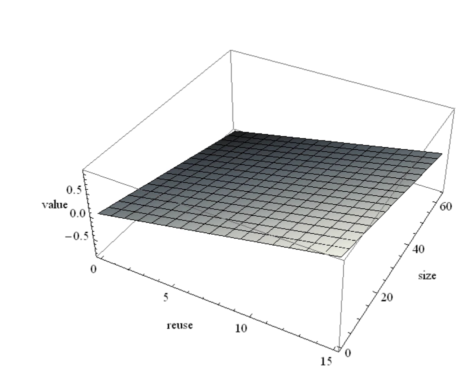
•Policy 2: Min-Eviction

Insight: Keeping multiple compressible blocks with less reuse may be more valuable than a single uncompressible block of higher reuse

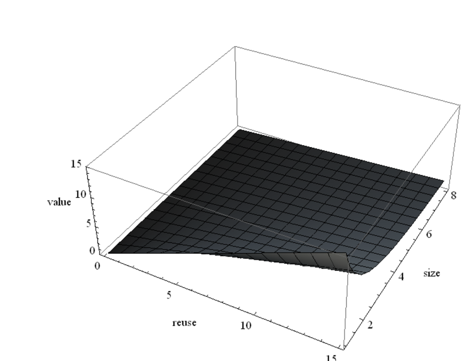
Key Idea: Assign a value based on reuse and compressibility to all blocks and on replacement, evict the set of blocks with the least value

Assigning Values to Block

- **Value function: f(block reuse, block size)**
- Monotonically increasing with respect to block reuse
- Monotonically decreasing with respect to block size
- **Plane** (see figure) achieves these goals, but is complex to implement in hardware
- **Reuse/Size** (see figure) approximates plane and is less complex
- **Probability of reuse predictor: RRIP [Jaleel et. al., ISCA'10] derivative**



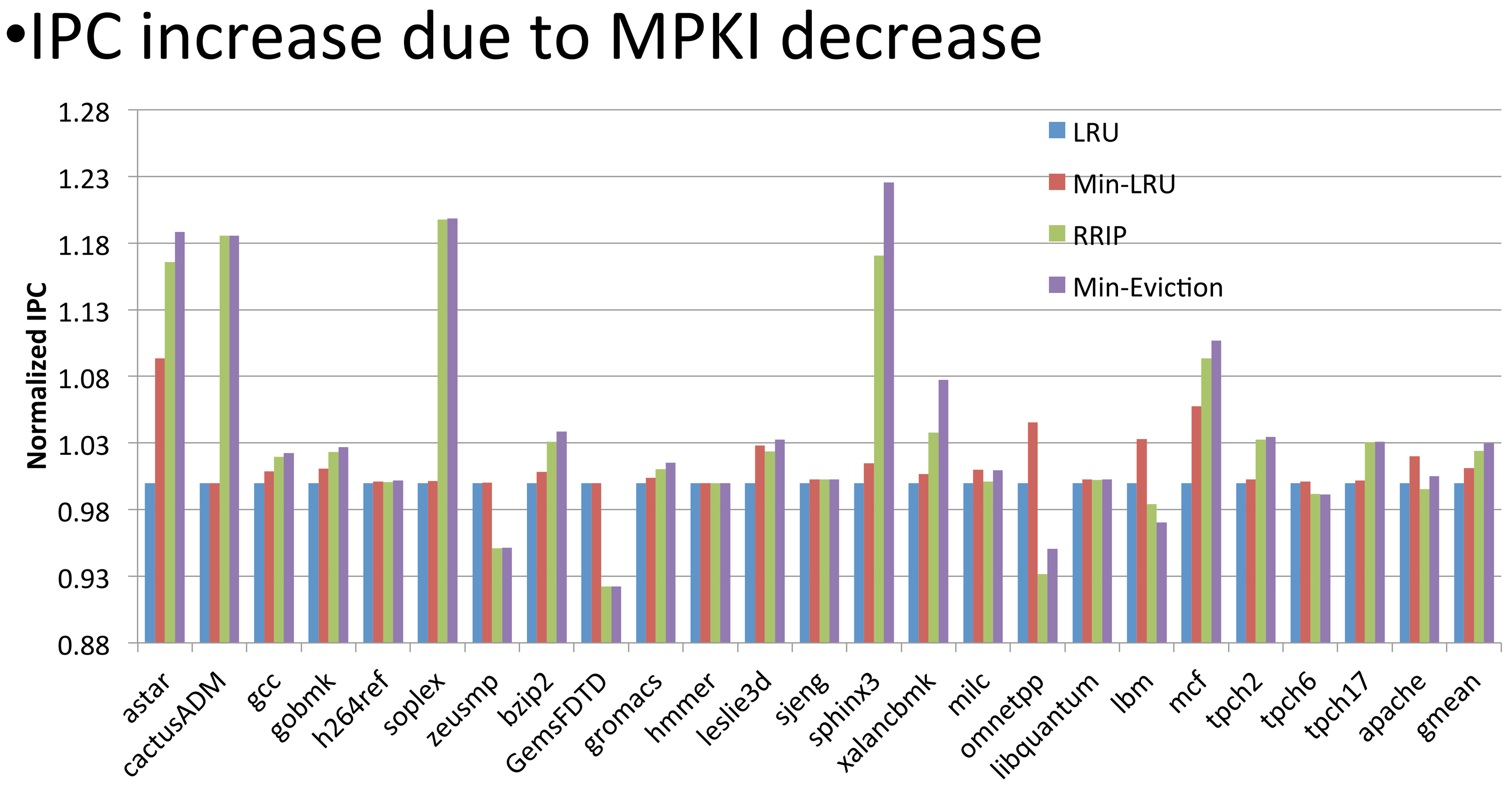
3D Plane



Reuse/Size

Results

- Min-LRU: 1% increase in IPC over LRU
- Min-Eviction: 3% increase in IPC over LRU
- IPC increase due to MPKI decrease



2MB cache size, Base-Delta-Immediate compression scheme, 4Ghz x86 in-order, 1B instructions

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