**Abstract**

Recent literature on cache compression has shown great potential for increasing the effective cache capacity on chip. Specifically, a technique called Base-Delta (BD) compression has presented excellent compression (about 1.4X) and improvements in overall performance. However, BD suffers from poor compressibility when adjacent data in memory have a high range in value.

We show here, as proof of concept, that existing techniques such as Data Splitting and Memory Pooling can enhance the BD compressibility of data in memory. Our simulations over various micro-benchmarks show that BD with pooling results in an 8% reduction in MPKI, and a compression ratio of 2.6X over the baseline.

**Introduction**

The memory bottleneck is a well known problem in computer architecture. Caching has become a standard for alleviating contention for data, the bus, and memory. As we trend to more cores, more applications, and larger computing problems, there is a much greater demand for data. Simply scaling cache size to compensate is too expensive, both in power and chip area.

Data compression in the cache is a promising alternative to increasing effective on chip cache capacity. For the same physical cache space, we can store more blocks per set. The ideal cache compression implementation would be fast, simple, and offer a high compression. Many ideas from older literature on cache compression suffer from either poor compression or incur high hardware complexity or long decompression latencies.

Why is fast decompression more important than fast compression? Decompression is on the critical path for a read. In order to supply the requested word, we must decompress the cache line. During a cache fill, compression can occur in the background while we bypass the requested word.

Recently a technique called Base-Delta compression

Recently, a technique called Base-Delta-Immediate compression [Pekhimenko et. al., PACT’12] was suggested. The first technique presented was referred to as Base+Delta (BD) compression. They later iterate on BD to represent cache lines with multiple bases – one that is computed and another that is always zero. This is referred to as Base-Delta-Immediate (BDI) compression.

**Motivation**

This paper focuses on BD compression over BDI compression for a couple reasons. The first is simplicity. Encoding cache lines with one base requires less metadata in the tag store and simpler hardware. Second, we hypothesize that applying techniques such as data splitting and memory pooling will alleviate many of the low compressibility cases that BD suffered on the benchmark tests.

The