# Enhanced Base-Delta Compression with Data Splitting and Memory Pooling

Aditya Bhandaru (akbhanda@andrew.cmu.edu) Gennady Pekhimenko (akbhanda@andrew.cmu.edu)
Onur Mutlu (akbhanda@andrew.cmu.edu)

## Abstract

*Recent literature on cache compression has shown great potential for increasing the effective cache capacity on chip. Specifically, a technique called Base-Delta (B+Δ) compression has presented excellent compression (about 1.4X) and improvements in overall performance. However, B+Δ 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 B+Δ compressibility*** *of data in memory. Our simulations over various micro-benchmarks show that B+Δ with pooling results in an 8% reduction in MPKI, and a compression ratio of 2.6X over the baseline.*

## 1. 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. In general, compression algorithms look for patterns in data to exploit. Therefore, they are very sensitive to how data is laid out in memory and the kind of data a program manipulates.

The ideal cache compression implementation would be fast, simple, and offer a high compression. These design points are largely at odds with one another. For example, 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.

### 1.1. Motivation

Recently, a paper on Base-Delta-Immediate compression [6] suggested technique called Base+Delta (B+Δ) compression. Base-Delta-Immediate (BΔI) compression is the final iteration of the technique. In comparison to preceding work on cache compression, it hits the fast-simple-high-compression trifecta nicely.

Their work yield largely positive results, making B+Δ compression worthy of further study and improvement. In particular, we observe that improving data layout in memory to leverage B+Δ compression in hardware may result in substantial performance gains. This paper focuses on two existing techniques: data splitting and memory. As a proof of concept, we show that applying these transformations to programs running on B+Δ architectures will realize considerable gains.

## 2. Related Work

There is a substantial body of work on optimizing the arrangement of data in memory to improve temporal and spatial locality. Notably these approaches often involve compiler hints or runtime directives to a runtime library [2] or allocator [1] for cache conscious data placement.

The idea of cache conscious data placement is not new. Further studies on dynamic analysis using profiling [3] and even runtime structure splitting [7] were introduced to reduce programmer effort. The latter technique offers great adaptability to the dynamic needs of the program, but incurs some overhead due to safety checks.

These findings enumerate various methods for **safely** and **efficiently** controlling data placement in memory, and performing the necessary pointer transformations in the compiler. *However, none of this prior work has targeted B+Δ compression specifically.*

B+Δ compression leverages that observation that for many cache lines, the values contained have a low dynamic range. That is, they could be encoded using a common base-value and a number of much smaller delta-values. For architectures that implement B+Δ cache compression, a cache conscious placement policy would then place values with low dynamic range together, therefore minimizing the deltas.

We investigate two existing mechanisms that place similar values together in memory: **data splitting and memory pooling**. As a proof of concept, this paper does not address the non-trivial implementation challenges for splitting and pooling data. There is however, compelling work in the area such as MPADS [5] and Forma [4].

**Why Base-Delta and not Base-Delta-Immediate?** This paper focuses on BD compression over BDI compression for a couple reasons. First, we suspect that applying techniques such as data splitting and memory pooling will alleviate many of the low compressibility cases that BD suffered on the bench-

mark tests. The second is simplicity. Encoding cache lines with one base requires less metadata in the tag store and simpler hardware.

## References


[1] S. John B. Calder, K. Chandra and T. Austin, "Cache-conscious data placement." *Proc. ASPLOS-VIII*, 1998.

[2] O. Mutlu M. Kozuch P. Gibbons T. Mowry G. Pekhimenko, V. Seshadri, "Base-delta-immediate compression: Practical data compression for on-chip caches," *PACT*, 2012.

[3] Chris Lattner and Vikram Adve, "Automatic pool allocation: Improving performance by controlling data structure layout in the heap." *PLDI*, 2005.

[4] Y. Gao R. Silvera J. Amaral P. Zhao, S. Cui, "Forma: A framework for safe automatic array reshaping," *ACM*, 2007.

[5] J. Amaral Y. Gao S. Cui R. Silvera R. Archambault S. Curial, P. Zhao, "On-the-fly structure splitting for heap objects," *ISMM*, 2008.

[6] M.D.Hill T.M. Chilimbi and J. R.Larus, "Cache-conscious structure layout." *PLDI*, 1999.

[7] PC. Yew J. Li D. Xu Z. Wang, C. Wu, "On-the-fly structure splitting for heap objects," *ACM*, 2012.