# BCD Deduplication: Effective Memory Compression Using Partial Cache-Line Deduplication

***Extended Abstract***

Sungbo Park[1], Ingab Kang[2], Yaebin Moon[2], Jung Ho Ahn[2], and G. Edward Suh[1]
[1]Cornell University, [2]Seoul National University

## 1. Motivation

Memory-intensive workloads such as in-memory databases have large working sets, and their performance is often quite sensitive to memory capacity. For example, if an application's working set does not fit in memory, disk accesses can significantly degrade performance. Unfortunately, the main memory (DRAM) cost is already a significant portion of the total cost of modern data centers, and it is costly to simply increase physical memory capacity. Moreover, large-capacity DRAM modules consume a significant amount of energy; DRAM consumes 20∼30% of the system energy in data centers [3]. Thus, reducing the physical DRAM capacity needed for a given workload is important to save DRAM cost and energy consumption.

## 2. Limitations of the State of the Art

Hardware-based memory compression techniques have the potential to significantly reduce the DRAM cost and improve the performance of memory-constrained applications by allowing more data to be stored in given physical memory. However, memory compression techniques need to carefully balance complexity and compression ratio; complex compression algorithms often achieve a higher compression ratio but may lead to significant performance degradation due to additional memory accesses and decompression latency.

**Pattern-based Cache-Line Compression:** Since additional memory accesses may significantly hinder performance due to the long latency, most memory compression techniques operate at a cache-line granularity. These techniques have low latency, but they often have relatively low compression ratios because they cannot exploit redundancy beyond one cache line.

**Dictionary-Based Compression:** Since a dictionary-based approach requires additional accesses to the dictionary, only a few schemes, including IBM's Memory Expansion Technology (MXT) [2], have been proposed for main memory. While MXT has a relatively high compression ratio, it has long latencies to compress and decompress a 1-KB block at a time. To hide the latencies, MXT uses a 32-MB cache in front of the compressed memory. Even with the large cache, the performance overhead would be significant for applications with working sets larger than the large cache.

**Memory Deduplication:** Memory deduplication is a special form of compression that stores only one copy of identical memory blocks by adding a level of address translation. While deduplication exploits redundancy across blocks, it is effective only when multiple blocks are exactly the same.

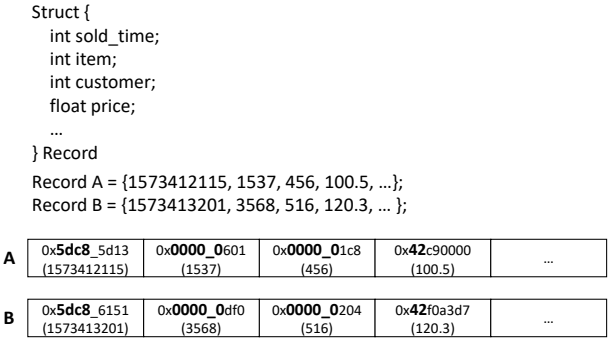


**Figure 1: An example of cache lines that cannot be compressed by cache-line compression.**

## 3. Key Insights

A variable in typical programs tends to have values in a limited range. This means that high-order bits of 4B or 8B data are often identical for multiple variables in an array. For example, if an array that has 16 4-B integers is stored in a cache line and all the integers represent customers' ages, they are likely less than 150. Previous cache-line compression techniques such as BDI [6] and Compresso [4] can leverage this redundancy if a cache line contains a single type of variables. However, if a cache line contains a structure or a class with multiple types of variables, each with a different range, the line may not be compressible in a traditional scheme. Figure 1 represents an example of two cache lines where each contains a structure with the same type. In this case, the bold high-order bits are identical in both lines, but they cannot be reused by the traditional compression, deduplication, or straight combination of the two. Our experiment results show that there is partial redundancy (one half of the block is identical to another block) for more than 60% of blocks in TPC-DS and TPC-H and 35% in the SPEC2017 benchmarks. For these blocks, at least one half can be reused across blocks.

## 4. Main Artifacts

To exploit the partial match among memory blocks (redundancy in high-order bits), we propose Base and Compressed Difference (BCD) deduplication, a novel combination of memory compression and deduplication that increases the effective capacities of both last-level cache (LLC) and main memory.

Figure 2 presents an overview of BCD. On a write, BCD first performs the traditional memory deduplication to compress a data block that is identical to another one (full-match). If

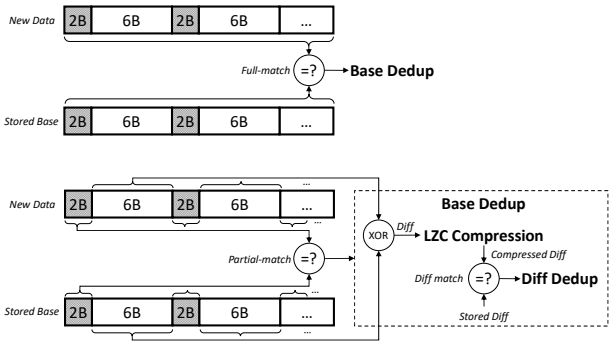


**Figure 2: An overview of BCD deduplication.**

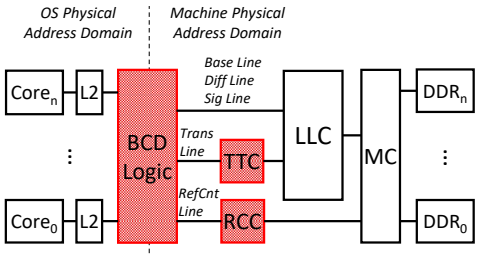


**Figure 3: A processor architecture with BCD compressed LLC and memory.**

there is no block with a full match, BCD checks for a partial match where the high-order 2B out of every 8B chunk within a block is identical. For a partial match, the matched part is deduplicated using the existing block as the base. The difference between the new block and the base is calculated using XOR operation. Then, the difference is compressed using leading-zero count (LZC) [5] compression. Finally, if there is the same compressed diff in memory, the diff is deduplicated. In summary, when multiple blocks have the same high-order bits, only one is fully stored in memory. For the rest, only the differences are stored in a compressed and deduplicated format.

While the scheme may look complex at a glance, BCD can be implemented as an extension to the traditional deduplication. The overhead is also still low; a read requires at most three memory accesses, and a write can be optimized by selectively applying different types of compression: base full/partial deduplication, difference deduplication, and difference compression.

The main challenge in implementing BCD lies in reducing potential performance overhead from additional meta-data accesses. To avoid the overhead, the proposed architecture makes three key design decisions and optimizations. Figure 3 shows the architecture overview of BCD. First, BCD is applied at the boundary of the L2 caches and the LLC. In this way, most meta-data are cached in the LLC and can often be accessed without off-chip accesses. This design also increases the effective capacity of the LLC and can improve performance. Second, the architecture customizes caches to match meta-data characteristics, including the translation table cache (TTC), and the reference count cache (RCC). Finally, we introduce a selective deduplication scheme to avoid deduplication overhead when applications do not benefit from added deduplication steps.

To understand the effectiveness of BCD for a broad set of applications, the compression ratios are first calculated from memory dumps of SPEC2017 [1], TPC-DS [8], and TPC-H [7]. Also, the performance impact of BCD is evaluated using Zsim [9] with SPEC2017.

## 5. Key Results and Contributions

**Compression Ratio from Memory Dumps:** In our experiment with memory dumps, BCD achieves an average 1.97× memory compression for SPEC2017, TPC-DS, and TPC-H. This compression ratio is 50.9% and 29.7% higher than that of Compresso, which represents the state-of-the-art cache-line compression scheme, and the combination of Compresso and deduplication, respectively. In comparison, the naïve combination is only 8.8% better than Compresso. MXT is 17.3% better than BCD, but performs compression/decompression at a 1-KB granularity and needs a large uncompressed cache to avoid significant performance overhead. BCD achieves compression ratios close to MXT while only compressing/decompressing one cache block at a time.

**Performance Simulation:** From the cycle-level simulation, though BCD without selective dedup has a 3.4% slowdown on average, BCD with selective dedup improves performance by 2.7% on average. Therefore, BCD provides a significant capacity gain with no noticeable performance overhead. If the selective dedup is applied, BCD can even improve performance by reducing LLC misses. In fact, the 2.7% speed-up is comparable to the baseline with a 1.625x larger LLC (2.6% speedup). Moreover, the speedup is significant (over 50%) for a few benchmarks, and selective dedup effectively limits the slowdown for some benchmarks.

The following summarizes the main contributions:

- We identify partial data redundancy across compression blocks that cannot be exploited by today's cache-line compression or memory deduplication, or the straightforward combination of the two.
- We propose Base and Compressed Difference (BCD) deduplication, which is a new deduplication scheme that leverages the partial inter-block data redundancy effectively. BCD achieves main memory saving of 49.3% on average for SPEC2017, TPC-DS, and TPC-H, which is 110% and 66% higher than the state-of-the-art memory compression and the naïve combination of the compression and deduplication.
- We introduce a hardware design that can effectively realize the BCD scheme for both LLC and main memory with low area overead, and achieve a performance improvement, which corresponds to a 62.5% larger LLC for SPEC2017.

## 6. Why ASPLOS

Memory management is one area where hardware architecture and an operating system are tightly coupled. Although BCD is almost transparent to OS, it requires OS support for data structure initialization, overflow, and selective deduplication.

# References


[1] Spec cpu2017 home page. https://www.spec.org/cpu2017. Accessed: 2020-08-15.

[2] B. Abali, H. Franke, D. E. Poff, R. A. Saccone, C. O. Schulz, L. M. Herger, and T. B. Smith. Memory expansion technology (mxt): Software support and performance. *IBM Journal of Research and Development*, 45(2):287–301, March 2001.

[3] Luiz André Barroso, Jimmy Clidaras, and Urs Hölzle. *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines, Second Edition*. Morgan & Claypool Publishers, 2013.

[4] Esha Choukse, Mattan Erez, and Alaa R. Alameldeen. Compresso: Pragmatic main memory compression. In *Micro*, 2018.

[5] Giorgos Dimitrakopoulos, Kostas Galanopoulos, Christos Mavrokefalidis, and Dimitris Nikolos. Low-power leading-zero counting and anticipation logic for high-speed floating point units. *IEEE transactions on very large scale integration (VLSI) systems*, 16(7):837–850, 2008.

[6] Gennady Pekhimenko, Vivek Seshadri, Onur Mutlu, Phillip B. Gibbons, Michael A. Kozuch, and Todd C. Mowry. Base-delta-immediate compression: Practical data compression for on-chip caches. In *PACT*, 2012.

[7] Meikel Poess and Chris Floyd. New tpc benchmarks for decision support and web commerce. *SIGMOD Record*, 29(4), 2000.

[8] Meikel Poess, Bryan Smith, Lubor Kollar, and Paul Larson. Tpc-ds, taking decision support benchmarking to the next level. In *ACM SIGMOD International Conference on Management of Data*, 2002.

[9] Daniel Sanchez and Christos Kozyrakis. Zsim: Fast and accurate microarchitectural simulation of thousand-core systems. In *ISCA*, 2013.