GBDI: GLOBAL BASE-DELTA-IMMEDIATE MEMORY COMPRESSION

PRESENTED BY: TingHung Chiu and Promit Panja

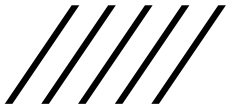
ECE5504

Outline

- Introduction
- Memory Compression Background
- Base-Delta-Immediate (BDI) Compression
- Global Base-Delta-Immediate (GBDI) Compression
- GBDI Implementation
- Results
- Future Work
- Conclusion

Introduction

- Due to exponential growth in processor performance there has been exponential growth in demand for memory bandwidth.
- High-bandwidth memory (HBM) can solve the issue but are **expensive**.
- Large caches can help but at significant cost.
- Cache compression enables larger cache at **lower cost**.
- As a result lossless memory compression algorithms are lucrative option for increasing memory bandwidth.



Memory Compression

- Memory compression is a technique used to increase the effective size of a computer's physical memory by compressing the data stored in memory.
- Memory compression algorithms can be **hardware** or **software-based**.

  -**Hardware-based** use dedicated hardware on the memory controller to compress and decompress.

  -**Software-based** use software algorithms to compress and decompress.

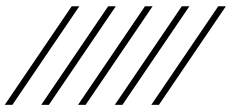
- Memory compression techniques abound to **maximize the compression ratio (CR)** while maintaining a **low latency**.

BDI (Base-Delta Immediate)

- **Base-Delta-Immediate (BΔI)** compression is a practical technique for compressing data in on-chip caches.
- Based on two observations:

  -The data values stored within the cache line have **small** relative difference between values.

  -Deltas can be encoded compactly.
- In such cases, the cache line can be represented in a compact form using a common **base** value plus an array of relative differences ("**deltas**"), whose combined size is much smaller than the original cache line.



BDI (Base-Delta Immediate)

Zero Values: initialization, sparse matrices, NULL pointers

0x00000000	0x00000000	0x00000000	0x00000000	…

Narrow Values: small values stored in a big data type

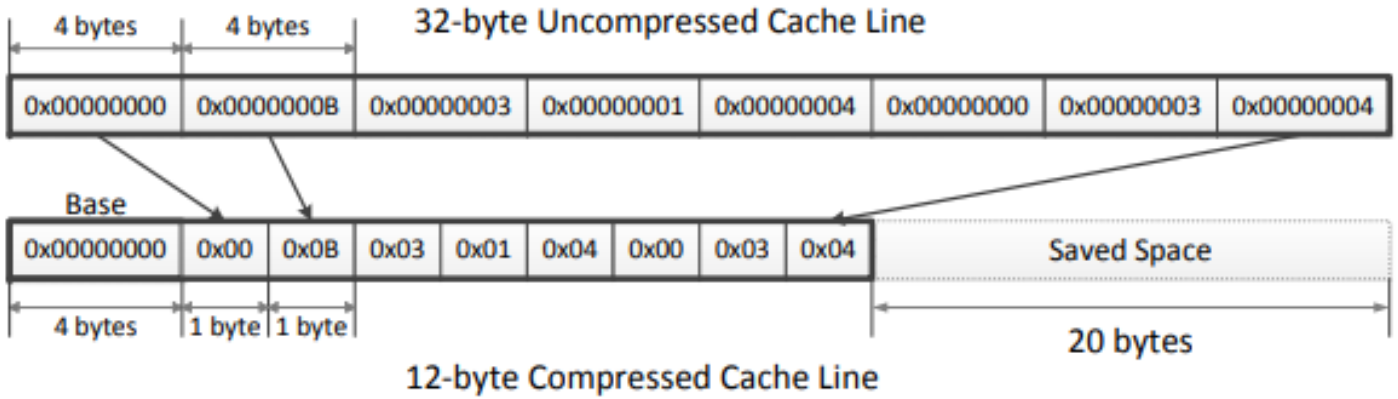
0x*00000000*	0x*000000***0B**	0x*00000000*	0x*00000000*	…

Other Patterns: pointers to the same memory region

0x*C04039***C0**	0x*C04039***C8**	0x*C04039***D0**	0x*C04039***D8**	…

*This slide has been taken from the original BDI presentation

BDI (Base-Delta Immediate)

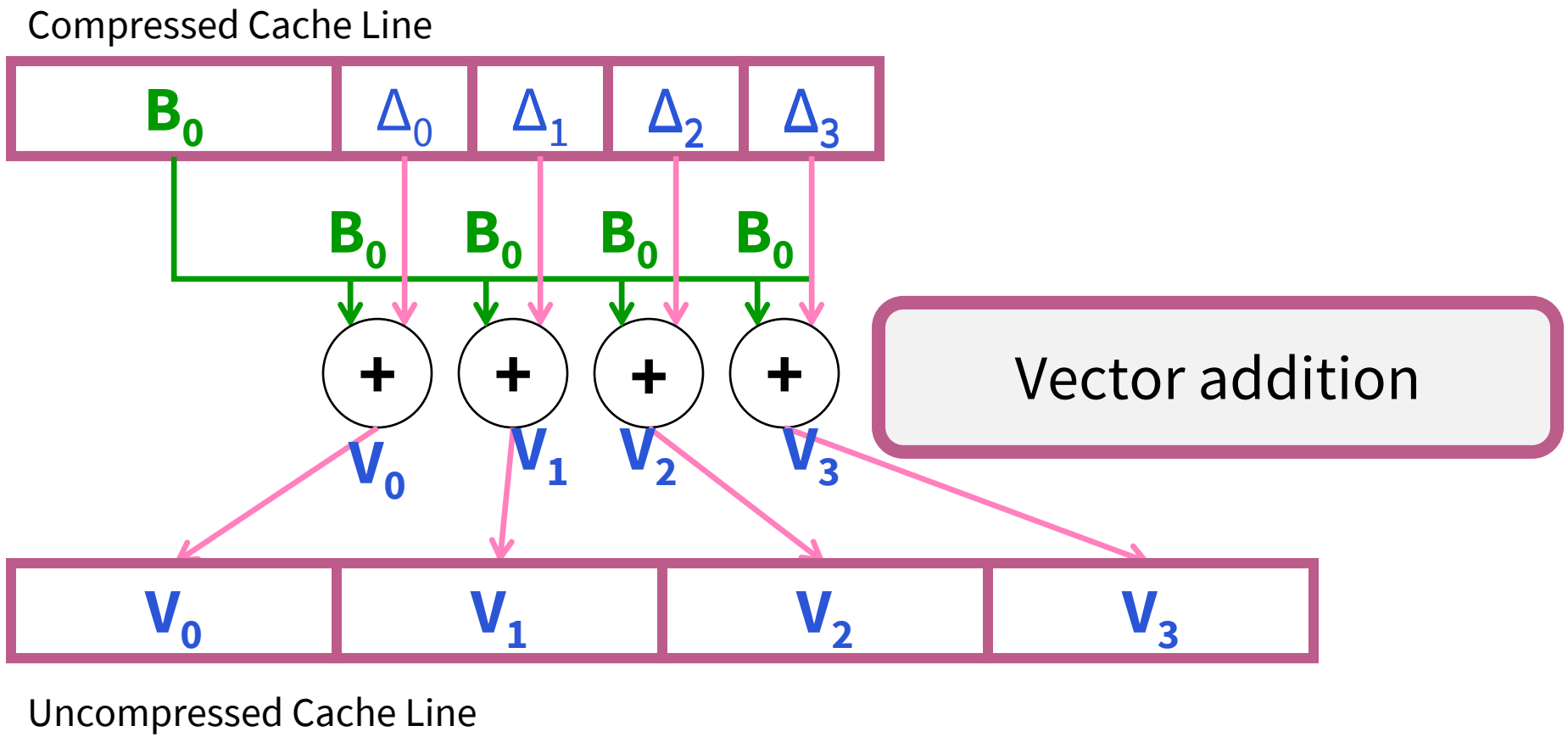


To decompress a compressed cache line, The value $\mathbf{v_i}$ is simply given by

$$\boldsymbol{v_i} = \boldsymbol{B}^* + \Delta_{\boldsymbol{i}}$$

As a result, the values in the cache line can be computed in **parallel** using a **SIMD**-style **vector adder**. Consequently, the entire cache line can be decompressed in the amount of time it takes to do an integer vector addition, using a set of simple adders.

BDI (Base-Delta Immediate)



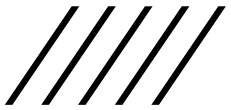
*This slide has been taken from the original BDI presentation

GBDI (Global Base-Delta Immediate)

- **Global Base Delta Immediate** compression (**GBDI**)– is a generalization of Base-Delta Immediate (BDI).
- The observations– intra-block values are numerically similar and deltas can be encoded compactly which are the basis of BDI are not true, in general, especially not for **floating-point** values.
- Select ***global*** base values across memory blocks, shared by all input data, instead of a single intra-block base value as in BDI.
- **Global bases** are established through a **data analysis** phase, by **software** algorithms, in the **background**.

GBDI (Global Base-Delta Immediate)

- GBDI aims at selecting a number of global bases, across all targeted data, that **minimizes** deltas when each value is encoded with a pointer to the closest global base and a delta with respect to that base.
- A **data analysis** methodology that, apart from selecting global bases additionally, on a per-block basis, selects the best among **BDI** and **GBDI**.
- The data analysis phase is triggered when the compression ratio **drops** below a given threshold.
- As this can be done in the background and is computationally light-weight, it is done in software.



Histogram Binning

Establishing Global Base Values:

- The authors propose **histogram binning** (**HB**), which empirically yields a good **CR** at a **lower** computational cost (time complexity and wall clock time).
- In **HB**, we construct a histogram of all values, with a number of equal-ranged bins, ***N***. Among the bins, we identify the ***B (B < N)*** bins that contain most values.
- Within each of the ***B*** bins, we locate the **single** value with the **highest** occurrence, and choose that as a **base** resulting in ***B*** base values.

Histogram Binning

Establishing Maximum Deltas:

- We set the maximum delta for each global base by considering all values that are closest to that base.
- The maximum delta is the number of bits needed to establish the distance to the furthest of these values.
- To improve compression, the authors establish an upper bound on the maximum delta.
- If the maximum delta is greater than the upper bound, it is lowered to be equal to the upper bound.

$$16 - log_2(B)$$

Where B is the number of global bases

GBDI Compression Algorithm

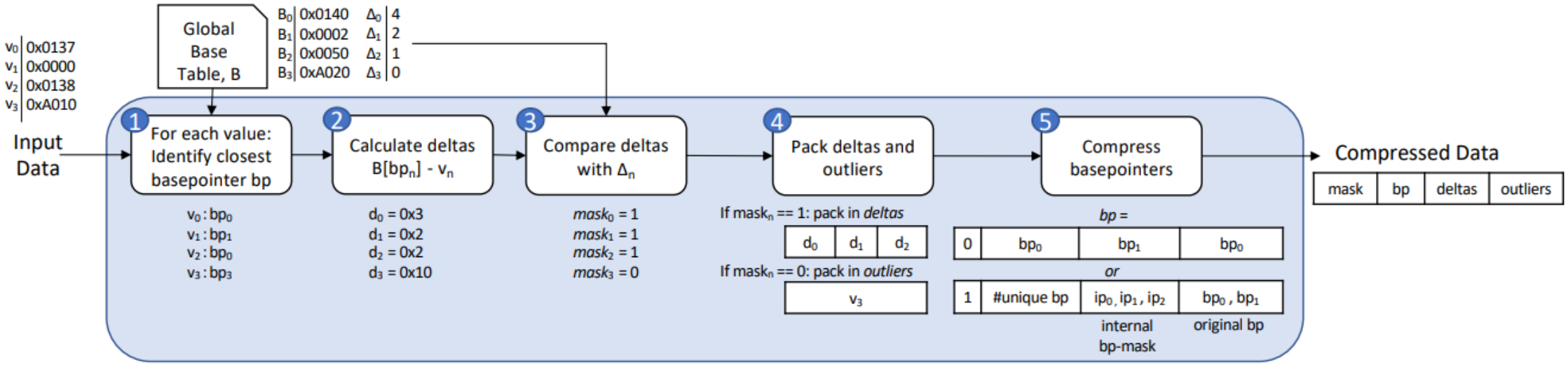


Figure 4: Overview of the main steps in the GBDI compression algorithm.

GBDI Implementation

- We implemented the GBDI memory compression in software using C.
- The three functions **establish_global_base_set_hb(), gbdi_compress(),** and **main()** make up the GBDI implementation.
- **establish_global_base_set_hb**: This function calculates and initializes the global base set used in GBDI compression. It takes input data, data count, a global base set pointer, and the maximum number of bases as input. It computes the most common deltas and adds them to the global base set.

GBDI Implementation

- **gbdi_compress**: This function takes uncompressed data, data count, a global base set, an output buffer, and an output size pointer as input. It compresses the input data using the GBDI algorithm, leveraging the global base set. It outputs the compressed data into the output buffer and updates the output size.
- **main**: The main function reads an ELF file, extracts memory segments, and processes cache lines. It calls extract_cache_lines to create a cache lines buffer from the segments. It then compresses the cache lines using GBDI compression . Finally, it prints the compression statistics.

Results

Benchmark	Original Size (Bytes)	Compressed Size (Bytes)	Compression Ratio (CR)
parsec_freqmine5dump	590430208	157331013	3.752789719
parsec_fluidanimate5dump	550227968	74133444	7.422128776
620.omnetpp_s_5.dump	231632896	41464076	5.586351327
600.omnetpp_s_5.dump	205651968	43178335	4.762850814

- As we can see the results currently obtained are incorrect and are far from the expected results as mentioned by the authors in the paper.
- The obtained result is incorrect because we have not excluded the all-zero pages and are still working on it to resolve the issue.

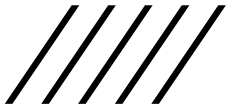
Future Work

- Improve the results by excluding all-zero pages.
- Include optimizations like- Huffman encoding to compress the base pointers.
- Testing the algorithm with different performance benchmarks.

Conclusion

From our implementation and testing of GBDI memory compression algorithm so far we have observed:

- Currently our results are incorrect as the compression ratios are unusually high.
- This can be resolved by excluding all-zero pages from compression, which we are still working on.



Thank You!