FastLanes: A SIMD-friendly Composable Compression Library

Azim Afroozeh and Peter Boncz
{azim, boncz}@cwi.nl

Abstract

In this talk, we first analyze the current compression schemes and try to address the shortcomings of the current ones. Then, we introduce a new library, FastLanes, designed to overcome these shortcomings.

We argue that the shortcomings of the state-of-the-art compression schemes could be categorized into four categories:

- Block-based compression: we call a compression scheme block-based if a block of data read into memory has to be decompressed altogether before being processed. These schemes, such as gzip, do not allow fine-grained decompression of tuples, forcing the database to decompress the whole block to access a single tuple [3]. This inhibits database systems from executing on the compressed data.
- Inefficient decompression: decompression can be considered a computation-intensive task. We argue that for a CPU database, the only way to make decompression efficient is using the SIMD capabilities of CPUs. However, using SIMD to speed up decompression is non-trivial. RLE and Delta encoding are two examples of schemes that have dependencies between tuples [2]. Therefore, it has not been possible to fully SIMDize their decoding [1].
- Inflexible decompression: compression schemes are primarily implemented as hard-coded functions that decompress data in one pass. A hard-coded function that combines multiple functionalities takes away flexibility. For instance, patched encodings could be seen as an enhancement (patching method) glued to traditional compression schemes such as FOR, Delta, DICT to make them less vulnerable to the outliers. However, several types of patching methods exist that each only work best for data with specific characteristics. Choosing one specific patching method leads to space and decompression speed overhead as it is not the best option for all cases. On the other hand, implementing all possible combinations makes the source code hard to maintain and update [2].
- Non-recursive compression: none block-based compression schemes often end up with compressed data that exhibit lots of similarities and potentially could be repeatedly encoded. For example, exception values that belong to patched encodings could be compressed as well [2]; however, previous compression libraries would not allow such recursive compression.

To overcome these shortcomings, we propose a new library, FastLanes, built upon three main principles. In FastLanes, firstly, all compression methods support compressed execution, after a (very fast) SIMDized bit-unpacking step. Secondly, we modify the RLE and Delta schemes to allow us to fully SIMDize the decompression part using transposed and interleaved value layouts. The basic idea behind these new layouts is to reorder the tuples to break the data dependencies into smaller parts and decompress each piece in parallel. Thirdly, we decompose the chosen compression schemes into separate primitives. By combining primitives in a hard-coded manner, we achieve the desired flexibility and recursivity.

Our preliminary experiments show up to 20 times faster decompression speed while having a comparable compression ratio.

References

[1] D. Abadi, S. Madden, and M. Ferreira. Integrating compression and execution in column-oriented database systems. pages 671–682, 01 2006.

[2] A. Afroozeh and P. Boncz. Towards a new file format for big data: Simd-friendly composable compression. *Master thesis*, 2020.

[3] M. Zukowski, S. Heman, N. Nes, and P. Boncz. Super-scalar ram-cpu cache compression. pages 59– 59, 05 2006.