# Byte-Aligned Bitmap Compression
## (abstract)

Gennady Antoshenkov
*Oracle Corporation*
*110 Spitbrook Road, Nashua, NH 03062, USA*

Bitmap compression reduces storage space and transmission time for unstructured bit sequences like in inverted files, spatial objects, etc. On the down side, the compressed bitmaps loose their functional properties. For example, checking a given bit position, set intersection, union, and difference can be performed only after full decoding, thus causing a many-folded operational speed degradation. The proposed Byte-aligned Bitmap Compression method (BBC) aims to support fast set operations on the compressed bitmap formats and, at the same time, to retain a competitive compression rate.

To achieve this objective, BBC abandons the traditional approach of encoding run-lengths (distances between two ones separated by zeros). Instead, BBC deals only with byte-aligned byte-size bitmap portions that are easy to fetch, store, AND, OR, and convert. The bitmap bytes are classified as *gaps* containing only zeros or only ones and *maps* containing a mixture of both. Continuous gaps are encoded by their byte length and a *fill* bit differentiating between zero and one gaps. Stretches of map bytes encode themselves and are accompanied with the stretch length in bytes. A pair (gap, map) is encoded into a single *atom* composed of a control byte followed by optional gap length and map. Single map byte having only one bit different than the others is encoded directly into a control byte using a "different bit" position within this map byte.

All set operations can be performed on BBC-encoded bitmaps using a single generic *merge* procedure that does only partial decoding of the operand streams into the gap/map byte sequences and then does simple byte skipping or AND/ORing. We ran benchmarks between BBC, δ encoding, and uncompressed sorted vectors of 32-bit integers.

| *Run-length:* | 1-1 | | 1-1 | | 1-3 | | 1-11 | | 1-10001 | |
|---|---|---|---|---|---|---|---|---|---|---|
| *Operands:* | 98% | 99% | 40% | 60% | 40% | 60% | 40% | 60% | 40% | 60% |
| | ∪ | ∩ | ∪ | ∩ | ∪ | ∩ | ∪ | ∩ | ∪ | ∩ |
| BBC : | 0.33 | 0.35 | 0.32 | 0.36 | 0.53 | 0.66 | 2.99 | 2.39 | 6.19 | 3.64 |
| δ: | 11.27 | 11.12 | 7.43 | 4.64 | 5.86 | 4.08 | 6.56 | 4.55 | 6.47 | 4.26 |
| IntVec: | 1.04 | 0.83 | 0.74 | 0.51 | 0.72 | 0.51 | 0.72 | 0.52 | 0.74 | 0.50 |

*In this test, sequences of one million integers were generated with run-lengths randomly distributed over intervals 1-n. Then two operands extracting X% and Y% from the sequence were produced and encoded for BBC and δ cases. Then ORing (∪) and ANDing (∩) of the two operands were performed using three different methods. In the table, merge time is given in CPU seconds running on VAX 6500.*

On dense bitmaps, BBC outperforms δ (up to 30 times) and even uncompressed integer merge (up to 3 times). BBC is always faster than δ.

We also introduced a simple extension mechanism for existing methods to accommodate a dual-gap (zeros and ones) run-length encoding. With this extension, encoding of long "one" sequences becomes as efficient and better than arithmetic encoding.