**Meta Thrift API Acceleration using Java VectorAPIs**

------------------------------------------------------------

Thrift is using a variable length encoding where in a payload is stored in least significant 7 bits and MSB is used as a continuation bit. Thus a 32-bit integer will need 7 \* 4 + 1 = 5 byte of storage using following layout.

C|000EEEE C|DDDDDDD C|CCCCCCC C|BBBBBBB C|AAAAAAA

This is opposed to fixed length bit packing encodings used by DB storage formats like parquet. A fixed length encoding demands pre-processing to estimate max-min value needed to estimate the number of encoding bits, whereas a variable length encoding save additional preprocessing at the cost of meta data bit[s].

|  |  |
| --- | --- |
| Bits used | Number of bytes for encoding |
| 1-7 | 1 |
| 8-14 | 2 |
| 15-21 | 3 |
| 22-28 | 4 |
| 29-32 | 5 |

Decoding of 32-bit integer values: -

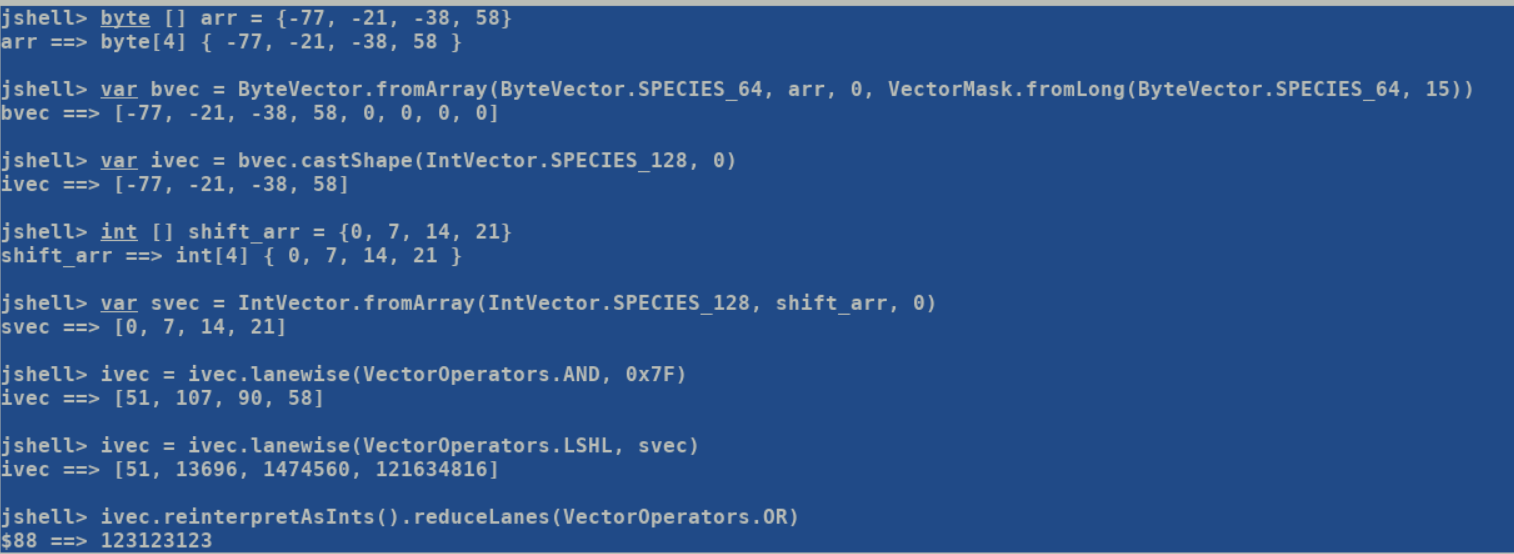
Input:



Steps for pre vector computation: -

1. Read 128bit inputs which accounts for 16 bytes of data.
2. Upcast byte lanes to integer lanes.
3. Compute mask using .LT. 0 predicate, a mask bit is set if continuation bit i.e. MSB of a byte is 1.
   * True count is equal to number of bytes read, skip step d – g.
4. Flip the computed mask, this shall provide us with total number of encoded integral values.
5. Use VectoMask.firstTrue to compute the hop window b/w consecutive set bits.
6. Use predicated lanewise AND, LSHL operations to influence only the lanes of interest.
7. Use predicated reduction OR operation to compute the decoded integer value.

APIs for vector computations (step f and g): -



Encoding of 32-bit integer values: -

Steps:

1. Read 256bit input which accounts for 8 integer values. Since each integer may consume maximum 5 bytes of encoding storage, hence we may need to upcast 32-bit lane to a 64-bit value.
2. Estimate maximum number of encode bits by performing vector comparison starting with 0x0FFFFFFF, 0x1FFFFF, 0x3FFF and 0x7F. Idea here is to find the maximum number of bits needed to encode lane holding largest value. For all the other lanes some of the vector operations will be redundant and will be suppressed using predicate mask.
3. Re-assemble the integral lanes into constituent byte encodings.
4. Reinterpret the integral lanes as byte lanes and pack the bytes of interest from each original lane using byte vector compression API.