CERN, SCCS Technische Universität München

Fast and Efficient Entropy Compression of ALICE Data using ANS Coding

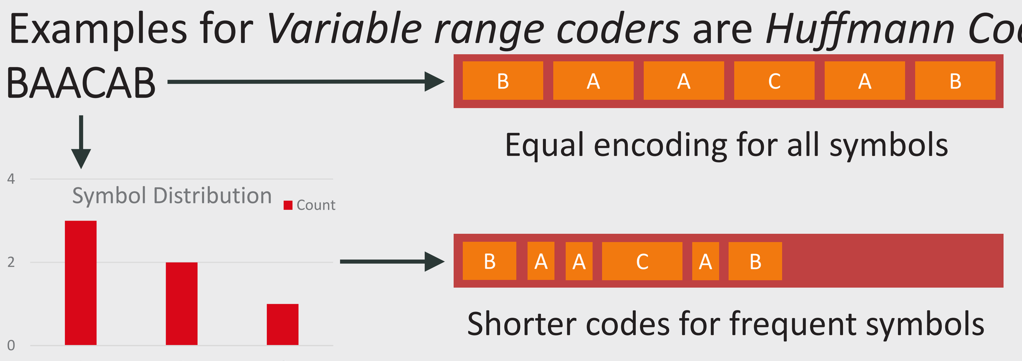
Michael Lettrich for the ALICE collaboration, michael.lettrich@cern.ch

Objective

- Provide state of the art lossles entropy compression in the ALICE O^2 framework that minimizes permanent storage requirements of detector data in LHC run 3.
- Preprocessed detector data are provided as *Compressed Time Frames* as flat, numeral arrays to be written out as a binary stream to permanent storage at a target data rate of ~90GB/s [1]
- Coder must efficienlty deal with source alphabets of up to 25 bits per symbol.
- If possible, exploit CPU and GPU resources of heterogenious ALICE O^2 farm.
- Maintain compatibility between data created by CPU and GPU.

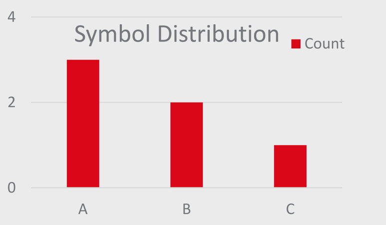
Asymetric Numeral Systems Coders

- *Variable range coders* use symbol distribution of source data for compression.
- Examples for *Variable range coders* are *Huffmann Coding* or *Arithmetic Coding.*



- ANS by Jarek Duda [2] adds new family of *variable range coders*.
- rANS as fast entropy coder for large alphabets based on arithmetic operations.
- Encodes symbols into single, infinite precision integer state in LiFo order.
- Decoder unwinds LiFo stack, reversing coder steps.

Message: BAACAB, length 6 $=: m$



$$b_s := \sum_{i<s} l_i$$

Symbol LUT	A	B	C
Count l_s	3	2	1
Cumulative b_s	0	3	5

rANS Coder $C(s,x) = m\lfloor x/l_s \rfloor + b_s + \mathrm{mod}(x, l_s)$

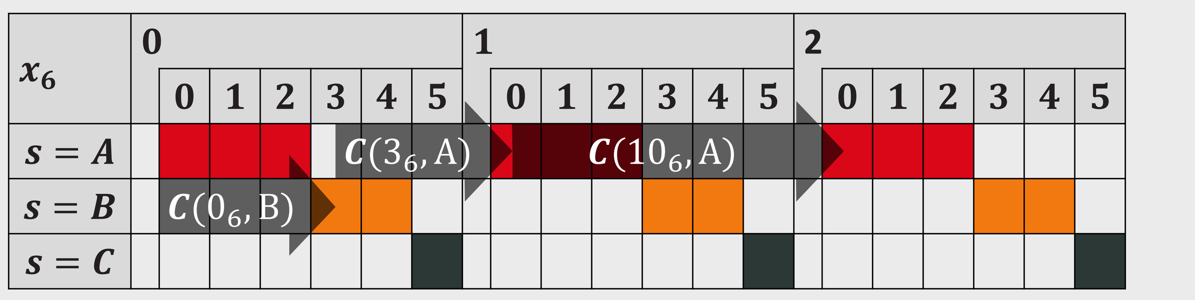
rANS Decoder $D(x) = (s, l_s\lfloor x/m \rfloor - b_s + \mathrm{mod}(x, m))$

$$D(C(s,x)) = (s,x)$$

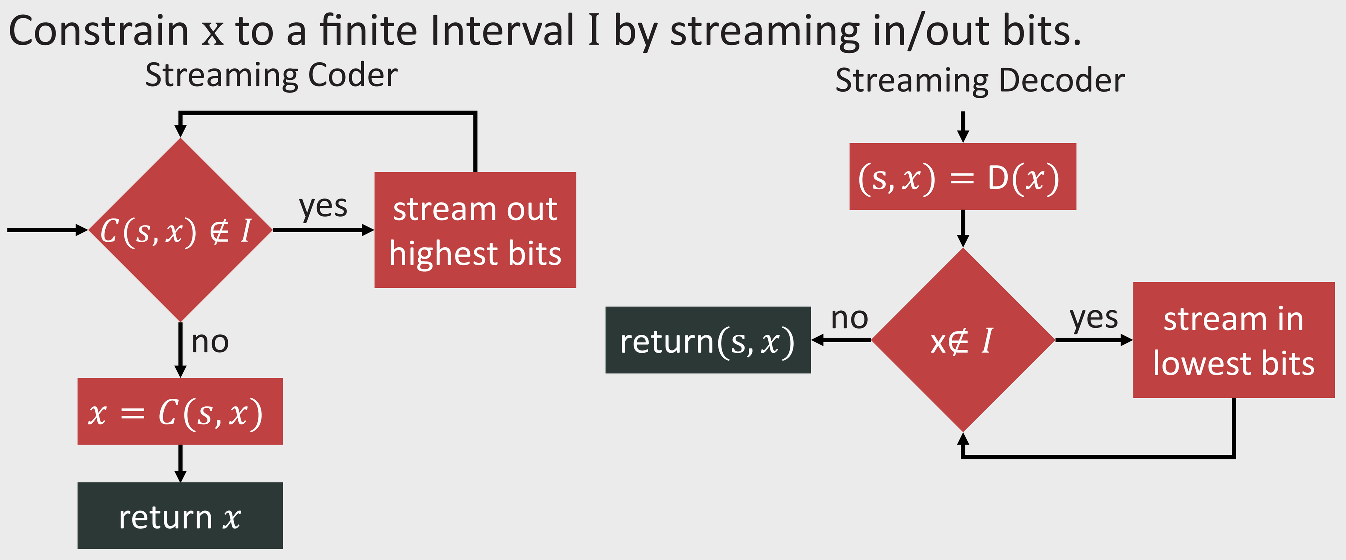
$C(s,x)$	$C(0,B)$	$C(3,A)$	$C(6,A)$	$C(12,C)$	$C(77,A)$	$C(152,B)$
x	3	6	12	77	152	459
x_6	3_6	10_6	20_6	205_6	412_6	2043_6

Direction of Coder → ← Direction of Decoder

- State interpretable as a cyclic, *asymetrized base-m numeral system.*



- State grows with probability of encoded symbol $x_{n+1} \approx x_n/\Pr[s]$.
- overcomes 1Bit/symbol limitation of Huffman, on pair with Arithmetic Coding.
- Constrain x to a finite Interval I by streaming in/out bits.



Performance Considerations

- Rescaling $m := 2^p$ allows `mult`, `div` be replaced by shifts and `mod` by bitops.
- Replace divisions by multiplication with precalculated inverse.
- rANS en-/decoding faster then Arithmetic Coding, slower then Huffman Coding.

rANS and Parallelism

- Coder and Decoder are exact inverse functions: $D(C(s,x))=(s,x)$.
- Allows to write incompressible bits or parallel coders on same stream, see [4].
- Still requires operations by $C(s,x)$ to remain exactly reversible by $D(s)$.

Conclusion

- rANS is a powerful entropy coding algorithm for large alphabets.
- Capable of handling close to optimal compression of all tested ALICE datasets.
- Fast encoding performance, currently lacks in decoding with large alphabets.

Outlook

- Improve code robustness and decoding speeds by optimizing LUT.
- Finish Implementation of SIMD and GPU features.
- Integration into ALICE O^2.

Acknowledgments

Special thanks to Jarek Duda and Michael Bader for advise, feedback and fruitful discussions.

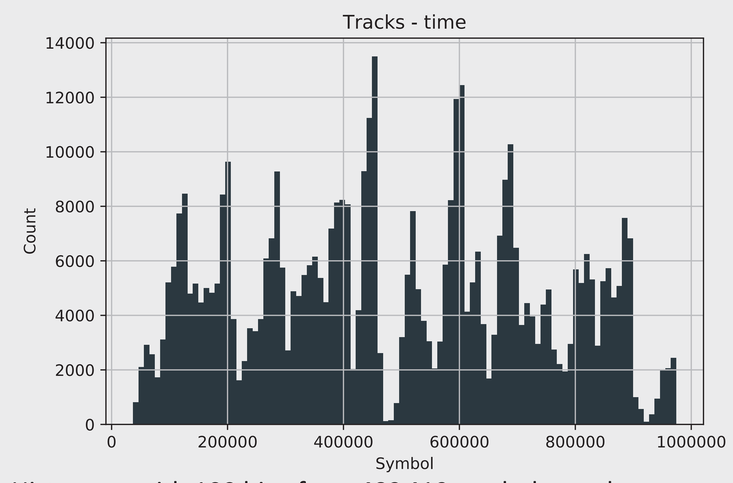
Case Study : Compressing ALICE TPC Data

ALICE Time Projection Chamber (TPC) Data

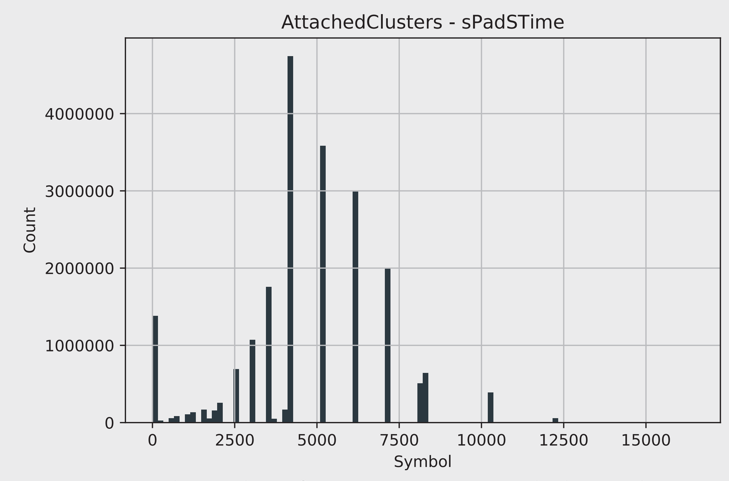
- Alice TPC responsible for >90% of data in ALICE timeframe for LHC run 3 [1].

Sample Source Data

- Flat structure of arrays, interpretable as four tables with multiple columns.
- Source symbol value range of 8-25 Bits unsigned integers per column.
- 130 simulated Pb-Pb collisions provide $O(10^7)$ symbols for good statics.
- Non standard, patchy distribution of source symbols.



Histogram with 100 bins from 482 419 symbol samples. Found 344'987 unique values in 24 Bit symbol range. Entropy of sample data is 18.24 Bit

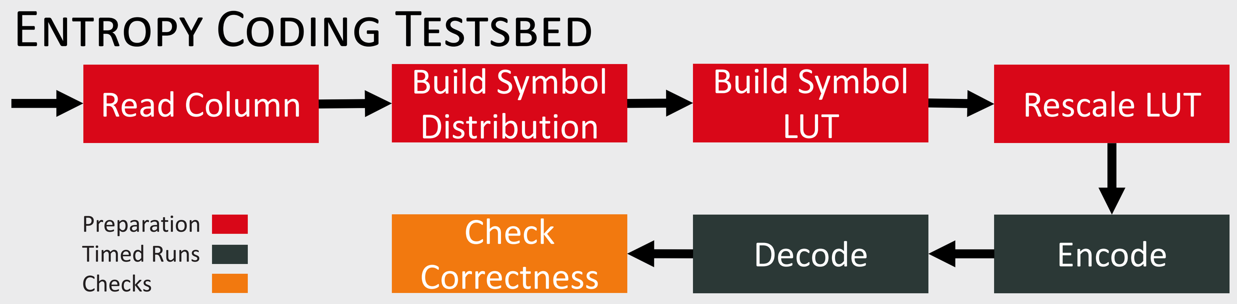


Histogram with 100 bins from 21 072 849 symbol samples. Found 419 unique values in 16 Bit symbol range. Entropy of sample data is 6.06 Bit

- Cross checks with detector data from LHC run 2 show high similarity.
- Assume static distribution of source symbols within each column.
- Calculation of entropy (H) as lower bound for compression.
- Concatenation of symbols in correlated columns to reduce overall entropy.

Table	Rows	Cols	Bits/row	H [Bits/row]	H concat [Bits/row]
AttachedClusters	21 072 849	4	41	17.15	15.75
AttachedClustersReduced	20 590 430	4	55	17.60	17.59
Tracks	482 419	5	73	53.90	53.90
UnattachedClusters	50 745 911	5	81	39.77	38.37

Entropy Coding Testbed



Only raw (de)coding speed is measured. LUTs are staticaly kept in memory for production system.

Coded data is decoded afterwards and compared to source data for correctnes.

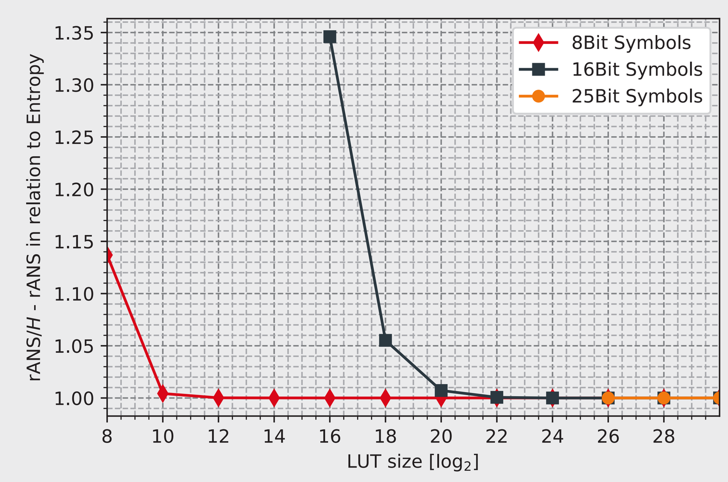
- C++ rANS code based on reengineered *ryg_rans* [3] coder/decoder.
- Bandwidth: Single thread, (`Bits/row×Rows`) `/time`, avg. over 5 runs.

Attainable Level of Entropy Compression

Table	Bits/row	H [Bits/row]	rANS [Bits/row]	rANS/H	BW [MiB/s]
AttachedClusters	41	15.75	15.75	1.00	649.72
AttachedClustersReduced	55	17.59	17.64	1.00	776.06
Tracks	73	53.90	53.90	1.00	365.56
UnattachedClusters	81	38.37	38.37	1.00	589.80

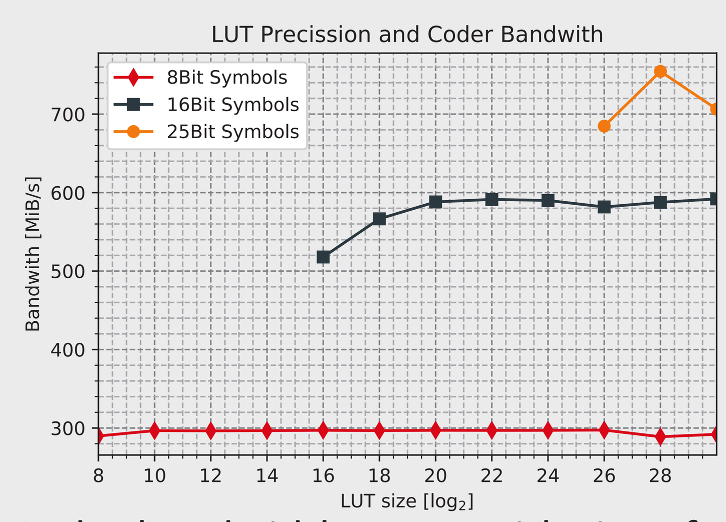
- rANS coder is close to entropy for all tables at high encoding speeds.

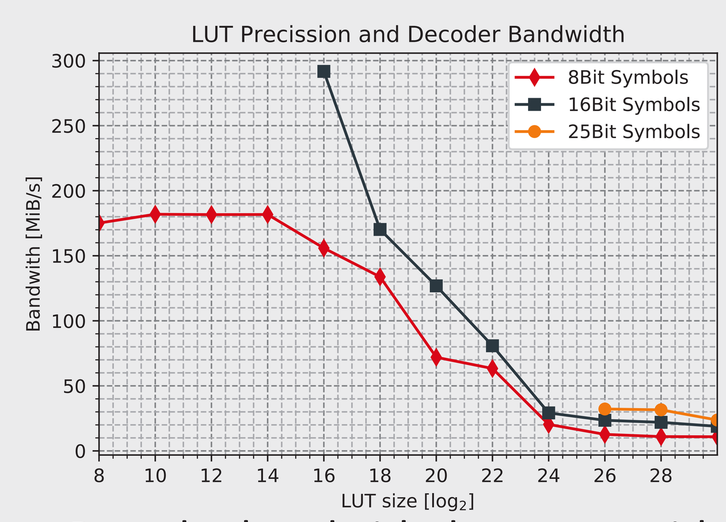
Compression and LUT Precision



- Rescaling LUT to 2^p for perfromance.
- LUT must map source distribution well enough for good compression.
- Trivialy: |LUT|>|SourceAlphabet|, i.e larger LUTs for larger alphabets.
- Must be preserved for decoding.
- Additional Inverse LUT (iLUT) for decoder |iLUT|≥|LUT|, constructed from LUT.
- For large source alphabets only sensible for static distributions, where LUTs can be reused or for extremely large datasets.
- Requires algorithmic improvements to decrease memory requirements.

Bandwidth Measurements





- Coder bandwidth grows with size of source alphabet.
- Same operations executed on larger input data (16Bit ~2x faster then 8Bit).
- Profits from sparse distributions.

- Decoder bandwith decreases with increasing LUT size.
- Performance bound by lookups in iLUT that does not fit into cache.

References:
[1] Technical Design Report for the Upgrade of the Online-Offline Computing System, CERN-LHCC-2015-006, 04/2015
[2] Jarek Duda, Asymmetric numeral systems: entropy coding combining speed of Huffman coding with compression rate of arithmetic coding, arXiv:1311.2540, 11/2013
[3] Fabian Giesen, https://github.com/rygorous/ryg_rans, last checked 10/2019
[4] Fabian Giesen, Interleaved entropy coders, arXiv:1402.3392, 02/2014