Fast Entropy Coding for ALICE Data Compression in Run 3

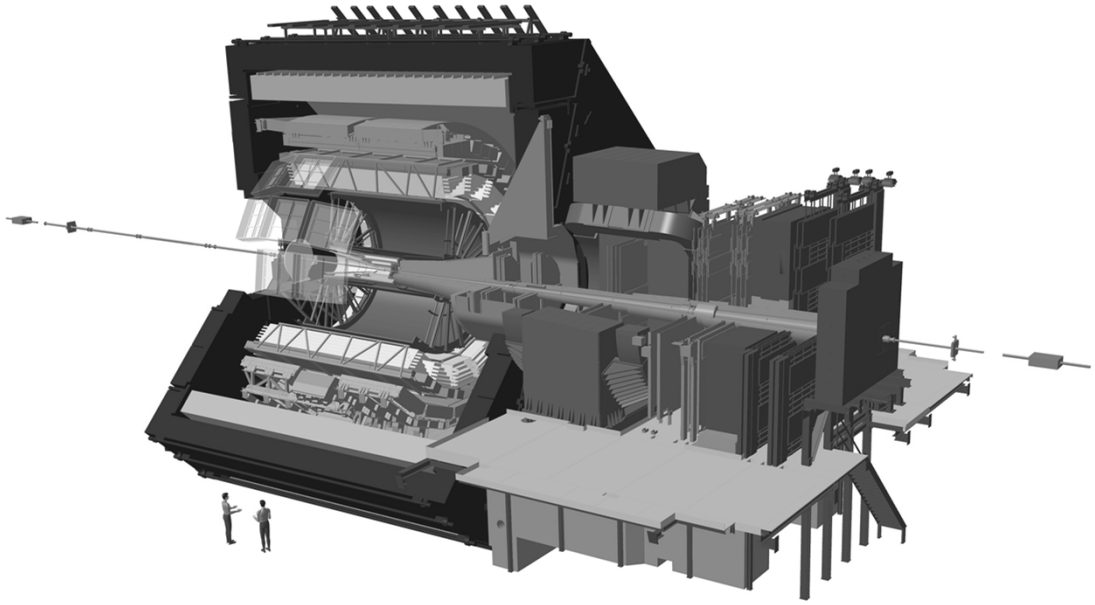
Michael Lettrich (CERN, Technische Universität München) for the ALICE collaboration

30.07.2020

Overview

- Context – ALICE Data Reduction in LHC Run 3
- Entropy Coding for ALICE Run 3
  - Properties of Source Data
  - Encoding Strategy
  - Entropy Coding with rANS
  - Extensions to rANS for ALICE Data
- Conclusion and Outlook

A Large Ion Collider Experiment

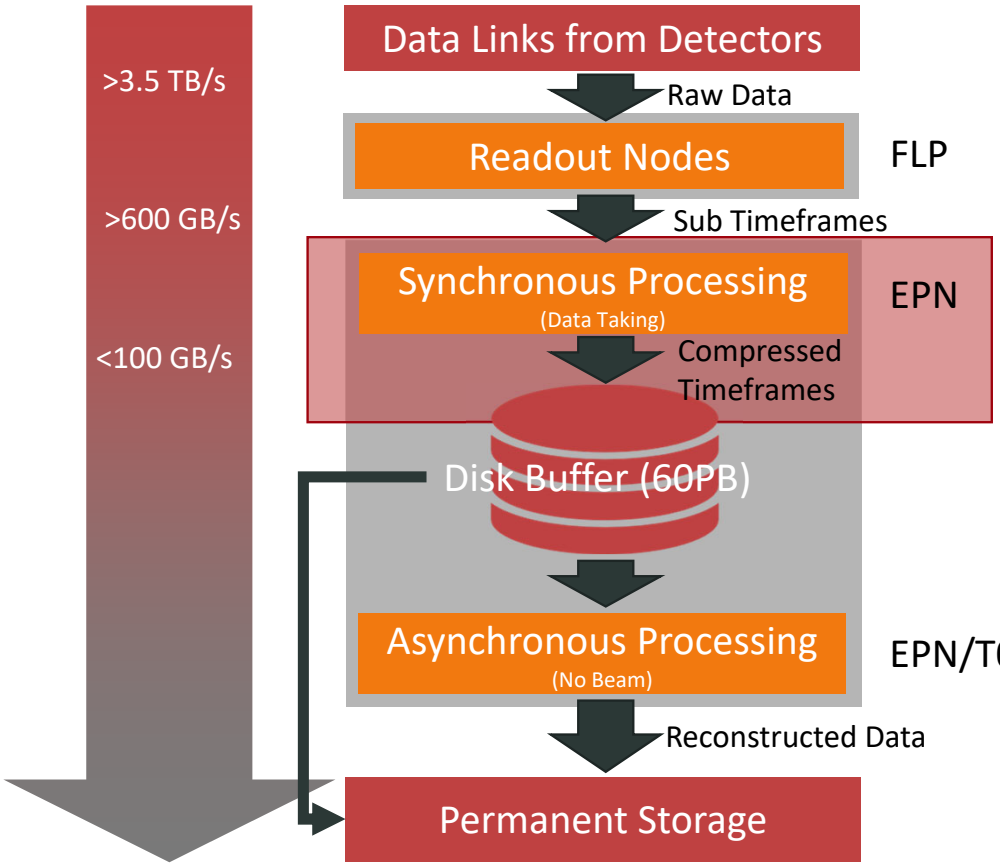


- Heavy-ion experiment at the LHC
- Goal: study properties of strongly interacting matter at extreme energy densities
- Precision tracking even at high multiplicities

For LHC Run 3: 50 kHz continuous readout

See C. Zampolli: ALICE in Run 3 and Run 4

Compression for ALICE Run 3



- Continuous Readout at 3.5 TB/s
- Processed as timeframes of 10-20ms
- Two stage synchronous-asynchronous data processing
- 35x data reduction during synchronous processing
- Entropy coding as last step in online processing

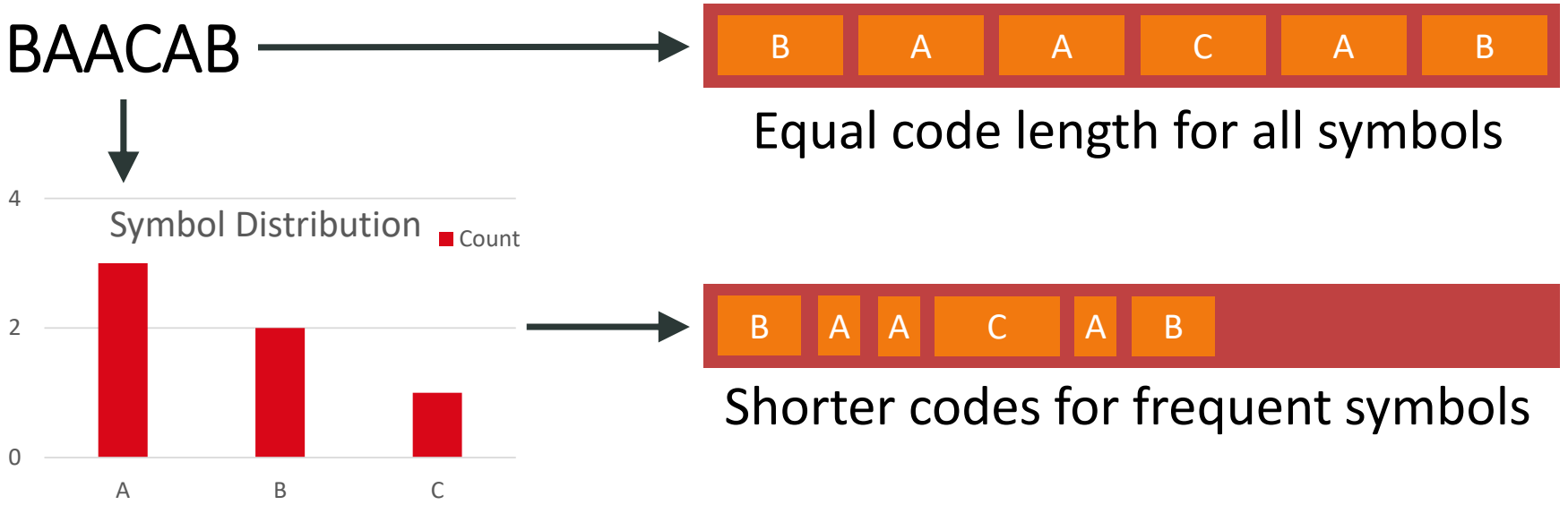
See C. Zampolli: ALICE in Run 3 and Run 4
See M. Concas: GPU based online-offline reconstruction

Objective of Entropy Coding

Lossless data compression by factor ~3

Time limit 35 s

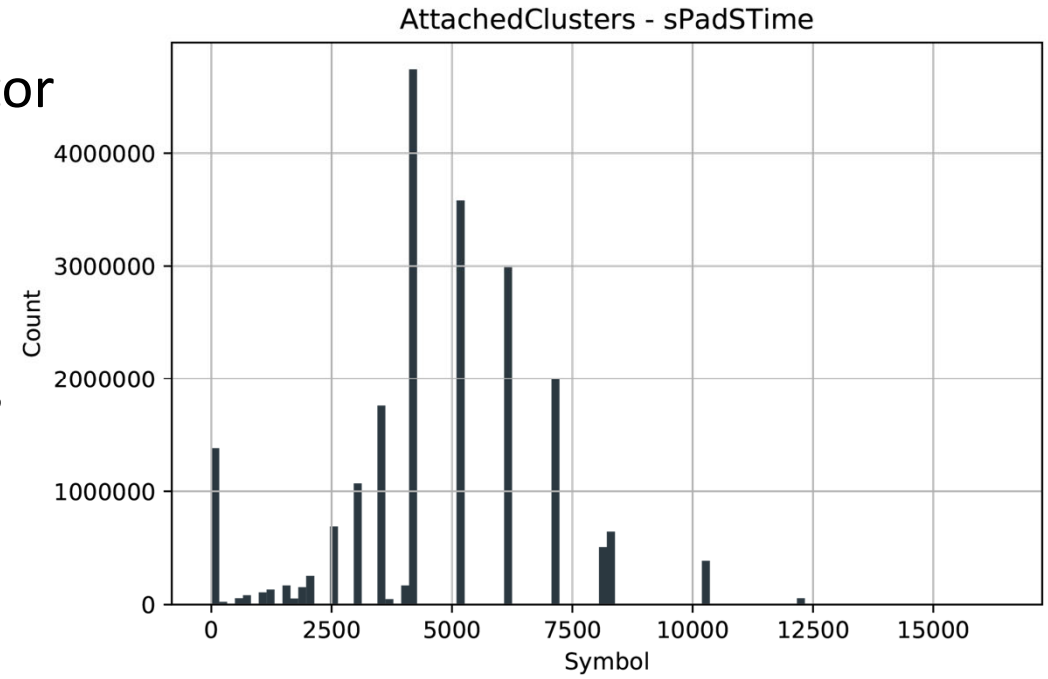
Entropy Coding Basics



- Lossless compression by more efficient representation of source data based on their frequencies
- Entropy (H) as lower limit

Properties of Source Data

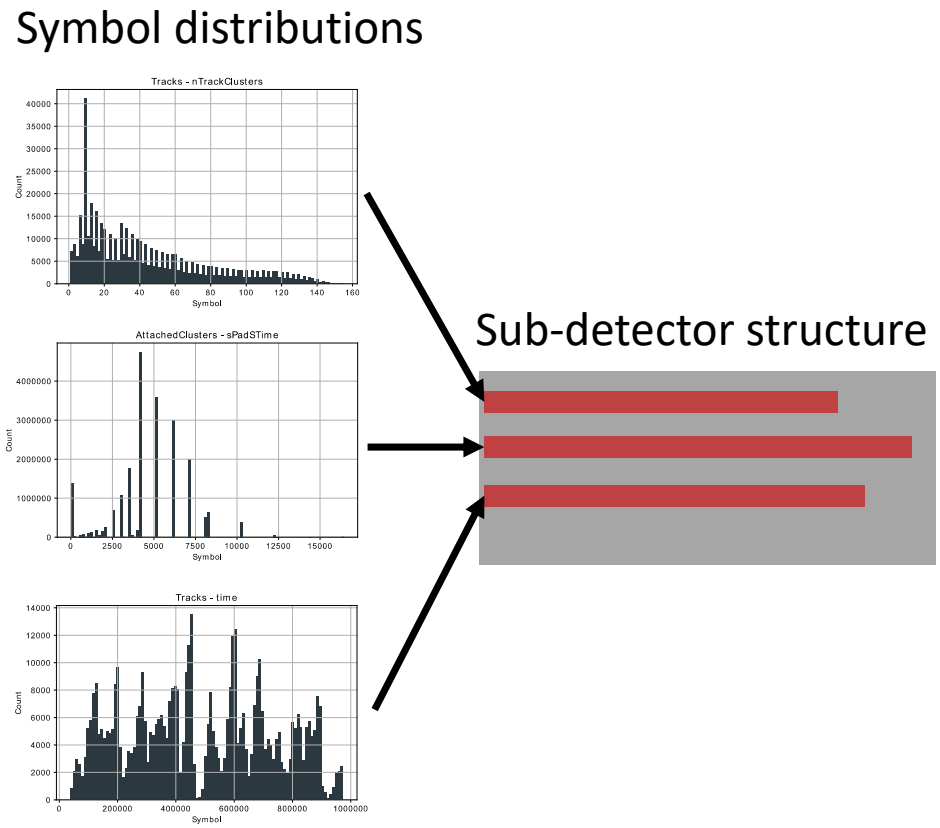
- Flat structure of arrays per sub-detector as result of data reduction
- Data volume dominated by TPC
- Data range up to 25 Bits/value
- Sparse, clustered symbol distributions
- Distributions stay static [3]
- Some data with repeating symbols



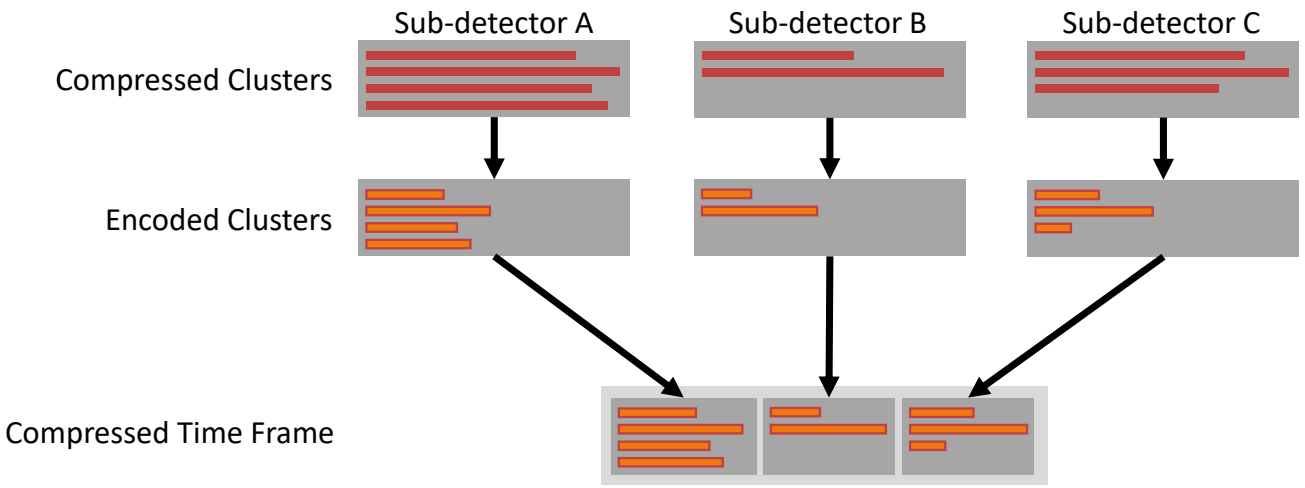
Entropy Coding in ALICE

- Individual treatment of each array in flat structure
- Use distribution of source data

- Better Compression
- Independent, parallel execution



Entropy Coding in ALICE



- Dedicated encoding process per sub-detector
- Merge into compressed time frame

Asymetric Numeral Systems Coders [1,2]

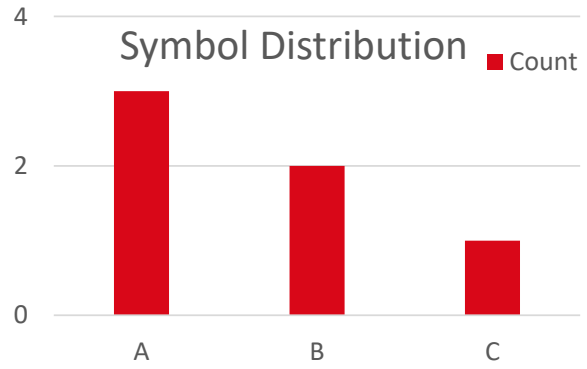
- Encode all symbols of message into single integer state $x \in \mathbb{N}$
- Based on arithmetic operations

Example $C(\text{BAACAB}) = 459$

$C(s,x)$	$C(0,B)$	$C(3,A)$	$C(6,A)$	$C(12,C)$	$C(77,A)$	$C(152,B)$
x	3	6	12	77	152	459

Direction of Coder

Direction of Decoder



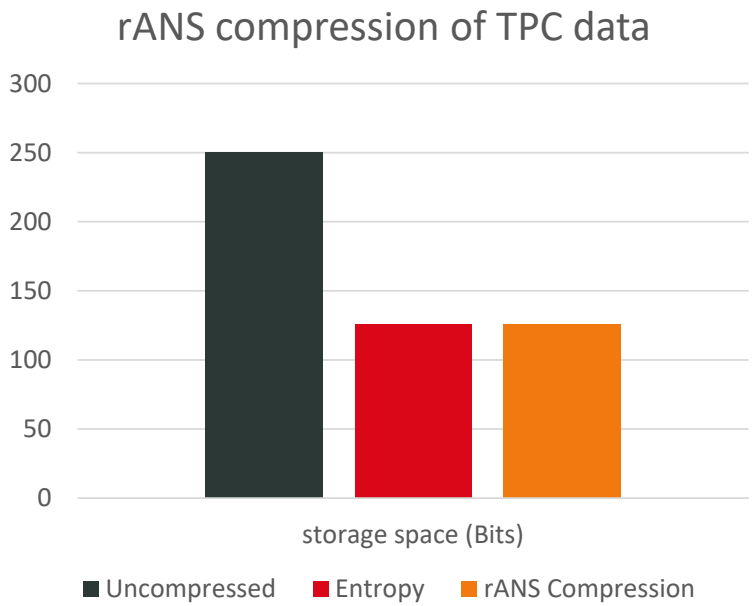
- Encoder grows x inversely proportional to symbol probability:

$$x_n \approx {x_{n-1}}/{P[s_n]}$$

- Decoder is inverse function of encoder: $D\big(C(s,x)\big) = (s,x)$

Performance of rANS 4

- rANS compresses close to entropy H at high bandwidths



Compression ratio	1.99
Efficiency (Compression/Entropy)	1.00
Bandwidth (avg)	595.28 MiB/s

Bandwidth: Single Thread, avg over 5 runs on Core i7 8700

Compressed TPC data from 130 Pb-Pb events from MC simulation with $O(10^7)$ samples

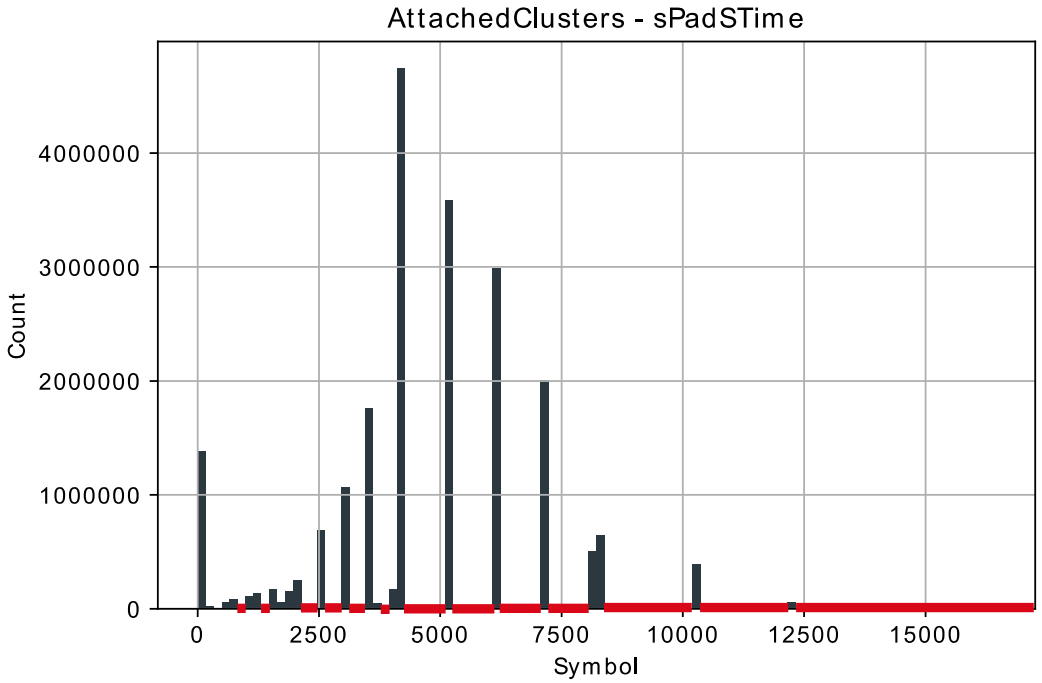
Adapting rANS for ALICE

Adaptations of rANS needed for:

- Data range up to 25 Bits/value
- Sparse, clustered symbol distributions
- Static distribution of source symbols across timeframes
- Repeating symbols

Adapting rANS for ALICE

- Optimized datastructures for symbol tables of large, sparse alphabets
- Static symbol tables across time frames with treatment of rare symbols



only 439 out of 2^{16} values are used, but all are possible

Adapting rANS for ALICE

Space efficient treatment of duplicates in source data

Example:

ABAABCBABCCCC = 1**A**1**B**2**A**1**B**1**C**1**B**1**A**4**C** (classical run length encoding)

ABAA**BCBABCCCC** = **ABAABCBAB**4**C**

Calculate if encoding is more efficient than entry in a lookup table

1. No duplicates: encode normally
2. Duplicates but more efficient to encode normally
3. Duplicates and more efficient to remove them

Current Status

- Core rANS implementation with adaptations
- Flexible configuration of entropy coder according to source data
- Integration of subdetectors TPC, ITS, MFT, FT0 (>95% of data volume)

Conclusion and Outlook

- rANS for fast, efficient and flexible entropy coding in ALICE Run 3

- Outlook
- Integration of remaining detectors
- Finalize data format
- Integrate static distributions in **C**ondition and **C**alibration **D**ata **B**ase (CCDB)
- Improve integration of components
- Benchmarking and performance tuning on system scale

Thank you for your attention

Questions?

Sources

1. J. Duda, Asymmetric numeral systems: entropy coding combining speed of huffman coding with compression rate of arithmetic coding (2013), 1311.2540
2. J. Gibbons, Coding with Asymmetric Numeral Systems, (2019) 10.1007/978-3-030-33636-3_16
3. J. Berger, U. Frankenfeld, V. Lindenstruth, P. Plamper, D. Röhrich, E. Schäfer, M. W. Schulz, T. M. Steinbeck, R. Stock, K. Sulimma, A. Vestbø, A. Wiebalck, TPC data compression, (2002), 10.1016/S0168-9002(02)00792-1
4. M. Lettrich, Fast and Efficient Entropy Compression of ALICE Data using ANS Coding, CHEP 2019