

ROOT Compression with ZSTD

GSoC project “Novel Applications of Zstandard (ZSTD) compression algorithm to ROOT”

*Alfonso Luis Castaño Marín, Universidad de Murcia,
Spain*

Mentors: Brian Bockelman, Oksana Shadura

ROOT

Due to the huge amount of data processed at CERN, compression is fundamental.

Depending on the case we will care more about compression or speed.



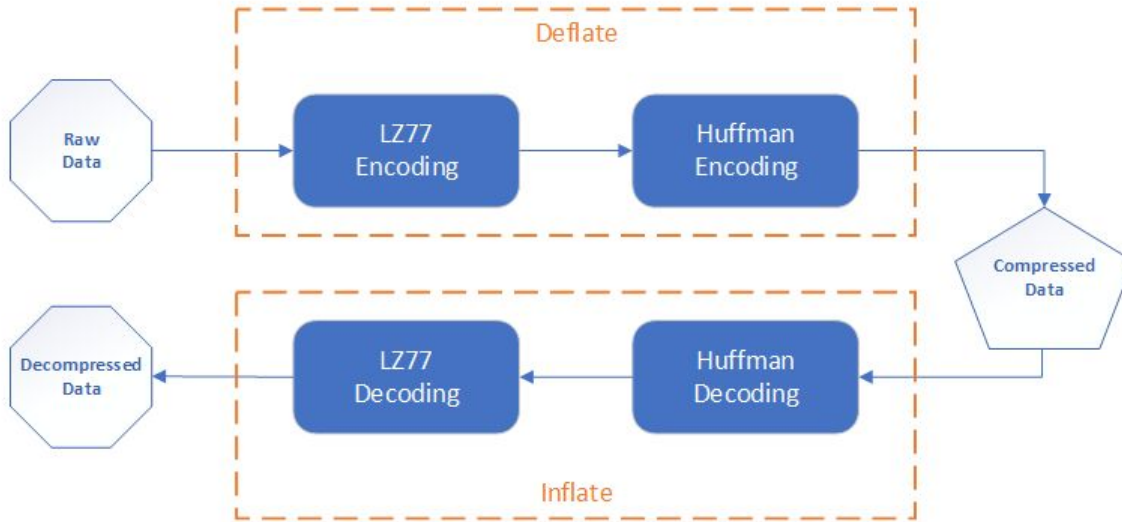
ZSTD

Promising alternative for cases where a balance between compression and speed is required. <https://github.com/facebook/zstd>

What ZSTD promises:

- Better than ZLIB in all metrics: compression speed, decompression speed, and compression ratio.
- Decompression speed should be constant regardless of compression level.
- High dynamic range in tradeoff between compression speed and ratio.
- Does not achieve compression ratio of LZMA, neither speed of LZ4.

ZLIB: LZ77 + Huffman Encoding



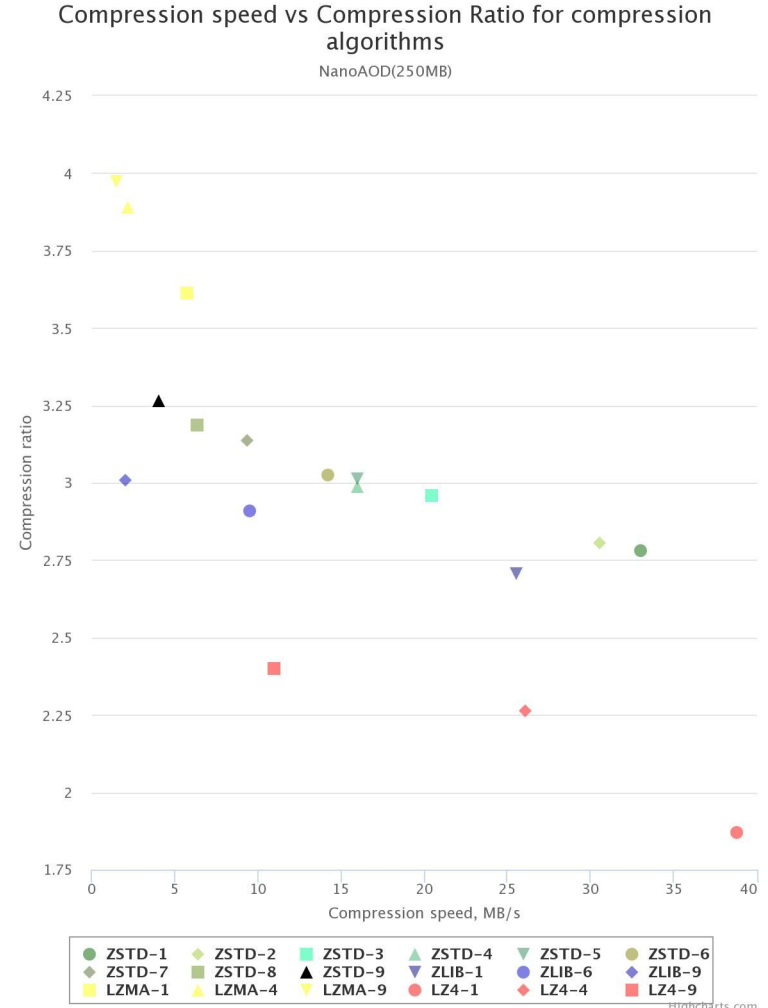
ZSTD

ZSTD follows the idea of ZLIB but introduces new techniques and improvements:

- Finite State Entropy (Arithmetic encoding)
- Dictionary Training
- Bigger Window Size
- Branchless design style
- Parallel decoding in single core
- And many more

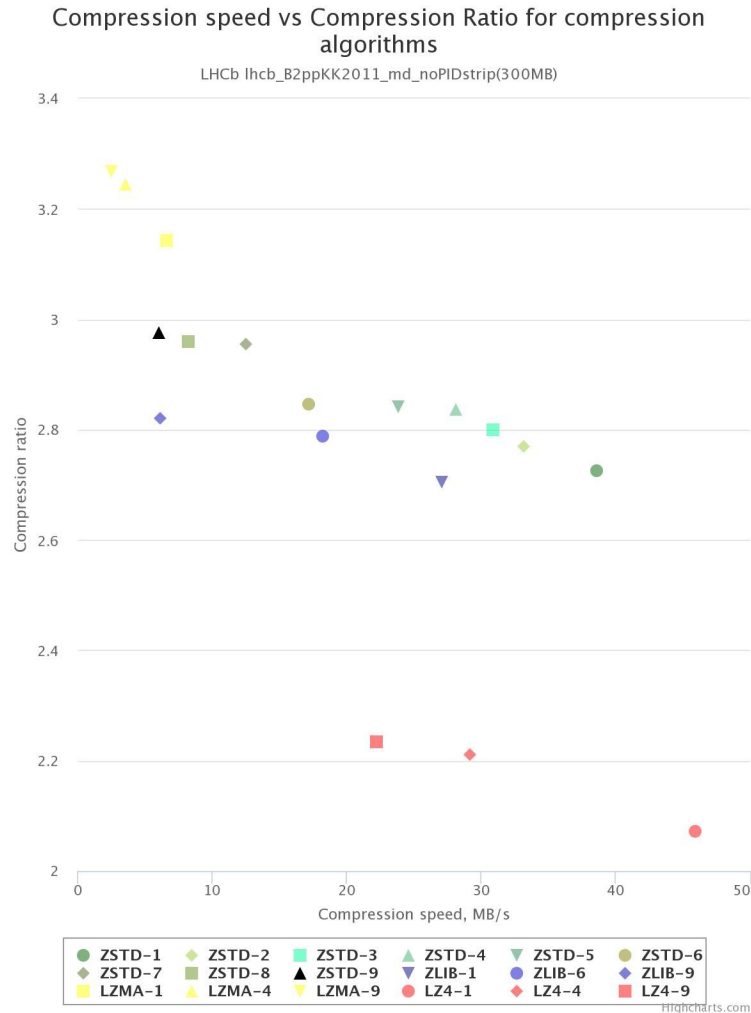
Results: CMS

File from 2017, probably different schema than current NanoAOD, currently testing in newer file (3GB).

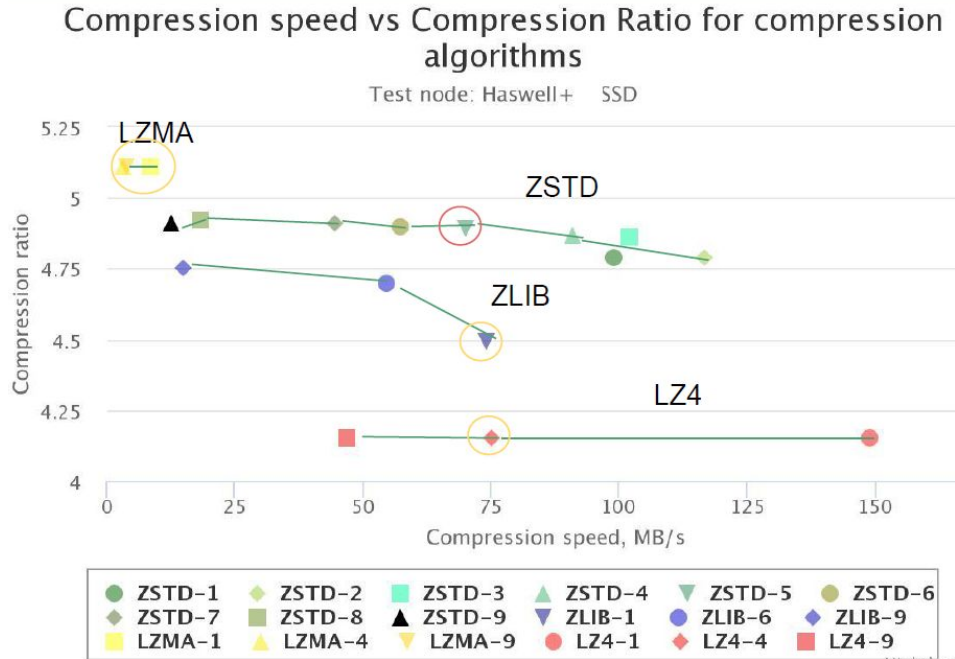
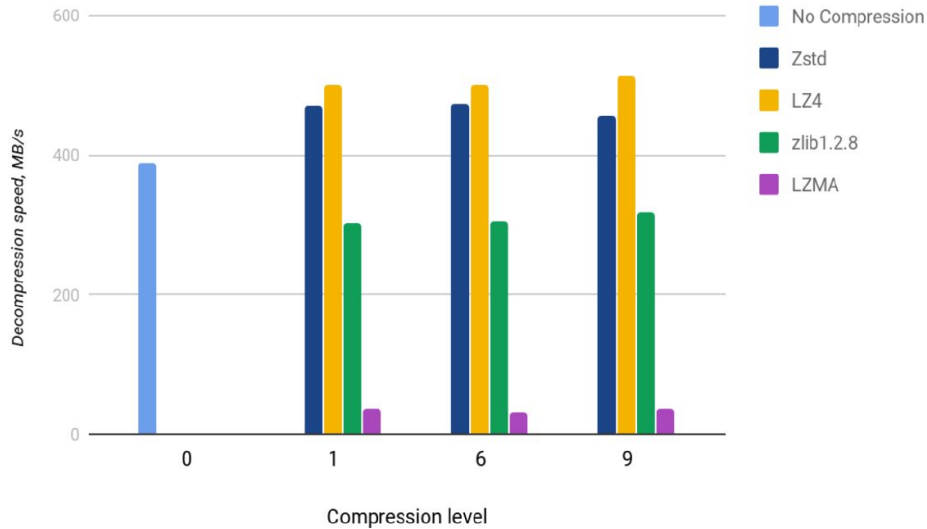


Results: LHCB

Uncommon branch distribution, several trees with few baskets.



Results: 2000Events



Current status of ZSTD: Ready to merge

PR 3947: Integrate ZSTD in ROOT github.com/root-project/root/pull/3947/

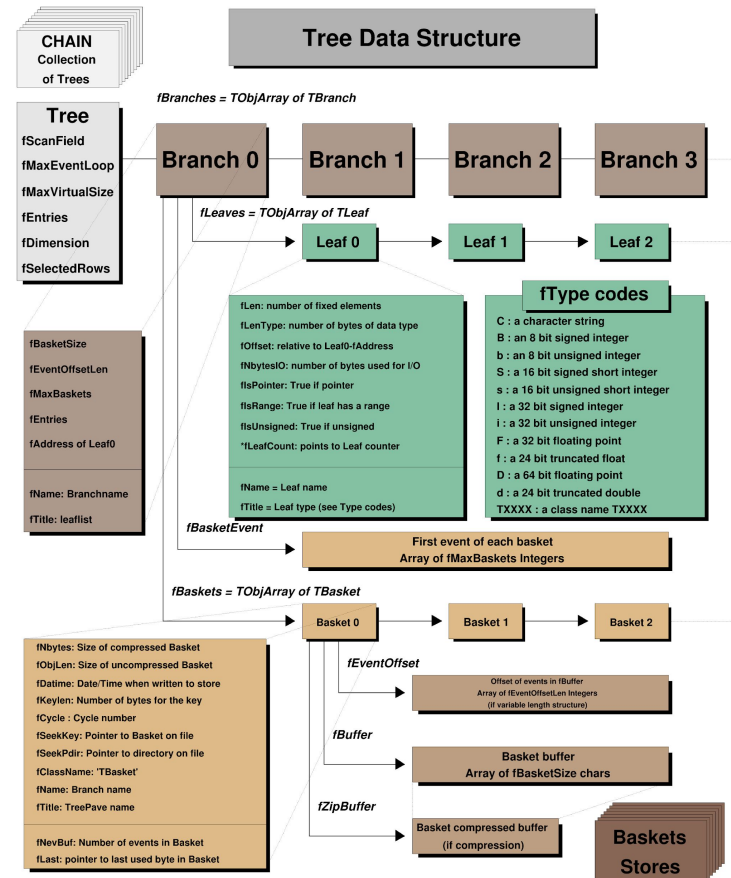
PR 352 : Integrate ZSTD in Roottest github.com/root-project/roottest/pull/352

ZSTD: Advanced Compression

- The advanced API of ZSTD allows to develop novel optimized solutions that have never been tried before: Compression Engine and Dictionaries Reusing.
- A Compression Engine is a class that will process all compressions requests. This centralization provides multiple benefits like reusing resources, one-time initializations and find synergies between compressed chunks.
- The dictionaries that are generated for a given data can be reused for similar data, reducing significantly the overhead of the dictionary size.

A little of background

- ROOT files have mostly Trees and Tuples.
- Inside them we have branches.
- A branch usually stores values of the same variable.
- A branch is divided in memory buffers called baskets.
- Each basket is compressed separately!



Branch compression

Instead of this:



We have this:



Synergic compression

Since baskets within a branch hold similar data, using a common dictionary could be very efficient.

The creation of the dictionary can follow two different approaches: Dictionary Training or Dictionary Reutilization

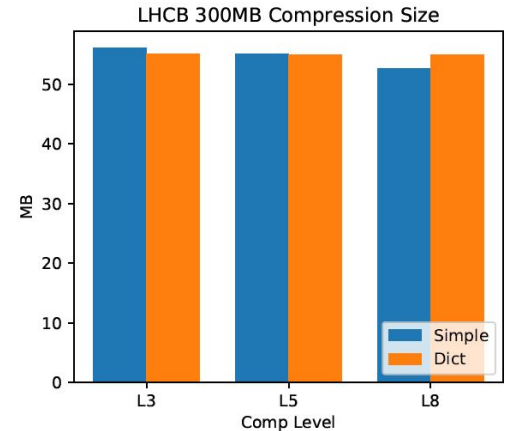
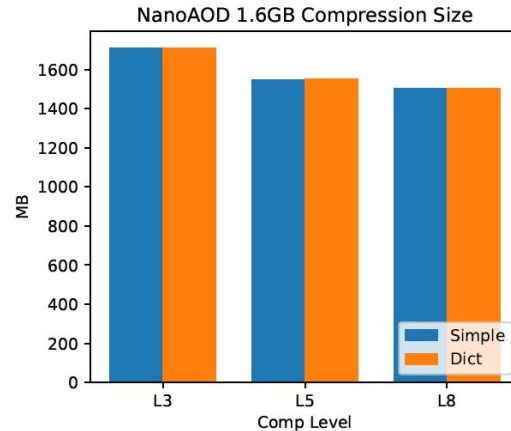
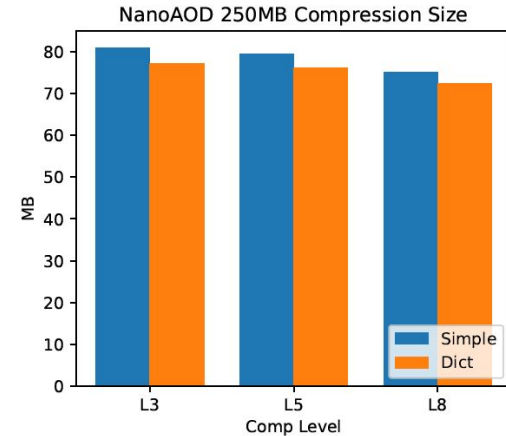
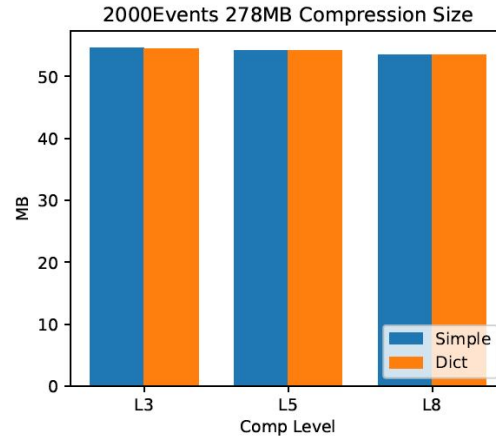
Dictionary Reutilization seems to offer the highest potential and flexibility for ROOT.

Trade-offs of Dictionary Reutilization

- **Reduce overhead of storing the Dictionary:** The amount of dictionaries that should be stored will drop from N (baskets) to 1. The improvement in the file size reduction will be determined by this.
- **Improve compression speed:** Dictionary will be generated only once, it could speed up the process.
- **Worse compression ratio if first basket is not representative:** We don't have any more customized dictionaries per basket.

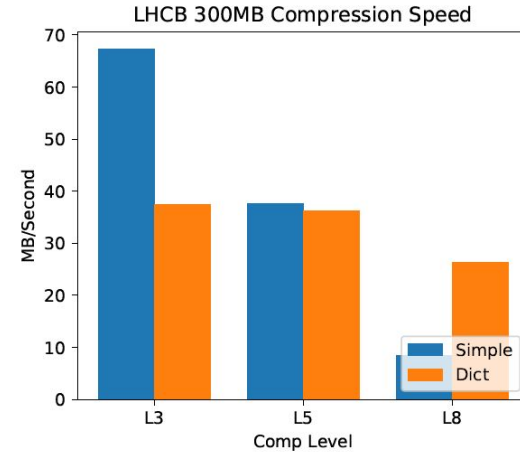
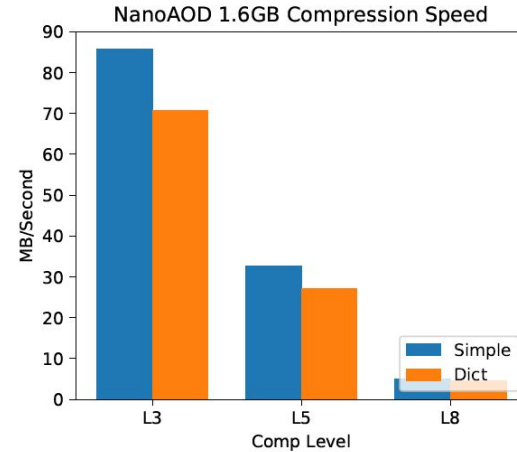
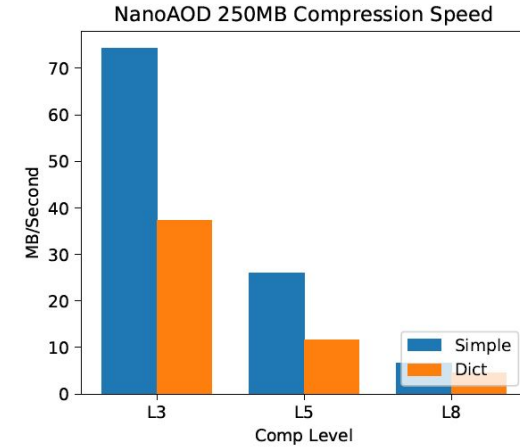
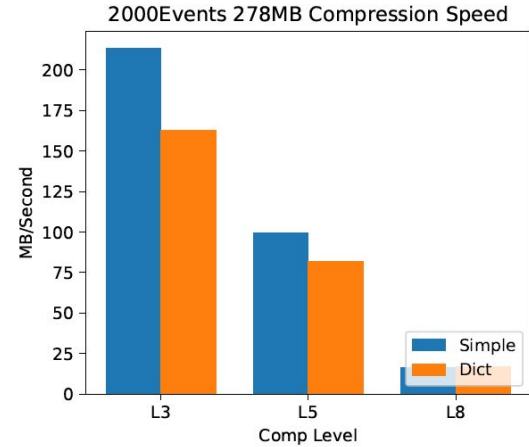
Results

- Same or slightly better compression ratios thanks to reduction of dictionary overhead.
- Highly dependent on the structure of the Tree.



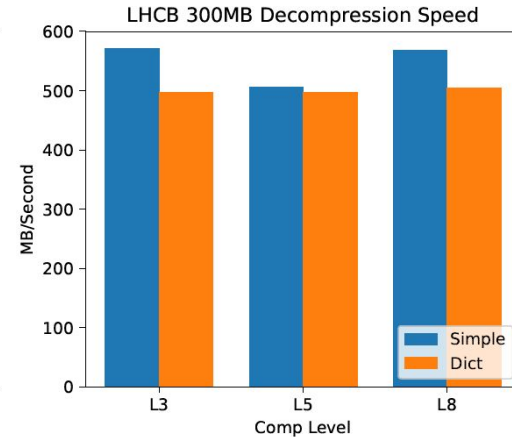
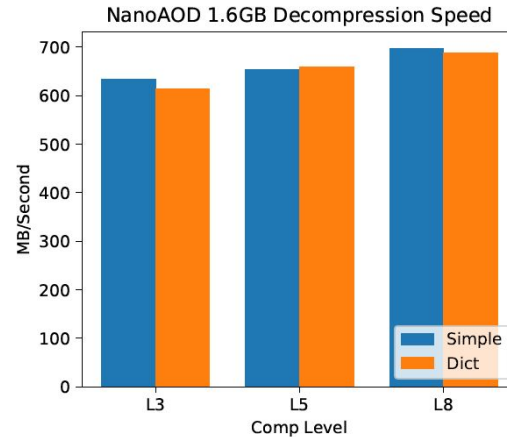
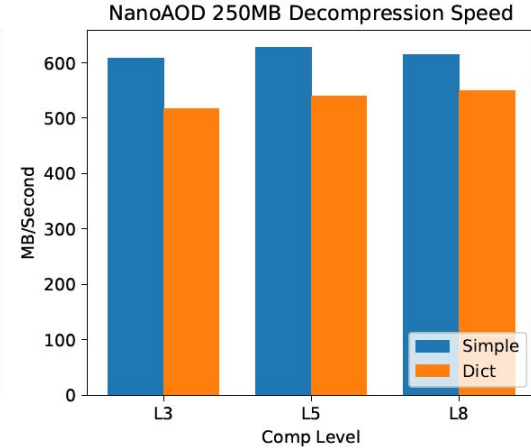
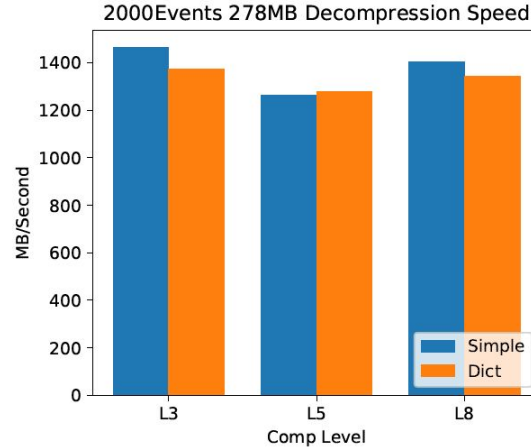
Results

- Regression in compression speed in some scenarios.
- Requires further investigation, there should be potential to improve speed over baseline.



Results

- More uniform results than in compression.
- Decompression speed is mostly determined by file size in ZSTD.
- Hence, limited room for improvement over the baseline without reducing the file size.



Conclusions

ZSTD: Fundamental Compression

- Outperforms in all metrics ROOT's default compression algorithm.
- Fully tested, integrated in ROOT and ready to merge.
- Still novel algorithm, new improvements will keep appearing.
- Unlocks the use of advanced compression dictionaries across ROOT projects.

ZSTD: Advanced Compression

- Infinite new possibilities to find synergies across data structures.
- Already obtaining compression improvements in certain scenarios.
- Deep investigation in process to understand in detail the structure of the data generated by main CERN experiments.