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 that ROOT I/O deals with, compression is fundamental.

However, depending on the case we will care more about compression or speed.

ZSTD

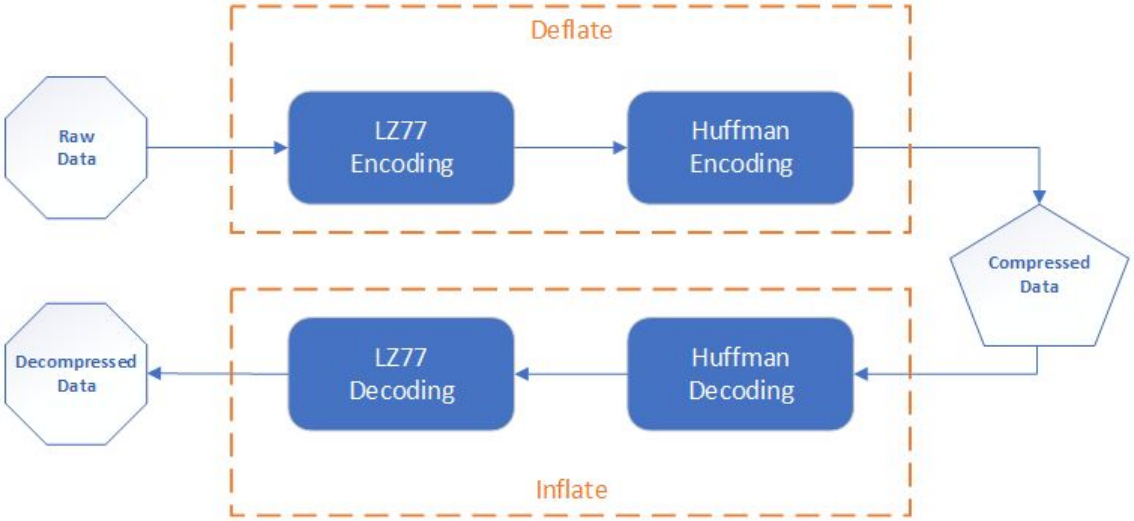
Promising alternative for cases where a balance between compression and speed is required. https://github.com/facebook/zstd

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.

But is not ZLIB already doing that?

ZLIB: LZ77 + Huffman Encoding

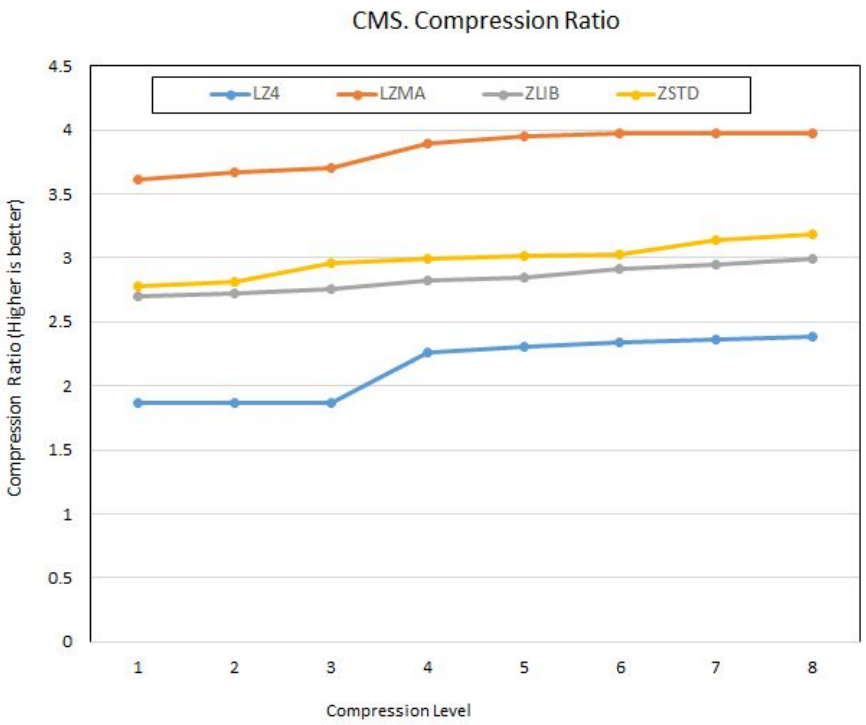


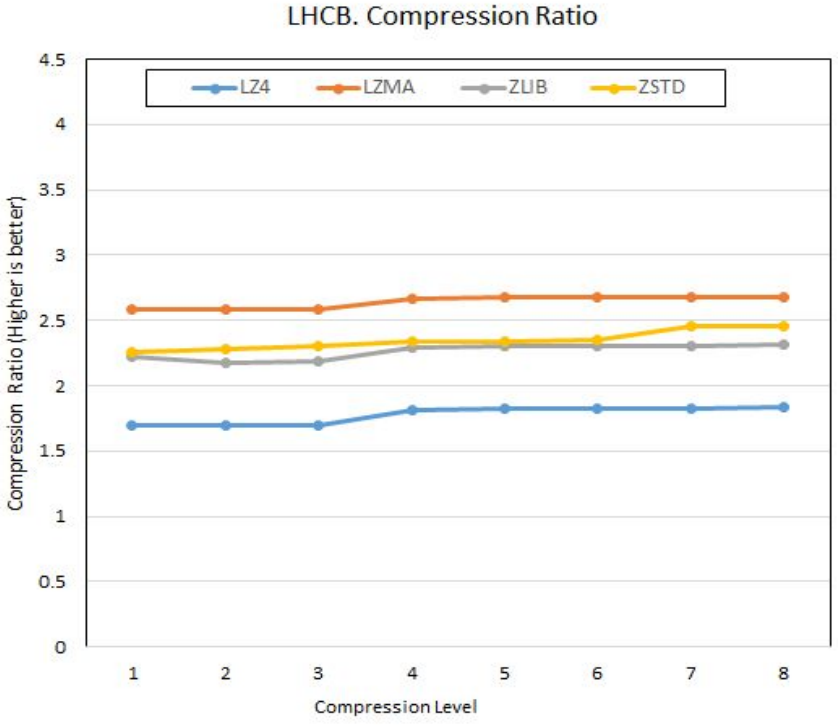
ZSTD

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

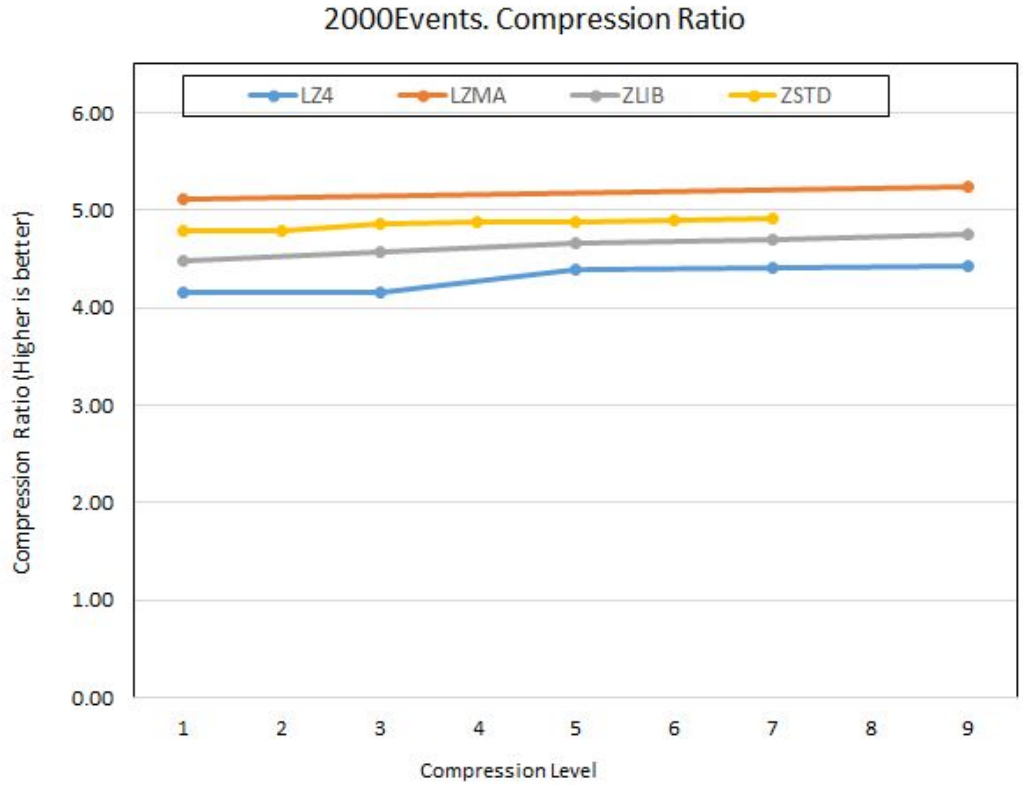
- Finite State Entropy (Arithmetic encoding)
- Dictionary Training
- Bigger Window Size
- Branchless design style
- Parallel decoding in single core
- And many more

Results

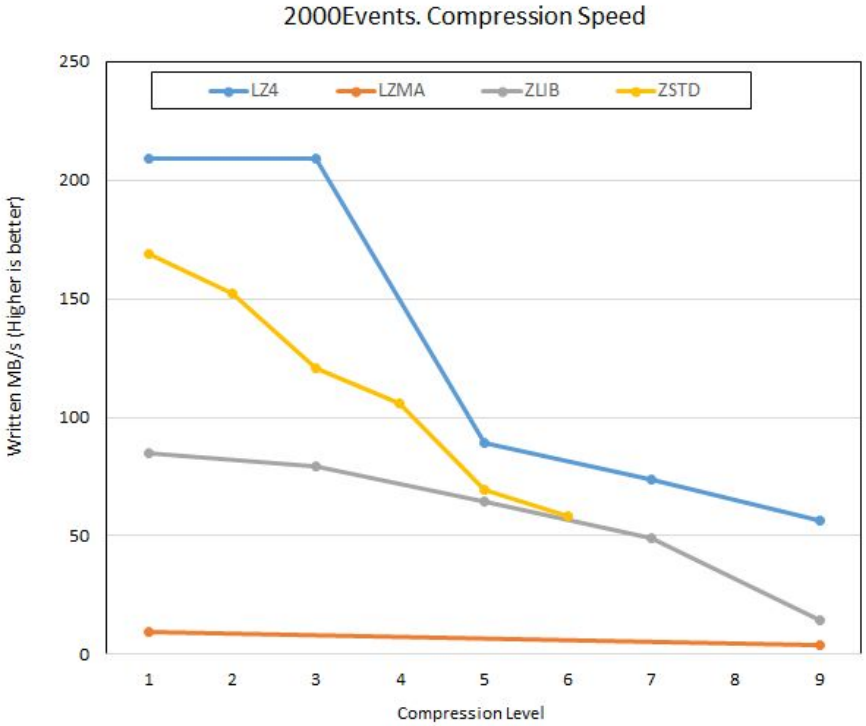

CMS. Compression Ratio
LZ4
LZMA
ZLIB
ZSTD
Compression Ratio (Higher is better)
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
1
2
3
4
5
6
7
8
Compression Level

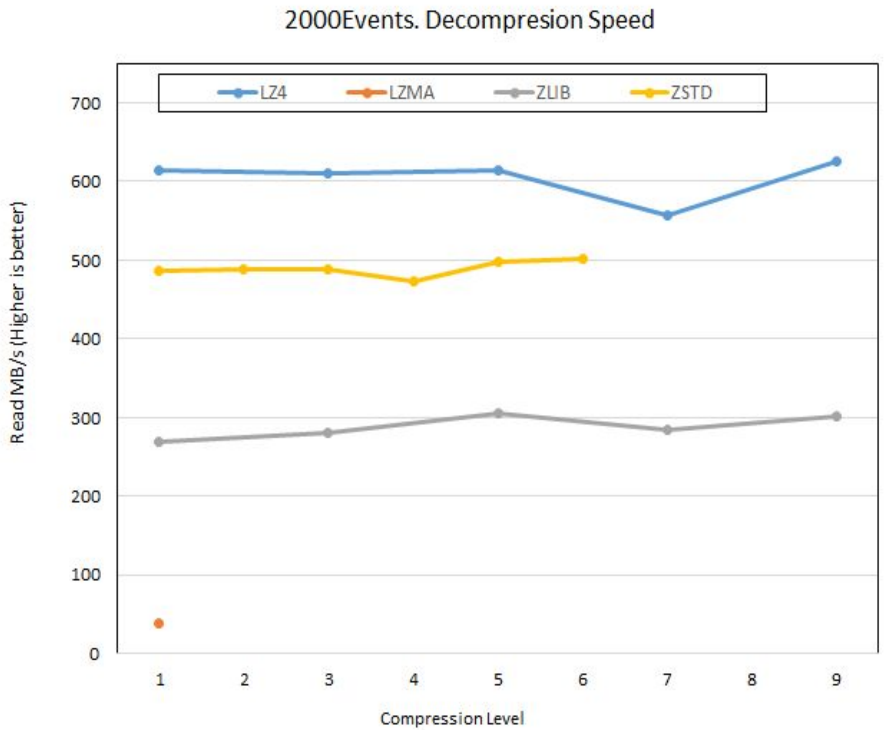

LHCB. Compression Ratio
LZ4
LZMA
ZLIB
ZSTD
Compression Ratio (Higher is better)
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
1
2
3
4
5
6
7
8
Compression Level

Results

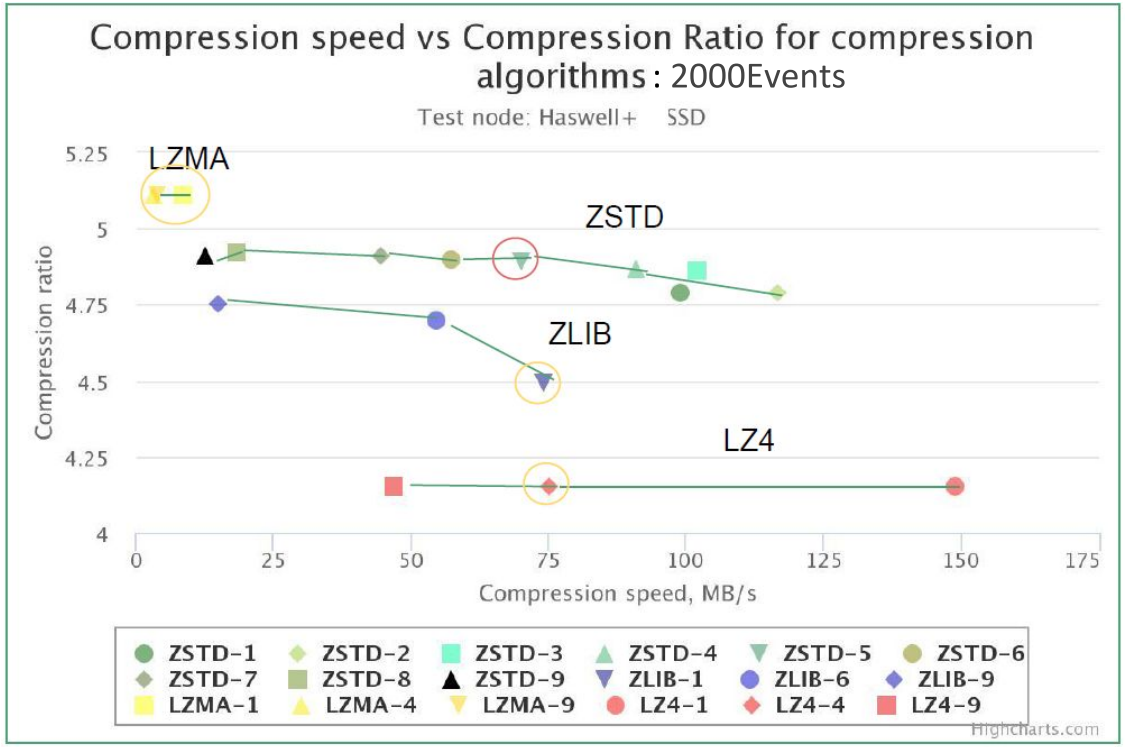

2000Events. Compression Ratio
LZ4
LZMA
ZLIB
ZSTD
Compression Ratio (Higher is better)
6.00
5.00
4.00
3.00
2.00
1.00
0.00
1
2
3
4
5
6
7
8
9
Compression Level

Results


2000Events. Compression Speed
LZ4
LZMA
ZLIB
ZSTD
Written MB/s (Higher is better)
250
200
150
100
50
0
1
2
3
4
5
6
7
8
9
Compression Level


2000Events. Decompresion Speed
LZ4
LZMA
ZLIB
ZSTD
Read MB/s (Higher is better)
700
600
500
400
300
200
100
0
1
2
3
4
5
6
7
8
9
Compression Level

Results


Compression speed vs Compression Ratio for compression algorithms : 2000Events
Test node: Haswell+ SSD
LZMA
ZSTD
ZLIB
LZ4
Compression ratio
5.25
5
4.75
4.5
4.25
4
0
25
50
75
100
125
150
175
Compression speed, MB/s
ZSTD-1 ZSTD-2 ZSTD-3 ZSTD-4 ZSTD-5 ZSTD-6
ZSTD-7 ZSTD-8 ZSTD-9 ZLIB-1 ZLIB-6 ZLIB-9
LZMA-1 LZMA-4 LZMA-9 LZ4-1 LZ4-4 LZ4-9
Highcharts.com

Current status of ZSTD: Ready to merge

- Work started on top of Brian's implementation, which included more advanced ideas like a compression engine. github.com/oshadura/root/tree/brian-zstd
- We have splitted this work and prepared ZSTD to be added as a regular compression algorithm in ROOT.

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

Future ideas for compression

- ZSTD has also brought the idea of improving the way compression is done in ROOT, mostly motivated by some of its new features like “*Training Dictionaries”*.

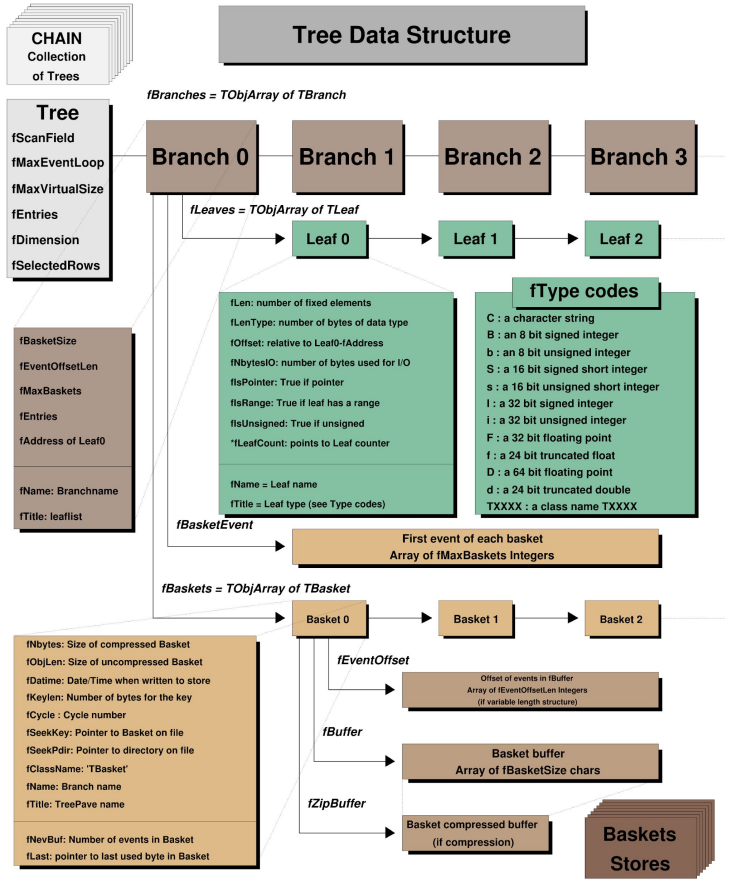
- Brian Bockelman proposed the idea of using a *Compression Engine* to develop a more integral approach to compression instead of compressing every single chunk of data in ROOT separately.

- A Compression Engine is a class that deals with all the compressions and decompression requests. By centralizing the compression work, it can provide multiple benefits like reusing resources, one-time initializations and find synergies between compressed chunks.

However there are many ways of approaching this idea.

A little of background

- In ROOT we 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:

Basket compression

- Baskets are not contiguous in memory so they must be compressed separately.
- Each basket is compressed from scratch and has a custom dictionary.
- So we have to build and store one dictionary per basket.
    - This impacts both the compression ratio and speed.

But baskets must be similar between them so why not using the same dictionary for them all?

Benefits of one Dictionary per Branch

Sharing the same dictionary for all the basket of a branch offers multiple benefits:

- Less overhead due to the dictionary size
  - Although is not yet clear how big the overhead is *(we need to benchmark it)*

- Faster compression: We only build one dictionary, then we just reuse it.

But how to do it?

Use Dictionary Training ZSTD

One of the most promising ideas of ZSTD but not so interesting as it looks like:

- Train with all the baskets of a branch
  - *Too much time*

- Train with the first basket of a branch
  - *Does not work, it requires multiples samples for training*

- Train with basket of all branches and use a dictionary for all the tree
  - *Too much time and the same dictionary can't be good for all the variables.*

Best scenario for ZSTD Training

Facebook is a good example of how training can bring a huge benefit:

- Lots of small chunks of data (comments, posts, etc)
- For so small chunks the dictionary would take a lot and would not have even enough data to be effective.
- Create a universal dictionary trained with all the text of their dataset.
- Dictionary good enough for almost everything and does not need to be stored.
- Does not apply to ROOT because our chunks of data are that small, neither all our data follows the same pattern.

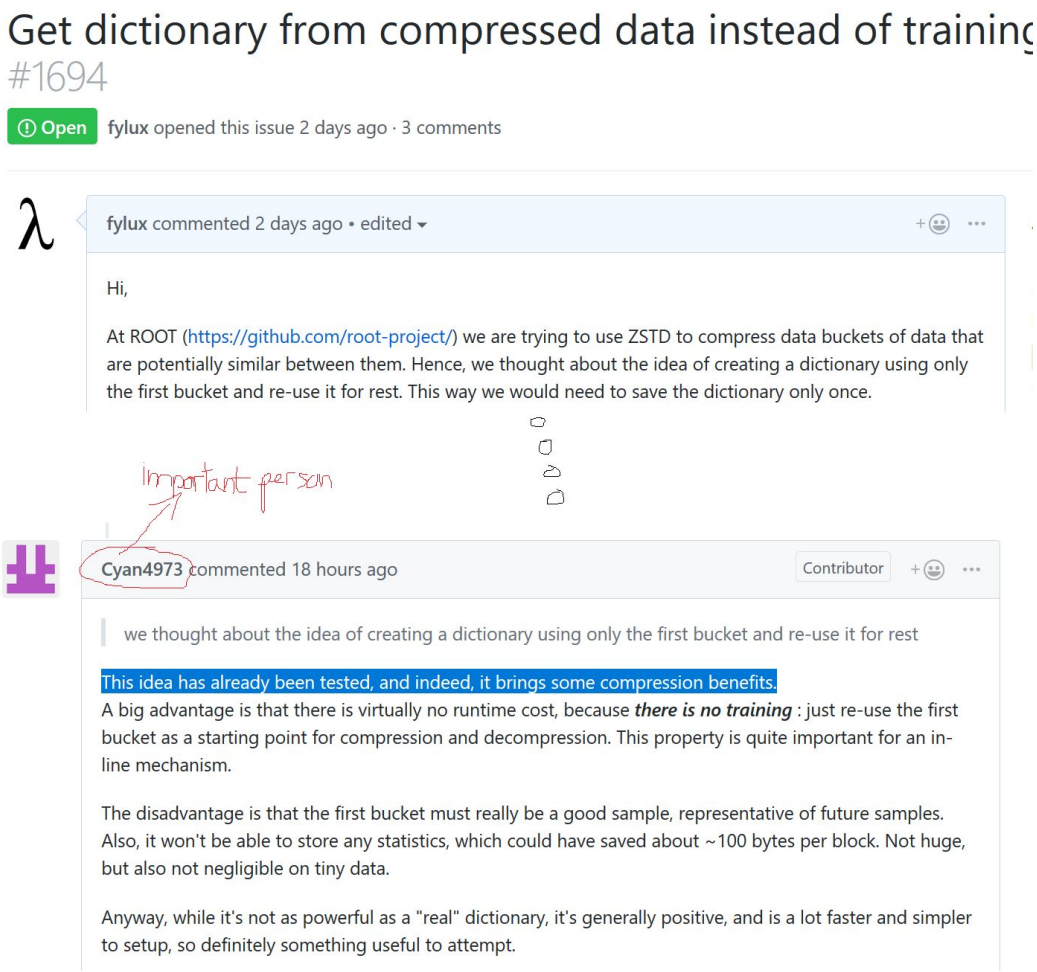
facebook.github.io/zstd

Let’s think simple

- We don’t need to go to the most advanced features for our problem.
- Baskets of the same branch should be similar right?
- Let’s reuse the dictionary that was created for the first basket.

But is this possible?

Let’s ask ZSTD’s author



Trade-offs of this approach

- **Reduce overhead of storing the Dictionary:** The improvement in the file size reduction will be determined by this.

- **Improve compression speed:** Dictionary will be generated only once, it should speed up considerable the process.

- **Worse compression ratio if first basket is not representative:** We don’t have any more customized dictionaries per basket.

Conclusions

- ZSTD can be a great addition for ROOT, both as a standalone compression algorithm and as the base of a powerful compression engine.

- Lot of potential in the compression engine: TTrees and RNtuples can benefit from all the exposed improvements.