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.

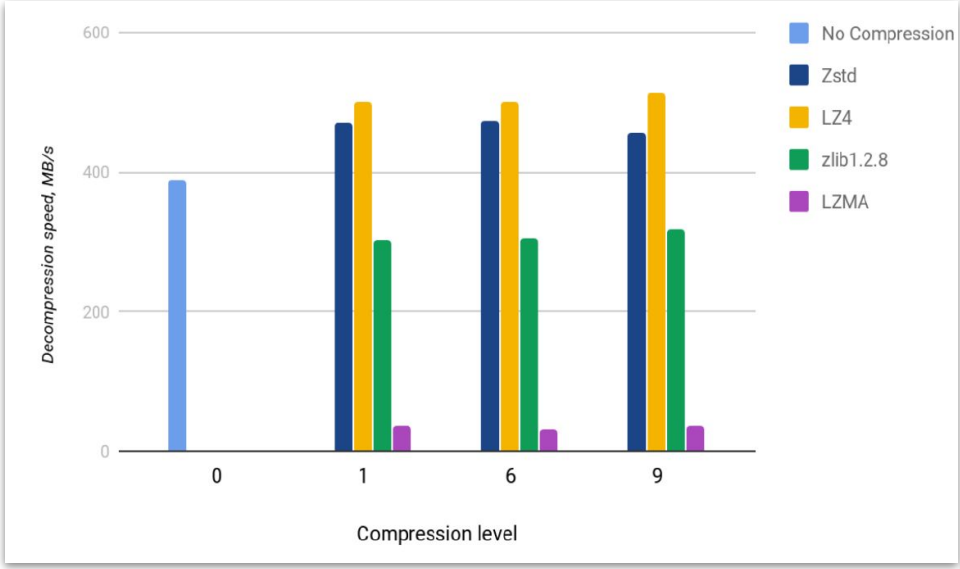
ZSTD

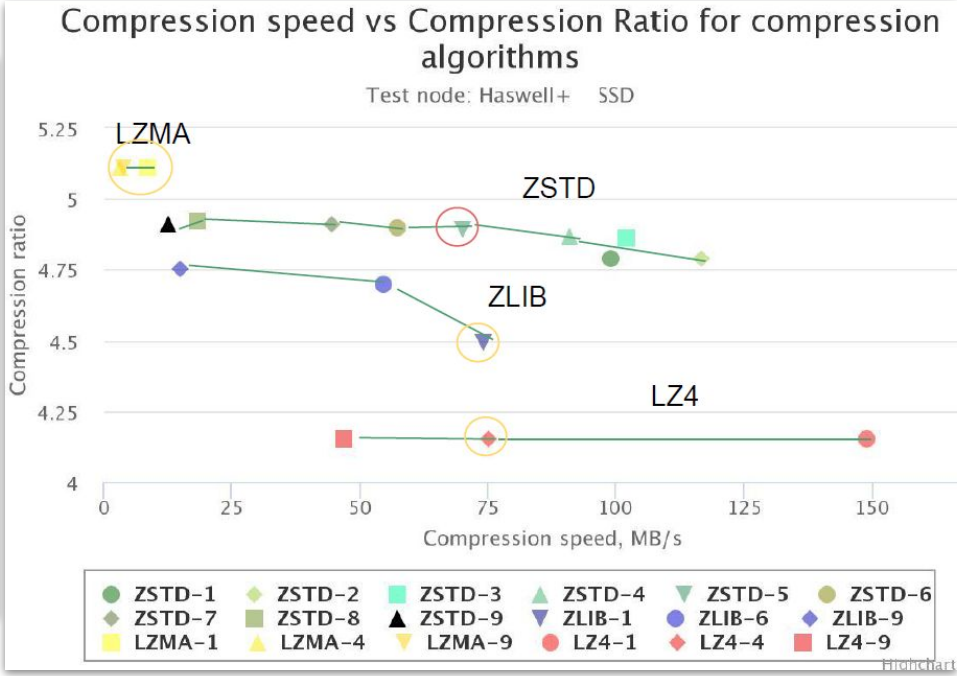
Promising alternative for cases where a balance between compression and speed is required. https://github.com/facebook/zstd

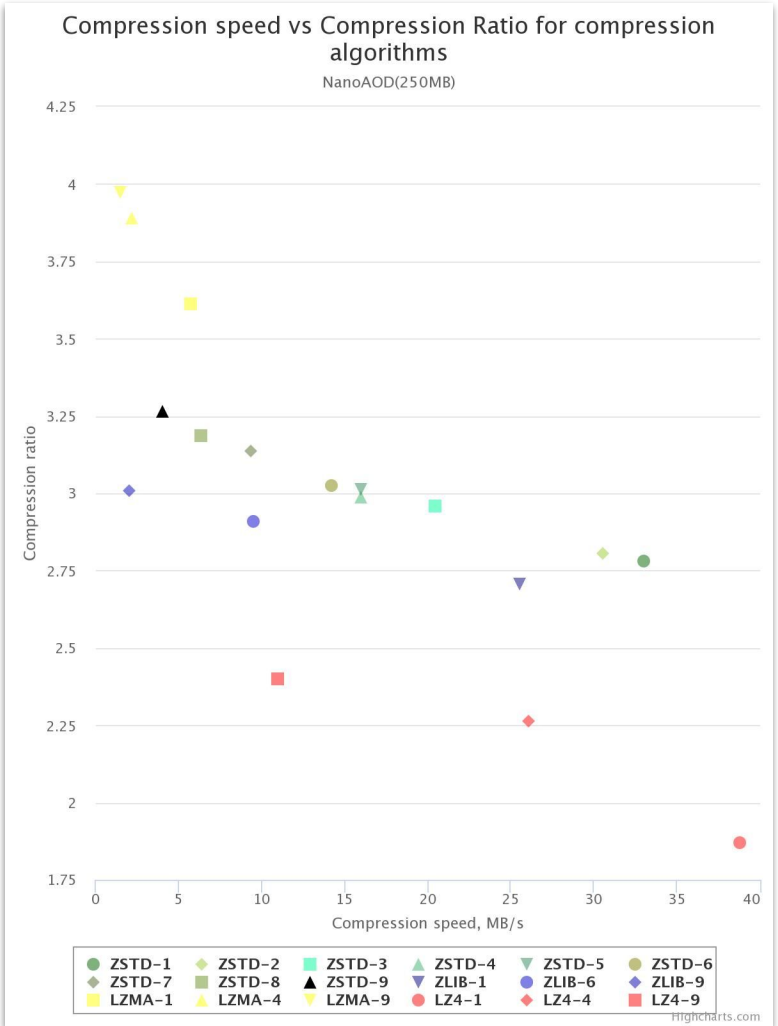
What ZSTD promises:

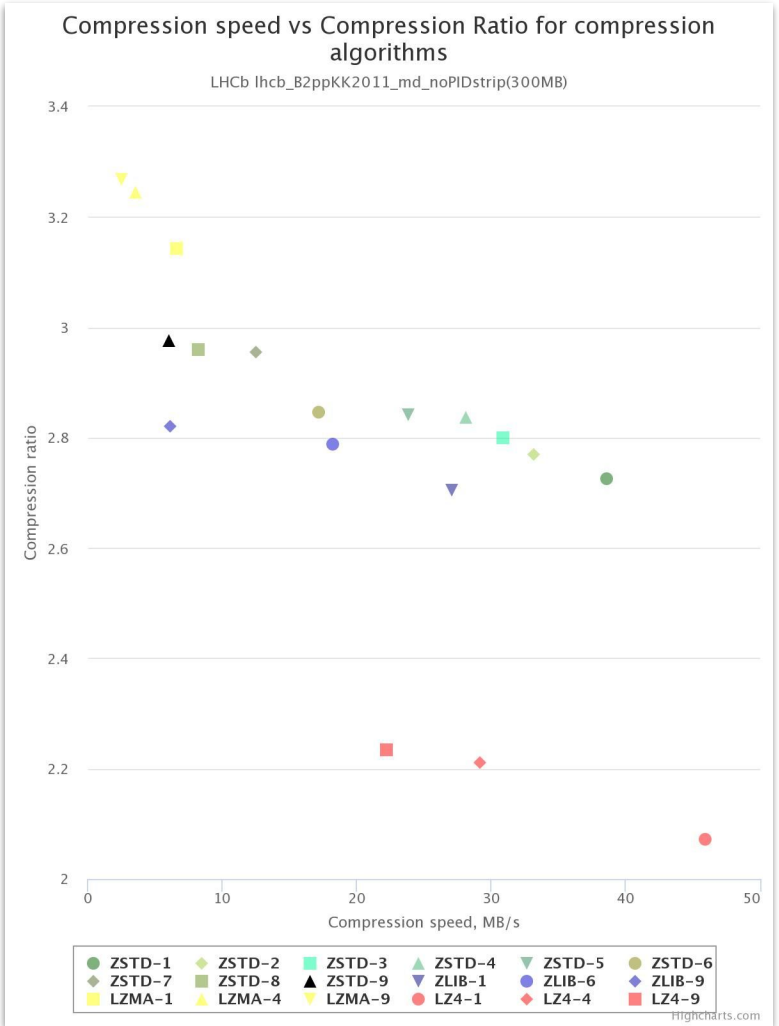
- Better than ZLIB in all metrics: compression speed, decompression speed, and compression ratio.
- Decompression speed should be constant regardless of compression level.
- High dynamic range in tradeoff between compression speed and ratio.
- Does not achieve compression ratio of LZMA, neither speed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

Results


600
400
200
0
Decompression speed, MB/s
0
1
6
9
Compression level
No Compression
Zstd
LZ4
zlib1.2.8
LZMA


Compression speed vs Compression Ratio for compression algorithms
Test node: Haswell+ SSD
LZMA
ZSTD
ZLIB
LZ4
5.25
5
4.75
4.5
4.25
4
Compression ratio
0
25
50
75
100
125
150
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


Compression speed vs Compression Ratio for compression algorithms
NanoAOD(250MB)
Compression ratio
Compression speed, MB/s
4.25
4
3.75
3.5
3.25
3
2.75
2.5
2.25
2
1.75
0
5
10
15
20
25
30
35
40
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


Compression speed vs Compression Ratio for compression algorithms
LHCb lhcb_B2ppKK2011_md_noPIDstrip(300MB)
Compression ratio
Compression speed, MB/s
3.4
3.2
3
2.8
2.6
2.4
2.2
2
0
10
20
30
40
50
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

ZSTD: Advanced Compression

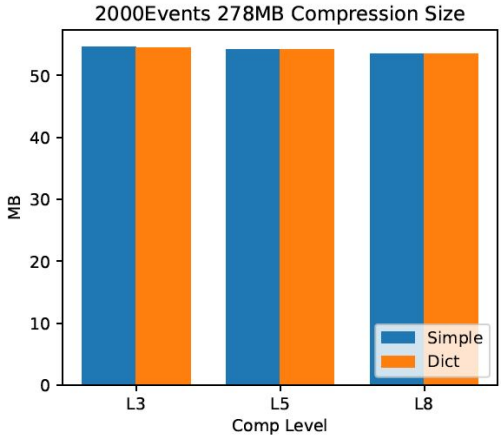
- The advanced API of ZSTD allows to develop novel optimized solutions that have never been tried before: Compression Engine and Dictionaries Reusing.

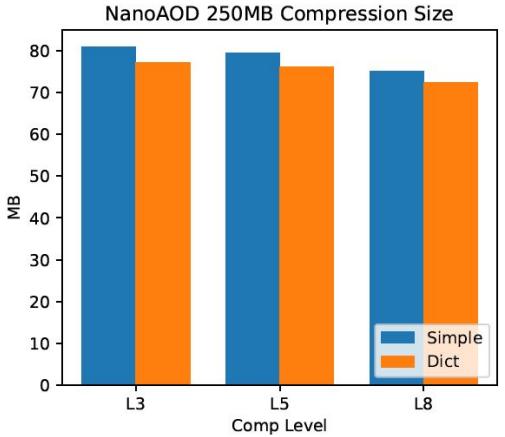
- A Compression Engine is a class that will process all compressions requests. This centralization provides multiple benefits like reusing resources, one-time initializations and find synergies between compressed chunks.

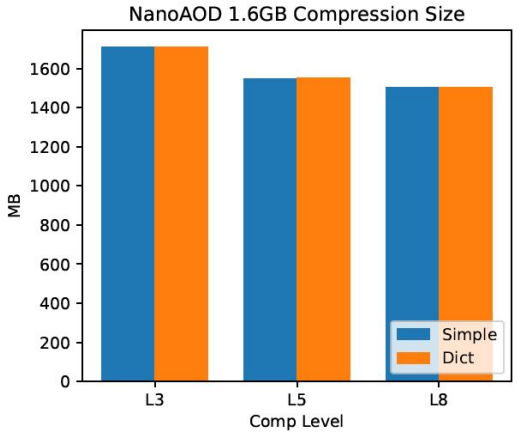
- The dictionaries that are generated for a given data can be reused for similar data, reducing significantly the overhead of the dictionary size.

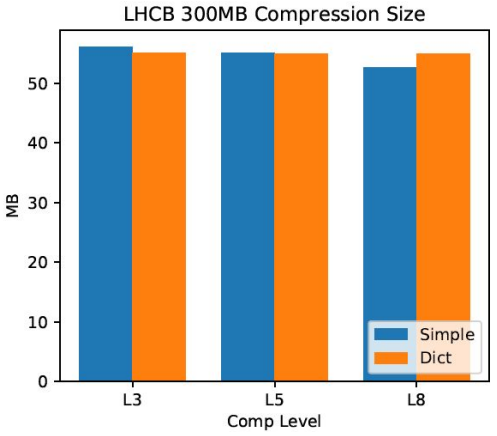
Results

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









Work Done

PR 93: Compression Benchmarks github.com/root-project/rootbench/pull/93

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

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

PR 4096: ROOT I/O Compression refactoring github.com/root-project/root/pull/4060

PR 4248: ZSTD Compression with Dictionaries github.com/root-project/root/pull/4248

Conclusions

ZSTD: Fundamental Compression

- Outperforms in all metrics ROOT's default compression algorithm.
- Fully tested, integrated in ROOT.
- 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.