ROOT I/O Compression

Compression in ROOT I/O

- Most time consuming part of ROOT I/O
  - Often between 30% and 60% of the total I/O time.

- Per buffer/basket basis.
- 2 bytes marker/prefix to indicate the compression algorithm
- Requires to build **all** compression algorithms on **all** platforms
  - Not doing so would mean some ROOT file are not readable on some platforms.

Currently Supported

- Default: post 2004 Zlib algorithtm
- LZMA
  - *Best compression ratio CPU time can buy*
    - *At least at the time we added it*
- And pre-2004 zlib algorithm for backward compability only
- And, of course, no-compression ☺

Currently Supported – Compared to field

- Default: post 2004 Zlib algorithtm
- LZMA
  - *Best compression ratio CPU time can buy*
    - *At least at the time we added it*
- And pre-2004 zlib algorithm for backward compatibility only

- *GitHub requests to add:* lzo, **lz4**, zopfli, and brotli
- And many more choices: zstd, snappy, lzfse, xz, PAQ, LZW, LZHAM, LZSS, etc ….

Zhe’s look at LZ4

Cost of each compression algorithm

- Maintain each accessor functions across algorithm version update
  - More code to maintain and monitor (eg. Security issues with compression algorithm)
- Insure that the compression algorithm builds on each platforms
- Add/run test with files with each compression algorithm
  - Tweak those tests for change in behavior across algorithm versions.
- Add test mixing compression algorithms

- More options means:
  - Feature creep
  - More confusion (which one should I choose?)
  - Higher complexity when reproducing issues
  - Potential difficulty/challenge to use alternate route (eg. Intel (de)compression ASICs).

- Lose ‘forward’ compatibility when used
  - i.e. file can not be read by older releases.

LZ4 sweet spots?

- Much faster decompression than zlib and lzma (4 to 7 times)
- Similar compression time as zlib
- But smaller to close to even compression ratio as zlib
  - Depending on the basket size and content.

Questions

- Is there any realistic use case for LZ4?
  - Who/When/How often?
- Would it be more used than no-compression?
- Does it justify the ‘expense’?
- Would the target audience know to enable this not-default option?

LZ4 Summary

Faster decompression time

Similar compression time

Larger files (depends on basket size and content)

Questions

- Is there any realistic use case for LZ4?
  - Who/When/How often?
- Would it be more used than no-compression?
- Does it justify the ‘expense’?
- Would the target audience know to enable this not-default option?

LZ4 Summary

Faster decompression time

Similar compression time

Larger files (depends on basket size and content)

Do you have ***experience*** with other algorithm and know one that would be significantly better in at least one dimension on our types of data?