Exploring Compression Techniques for ROOT IO

Zhe Zhang, Brian Bockelman
University of Nebraska-Lincoln
zhan0915@huskers.unl.edu, bbockelman@cse.unl.edu
Agenda

• Introduction
• Motivation
• Approaches
• Results
• Conclusions
Introduction

• ROOT is a software tool to store large amount of objects and help researchers to analyze and visualize the data.
  ➢ ROOT uses tree to store objects.
  ➢ Trees allow user to sequentially and randomly access entries.
  ➢ Trees allow user to access sub-branches.
Event Structure:
class A {
    class B {
        class D;
        class E;
    };
    class C {
        class F;
        class G;
    };
};
Trees are serialized by branches:

- Easy comparison across branches.
- Redundant data are close together.
Motivation

- ROOT works well for the common case (reading sequentially through a fixed set of branches) – can we beat its performance for various other use cases?
  - Alternative compression algorithms
  - Performance of random reads
  - Comparison to naïve approach
Alternative Compressions:

- Zlib vs LZMA vs LZ4 (right table)
- Tradeoffs between compression ratio and decompression speed

<table>
<thead>
<tr>
<th></th>
<th>Zlib</th>
<th>LZMA</th>
<th>LZ4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fast compression speed</td>
<td>✗</td>
<td></td>
<td>✗</td>
</tr>
<tr>
<td>Medium compression speed</td>
<td>✗</td>
<td>✗</td>
<td></td>
</tr>
<tr>
<td>Slow compression speed</td>
<td></td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Fast decompression speed</td>
<td></td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Medium decompression speed</td>
<td>✗</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Slow decompression speed</td>
<td>✗</td>
<td>✗</td>
<td></td>
</tr>
<tr>
<td>High compression ratio</td>
<td></td>
<td></td>
<td>✗</td>
</tr>
<tr>
<td>Medium compression ratio</td>
<td></td>
<td>✗</td>
<td></td>
</tr>
<tr>
<td>Low compression ratio</td>
<td></td>
<td>✗</td>
<td></td>
</tr>
</tbody>
</table>
Test Setup:

- CMS file contains 9 trees.
- The tree of **Events** has 213 branches and 6500 entries.
- Raw data file is 6.4 GB.
Read performance:

- LZ4-HC performs slightly better than Zlib-6 but not significant.
ROOT pipeline:

- Each branch has a basket.
- Each basket is filled with events.
- Each basket is compressed and flush to disk once it is full.
ROOT pipeline:

- Each branch has a basket.
- Each basket is filled with events.
- Each basket is compressed and flush to disk once it is full.

What if we only want to read a single event with several bytes out from a basket of 32 KB?
Random Access Compression:

Add array of offsets for compressed entries in TKey. ROOT searches the offset before decompress the basket; can decompress individual events.
Handcrafted objects:

- **Tlarge (4MB):**
  - is an array which contains 1,000,000 elements of which each is 4 bytes, and each random float number is repeated 60 times.

- **Tsmall (4KB):**
  - is an array which contains 1,000 elements of which each is 4 bytes, and each random float number is repeated 6 times.

- **Tint (39B):**
  - is an Int array which contains 6 integer with the same value.
RAC Compression Ratio:

- Tiny events decrease compression ratio.
- Once the size of single event goes beyond basket size, RAC is basically the same thing with non-RAC.
RAC Read performance:

- For small events, RAC requires more IOs to read data, but it still needs less CPU time to decompress data.
- Once data are in cache, RAC becomes much faster.
External Compression:

Dividing uncompressed ROOT file into equal size of blocks on disk and independently compress each block. Done with no knowledge of actual file contents, using SquashFS.
Compression Ratio:

- ROOT gives better compression ratio than SquashFS.

*ROOT Optimal* indicates the default setting in current ROOT where each basket size is dynamically configured by input data.
Read Performance with Hot Cache:

- ROOT understands the data layout better.
- User-space decompression degrades read speed.
Conclusions:

- LZ4 does not have significant improvement over Zlib.
- RAC can dramatically accelerate random read speed for small objects although it needs more storage.
- ROOT decompresses basket in user space where random reads in memory might slow down.
- ROOT has a lot of good features: fast sequential access, sub-branches accesses, good knowledge of event structure, etc. but there is still some room to improve.
Questions?
References:

• Alternative compression algorithms
  - https://github.com/root-mirror/root/pull/81
• Additional compression algorithms from github@pseyfert
  - https://github.com/root-mirror/root/pull/177
• RAC (not backward compatible):
  - https://github.com/root-mirror/root/pull/152
Write Performance:

- LZ4 Compression is faster than Zlib and LZMA in general and has similar performance with Zlib-1
- LZMA is slowest

Compression Ratio:

- LZMA has the best compression ratio.
- LZ4 is worst
- LZ4-HC sits between Zlib-1 and Zlib-6 (Default by ROOT)
Appendix

Cold Cache:
Read all events sequentially &
Read every 10th events
Appendix

Hot Cache:
Read all events sequentially &
Read every 10\textsuperscript{th} events
Appendix

Read every 100\textsuperscript{th} events with cold/hot caches
Appendix

RAC Random Read performance
Appendix

RAC Sequential Read performance
Appendix

Comprehensive RAC Read performance