GSoC End of Program Presentation
(2020 Sep. 3)

Keisuke Kamahori

Who am I & What am I doing

Keisuke Kamahori (https://github.com/kamahori)

Sophomore at University of Tokyo

Working on pre-conditioners of compression algorithms as a GSoC student.
(***Mentors: Oksana Shadura, Brian Paul Bockelman, Ken Bloom***)

1. Integrate Bitshuffle in ROOT's compression layer
2. Performance benchmarks

Bitshuffle

- A possible pre-conditioner for ROOT I/O
- Transpose input bytes prior to applying the compression algorithm (LZ4)
- Improve compression performance for typed binary data (size and speed), especially when adjacent values are highly correlated

Bitshuffle in a nutshell:

```
11010010          1111 1111
11010011 -->      0000 1111
11010100          0000 0011
11010101          1100 1010
```

Progress about LZ4+BS

- Fixed a bug about LZ4+BS decompression
- Added appropriate trampoline between SSE/AVX2 (enabled vectorisation)
- Wrote performance benchmarks in *rootbench.git*
- In final implementation,
  - Source size must be the **multiple of 4** in order for BS to work properly ([code](#))
  - Then take **comparison approach**: try with both LZ4 / LZ4BS, and choose better one
  - **Compression speed is a bit reduced**: ~30% longer at most
  - **Decompression speed is almost the same as LZ4**

Performance in TTree

Red: ZLIB / Green: LZ4 / Yellow: LZ4BS / Black: ZSTD
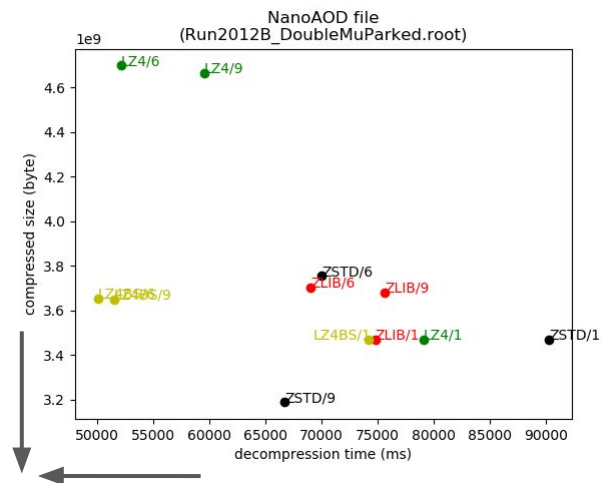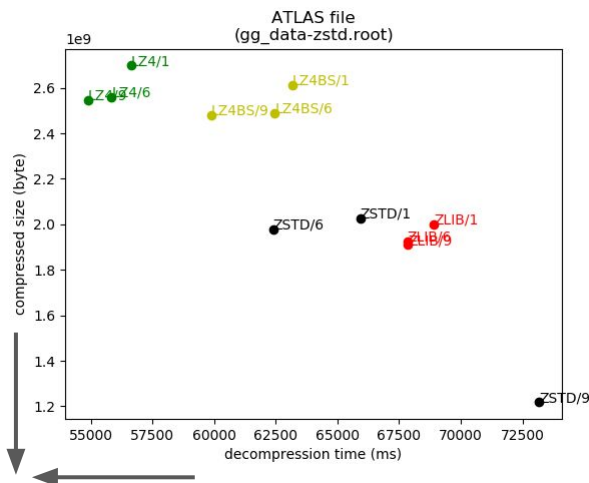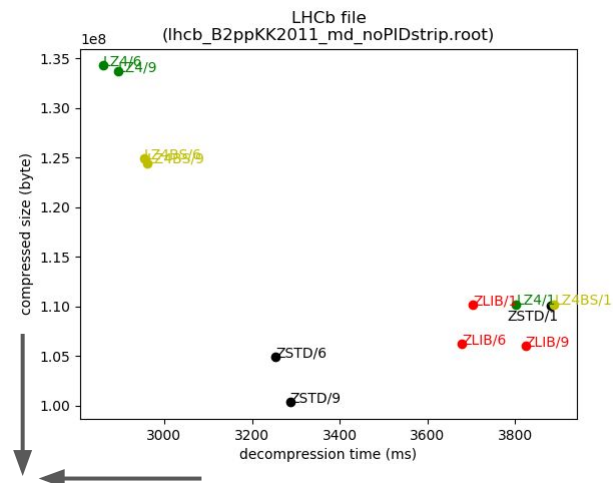
Performance in RNTuple
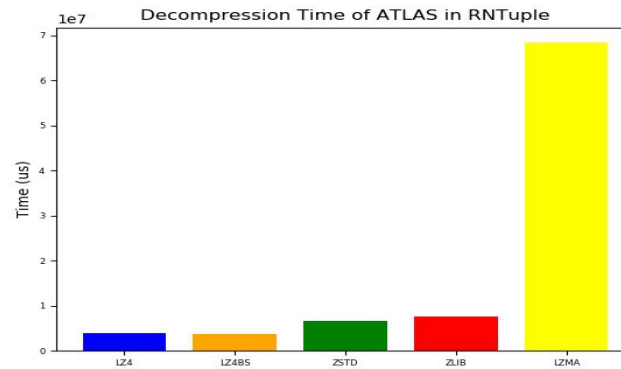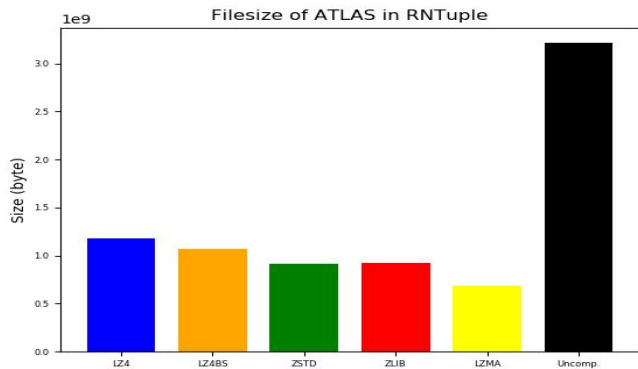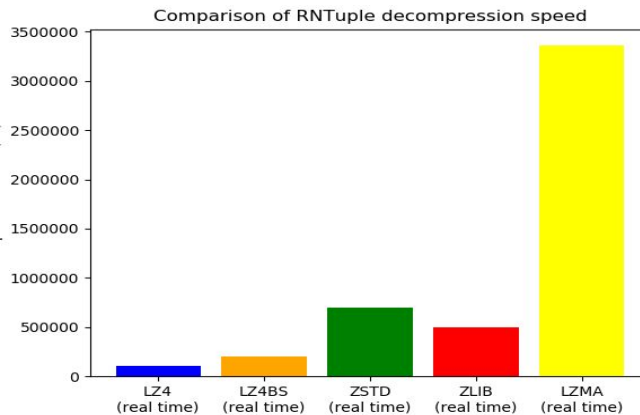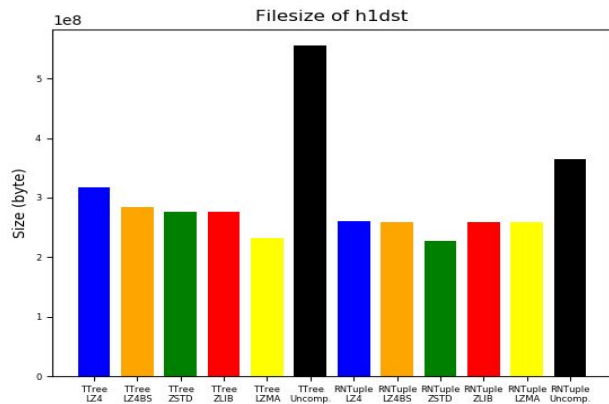
Compression Ratio by Branch / Page ([data](#))

- Those consist only of **small & positive ints** have great compression ratio in LZ4BS
  - There are many consecutive zeros when transposed!
- **Floats** tend to have worse compression ratio than ints
  - Maybe because mantissa bytes are unlikely to correlate
- **Bools** have bad compression ratio
  - Because adjacent bytes are unlikely to correlate

Final report

More details in GSOC final report:

- https://kamahori.github.io/gsoc2020/

Contacts:

Keisuke Kamahori [keisuke1258@gmail.com]