LZ4, BulkIO, and offset removal performance

Jim Pivarski

Princeton University – DIANA

October 11, 2017

Motivation for this study

Three updates to ROOT I/O are aimed at speeding up or reducing file size for end-user analysis:

- new compression algorithm: LZ4 (speed)
- reading TBasket data directly into arrays: BulkIO (speed)
- removing offset data from TBranches that have a counter (size)

Motivation for this study

Three updates to ROOT I/O are aimed at speeding up or reducing file size for end-user analysis:

- new compression algorithm: LZ4 (speed)
- reading TBasket data directly into arrays: BulkIO (speed)
- removing offset data from TBranches that have a counter (size)

Focus on CMS NanoAOD in particular because

- it is aimed at end-users (1–2 kB/event)
- it is broadly intended for 30–50% of analyses (not an individual user's ntuple)

Motivation for this study

dianahep

Three updates to ROOT I/O are aimed at speeding up or reducing file size for end-user analysis:

- new compression algorithm: LZ4 (speed)
- reading TBasket data directly into arrays: BulkIO (speed)
- removing offset data from TBranches that have a counter (size)

Focus on CMS NanoAOD in particular because

- it is aimed at end-users (1–2 kB/event)
- it is broadly intended for 30–50% of analyses (not an individual user's ntuple)

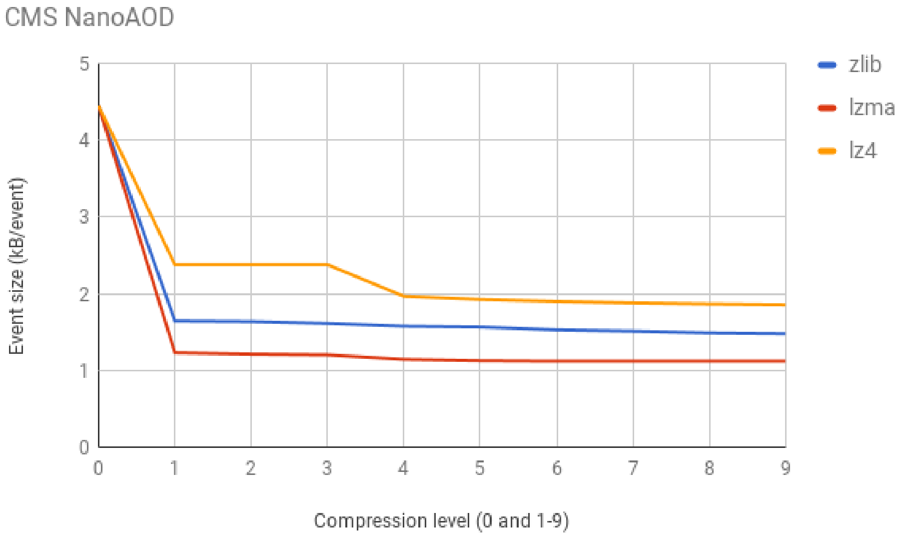
Also including studies of LHCb (thanks, Oksana!).

No ATLAS files because I can't generate new ones or `TTree::CopyTree` old ones.

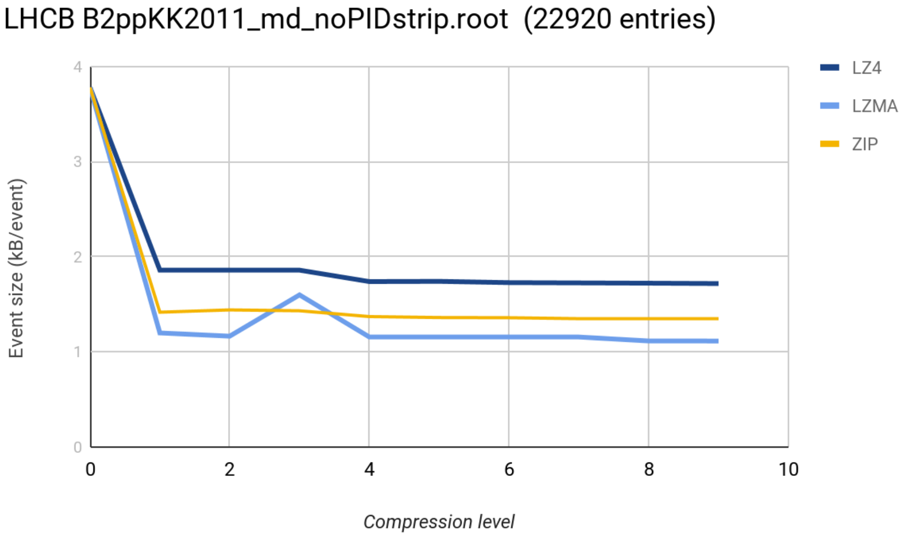
Parameters of the NanoAOD studies

- AWS instance with a fast SSD disk (i2.xlarge).
- No resource contention because I paid for exclusive access.
- "Writing" means a `TTree::CopyTree` with new TFile compression.
- "Reading" means filling a class made by MakeClass.
- "BulkIO" means filling arrays through `GetEntriesSerialized`.
- Always *reading* from warmed cache.
- Five repeated trials; standard deviations are small compared to trends.

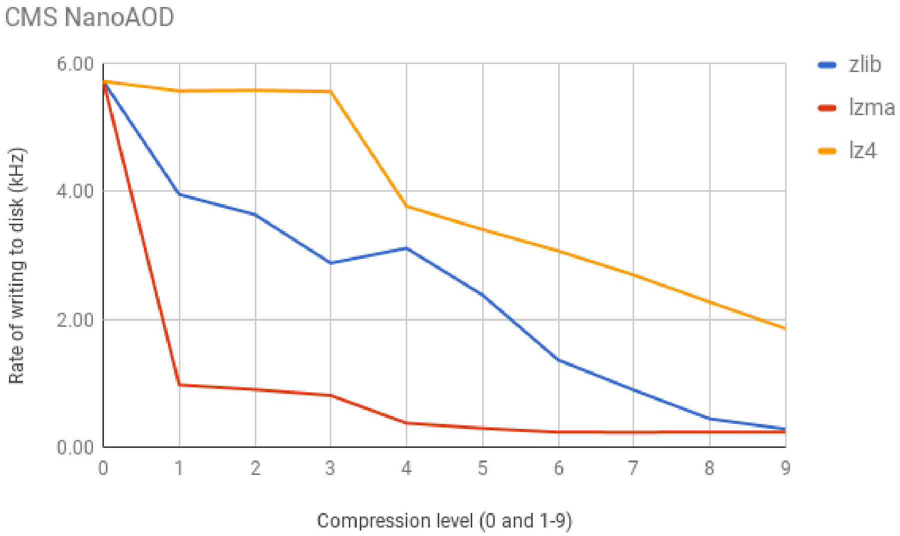
LZ4 doesn't compress as well as ZLIB, LZMA



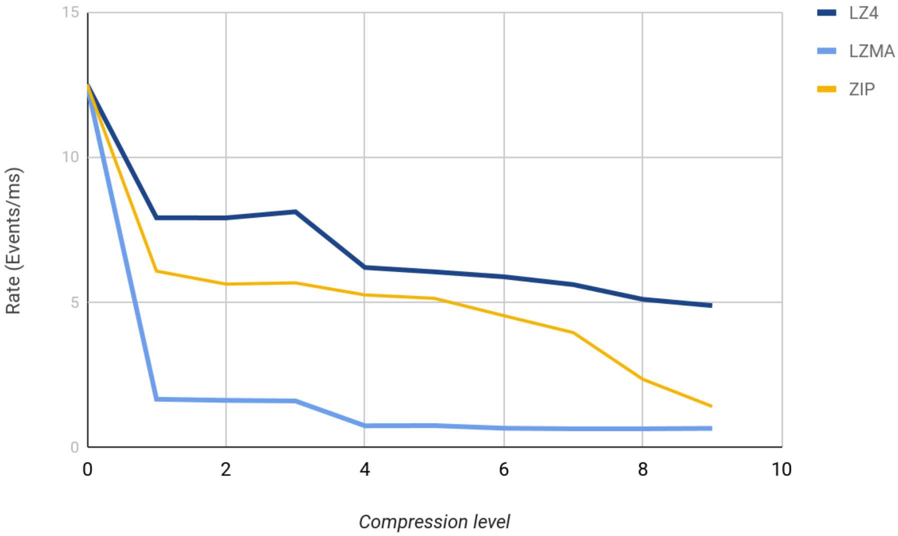
. . . same for LHCb



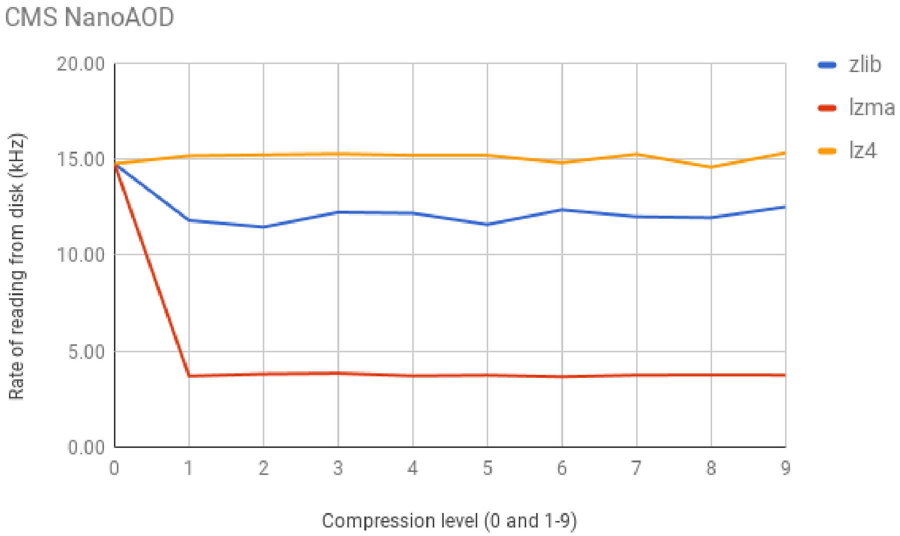
But it's faster: levels 1–3 are as fast as writing uncompressed



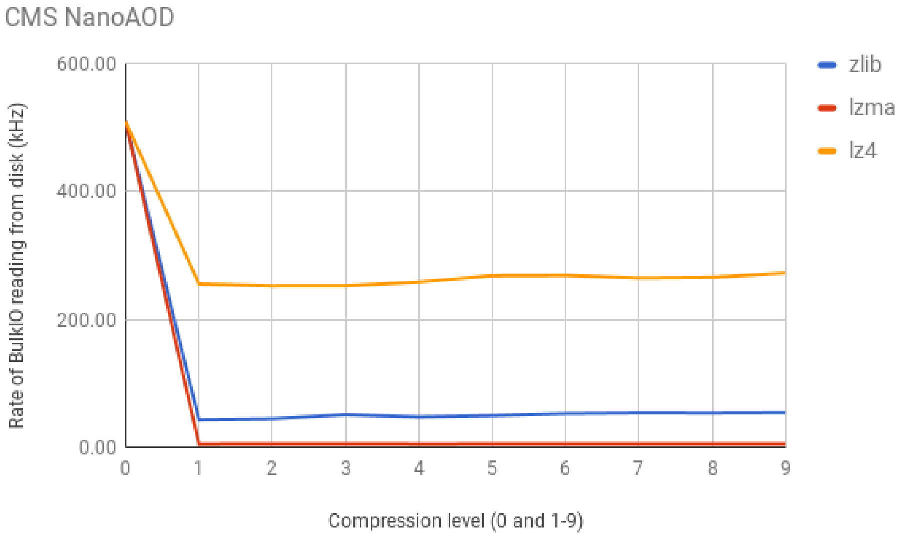
. . . same for LHCb



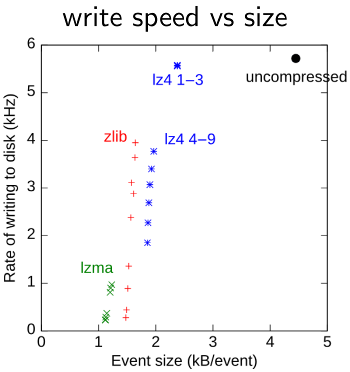
More importantly: reading is as fast as uncompressed

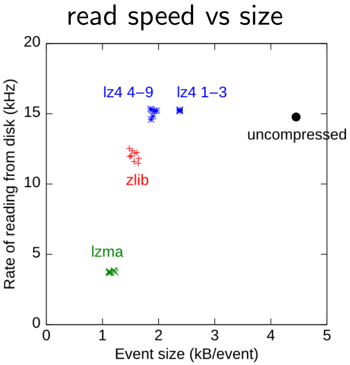


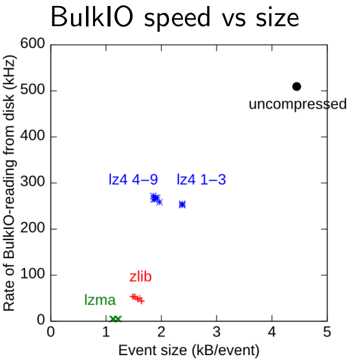
And BulkIO reading is super-fast: serious penalty for LZMA



Speed vs. size trade-offs







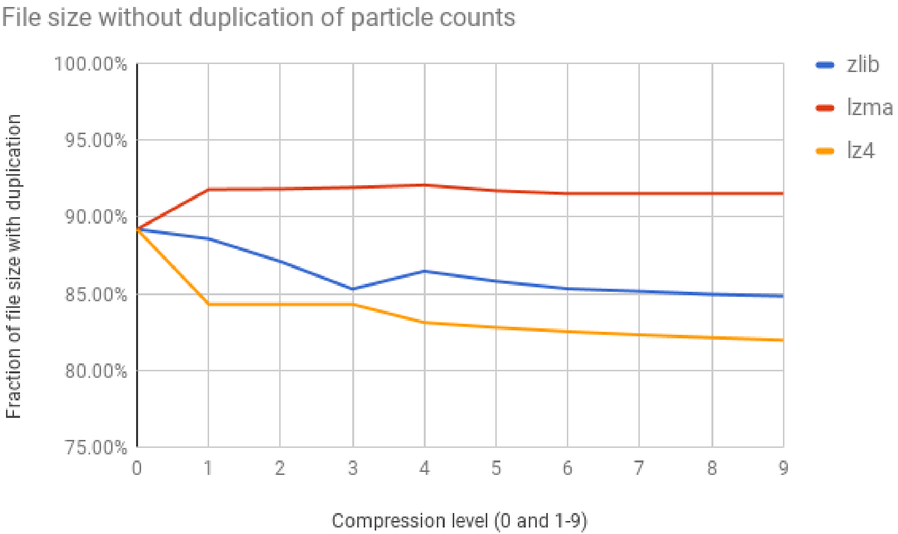
Removing unnecessary offsets

TBranches for variable-sized data contain offsets indicating where each entry starts.

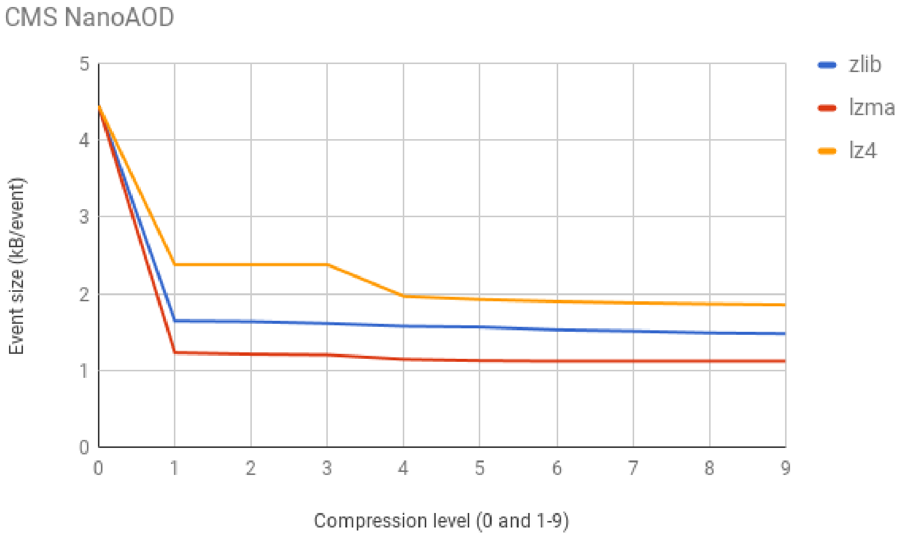
- This is unnecessary for branches with counters (e.g. `"Muon.pt[nMuons]/F"`).
- A fix is in progress (PR #1003) to optionally not write these offsets.
- May also write counts, instead of offsets, since repeated values might be more compressible.

My study pre-dated (inspired) this PR; I constructed a copy of NanoAOD without offsets by putting all muon data into a flat TTree, all jet data into a flat TTree, etc.

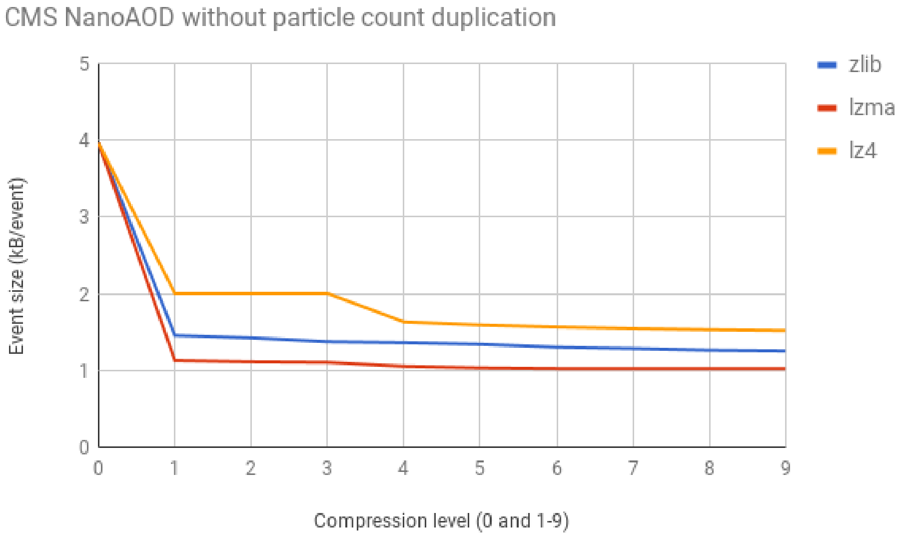
After compression, this saves 8–18%



And it closes the LZ4/LZMA gap to a factor of 1.5×



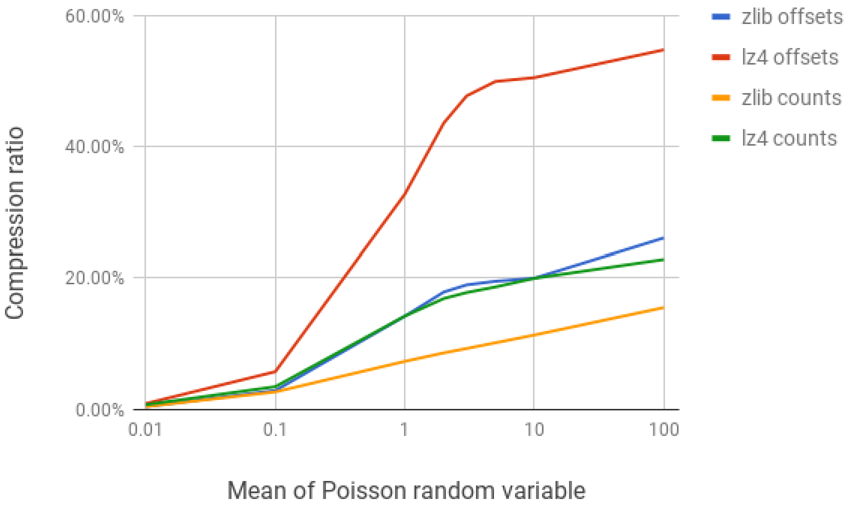
And it closes the LZ4/LZMA gap to a factor of 1.5×



Do offsets vs. counts matter? Yes for LZ4.

Synthetic test:

I generated Poisson-random counts and integrated them to make offsets, then ZLIB and LZ4 compressed them.



Conclusions

LZ4 is as fast as uncompressed data for traditional `GetEntry` jobs.

BulkIO is an order of magnitude faster than `GetEntry`, especially with LZ4.

Unnecessary offsets add $\sim$10% to file size; may be removed.

Counts compress better than offsets, especially for LZ4.