Integration of RNTuple in ATLAS Athena

Florine de Geus[1,2], Javier Lopez-Gomez[1], Jakob Blomer[1], Marcin Nowak[3] and Peter van Gemmeren[4]

May 8, 2023

[1] CERN
[2] University of Amsterdam
[3] Brookhaven National Laboratory
[4] Argonne National Laboratory

ROOT
Data Analysis Framework
https://root.cern

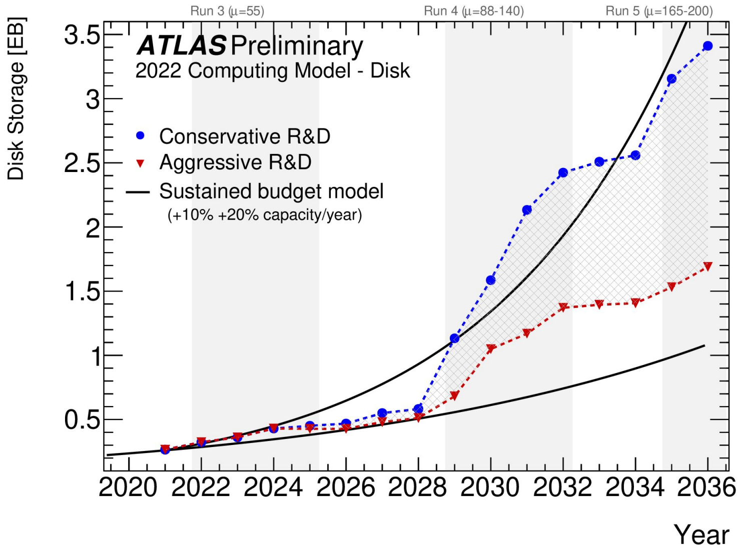
Background and motivation

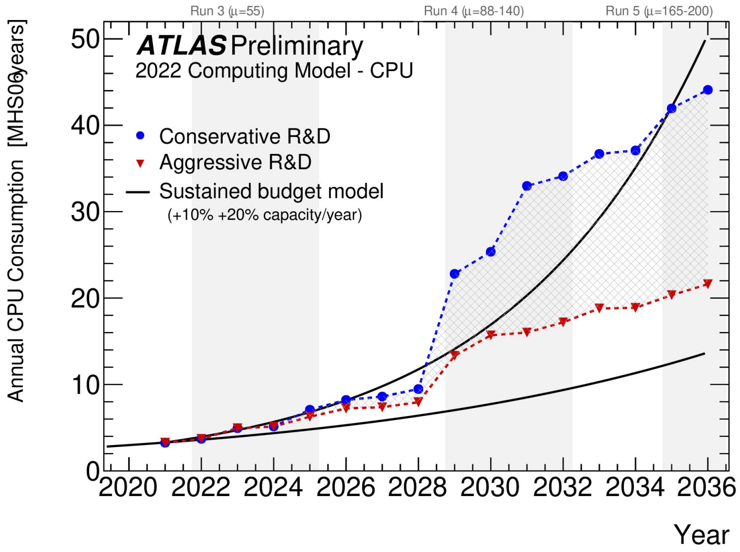
- **Athena:** ATLAS experiment software framework for data and MC processing
- For Run 3, **DAOD_PHYS** has been the common ATLAS wide analysis format
    - Produced by deriving primary AODs resulting from data/MC reconstruction
- For Run 4, **DAOD_PHYSLITE** will additionally be used
    - Centrally calibrated, which means it needs to store fewer variables

N.B. (D)AOD = *(Derived) Analysis Object Data*

Background and motivation

HL-LHC: (even) more data to store and process!





Source: ATLAS Software and Computing HL-LHC Roadmap

RNTuple: experimental evolution of ROOT's TTree columnar data storage

(See previous talk for more on RNTuple)

Getting RNTuple in shape for Athena

- Collection Proxies
  - Support for user-defined classes that behave as collections. These have an associated "collection proxy" that provides access to collection's elements
- Read rules
  - Act on standard ROOT I/O customization rules (i.e., `#pragma read`)
  - Enables custom post-read callbacks
- Late model extension
  - Allows for on-demand extension of RNTuple model with new fields after some entries have been written using the initial schema
  - Required for adding dynamic attributes during the derivation job

→ With these features, RNTuple-based DAOD_PHYS(LITE) production is fully possible in Athena

RNTuple for ATLAS DAOD_PHYS

Two central questions

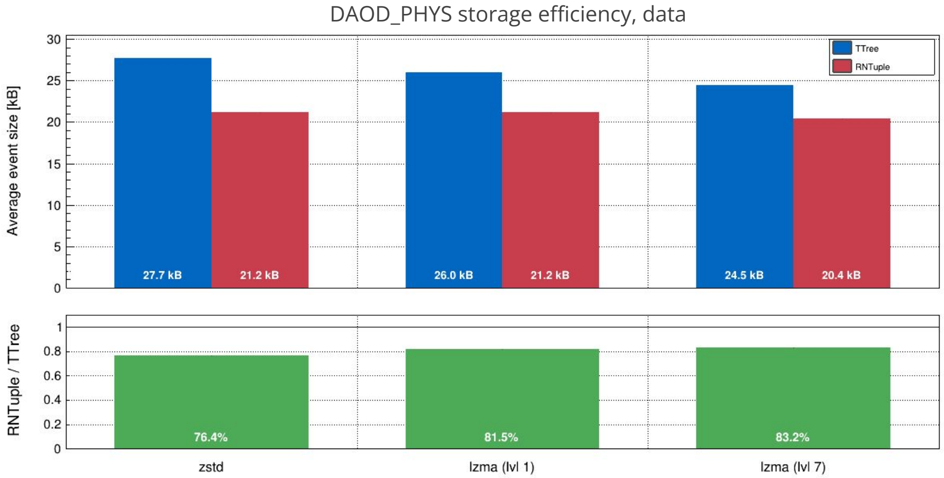
1. How does RNTuple perform compared to TTree?
2. How could it (ideally) be used in the future?

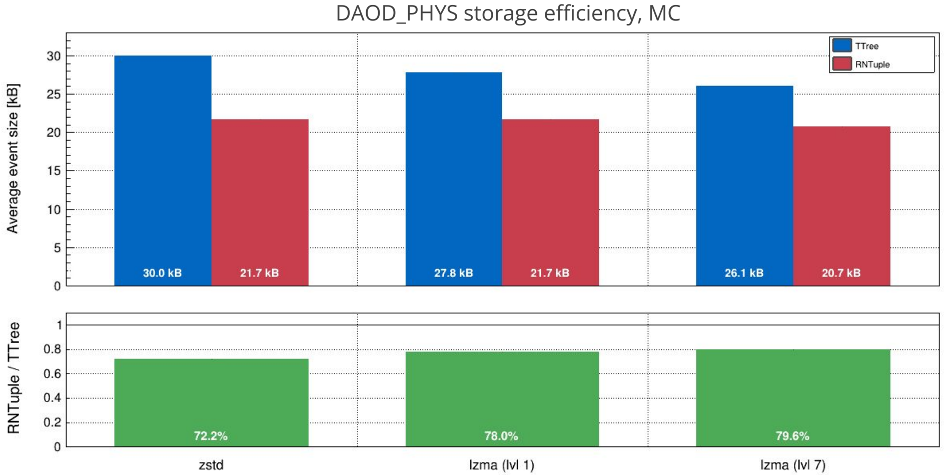
Evaluation of DAOD_PHYS

- Storage efficiency and read throughput
- Samples from real data and Monte Carlo
- RNTuples fully equivalent to TTrees, event-wise
  - Created using ROOT's `RNTupleImporter`
  - Default cluster and page configurations used

Storage Efficiency

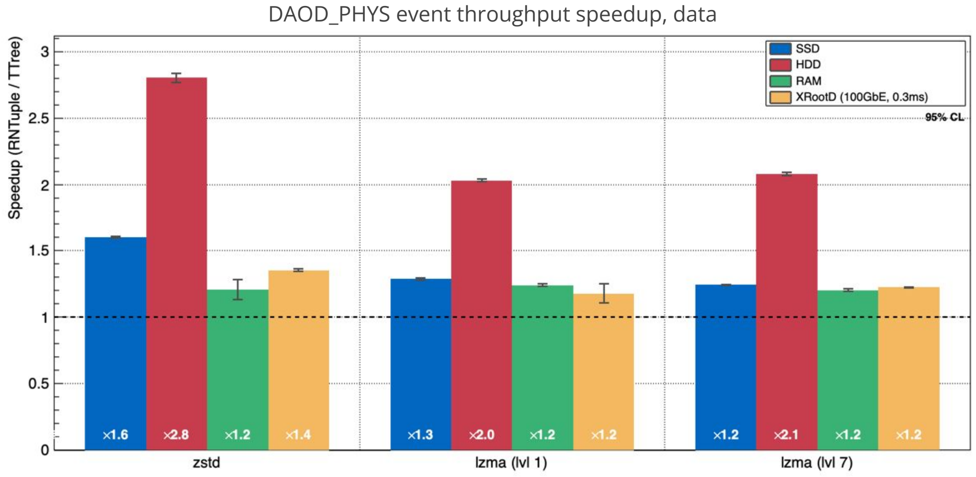
- DAOD_PHYS (almost) exclusively contain collections (both STL and custom), various levels of nesting
- Storage efficiency in line with other evaluations
  - See previous talk
- Selection flags are stored as `std::vector<char>`
  - Storing them as `std::vector<bool>` might lead to additional size reduction

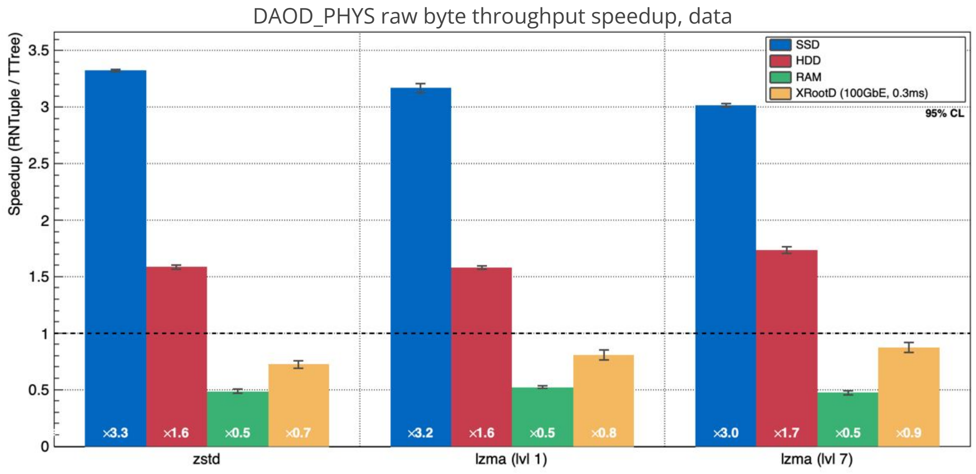




Read throughput

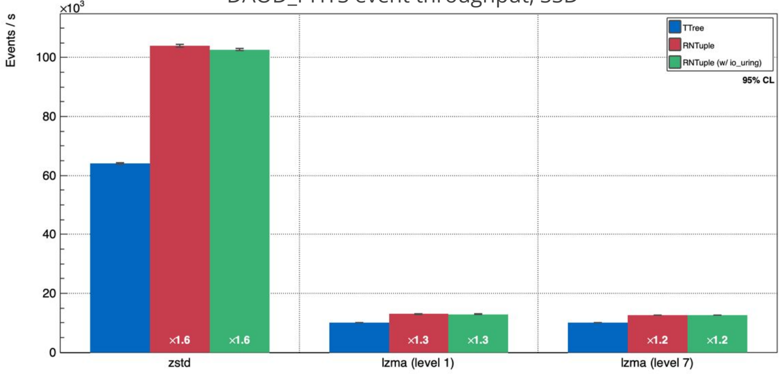
- Benchmark: (highly) artificial event loop using RDataFrame
  - Restricted to reading `std::vector<float>` fields
  - More representative benchmarks are planned, requires additional RNTuple support in Athena
- Depending on storage medium, performance may be CPU bound
- In general: faster time-to-plot
- Similar results with MC benchmark sample



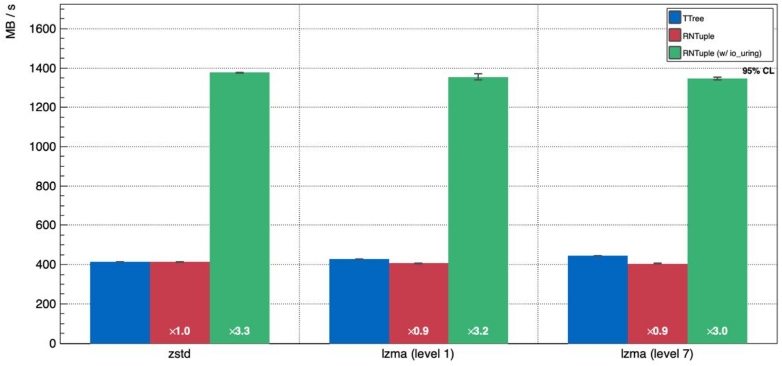


Read throughput

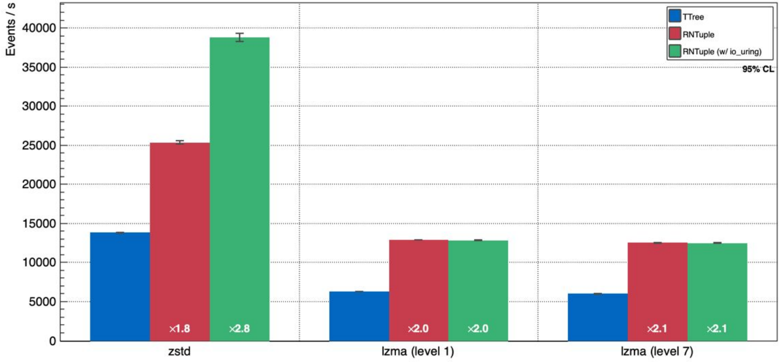
DAOD_PHYS event throughput, SSD



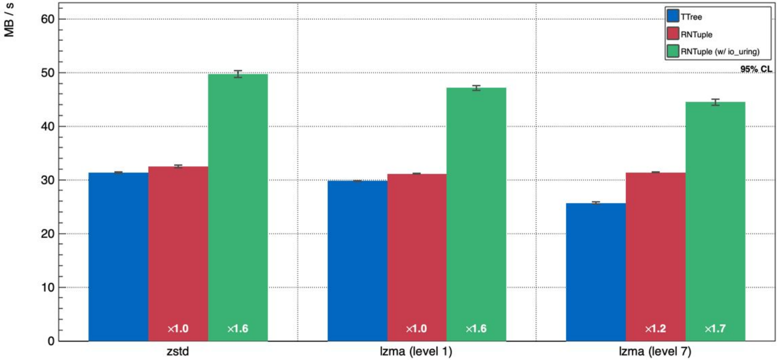
DAOD_PHYS raw byte throughput, SSD



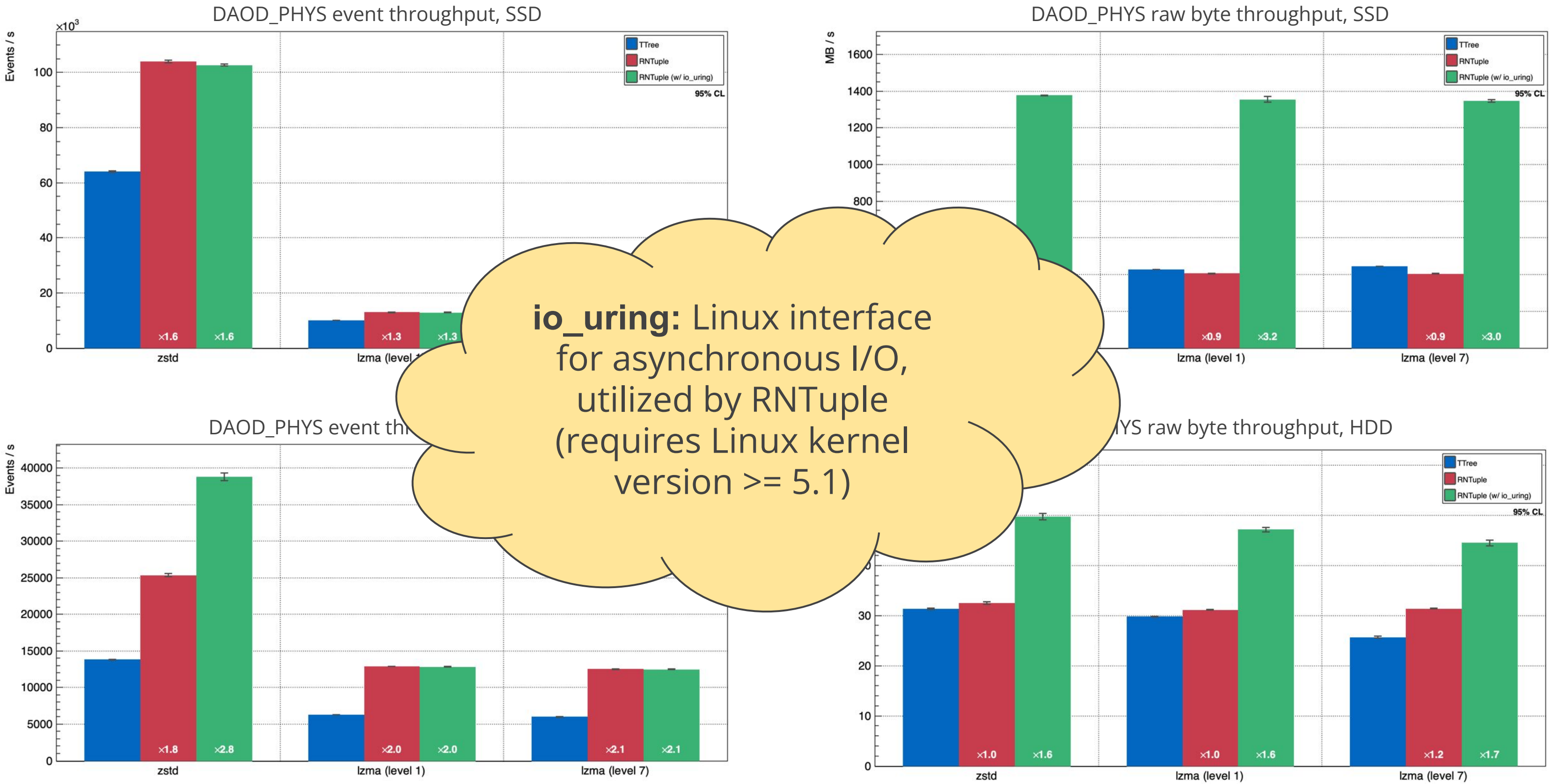
DAOD_PHYS event throughput, HDD



DAOD_PHYS raw byte throughput, HDD



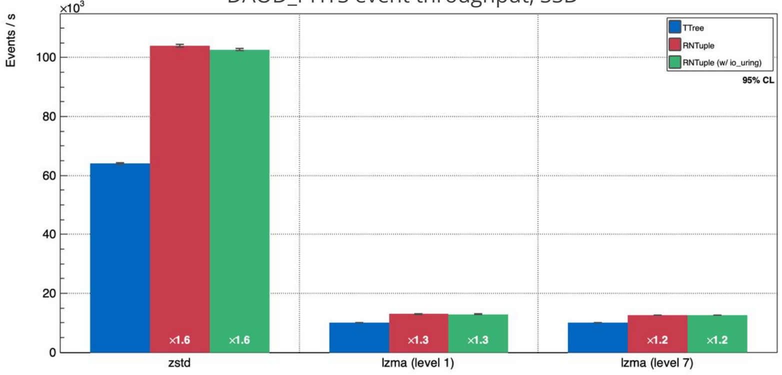
Read throughput



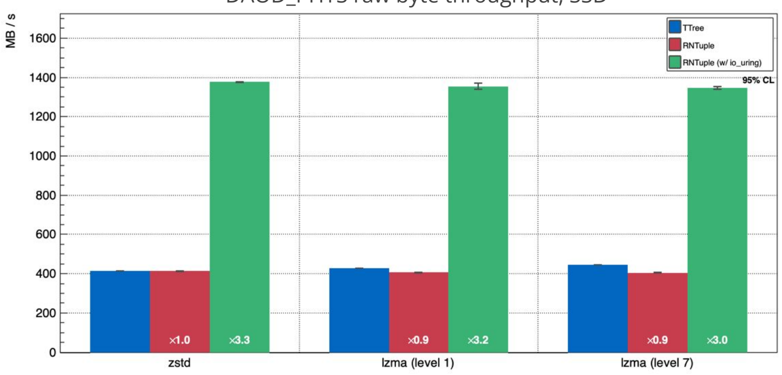
io_uring: Linux interface for asynchronous I/O, utilized by RNTuple (requires Linux kernel version >= 5.1)

Read throughput

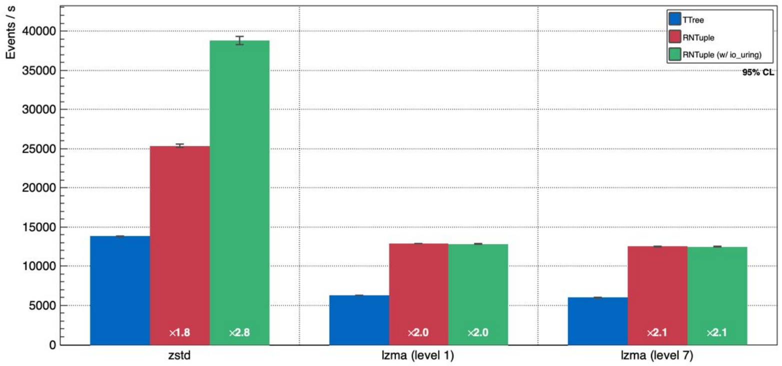
DAOD_PHYS event throughput, SSD



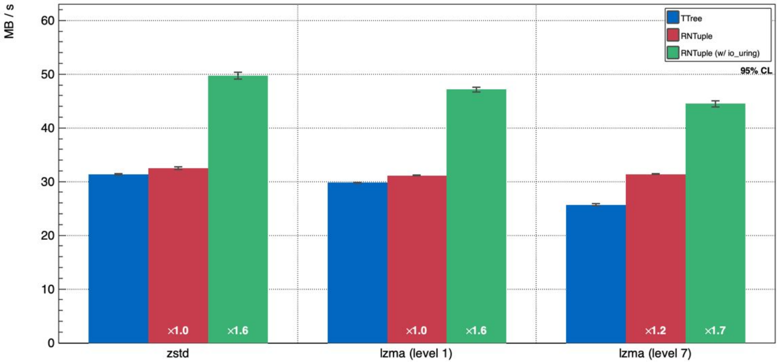
DAOD_PHYS raw byte throughput, SSD



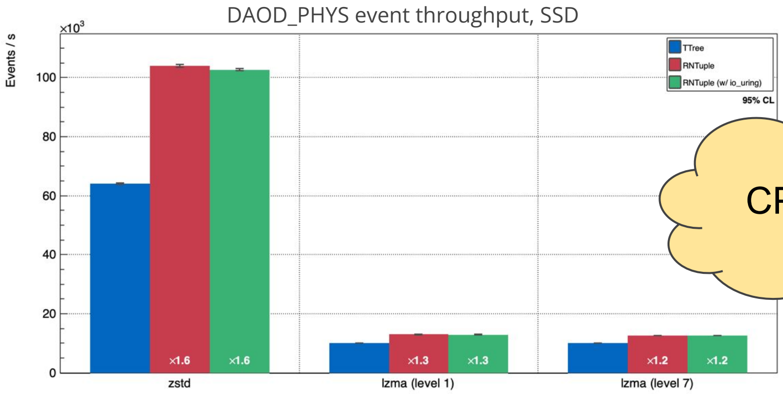
DAOD_PHYS event throughput, HDD

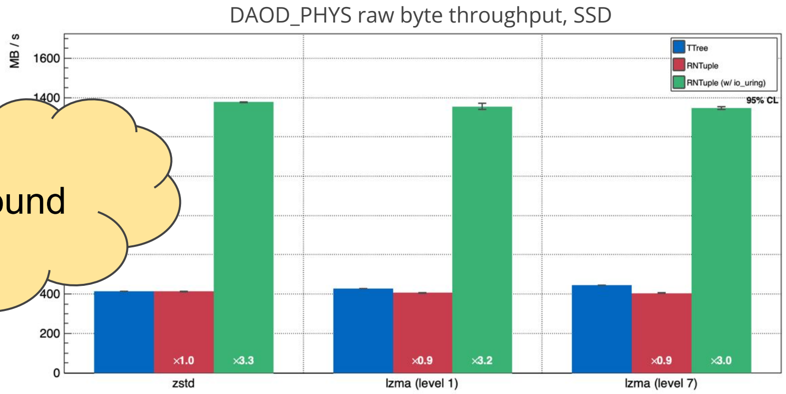


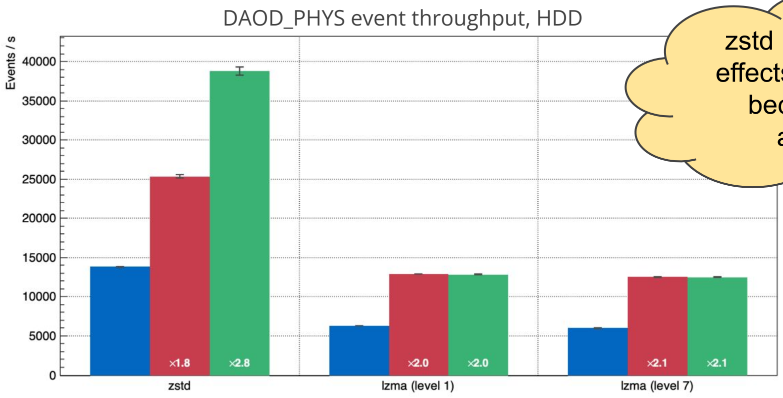
DAOD_PHYS raw byte throughput, HDD

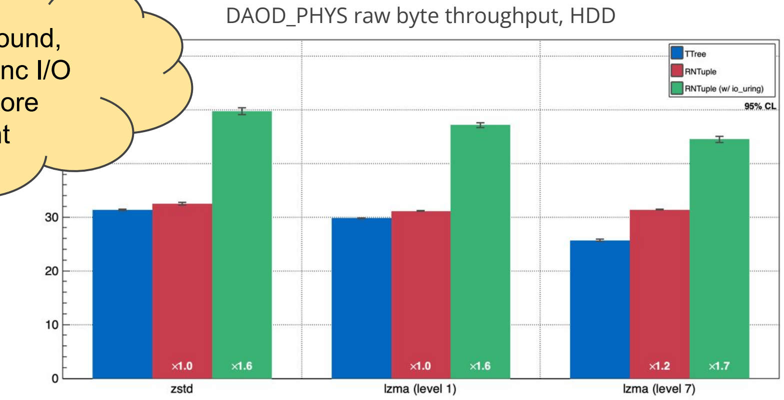


Read throughput









Next steps

1. Explore storage efficiency across more (different) files
2. Explore more of the benchmarking phase space
   - Compression: Lz4, lossy compression
   - Storage backends: Intel DAOS, S3
     - More on this during tomorrow's session
   - Data sources: more (XRootD) latency configurations
   - RNTuple parameter configuration: cluster and page sizes
     - Check out this poster on ML-based optimization of RNTuple I/O parameters
3. Evaluate (multiple) representative analyses
4. Evaluate DAOD_PHYSLITE
5. Bonus: Evaluate RNTuple use in other stages of the data production pipeline

Summary and concluding remarks

- Support for RNTuple in ATLAS Athena almost complete – validation ongoing
- RNTuple shows improvements in file size and read speed w.r.t. TTree for DAOD_PHYS
- Similar to TTree, zstd compression seems to outperform lzma in terms of read speed
  - File sizes seem to be comparable, need to validate with more data sets
  - Comparison to other compression methods is planned
- We need further evaluation and benchmarking to understand current (performance) bottlenecks

Backup

Benchmark setup

- Single-core “analysis” using RDataFrame
- For 8 object containers, read the pt, eta, phi and mass
  - 32 branches/top-level fields in total
- Calculate the invariant mass (using `ROOT::VecOps`) and fill a histogram with the result
- Repeated 10 times, outliers removed

Hardware and software

Hardware

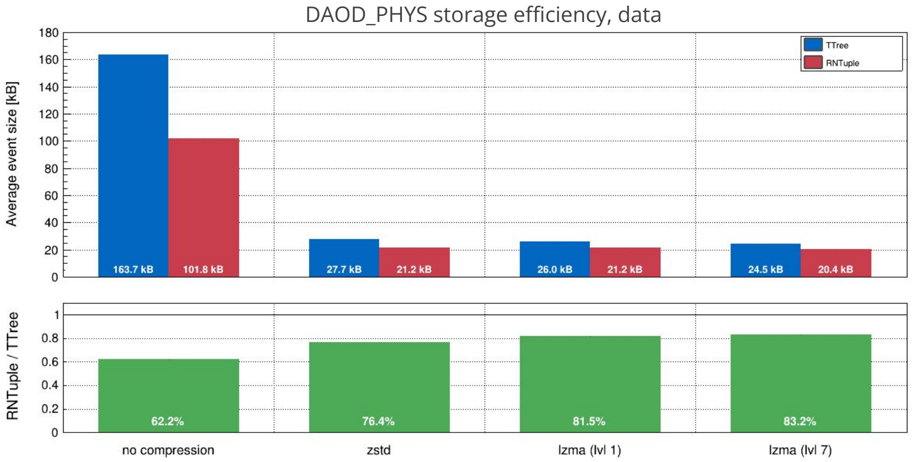
- CPU: AMD EPYC 7702P @ 2GHz, 128 logical cores
- RAM: 128GB DDR4 RDIMM 3200 MHz
- SSD: Samsung MZWLJ3T8HBLS-00007
- HDD: TOSHIBA MG07ACA14TE SATA, 7200 RPM
- Network: 100GBe

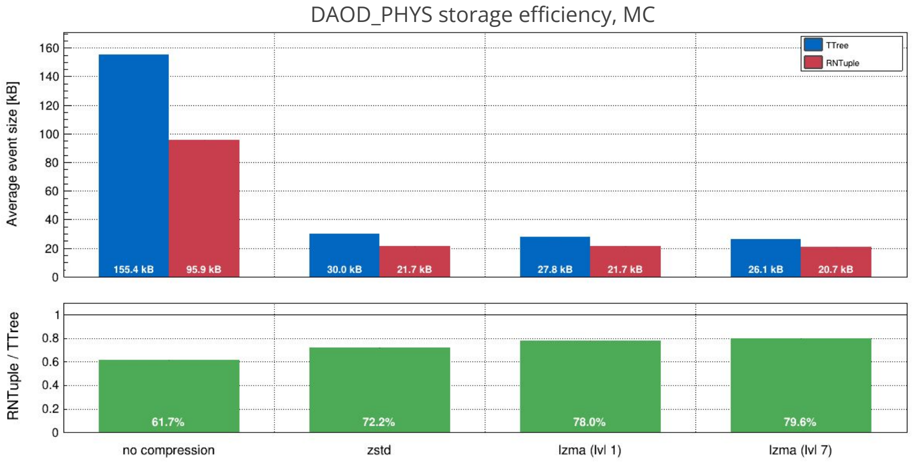
N.B.: *XRootD access from projects.cern.ch EOS instance (same datacenter)*

Software

- ROOT
- Benchmark code
- OS: AlmaLinux 9.1 with Linux kernel 6.3 from ELrepo (uring enabled)

Storage efficiency (incl. no compression)



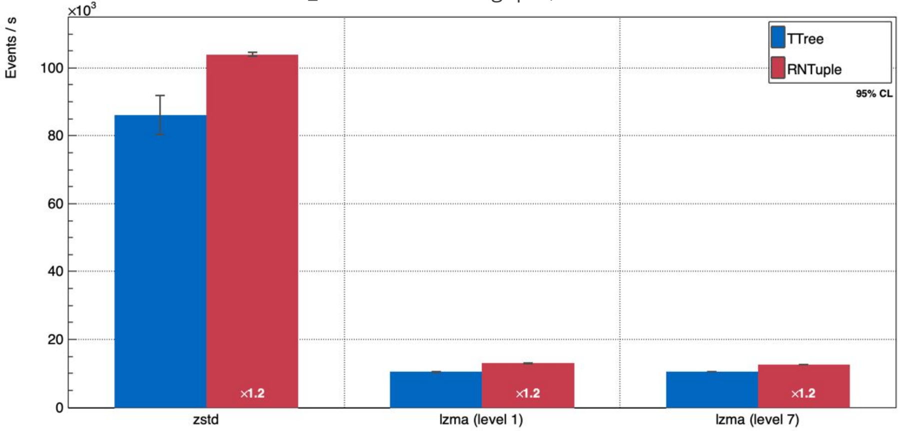


Why is the ratio RNTuple/TTree so much larger for uncompressed DAOD_PHYS?

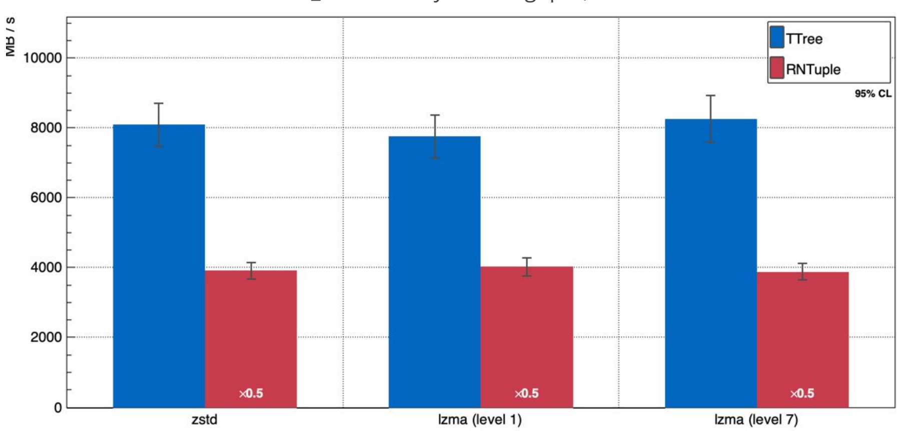
- DAOD_PHYS files contain a lot of std::vectors and other vector-like branches/fields
- Every std::vector<POD> needs 10 bytes more in TTree compared to RNTuple
  - Similar story for other types of STL(-like) collections
- Lots of redundant data, compresses away well – but not completely

Warm cache performance

DAOD_PHYS event throughput, warm cache



DAOD_PHYS raw byte throughput, warm cache



XRootD performance