Integration of RNTuple in ATLAS Athena

Florine de Geus\textsuperscript{1,2}  Javier Lopez-Gomez\textsuperscript{1}  Jakob Blomer\textsuperscript{1}  Marcin Nowak\textsuperscript{3}  Peter van Gemmeren\textsuperscript{4}

\textsuperscript{1}CERN  \textsuperscript{2}University of Amsterdam  \textsuperscript{3}Brookhaven National Laboratory  \textsuperscript{4}Argonne National Laboratory

CHEP 2023 – Norfolk, U.S.  May 8, 2023
Background and motivation

• **Athena**: ATLAS experiment software 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

[(D)AOD: (Derived) Analysis Object Data]
Background and motivation

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

ATLAS Preliminary
2022 Computing Model - CPU

- Conservative R&D
- Aggressive R&D
- Sustained budget model (+10% +20% capacity/year)

ATLAS Preliminary
2022 Computing Model - Disk

- Conservative R&D
- Aggressive R&D
- Sustained budget model (+10% +20% capacity/year)

RNTuple: evolution of ROOT's TTtree columnar data storage

Check out the previous talk for more
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 step

With these additions, RNTuple-based DAOD_PHYS(LITE) production is fully supported in Athena
RNTuple for ATLAS DAOD_PHYS

**Two central questions:**

1. How does RNTuple perform compared to TTree?
2. What else is needed for RNTuple to be fully adopted by Athena?
Two central questions:
1. How does RNTuple perform compared to TTree?
2. What else is needed for RNTuple to be fully adopted by Athena?

To answer these questions, we’ve evaluated DAOD_PHYS:
1. By creating RNTuple-based DAOD_PHYS files fully equivalent to their TTree counterparts...
   - Using a (sample of a) data set from real data and MC
   - Recompressing the original TTree-based samples with different algorithms using ROOT’s hadd
   - Converting these samples to RNTuples using ROOT’s RNTupleImporter
2. ...and comparing those in terms of storage efficiency and read throughput
Storage efficiency

![Graph showing storage efficiency for different compression methods and formats.](image)

**DAOD_PHYS storage efficiency, data**

- zstd: 76.4%
- lzma (lvl 1): 81.5%
- lzma (lvl 7): 83.2%

**DAOD_PHYS storage efficiency, MC**

- zstd: 72.2%
- lzma (lvl 1): 78.0%
- lzma (lvl 7): 79.6%
Storage efficiency

- Storage efficiency in line with other evaluations
  - DAOD_PHYS (almost) exclusively contains collections
- Potential optimization: using `std::vector<bool>` for selection flags instead of `std::vector<char>`
**Read throughput**

### Benchmarked with a (highly) artificial event loop using RDataFrame

- Restricted to reading `std::vector<float>` columns
- More representative benchmarks require additional RNTuple support in Athena

### Depending on storage medium, performance may become CPU-bound

- **SSD**
- **HDD**
- **RAM**
- **XRootD (100GbE, 0.3ms)**

**Throughput ratio (RNTuple / TTree)**

- DAOD_PHYS RNTuple/TTree event throughput ratio, data
Read throughput: SSD

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

---

**DAOD_PHYS RNTuple/TTree SSD event throughput**

- TTree
- RNTuple
- RNTuple (w/ io_uring)

**DAOD_PHYS RNTuple/TTree SSD raw I/O throughput**

- TTree
- RNTuple
- RNTuple (w/ io_uring)

Integration of RNTuple in ATLAS Athena

CHEP 2023 – Norfolk, U.S., May 8, 2023

8
Read throughput: SSD

CPU bound

DAOD_PHYS RNTuple/TTree SSD event throughput

DAOD_PHYS RNTuple/TTree SSD raw I/O throughput

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

DAOD_PHYS RNTuple/TTree HDD event throughput

- TTree
- RNTuple
- RNTuple (w/ io_uring)

95% CL

DAOD_PHYS RNTuple/TTree HDD raw I/O throughput

- TTree
- RNTuple
- RNTuple (w/ io_uring)

95% CL

Integration of RNTuple in ATLAS Athena
Read throughput: HDD

DAOD_PHYS RNTuple/TTree HDD event throughput

zstd is I/O-bound: effects of async I/O become apparent

DAOD_PHYS RNTuple/TTree HDD raw I/O throughput

Integration of RNTuple in ATLAS Athena

CHEP 2023 – Norfolk, U.S., May 8, 2023
Next steps

1. Explore storage efficiency across more DAOD_PHYS data sets
2. Explore more of the evaluation phase space
   ▶ Compression: lz4, lossy compression
   ▶ Storage backends: Intel DAOS, S3
   ▶ Data sources: more (XRootD) latency configurations
   ▶ RNTuple I/O parameters: page and cluster sizes
3. Benchmark with (multiple) representative analyses
4. Evaluate also with DAOD_PHYSLITE
5. Performing large-scale, distributed tests
6. Bonus: Evaluate RNTuple with other stages of the ATLAS data production pipeline
• Support for RNTuple in ATLAS Athena is almost complete: validation ongoing
• RNTuple shows improvements in file size and read throughput w.r.t. TTree for DAOD_PHYS
• Similar to TTree, zstd outperforms lzma in terms of read throughput
  ▶ File sizes are comparable, validation with more data sets is necessary
  ▶ Comparison to other compression methods still planned
• We need further evaluation and benchmarking to understand current (performance) bottlenecks...
Summary and concluding remarks

- Support for RNTuple in ATLAS Athena is almost complete: validation ongoing
- RNTuple shows improvements in file size and read throughput w.r.t. TTree for DAOD_PHYS
- Similar to TTree, zstd outperforms lzma in terms of read throughput
  ▶ File sizes are comparable, validation with more data sets is necessary
  ▶ Comparison to other compression methods still planned
- We need further evaluation and benchmarking to understand current (performance) bottlenecks...

...but the benefits of using RNTuple for ATLAS are already apparent
Read throughput benchmark setup

- Single-core “analysis” using RDataFrame
- For 8 object containers, read the $p_T$, $\eta$, $\phi$ and $m$ columns
  - Stored as `std::vector<float>`
  - Ergo, we read 32 branches (TTree) or top-level fields (RNTuple) in total
- Calculate the invariant mass (using `ROOT::VecOps`) and fill a histogram with the results
  - Done for every event, no cuts applied
- Each benchmark is repeated 10 times, outliers are removed
Hardware and software setup

**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**  100 Gbit/s Ethernet

*N.B.*  XRootD access from projects.cern.ch

EOS instance (same datacenter)

**ROOT**  [GitHub](#)

**Benchmarks**  [GitHub](#)

**OS**  AlmaLinux 9.1 with Linux kernel 6.3 from ELRepo (uring enabled)
Storage efficiency (with uncompressed)

DAOD_PHYS storage efficiency, data

DAOD_PHYS storage efficiency, MC

Integration of RNTuple in ATLAS Athena

CHEP 2023 – Norfolk, U.S., May 8, 2023
Storage efficiency (with uncompressed)

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

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

DAOD_PHYS storage efficiency, data
Read throughput: RAM

DAOD_PHYS RNTuple/TTree RAM event throughput

DAOD_PHYS RNTuple/TTree RAM raw I/O throughput
RNTuple uses a different code path than TTree, XRootD access is not yet optimized for this.

**DAOD_PHYS RNTuple/TTree XRootD event throughput**

- **zstd** (x1.4)
- **lzma (level 1)** (x1.2)
- **lzma (level 7)** (x1.2)

**DAOD_PHYS RNTuple/TTree XRootD raw I/O throughput**

- **zstd** (x0.7)
- **lzma (level 1)** (x0.8)
- **lzma (level 7)** (x0.8)