Status of ROOT I/O in ATLAS

Peter van Gemmeren (ANL)

Big picture

- Things concerning ROOT I/O in ATLAS work reasonably well
- But longer term things are changing and ATLAS ROOT I/O will have to evolve:
  - More fine-grained event processing (EventService, AthenaMP) could benefit from efficient event data sharing.
  - Utilize more remote data reading
  - Last but not least higher level of concurrency using multi-threaded processing
- Storage is becoming more the cost driver for ATLAS computing and needs to be addressed.
  - Including new compression settings/algorithms in ROOT

Event Data Sharing

- ATLAS uses multi-process jobs with shared reader/writer
- Rather inefficient due to additional serialization/de-serialization needed to share objects
- Would be beneficial is one could separate interface to read serialized objects.
  - ROOT knows about this request, but requires rather complex changes.

Move to multi-threaded framework AthenaMT

- For Run 3, ATLAS is moving to a multi-threaded version of its event processing framework: AthenaMT
- Currently, I/O (including ROOT) are not show stoppers (e.g. there appear no crashes), but because most of the execution is serial it may become a bottleneck for some workflows.

AthenaMT: Multiple events in flight processing

- AthenaMT will process many events simultaneously in flight.
- Together with on-demand reading of data, this will result in non-sequential access to branch entries in ROOT trees:
  - ATLAS reads data via: `branch->GetEntry(n);`
- For efficient reading, ATLAS relies on ROOT TTreeCache to avoid large number of small disk reads.
  - The ATLAS xAOD event data model fragments an event into container with static auxiliary stores and many dynamic auxiliary data members (~simple type vectors) that have to be written to individual ROOT branches (>1,000).
- In serial (or AthenaMP) access ATLAS I/O performance (including ROOT) is sufficient, but cache and buffer transitions may cause problems for non-sequential/backward reading.

AthenaMT: non-sequential data access

- TTreeCache reading and TBranch basket de-compression will likely not work efficiently with out-of-sequence branch access.
  - Could be improved by using overlapping caches or keeping multiple copies.
- Argonne has a DOE SULI summer student: David Clark who is starting to look into this