ATLAS Event Storage News

- 64bit integer support in TTreeFormula, TLeafs and TTreeIndex

  Marcin Nowak (BNL)

- Fixes for TTreePerfStats

  Peter Van Gemmeren (ANL)

- New event data storage format: xAOD

  Attila Krasznahorkay

TTreeIndex 64bit

- ATLAS is planning to extend the current data navigation model based on POOL Tokens with navigation based on Run+Event number (which are going to be 64bit values)
  - Why: POOL Tokens are generic object references that one needs to keep somewhere (TAG database)
  - No support for object relocation (file merging)
- We want to use TTreeIndex to locate events in a file
- TTreeIndex is a composite index built from two 31bit integer values (calculated from each TTree row using two TTreeFormulas) and then glued together into one 64bit int – based index
- ATLAS is contributring manpower to improve handling of 64bit ints in ROOT
- Current status of work: 64bit version of TTreeIndex is implemented
  - using 2 separate 64bit vectors for the 2 index values (not using 128bit ints)
  - code in a private ROOT branch (will be moved to github)
  - full schema evolution support needs to be added (soon)

TTreeFormula and 64bit ints

- T[Tree]Formula evaluates expressions using floating point numbers (64bit)
  - 64bit floating point format can not store 64bit integers!
    - It works up to ~62bits, errors are rare and unexpected
- ATLAS needs 64bit support for Event Index and for 64bit words used to store trigger decision
  - ATLAS TAG utilities are using a fixed version of TTreeFormula based on long double since for some time
- Proposal to use long double type in ROOT was not accepted (performance reasons) – decided to implement template specializations for double, long double and Int64

```
template<typename T> T  TTreeFormula::EvalInstance()
```

TLeaf and 64bit

- Fixing TTreeFormula only helps with evaluation of the expression, but data is read from TTree using the common GetValue() API that already converts to double
  - Solution: template specialization GetTypedValue<T>()
  - Required chagnes in:
    - Tleaf, TLeaf specializations, TLeafElement, TBranchElement, TStreamerInfo, TLeafInfo and about 15 TLeafInfo subclasses
- Status: changes implemented and in testing, awaiting evaluation
  - We hope to have the changes accepted soon for a dev release

Fixes for TTreePerfStats

- TTreePerfStat is a very helpful tool to monitor TTree performance.
- Unfortunately it is not quite correct:
  - Monitors I/O performance on TFile level, assuming that all reads are for a single TTree.
- Two step approach to improvement:
  1. Make TTreePerfStats work correctly for single TTree.
     - ROOT patch has been provided by ATLAS
  2. Support multiple TTreePerfStats.
     - Under development with limited effort resources…

The xAOD File Format

- ATLAS is working on a new file format for the output of its reconstruction. The new formats needs to:
  - Provide the ability to save the same information as the AODs used in Run 1
  - Provide the same flexibility with file content manipulation as the flat ntuples that we used for practically all physics analyses in Run 1
  - Provide support for (auto-)vectorization in the reconstruction software
- Done by creating separate data storage classes (auxiliary classes) and “interface classes” that provide a user-friendly interface to the information
  - The auxiliary classes are SoA (StructOfArray), while the interface objects provide access to the information in an AoS (ArrayOfStruct) manner

xAOD Writing/Reading

- Want to use these files up to a late stage in analysis
  - Require good performance for reading a small portion of the event data
    - Done by selectively & lazily reading the auxiliary information from the input
  - For this, we have to write the auxiliary properties either as individual simple (std::vector<T>) branches, or as a SoA object with a split level of 1
    - In both cases the created TTree may have O(10k) branches
- Handling this many branches in analysis is “doable”, as rest of the analysis job doesn’t use much memory usually.
  - However, need to be able to write this many branches with reasonable memory requirements in reconstruction type jobs as well.
- Current TTreeCache infrastructure seems to be adequate for serving such jobs
  - While not widely used, did use TTreeCache in some analysis jobs that ran on O(10k) branches.