ATLAS Event Store and I/O developments in support for Production and Analysis in Run 3

Marcin Nowak[1,*,†], *Peter* van Gemmeren[2], and *Jack* Cranshaw[2]

[1]Brookhaven National Laboratory, Upton, New York 11973, USA
[2]Argonne National Laboratory, Argonne, Illinois 60439, USA

Abstract. During the long LHC shutdown, ATLAS experiment is preparing several fundamental changes to its offline event processing framework and analysis model. These include moving to multi-threaded reconstruction and simulation and reducing data duplication during derivation analysis by producing a combined mini-xAOD stream. These changes will allow ATLAS to take advantage of the higher luminosity at Run 3 without overstraining processing and storage capabilities. They also require significant improvements to the underlying event store and the I/O framework to support them. These improvements include: 1) an overhaul of the Run 2 I/O framework to be thread-safe and minimize serial bottlenecks, 2) introduction of new immutable references for object navigation, which don't rely on storage container entry number so data can be merged in-memory, 3) using filter decisions to annotate combined output stream to allow for fast event selection on input and 4) selecting optimized compression algorithms and settings to allow efficient reading of event selections.

1 Introduction

The ATLAS[1] experiment is one of the 4 experiments collecting data from the LHC at CERN. The LHC operation schedule consists of periods of activity (Runs) interspaced with breaks intended for hardware maintenance and upgrades (Long Shutdowns). Each consecutive Run generates an increasing amount of physics data that needs to be processed. The experiment software has to evolve to keep up with the increasing processing needs. The upcoming LHC Run 3 and especially Run 4 will require significant software performance improvements. In this paper we describe the developments in the ATLAS Event Store and the I/O framework that were done to fulfil these needs.

ATL-SOFT-PROC-2020-015
18 February 2020

* Corresponding author: mnowak@bnl.gov

2 Major software improvements

Taking advantage of the currently on-going Long Shutdown 2, the software team is implementing 4 major improvements to the ATLAS offline event processing framework Athena[2] and analysis mode.

2.1 Thread safety and concurrency in the I/O framework

One of the biggest changes in the Athena framework for Run 3 is the introduction of parallel Event processing using multi-threading (AthenaMT)[3]. In AthenaMT multiple Events are processed at the same time, each in a dedicated Event Store, and then written out [Figure 1]. The I/O framework had to be adapted to accept overlapping write requests coming from the Stores, providing thread safety and allowing as much concurrency as possible in order to ensure good performance. Thread safety was achieved by identifying and protecting, with thread locking mechanisms, the critical sections of the components that have only once instance in the application. Examples are the central Conversion Service and accompanying POOL Service. It was also necessary to eliminate state information stored in these components. Concurrent execution was achieved by creating multiple copies of components that can operate independently and by allowing subcomponents, already existing in multiple instances (e.g. converters specialized by converted object type), to work in parallel.

Certain natural restrictions to concurrency come from the employed storage technology – in Athena writing to a given output file is done one Event at a time, to preserve integrity. Concurrent writing can still be performed on different files.

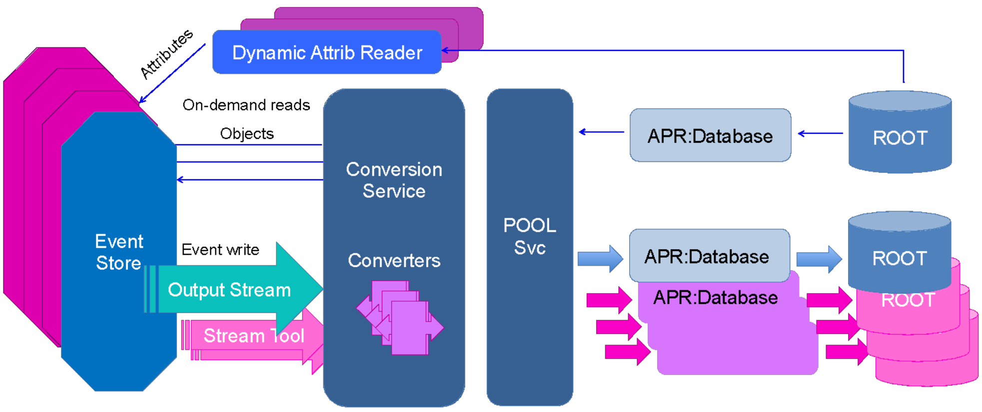

Figure 1. I/O Framework evolution from Single-Process (blue) to Multi-Threaded (all).

Reading Events from input is done sequentially in AthenaMT. However, certain data elements are read on-demand during processing and such requests can happen concurrently. If the requests demand access to the same file, they are serialized.

2.2 Object indexing in the storage layer

Athena persistent storage technology (POOL/APR) is based on the concept that every data object (in the C++ sense), when it is stored, it is assigned an immutable reference that can be used to retrieve it later. Internally, references are broken into 2 parts: the object ID in the storage container and all the reminding information, which is the same for all objects in a given container. That information is kept aside in the "Links" table as a means to reduce space usage. Object references are then represented as a pair of two numbers – object ID and the entry position to the Links table – called a POOL/APR Token [Figure 2].

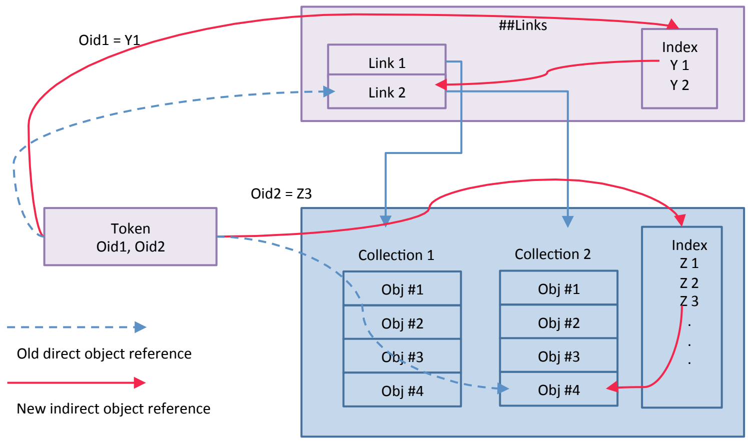

Figure 2. Indirect object access with indexing.

There are two production use cases where having direct object references is a disadvantage – file merging and concurrent writing to the same file. When merging files, the objects change their position and thus their object ID. During concurrent writing, the final object position is not known (due to buffering and delayed writing). To address both issues, we added an additional level of indirection for the Tokens in the form of in-file index (TTreeIndex). The index can be automatically updated by ROOT[4] during file merging, while the original Tokens stay unchanged. For concurrent writing, having an indirection layer allows to pre-assign logical object IDs that are later redirected to the real object IDs.

2.3 Combined output stream with Event tagging

The ATLAS Run 3 production model assumes using a single combined derivation output stream in order to avoid event data duplication, observed among Run 2 separate streams, and save disk space. Events in the common stream are marked as belonging to any of the derivation streams. For every Event additional values relevant to stream assignment

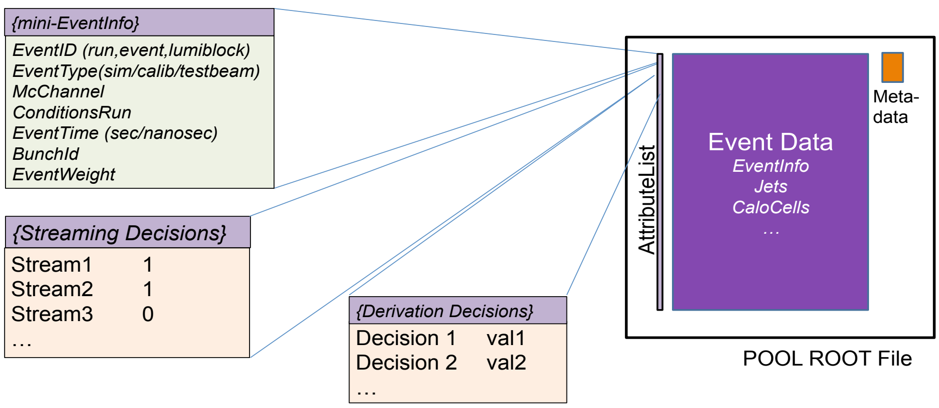

Figure 3. Out-of-band Event-level metadata in a data file for fast Event selection on input.

decision can be stored. The most essential Event identification information in the form of a mini-EventInfo is also present [Figure 3].

Keeping this information outside the main Event body in a simple, easily readable ROOT format, permits the Main Event Loop to efficiently read only events belonging to the desired derivation stream, with additional possibilities for selection using the stored attributes.

2.4 Compression optimization for efficient selective reading

The decision to combine all derivation streams together in the same file (described in the previous section) introduces a change in the reading pattern - from mostly sequential to selective, with varying levels of selectivity. Selective reading of objects from ROOT files can be less efficient, due to reading and decompressing entire buffers, which can contain multiple events, even when only a fraction of that data had been requested.

During Run 1 and Run 2, ATLAS was using the standard ROOT compression algorithm ZLIB. For Run 3, we evaluated another algorithm - LZ4 – supported by the recent ROOT releases and the default in the new releases. We found that LZ4, thanks to it's very fast decompression [Figure 4], makes it possible to read data at 40%-100% of the rate of pre-selected samples [Figure 5], much faster than with ZLIB, while retaining most of the size reduction advantages of ZLIB compression.

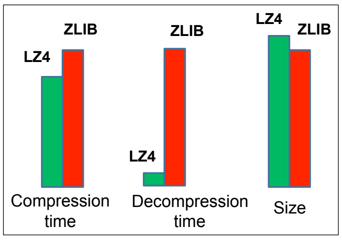

Figure 4. Relative performance comparison between LZ4 and ZLIB compression algorithms (in arbitrary units) for ATLAS Event data files

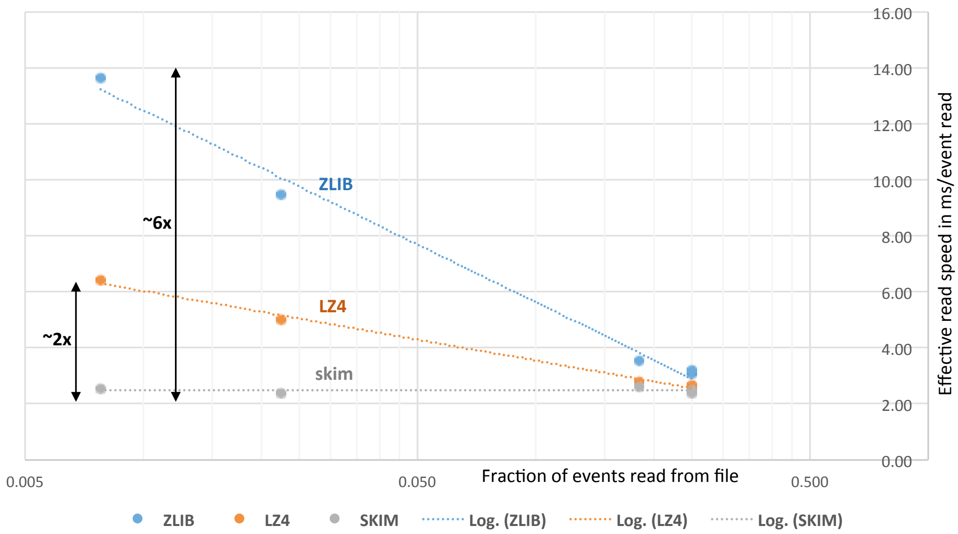

Figure 5. Effects of different compression algorithms on selective reading performance compared to 100% read (skim). The read time per event is constant for skimmed data file as it contains only preselected events. Read time per event goes up as selectivity increases for both ZLIB and LZ4 because there are less and less relevant events in every decompressed block.

3 Conclusions

ATLAS Computing is undergoing important changes to meet the challenges of LHC Run 3 data handling and processing. The offline software framework Athena, and in particular its I/O components, has evolved to support these changes. The framework foundations have achieved stability in the multi-threaded environment, which provides the ground for adaptation and testing of physics algorithms and other framework components. Data storage format was modified to achieve better balance between disk space and performance. Object referencing was made more robust. The work on the I/O layer continues to increase concurrency on writing and increase performance.

Acknowledgements

The work is funded in part by the U.S. Department of Energy, Office of Science and High Energy Physics contracts.

References

1. ATLAS Collaboration (2008) "*The ATLAS Experiment at the CERN Large Hadron Collider*", J. Inst. 3, S08003 https://iopscience.iop.org/article/10.1088/1748-0221/3/08/S08003
2. ATLAS Collaboration (2019) Athena (Version 22.0.1) Zenodo http://doi.org/10.5281/zenodo.2641997
3. G. A. Stewart et al (2016) "*Multi-threaded software framework development for the ATLAS experiment*" J. Phys.: Conf. Ser.**762** 012024 https://iopscience.iop.org/article/10.1088/1742-6596/762/1/012024
4. R. Brun, F. Rademakers (1996) "*ROOT: An Object Oriented Data Analysis Framework*" AIHENP'96 Workshop Nucl. Inst. & Meth. In Phys. Res. A 389 81-86. http://root.cern.ch