New developments in file-based infrastructure for ATLAS event selection

P van Gemmeren[1], D M Malon[1], M Nowak[2]

[1]Argonne National Laboratory, Argonne, Illinois 60439, USA

[2]Brookhaven National Laboratory, Upton, NY 11973-5000, USA

E-mail: gemmeren@anl.gov

Abstract. In ATLAS software, TAGs are event metadata records that can be stored in various technologies, including ROOT files and relational databases. TAGs are used to identify and extract events that satisfy certain selection predicates, which can be coded as SQL-style queries. TAG collection files support in-file metadata to store information describing all events in the collection. Event Selector functionality has been augmented to provide such collection-level metadata to subsequent algorithms. The ATLAS I/O framework has been extended to allow computational processing of TAG attributes to select or reject events without reading the event data. This capability enables physicists to use more detailed selection criteria than are feasible in an SQL query. For example, the TAGs contain enough information not only to check the number of electrons, but also to calculate their distance to the closest jet--a calculation that would be difficult to express in SQL. Another new development allows ATLAS to write TAGs directly into event data files. This feature can improve performance by supporting advanced event selection capabilities, including computational processing of TAG information, without the need for external TAG file or database access.

1. Introduction

In ATLAS, TAGs are event metadata records with about 200 attributes, together with persistent pointers to event data at various stages of processing that can be stored in various technologies, including ROOT [1] files and relational databases. TAGs are used to identify and extract events that satisfy certain selection predicates, which can be coded as SQL-style queries. ATLAS uses the LCG POOL [2] software as a foundation for TAG implementation, with sets of TAGs instantiated as POOL Collections. POOL Collections can use different storage technologies: ROOT files via POOL Collections, Oracle via POOL, and MySQL via POOL (limited). An overview of new developments for ROOT file-based TAGs is given in this paper.

2. TAG collection support for in-file metadata

It is often useful and sometimes necessary to associate metadata with collections of events. Since Version 2.8, POOL Collections support collection-level metadata. Support for POOL Collections in-file metadata was implemented in the AthenaPOOL I/O framework by extending the infrastructure supporting in-file metadata for event data files [3].

2.1. Stored as Key/Value map<string, string>

POOL Collections are a generic technology and are agnostic with respect to the data types they collect. For this reason, and to preserve persistence technology independence, POOL Collections store in-file metadata as key/value pairs in a STL `map<string, string>`.

2.2. Recording TAG file metadata in StoreGate

The Atlas software framework Athena [4] uses a transient store called `StoreGate` [5] to exchange information between different modules. In the Atlas Event Data Model a new data object class (`CollectionMetadata`) was introduced to allow in-file metadata to be recorded into the transient event store by the ATLAS event selector (`EventSelectorAthenaPool`).

2.3. AthenaPOOL fires incident to alert for begin TAG file, end TAG file

Correct processing of in-file metadata coming from a variety of sources, from TAG files as well as from event data files ("payload" files), requires a clear model for handling such metadata, and for distinguishing the sources. Because opening and closing new input (payload) files and TAG files are not state transitions in the Athena framework, ATLAS relies on firing incidents to notify potential clients that new files have been opened or closed. In addition to firing incidents for begin and end Input file, the event selector now fires incidents to alert for begin and end TAG file (see Figure 1).

When reading event data via TAGs, the event selector will open the Collection file, read the in-file metadata and record it into a transient metadata store, then a `BeginTagFile` incident is fired and clients which listen to these incidents can retrieve the data. Before opening a new Collection, an `EndTagFile` incident is fired to allow clients to retrieve the in-file metadata, before the store is cleared and the new data is read.

When reading via TAGs, event data files are opened implicitly by POOL when the first payload object (i.e. event `DataHeader`) is retrieved from that file. The event selector will detect event file transition by checking the GUID within the TAG and fires begin and end Input file incidents as needed.

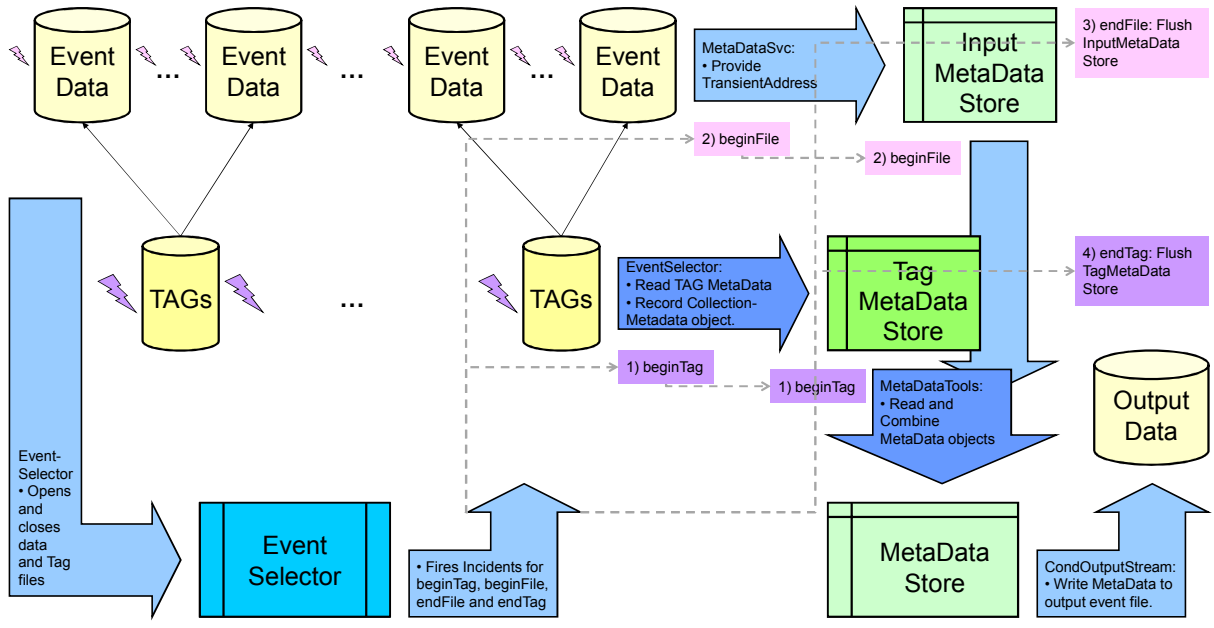

Figure 1. The AthenaPOOL framework has been augmented to support both event file and TAG file metadata.

3. Computational processing of TAG attributes

The ATLAS I/O framework has been extended to allow computational processing of TAG attributes, using C++ algorithms rather than just SQL-like queries, to select/reject events without reading the event data (not even `DataHeader` or `EventInfo`).

3.1. Preselecting events

This capability enables physicists to preselect events with more detailed selection criteria than are feasible in an SQL query. For example, the TAGs contain enough information to not only check the number of electrons, but also calculate their distance to the closest jet. Iterative and procedural computations are difficult in SQL. Data stored in TAGs suffice for invariant mass computations. A selection criterion could be whether an event contains a combination of jets with an invariant mass within a certain range. Computations like this are straightforward in C++, but combinatorics are very hard to implement with SQL queries.

3.2. Implementation

The details of the implementation are a bit specific and involved for a general paper. The basic concept is shown in Figure 2: The TAG is made available to selection tools at the beginning of the `next()` iteration of the event loop, prior to retrieval of any bulk event data or event root objects in `loadAddresses()`. If any of the extensible list of selection tools indicate that the event should be skipped, `the` event selector repeats the `next()` execution until an event is selected. Only at this point is the event data loaded and the event loop executed.

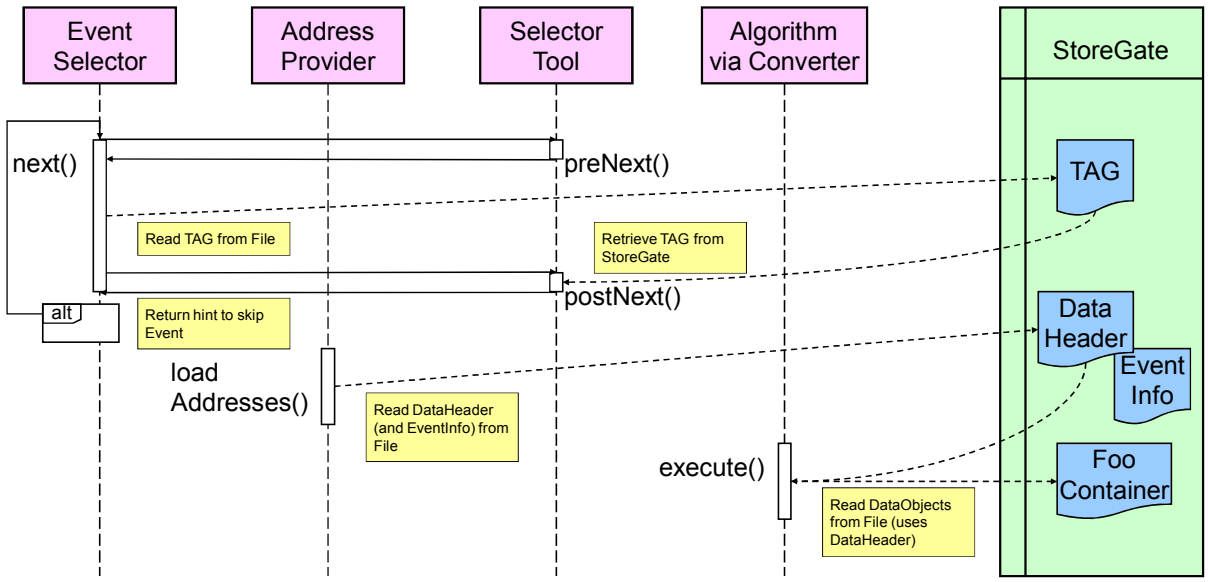

Figure 2. Computational processing of TAG attributes enables physicists to reject events using complex queries without reading the event data.

3.3. Performance gains

Preliminary studies show significant performance gains when using computational processing of TAG attributes versus using the AOD (or even ESD) content to make the same selection. Deselecting events based upon TAG content requires only reading of the TAG data, which is ~1KB/event and allows rejection of an event before entering the event loop.

Using event payload data for selection requires not only reading the containers needed to evaluate the selection criteria (for example, one needs to read the electron container to determine its size,

whereas this is a single attribute in the TAG), but in addition the `DataHeader` (which gives access to the persistent event data objects) and an `EventInfo` object need to be read; these containers have a size of ~5KB/event and ~1KB/event respectively.

A speedup of more than a factor of 10 for rejecting events has been observed in tests. The overall analysis speed improvement depends on the fraction of events rejected, and highly selective analyses show the greatest gains.

TAG selection SQL queries can be faster for simple criteria, as they only retrieve the attributes necessary to evaluate the given query. However, for more complex queries computational preprocessing becomes advantageous.

4. TAGs within event data files

Event selection with TAGs is becoming increasingly popular in ATLAS.

4.1. It would be nice to be able to apply the TAG tools directly to AOD

TAG selection functionality is rapidly gaining in importance as ATLAS analyses shift away from single stream Monte Carlo samples to more realistic mixed samples and ultimately real data. ATLAS has successfully prototyped machinery to write TAGs directly into the AOD files (see Figure 3), in addition to writing them into independent TAG datasets and relational databases. There are a number of advantages to doing this:

- Having TAGs written to AOD files provides the benefits of TAG functionality without auxiliary Collection files. For example, SQL queries or computational pre-processing could be done directly on the AOD.
- The current computing model foresees that TAGs are produced during the AOD merge step so a copy of the TAGs could be added to the merged AOD file upon production at no significant operational cost or significant change to the workflow.
- The small size of the TAGs compared to AOD (less than 1%) allows ATLAS to add them without significantly affecting disk requirements.
- ATLAS uses POOL Collections for TAGs and POOL's `RootStorageSvc` for event data.

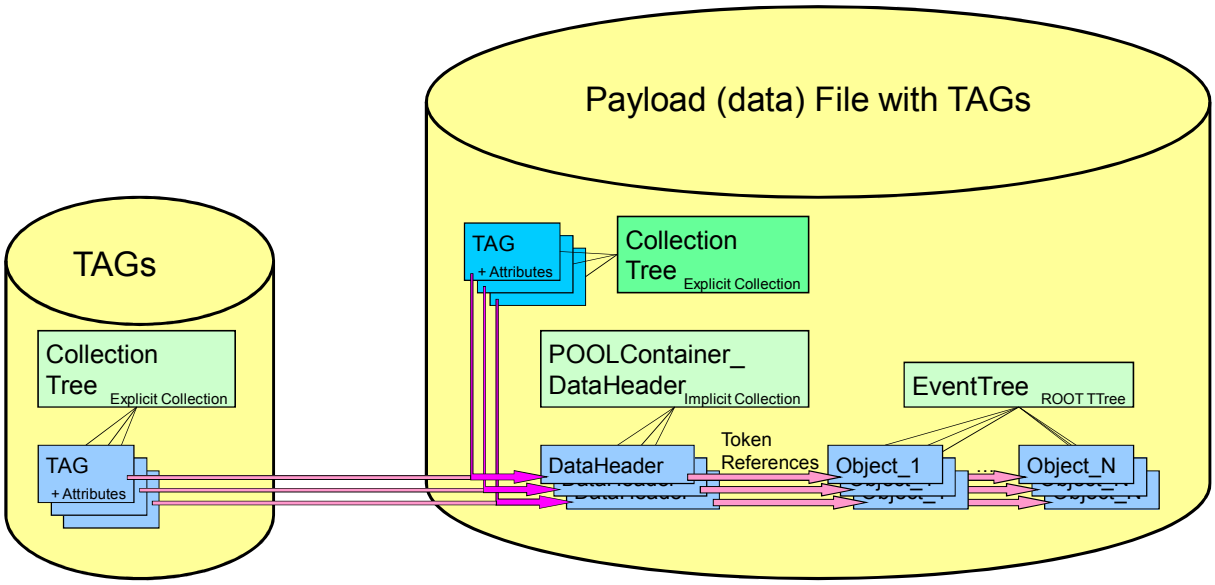

Figure 3. Writing TAGs into event data files allows using TAG event selection capabilities on the data file.

4.2. Issues are mainly technical

While ATLAS has successfully prototyped the writing of TAGs into AOD or other event payload files, this capability is not yet in ATLAS software releases. The open issues are mainly related to the synchronization of multiple writers to the same output file. A possible solution is to write TAG collections during finalize, after event data are done. A separate issue of distinguishability of payload and TAG Collection containers, which are currently both named 'CollectionTree' can be resolved by renaming the payload tree (e.g., to 'EventTree'). The name of the POOL container is configurable and hidden to the Athena client by the `DataHeader` so this change will be transparent.

5. Conclusion

New functionality for ROOT file-based TAGs has been added to ATLAS software. TAG files now support in-file metadata utilizing the same framework as in-file metadata for event data files. This allows provenance records to be propagated to event collections. Augmenting the existing capability of selecting events by issuing SQL-like queries on TAGs, a new computational selection tool allows implementation of selection criteria as C++ code. As a future development, ATLAS will add the capability of writing TAGs directly into event data files to allow transparent use of TAG functionality without requiring access to auxiliary files or databases.

References

[1] http://root.cern.ch

[2] http://pool.cern.ch

[3] D Malon, P van Gemmeren, R Hawkings and A Schaffer, *"An inconvenient truth: file-level metadata and in-file metadata caching in the (file-agnostic) ATLAS event store"*, 2008 *J. Phys.: Conf. Ser.* **119** 042022 (7pp)

[4] https://twiki.cern.ch/twiki/bin/view/Atlas/AthenaFramework

[5] P. Calafiura, C.G. Leggett, D.R. Quarrie, H. Ma, S. Rajagopalan, "*The StoreGate: A Data model for the Atlas software architecture"*, 2003 *CHEP-2003-MOJT008, ATL-SOFT-2003-009*

Acknowledgments

Argonne National Laboratory's work was supported by the U.S. Department of Energy, Office of Science, Office of Basic Energy Sciences, under contract DE-AC02-06CH11357.