Next-Generation Navigational Infrastructure and the ATLAS Event Store

P van Gemmeren¹, D Malon¹ and M Nowak² on behalf of the ATLAS Collaboration
¹Argonne National Laboratory, Argonne, Illinois 60439, USA
²Brookhaven National Laboratory, Upton, New York 11973, USA

E-mail: gemmeren@anl.gov

Abstract. The ATLAS event store employs a persistence framework with extensive navigational capabilities. These include real-time back navigation to upstream processing stages, externalizable data object references, navigation from any data object to any other both within a single file and across files, and more. The 2013-2014 shutdown of the Large Hadron Collider provides an opportunity to enhance this infrastructure in several ways that both extend these capabilities and allow the collaboration to better exploit emerging computing platforms. Enhancements include redesign with efficient file merging in mind, content-based indices in optimized reference types, and support for forward references. The latter provide the potential to construct valid references to data before those data are written, a capability that is useful in a variety of multithreading, multiprocessing, distributed processing, and deferred processing scenarios. This paper describes the architecture and design of the next generation of ATLAS navigational infrastructure.

1. Introduction

1.1. ATLAS Software & Event Data Model
The ATLAS experiment at CERN [1] uses the Athena software framework [2] to run simulation, reconstruction, and analysis by iterating over data associated with a single collision of proton bunches, called events. To accomplish these tasks, Athena runs a sequence of algorithms (which can use tools and services) that can exchange data via a transient store called StoreGate [3]. Having a common in-memory database (or "blackboard"), from which algorithms can access data using the type and a string key greatly reduces the coupling among software components (see diagram in figure 1).

StoreGate allows software modules to transparently use a data object that was created earlier in the same job or read from disk. A proxy defines and hides the cache-fault mechanism: Upon request, a missing data object instance can be created and added to the transient data store, retrieving it from persistent storage on demand.

Event data typically undergo a sequence of processing stages, each of which may yield an output data product. Event data from LHC collisions are reconstructed to produce event summary data, from which analysis object data are in turn extracted, from which a variety of analysis data products are derived.
1.2. Athena POOL / ROOT Persistency Framework

Athena software uses ROOT I/O [4] via the ATLAS POOL persistency framework [5], which provides high-performance and highly scalable object serialization to self-describing, schema-evolvable, random-access files. POOL, a product that originated as a common effort among the LHC experiments, is being directly incorporated into the Athena code base. POOL software employs a technology-independent approach, supporting both file-based and relational technologies with a single consistent API:

1. Object streaming (e.g., ROOT I/O), addresses persistency for complex C++ objects such as high energy physics event data
2. Relational databases (e.g., MySQL and Oracle) provide distributed, transactionally consistent, concurrent access to data that may need to be updated.

In practice, ROOT I/O is the only technology currently supported by POOL for object streaming into files. ATLAS stores its data in ROOT Trees.

2. References / Token

ATLAS I/O and persistence infrastructure supports extensive navigational capabilities [6].

2.1. Real-time back navigation to upstream processing stages

While a job is ordinarily configured to use data from a particular processing stage as input, any Athena job can be further configured to allow retrieval of data objects from earlier (“upstream”) processing stages on demand. This feature allows on-demand recalculation of certain quantities derived from such upstream data, or retrieval of different object versions (from different files) into StoreGate transparently. While this capability has not been widely used by physicists to date, with the widespread adoption of wide-area data access technologies such as FAX and xrootd [7], such on-demand access to upstream data, even when it resides offsite, becomes an increasingly viable option.
2.2. Externalizable data object references
ATLAS uses object and event references that can be represented as a string and exported to other applications, such as event TAGs [8]. Event TAGs are event-level metadata records, useful for event selection, that are uploaded to a database. Physicists can use the TAG database to apply event selection criteria (e.g., trigger decisions) of their choice and create a file with references to the selected events only. This output file can be used to process any data product to which the TAGs contain references in a way that is equivalent to directly processing the entire file-based dataset itself, except that only selected events are read.

2.3. Navigation from any data object to any other both within a single file and across files
The ATLAS event data model consists of a large and heterogeneous assortment of objects, with unidirectional (and potentially bi-directional) associations among them (e.g., between an electron object and a track object). During execution, these associations can be made using C++ pointers, but such pointers lose their meaning when the program terminates. For this reason, inter-object relationships are implemented in the ATLAS transient data model using special reference classes, which manage associations by tracking the StoreGate identification (class ID and key) of pointed-to objects, and additionally, for objects stored in a container, the index into it. Following such a link triggers the same object retrieval machinery as is triggered by a direct attempt to access an object in the transient store.

To support these navigational capabilities, Athena POOL uses references also known as Tokens. These Tokens store the storage technology, database identifier, persistent class identifier, container name, and two offsets (see Table 1). RAW data files are written in ByteStream format, which cannot be accessed via POOL, but Tokens are used to reference these events as well. The ATLAS navigational infrastructure supports a common model for referencing and accessing data in both technologies.

Table 1. Components of the Token in different technologies.

| Generic | ROOT (Tree) | ROOT (Dir) | RDBMS | RAW |
|---------|-------------|------------|-------|-----|
| Database | File        | File       | Database | File |
| Container | Tree/Branch | Directory  | Table   |     |
| Item ID | Entry Number | Unique key | Primary Key | File Offset |

For reasons of deployment optimization, two different types of Tokens are employed for file-based data. A Container Token stores the technology, database, and container name for each container in an internal table (called ‘##Links’). A Full Token additionally stores two additional values, a link number and an object identifier. The first represents the identifier for the row in the Links table for the corresponding container Token and the second gives the entry number into the container for the referenced data object.

![Diagram of Token and Link table](image)

Figure 3. POOL Token references and Link table.
3. Future Enhancements to the current navigational infrastructure

Until now, ATLAS references have been content-agnostic: Their structure does not depend upon the type of the pointed-to object, and even their content has not depended upon the object type, beyond recording a persistent class identifier. Whether a ROOT Tree contains events or calibrations, for example, one would point to the nᵗʰ object in the tree in exactly the same way.

The infrastructure is flexible enough to support the addition of reference types optimized for specific kinds of data. References to event entry points, for example, may be designed to use unique, immutable {run number, event number} pairs (a sort of primary key for event data, even when such data are resident in files) as object identifiers. Such content-based reference types allow optimization for a number of purposes.

Because {run number, event number} pairs are unique within a processing stage and data stream (i.e., no events share the same values, nor is the same event replicated multiple times) they are an appropriate foundation upon which to build an index. Because they are immutable (their values do not change from one processing stage to the next, nor during reprocessing of any kind) they are well suited for standard navigational use cases and support the potential for reference simplification, and even for forward referencing.

For data taking after 2015, ATLAS will use content-based indices in optimized references. References based upon immutable attributes of the event data content could be indicated in the Token in its technology attribute. The Token must be adapted to store values for the corresponding content (e.g., {run number, event number} pairs), rather than just a single offset (OID2). Also, the underlying storage technology (ROOT) will need to support efficient lookup with these attributes.

3.1. Enhancements to ROOT to support content-based indexing

Efficient implementation of content-based references requires the ability to implement an index in the underlying storage technology. Without this, such references might be made to work, but navigation could require time linear in the number of events (or objects, more generally) contained within a referenced file, and could further require domain-specific knowledge in order to locate the referenced object.

ROOT supports an in-file indexing capability that provides the foundation for implementation of content-based references to event data in ATLAS, with \(O(\log N)\) performance. Because ATLAS event numbers will be 64-bit integers in the coming runs, a number of enhancements are required, both to ROOT indexing and to its underlying formula evaluation infrastructure, in order to support \{run number, event number\}-based references. ATLAS will contribute these enhancements to the ROOT code base.

3.2. Immutable references for merged files

Merging of files is becoming more important as event processing moves to more and more distributed systems processing fewer events in an individual job or process, including the multi-processor version of Athena, called AthenaMP.

For event data stored in ROOT Trees, merging files with identical schema is done trivially by appending the Tree’s entries. Handling the Token references is more complex, as the offset identifiers need to be updated. Currently this is handled by a dedicated re-directional layer in POOL (called ‘##Sections’). The use of immutable attributes in the Token reference will make the ##Sections layer unnecessary, as offset identifiers will become irrelevant. Re-indexing within the merged file will be handled by the underlying storage technology. While the benefits in CPU time and memory footprint will be minor, this strategy will reduce complexity, increase robustness, and enhance portability.

3.3. Support for forward references

In a production environment, the storage technology type, the data type of an event entry point (a DataHeader in ATLAS), and the name of the Tree in which it resides, while flexible in principle, are standardized in practice. A consequence is that to construct a reference to an event in a standard
ATLAS data file one needs only to identify the file and to specify which event in the file is to be referenced, as all other attributes have predefined default values in standard collaboration-wide production. Because \{run number, event number\} is immutable, to refer to a given event one needs only the file’s unique identifier to complete the reference.

It is possible to pre-generate such file identifiers. When an Athena job creates an output file, it will use the identifier found in its file catalog if one has already been entered to correspond to the logical file name. Together these capabilities allow ATLAS the potential to define forward references to events in downstream data products that have not yet been created. For example, ATLAS TAGs can be produced centrally pointing to events in derived data products that have not yet been made.

4. Conclusion

During the current LHC shutdown, the ATLAS I/O framework will enhance its content-agnostic reference infrastructure for persistent data with the addition of content-aware references for event data. This will simplify file merging and support new capabilities such as forward references, and will provide a foundation for future event-level distributed processing.

5. References

[1] ATLAS Collaboration The ATLAS Experiment at the CERN Large Hadron Collider 2008 JINST 3 (2008) S08003.
[2] Calafiura P et al 2004 The athena control framework in production, new developments and lessons learned Computing in high energy physics and nuclear physics
[3] Calafiura P, Leggett C G, Quarrie D R, Ma H and Rajagopalan S, 2003 The StoreGate: A Data model for the Atlas software architecture Computing in High Energy and Nuclear Physics
[4] Brun R and Rademakers F 1996 ROOT: An Object Oriented Data Analysis Framework AIHENP'96 Workshop Nucl. Inst. & Meth. In Phys. Res. A 389 (1997) 81-86. See also http://root.cern.ch
[5] van Gemmeren P and Malon D 2009 The event data store and I/O framework for the ATLAS experiment at the Large Hadron Collider IEEE International Conference on Cluster Computing doi: 10.1109/CLUSTR.2009.5289147
[6] Malon D, van Gemmeren P and Schaffer A 2006 Sailing the Petabyte Sea: Navigational Infrastructure in the ATLAS Event Store Computing in High Energy and Nuclear Physics
[7] L. Bauerdick et al 2012 Using Xrootd to Federate Regional Storage J. Phys.: Conf. Ser. 396 042009 doi:10.1088/1742-6596/396/4/042009
[8] Malon D et al 2006 A flexible distributed event-level metadata system for ATLAS Computing in High Energy and Nuclear Physics

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.

Notice: The publisher by accepting the manuscript for publication acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes.