Supporting High-Performance I/O at the Petascale: The Event Data Store for ATLAS at the LHC

Peter van Gemmeren [#1], David Malon [#2], for the ATLAS Collaboration

[#] *Argonne National Laboratory*

9700 S. Cass Avenue, Argonne, IL 60439, U.S.

[1] gemmeren@anl.gov

[2] malon@anl.gov

***Abstract*—A key to delivering high-performance scientific data stores at multi-petabyte scales and beyond is to provide an infrastructure that allows scientists to select, read, and process only the data that they need.**

The ATLAS experiment at the Large Hadron Collider (LHC) has deployed an event data store and an I/O framework designed from their inception to support efficient selective reading at many scales, from event-level selections to partial retrieval of the complexly-structured event data objects themselves. In-file metadata is used to accomplish the bookkeeping needed to accompany these data reduction strategies.

This paper describes features of the ATLAS I/O and event data store infrastructure that make efficient selective reading possible.

I. Introduction

The Large Hadron Collider at CERN provides high energy proton-proton collisions to four large experiments. The ATLAS experiment is one of the two multi-purpose detectors with a wide physics program. It was designed for the discovery of new particles such as the Higgs boson and super-symmetric particles, and to improve precision measurements of gauge bosons and heavy quark properties. Almost 3000 scientists work on the ATLAS experiment, bringing a diverse range of physics objectives and strategies and a wide variety of data analysis use cases.

At the 2009 IASDS workshop, we presented an overview of the Event Data Store and I/O Framework for the ATLAS Experiment at the Large Hadron Collider [1]. At that time, the ATLAS experiment had not yet started to take real proton-proton collision data, and the software infrastructure was exercised mainly with simulation, calibration- and cosmic ray data.

Since then, ATLAS has recorded almost 1 billion events at 7 TeV, an energy more than three times higher than the previously record (for an example, see figure 1). These collisions have not only sparked worldwide interest, but also have provided the first real test for the Event Data Store and I/O framework.

Raw event data are written to "bytestream" files and serve as input to reconstruction which produces Event Summary Data (ESD) and Analysis Object Data (AOD) in object-aware POOL/ROOT files [2][3].

Furthermore, event data are skimmed, trimmed, thinned and slimmed (more on these operations later) to produce smaller more customized data products, called Derived Physics Data (DPD), which is input to particular performance studies and physics analyses.

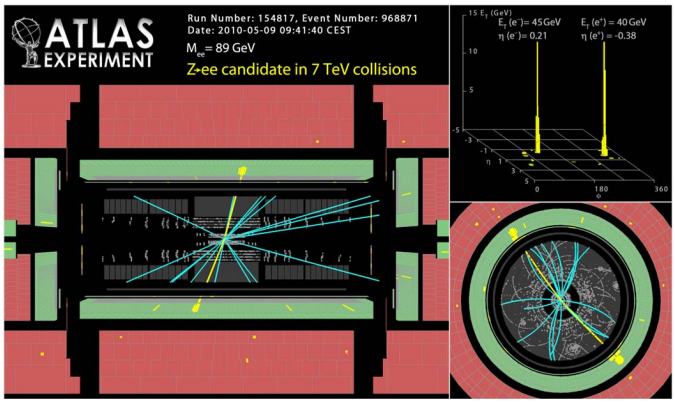

Fig. 1. Event display of the first ATLAS candidate for an event with a Z boson decaying to two electrons seen in 7 TeV collision data.

The primary event processing (reconstruction) occurs at the Tier-0 facility hosted by CERN, where one copy of the RAW data is archived. Another replica of the RAW data and the produced ESD are distributed to the approximately ten Tier-1 facilities around the world. The Tier-1s take custodial responsibility for the data, and provide reprocessing capacity and hosting for simulated data. Each Tier-1 serves several Tier-2 facilities, hosted by universities or laboratories and responsible for providing the simulation capacity for the experiment. Hundreds of smaller institutional resources, at the university or laboratory level, serve as Tier-3 facilities for physics analysis.

II. Event Data Store and I/O Framework

The data associated with a single collision of proton bunches is called an "event." In order to be usable for physics analysis, event data must be processed using complex software algorithms to find tracks, jets, vertices and clusters, to calibrate energies and to reconstruct the event. The event data store and I/O framework need to support these tasks and the underlying data model.

The event data store comprises a number of successively derived event representations. Raw data taken by the ATLAS detector and simulated data are reconstructed into streamlined event representations suitable for analysis (see figure 2).

Like most HEP experiments, ATLAS also uses an approximately matching amount of simulated events for many of their analyses to estimate detector response and efficiency. A generator program is used to generate events of a certain physics process (e.g., top pair production, W boson production with leptonic decays). The output of the simulation is stored as Raw Data Objects (RDO) and contains information similar to detected RAW data in addition to information about the underlying generated event.

In addition to being necessary for physics analysis, simulated data was used to exercise and validate the ATLAS computing model, including reconstruction and analysis. As simulated data is produced for individual physics processes separately, event filtering is less important than for real data. Many of the detector performance studies, which require very detailed information about individual detector components, are only done using real data events.

RDO, ESD, AOD, and most DPD are data with an object-oriented representation, and are stored via POOL/ROOT, which provides high-performance and highly-scalable object serialization to self-describing, schema-evolvable, random-access files.

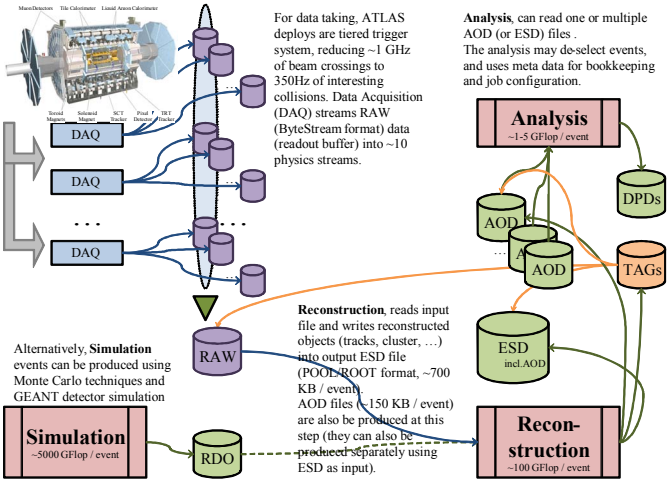

Fig. 2. Diagram of the ATLAS data flow: Simulation, Reconstruction and Analysis store data in POOL/ROOT file format. All event data is accessed via a DataHeader, which stores persistent addresses to all data objects and DataHeader of upstream processing steps (provenance information to allow back navigation). Some metadata describing the event sample is directly written to the event file (in-file metadata)

ROOT supports limited schema changes (e.g., adding or removing of s data member) automatically without breaking readability of old data. However, as the ATLAS experiment will take data for more than a decade (and analyze it for even longer) more significant changes of the transient event data model need to be supported. ATLAS software accomplishes this by deploying a dedicated persistent data model. As described in [1], the mapping layer between the transient and persistent data objects can be used to allow almost arbitrary schema changes without compromising readability of existing data. The simplified layout of the persistent classes also increases the I/O performance dramatically.

In addition to event payload data, ATLAS also uses event-level metadata records called TAGs [4]. TAGs contain event attributes chosen to support efficient identification and selection of events of interest to a given analysis. TAG data can be stored in ROOT files and uploaded into a relational database.

The ATLAS software framework, Athena, belongs to the blackboard family of architectures: data objects produced by algorithms are posted to common in-memory data stores from whence other modules can access them and produce new data objects. This approach greatly reduces the coupling among algorithms for analysis, reconstruction, and simulation, which share data objects.

StoreGate, the Athena implementation of the blackboard, allows modules to use transparently a data object created by an upstream module or read from disk. A proxy is used to define and hide the cache-fault mechanism. Upon request, a missing data object is created and added to the transient data store, with retrieval from persistent storage on demand. StoreGate identifies objects via data type and string key. It supports base-class and derived-class retrieval of data objects and containers, key aliases, and inter-object references.

Within an event, ATLAS arranges its data in top-level data objects and containers that are collections of objects of homogeneous type. The persistent format of ESD, AOD, and DPD reflect this transient layout (see figure 3). Persistent data objects are simplified representations of the transient data model, which uses the full expressive power of C++ (including inheritance, polymorphism and pointer).

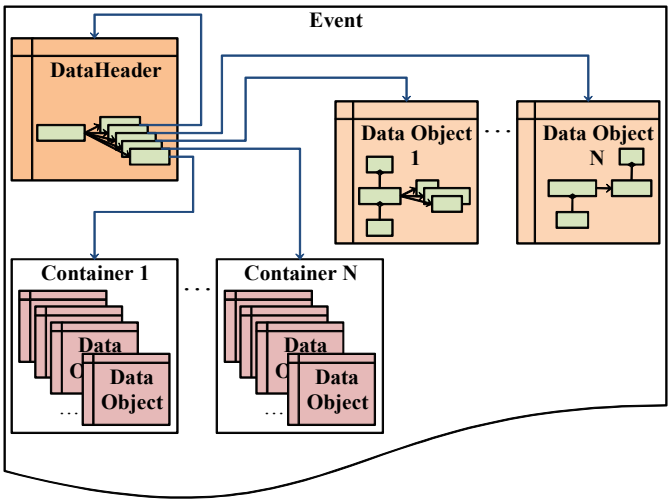

Fig. 3. Diagram of an ATLAS event: Persistent event data contains a DataHeader which servers as entry point to the event and provides persistent addresses and StoreGate states for all top-level Data Objects and Container used to store collections of Data Objects.

Athena persistence services require more than just the capability to write data objects and read them back again. The machinery by which StoreGate supports object identification, base-class and derived-class retrieval, aliases, and inter-object references must also be stored. Not only the states of physics data objects, but also their state in StoreGate must be saved and restored. The DataHeader provides the layout to save the

StoreGate state and the persistent address for all data objects. It is written automatically for each event and is used to load the stored data object from POOL (by using its persistent address or token), restore its StoreGate state (e.g., base and derived class information, key, and alias) and provide support object and event navigation.

III. Data Taking

Since the Large Hadron Collider at CERN began operations late last year, scientists and engineers have made rapid progress increasing energy and luminosity of the proton-proton collisions. On March 30th, LHC reached its initial operating energy with collisions at 7 TeV center of mass energy. Since then, almost 1 billion collisions have been recorded by the ATLAS detector.

These events are stored in different data products and are replicated multiple times. The event size for RAW data is about 1.6 MB, so the total storage space needed for two copies is 3 PB.

Most detector and software performance studies and physics analyses do not require RAW data, but use reconstructed data instead. ESD is the most complete set of reconstructed event data, and a single replica of ESD for all events has a total size of 700 TB, which is already too large for frequent reads. Therefore subsequent data products, such as AOD and DPD, are used to reduce the data size. AOD contains all information needed for most analyses and is produced for all events during reconstruction. Its total size of 150 TB is still too large for frequent end user analysis. Further reduction of data has to be customized for different users; for example, studies of tracking efficiencies require information different than an analysis of Z boson decays. For this purpose ATLAS uses several streams of Derived Physics Data.

IV. Production of Derived Physics Data

Derived Physics Data [6] is a reduced version of the full datasets (ESD, AOD). This reduction can be achieved in several ways:

- Skimming: Only events that are interesting for the specific types of performance studies or analyses are kept. For example, one might keep only events with two or more good electrons, for analyses looking for events containing decays of $Z \rightarrow e^+e^-$ or new high mass resonances.
- Trimming: Removal of entire containers or top-level data objects from all events. For example, one might remove the CaloCellContainer (calorimetry data), for performance studies of track reconstruction software.
- Thinning: Removal of individual objects from a container. For example, to study efficiencies for electron and photon reconstruction one needs to keep only CaloCells near a track.
- Slimming: Removal of parts of an object. For example, one might remove an error matrix from TrackParticles, as this very detailed information is not needed for many analyses, even when TrackParticle objects are used.

For DPD making these steps are done not only to reduce the disk size of the data representation, but also to increase the read speed of the data for any subsequent analysis or performance study.

Specific criteria for skimming, trimming, thinning, and slimming depend on the intended use of the Derived Physics Data (see figure 4). DPD for detector and software performance studies is made using ESD as input, because these tasks usually require detector-specific details that are not kept in the AOD. At the first reconstruction, such "performance" DPD (or dESD) is produced at the Tier-0 just after the ESD have been completed.

For most physics analyses, DPD can be made from AOD. Physics DPD (or dAOD) is produced at the Tier-1s.

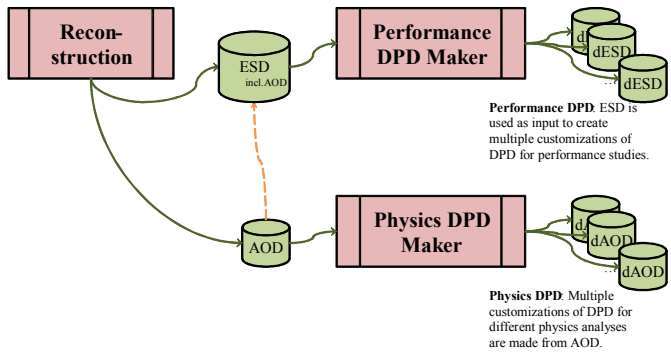

Fig. 4. DPD data can be categorized into two groups, as either dESD, made from ESD and used for performance studies or dAOD, made from AOD and used for physics analyses. Depending on the intended purpose, there are multiple different streams of dESD and dAOD.

Because of the large reduction in storage size, both dESD and dAOD can be copied to the Tier-2s to make them available to all ATLAS scientists. The DPD produced in this manner only contains information from ESD or AOD, and is called primary DPD. They are centrally produced by the ATLAS production system. Individual or groups of user may decide to produce even further derived data from the primary DPD, called secondary or tertiary DPD. Secondary DPD is written in POOL/ROOT format (same as ESD, AOD and primary DPD) and typically uses more stringent skimming, trimming, thinning, and slimming. Users can also decide to add their own content to secondary DPD, such as particle decay candidates like $Z \rightarrow e^+e^-$. Tertiary DPD is completely customized by the user and often can be written as an ntuple in ROOT.

V. Data Reduction

A recurring tradeoff in the operation of a scientific data store as large as the ATLAS event store is between extraction of a sample corresponding to a specific selection as a physically materialized dataset, and support of such a sample as a virtual selection via navigational infrastructure. A globally distributed physics working group may, for example, agree upon common event selection criteria for a shared analysis, and the events that satisfy the selection criteria may be "skimmed" from the original data (and possibly thinned and slimmed and trimmed to include from the selected events

only the subset of event information useful to the specific analysis), with copies possibly replicated to participating institutes; alternatively, the group may build a shared, agreed-upon event list, and analyze exactly those events and no others on sites that host the bulk data by using direct navigation via persistent references to the selected events. Replication increases the storage requirements of an already-multi-petabyte data store, but may allow physicists to tap CPU resources that otherwise might not be available to their analyses, or may improve turnaround time for distributed analysis jobs.

The ATLAS event store and I/O infrastructure have been designed so that the choice between materialized and virtual event and sub-event selections may be based purely upon operational, not functional, considerations: whether an event selection has been materialized or not is transparent to the analysis jobs themselves. In practice, both explicitly-materialized selections and virtual selections are employed by ATLAS physicists.

Virtual event selections utilize event collections and navigational infrastructure. Virtual sub-event selection relies on the StoreGate proxy mechanism, supported by the persistent DataHeader. Still, virtualization of data selection has its limits. The virtual selection of individual data members of an object (without reading the rest) is not supported by the Athena framework; even though the code may access only certain data members, the entire object is retrieved from storage. However, ROOT data is stored column-wise, and AthenaRootAccess, a lightweight version of the Athena framework designed for DPD analysis, uses this structure for efficient reading of single attributes (see figure 5).

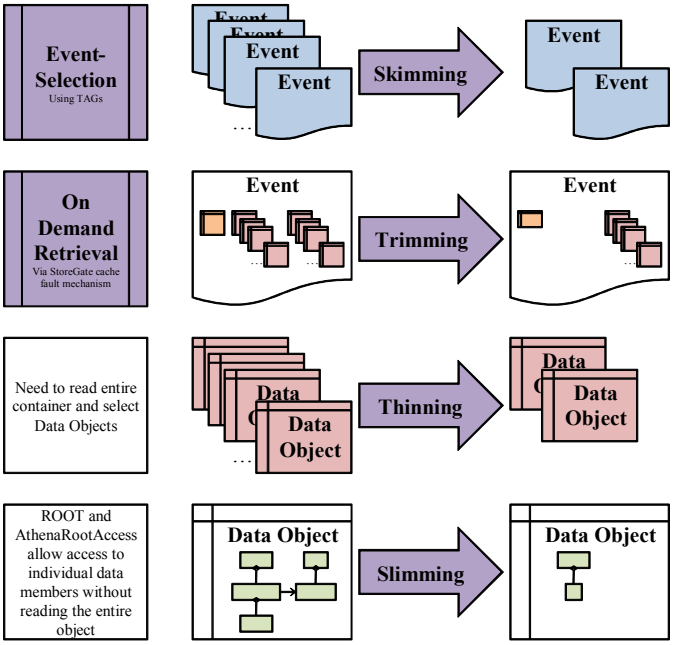

Fig. 5. Skimming, Trimming, Thinning and Sliming are used to reduce the amount of data stored in derived data representations.

A. *Skimming*

Skimming, the selection of interesting events, can be done using TAGs (by means of an SQL-style query or computational processing) or veto/accept filter algorithms for Output Streams.

TAGs allow jobs to process only events that satisfy a given predicate. Less complex predicates can be applied as SQL-style queries on the TAG database using a web browser interface, which produces a TAG file with references to the selected events. These TAG files [5] can then be used as input to a job in a manner similar to direct use of event data payload files as input. SQL style queries can also be applied on TAG files directly. For more complex criteria, C++ modules can be written using the TAG attributes for event selection. In either case, no payload data is read for unselected events, and data files containing only unselected event are not accessed.

The selected events can then be written to a single output file. It is also possible to have the same job produce several different output event selections from the same input files. For that purpose, different filter tools are attached to separate output streams, which then produce output files with events according to their selection criteria.

Skimmed output files are smaller than the original files and allow reading of the selected event without need to reapply the selection criteria. Of course, event selection can also be done when reading the original file. However, there can be sizeable overhead as ROOT may store data for subsequent events in the same "baskets," which are read entirely (see figure 6). When tuning the size of ROOT baskets, one of many adjustable parameters for I/O optimization, ATLAS must balance the performance implications for both use cases, reading all or reading selected events.

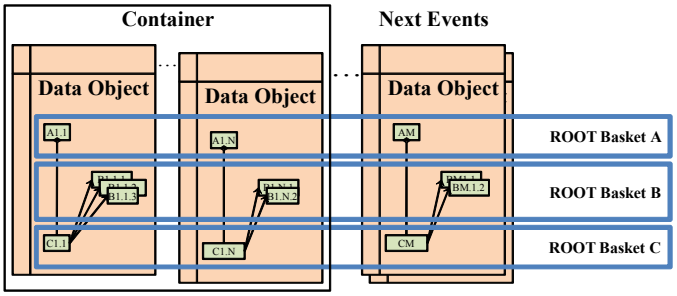

Fig. 6. In ROOT, data is stored in baskets by attributes. Attributes of different data objects in a container or even from subsequent events can be stored in the same basket. This column-wise storage model has advantages for file compression and when reading individual attributes for many objects.

B. *Trimming*

Trimming, the selection of top-level data objects and containers, is done via job configuration. When fewer top-level data objects are written, there are fewer entries needed in the DataHeader and fewer proxies are created when reading the file.

A single job can have multiple output streams, each configured to write a separate set of data objects into its output file. The resulting output files will be, to a degree depending upon the container selection, smaller than the input

file. As fewer containers are written, there are fewer entries needed in the DataHeader and the numbers of proxies in StoreGate is reduced.

On input, container and top-level data objects are read on demand and there is little performance overhead due to the reading of DataHeaderElements and the creation of proxies for all data objects. Proxy creation could be delayed until on demand retrieval of the data object, but the performance impact is not significant so at this point this has not been implemented in Athena.

C. Thinning

Thinning, the selection of particular data objects in a container, is done using configurable algorithms that retrieve the original container and copy the selected objects into a new container, which then is written to the output file. Any persistent references to the thinned container need to be updated; for example, a reference from another object to the ith element of the unthinned container may be adjusted to point instead to the jth element of the thinned container. Such reference updates are handled automatically.

On input, removing individual objects from a container cannot be done prior to retrieval. Athena will read all objects of a container, even if only some objects in it are needed, in order to rebuild the container object in the transient data store. The retrieval granularity for StoreGate corresponds to top-level objects and to containers.

Even if StoreGate supported partial retrieval of container, most of the data would still have to be read, because ROOT will store the attributes of multiple objects of a single container in the same baskets (see figure 6). However, by choosing which data objects are stored at top-level and which to collect in a container, one can define an event data model that corresponds to predictable and common partial retrieval use cases.

D. Slimming

Slimming, the selection of specific data object members, can be done in several ways:

1. The ATLAS software framework supports persistent associations among data objects in different containers (even over file boundaries). When data members are components of an object by association, it is possible to slim DPD by removing associated objects and not writing the host container of the target of such an association. When a data object with an association is retrieved, the host container of the target object is read only on demand, using StoreGate proxies and the DataHeader, so it is possible to avoid needless retrieval. This method is very similar to trimming of the host container and can be accomplished as a virtual selection without significant performance penalty.
2. Changes to the inheritance hierarchy, by writing a base class rather than a derived class, can reduce the number of data members and therefore the size of the output file. StoreGate supports base-class retrieval, but the conversion service will still read all data members of the derived class. In a future extension of the I/O framework we will consider implementing reading and conversion of the base class only.
3. Reducing (or eliminating) the number entries in a member container (such as an STL container), does not affect the class layout, but will reduce the data size. From the I/O perspective this is a change of data content and there is no equivalent functionality to handle such a data reduction virtually.

E. Comparisons with SQL

In relational database terms, events might correspond to rows and top-level objects and containers could be columns of a table. So skimming can be compared to an SQL 'WHERE' clause in a 'SELECT' statement and trimming would be the 'SELECT' clause (the column name selection). Skimming with TAG attributes, utilizes event level metadata rather than record data itself. TAGs contain references to different event representations akin to foreign keys in a database. Cells, however, do not just contain a single piece of data (as in a table), but a representation of a complex C++ object or a collection of such. Thinning and slimming are operations on the data content within one cell.

Within some object-oriented database systems, objects are represented as rows in a table. So thinning could be compared to the WHERE clause and slimming to the SELECT clause in a SELECT statement on an object-oriented database system.

The ATLAS event store and I/O infrastructure is able to execute virtual skimming and trimming which correspond to relational database operations, typically with only small performance overhead. However, virtual thinning and slimming, the finer grained object-oriented database operations, are not fully implemented at this point.

VI. Metadata Implications

As skimming reduces the number of events stored in the DPD, metadata information about unselected events, and about the event collections from which the selections were made, must be kept. This information is necessary for many physics analyses (e.g., when calculating branching ratios or exclusion limits).

Event bookkeeping is part of DPD production, implementing counters of input and output events, recording information about the domain of the selection, and storing the results as in-file metadata in the DPD itself (see figure 7). In addition, metadata about luminosity and detector conditions needs to be propagated from the input files. Automatic job configuration uses in-file metadata, so DPD needs to contain all information needed to set up analysis jobs correctly.

Propagation of in-file metadata has to provide the capability to aggregate and summarize information from multiple input files. The rules to merge metadata depend upon the type of metadata (e.g., merging a set of run numbers is different from adding luminosities). Therefore, the Athena framework allows for dedicated metadata tools to accomplish this task. These tools are called automatically when opening or closing a new input files, so they can retrieve their metadata and combine it.

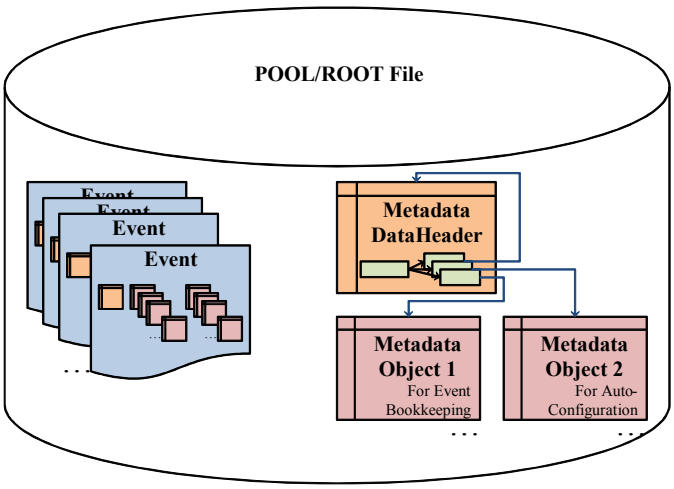

Fig. 7. ATLAS event data files store in-file metadata objects which can be used for job auto configuration and event bookkeeping needed when skimming events.

For stringent event selections, it is possible that none of the input events in a given file or set of files pass the selection criteria. Still an output file must be written containing all the metadata, event bookkeeping, and auto-configuration information, since this information is as necessary to many physics analyses as the corresponding information when events do happen to meet the selection criteria.

VII. Conclusion

As scientific data stores grow in volume, machinery to support reading only the data that one needs becomes ever more important. In some disciplines this may be straightforward to achieve by appropriate data organization (e.g., in the retrieval of spatial data by region); in others, such as high energy physics, data of interest may be distributed throughout the store. While regrouping of the data may be effective in some cases, in large-scale collaborations with multiple scientific objectives it is impossible to group the data to satisfy all important access patterns simultaneously. Even when data reorganization is possible, the machinery to support efficient navigation to and extraction of the data of interest in order to build the reorganized or extracted samples is essential.

The problem is more difficult in disciplines with complicated, inhomogeneous, object-oriented data models.

For data stores of homogeneous, simply-structured data (such as raw image data from homogeneous pixel arrays), it may be straightforward to organize the store and design the I/O framework to extract only the attributes one needs.

For experimental high energy physics, where even the RAW data is the heterogeneous output of almost 100 detectors, and the physics data derived from it are as complicated as C++ allows objects to be, optimized selective retrieval is a challenge.

In this paper we have described how the ATLAS event data store and I/O framework and operational model together support high performance at petabyte scales by providing machinery for selective data access at multiple levels.

A number of challenges remain, but the ATLAS infrastructure is demonstrably supporting the needs of a community of 3000 scientists for access to a globally distributed multi-petabyte scientific data store in these early and exciting times at the frontiers of particle physics at the Large Hadron Collider.

VIII. Acknowledgment

The ATLAS event store is the work of many people. The authors would like in particular to acknowledge the contributions of Jack Cranshaw (Argonne National Laboratory), Marcin Nowak (Brookhaven National Laboratory), and Arthur Schaffer (Laboratoire de l'Accélérateur Linéaire, Orsay).

Notice: The submitted manuscript has been created by UChicago Argonne, LLC, Operator of Argonne National Laboratory ("Argonne"). Argonne, a U.S. Department of Energy Office of Science laboratory, is operated under Contract No. DE-AC02-06CH11357. The U.S. Government retains for itself, and others acting on its behalf, a paid-up nonexclusive, irrevocable worldwide license in said article to reproduce, prepare derivative works, distribute copies to the public, and perform publicly and display publicly, by or on behalf of the Government.

References

[1] P. van Gemmeren, D. Malon, "The event data store and I/O framework for the ATLAS experiment at the Large Hadron Collider," in *IEEE International Conference on Cluster Computing and Workshops,* 2009, pp.1-8.

[2] R. Brun and F. Rademakers, "ROOT – An Object Oriented Data Analysis Framework", Nucl. Inst. & Meth. in Phys. Rev. A 389 (1997) 81-86. See also http://root.cern.ch

[3] D. Duellmann, "The LCG POOL project: General overview and project structure", Proceedings of Conference for Computing in High-Energy and Nuclear Physics (CHEP), 2003

[4] J. Cranshaw et al, "Event selection services in ATLAS," in *J. Phys.: Conf. Ser.,* vol. 219, 042007, 2010.

[5] P. van Gemmeren, D. Malon, M. Nowak, "New developments in file-based infrastructure for ATLAS event selection," in *J. Phys.: Conf. Ser.,* vol. 219, 032061, 2010.

[6] K. Koeneke et al, "A new data format for the commissioning phase of the ATLAS detector," in *J. Phys.: Conf. Ser.,* vol. 219, 032047, 2010