A FLEXIBLE, DISTRIBUTED, EVENT-LEVEL METADATA SYSTEM FOR ATLAS*

D. Malon, J. Cranshaw, K. Karr, Argonne National Laboratory, Argonne, IL, USA
J. Hrivnac, A. Schaffer, LAL, Orsay, France

Abstract

The ATLAS experiment will deploy an event-level metadata system as a key component of support for data discovery, identification, selection, and retrieval in its multi-petabyte event store. ATLAS plans to use the LHC Computing Grid (LCG) POOL collection infrastructure to implement this system, which must satisfy a wide range of use cases and must be usable in a widely distributed environment. The system requires flexibility because it is meant to be used at many processing levels by a broad spectrum of applications, including

- primary reconstruction,
- creation of physics-group-specific datasets,
- event selection or data mining by ordinary physicists at production and personal scales.

We use to our advantage the fact that LCG collections support file-based (specifically, ROOT TTree) and relational database implementations. The ROOT trees provide a simple mechanism to encapsulate information during collection creation, and the relational tables provide a system for data mining and event selection over larger data volumes. By several measures, the event-level metadata system is the collaboration's most demanding relational database application. ATLAS also uses the ROOT collections as local indexes when collocated with associated event data.

Significant testing has been undertaken during the past year to validate that ATLAS can indeed support an event-level metadata system with a reasonable expectation of scalability. In this paper we discuss the status of the ATLAS event-level metadata system, and related infrastructure for collection building, extraction, and distributed replication.

TAGS AND THE COMPUTING MODEL

The ATLAS experiment will collect data at a rate of approximately 200 events per second, for a total of approximately 2 x 10^9 events per year. At 1.6 MB/event, this results in more than 3 petabytes of raw data per year, passing through a reconstruction phase to yield 1 petabyte of Event Summary Data (ESD) and 200 terabytes of Analysis Object Data (AOD) annually. From information contained in the AOD, ATLAS proposes to extract metadata describing each event—information to support event-level selection—for insertion into a queryable tag database. The current budget for such tags is 1 KB/event, so a tag database will accumulate approximately 2 terabytes of payload data annually. Tag storage requirements may be two or three times this amount, to allow for indexes and navigational infrastructure support, but the scale remains small—less than 10 terabytes—relative to the overall scale of the ATLAS event store.

In the ATLAS computing model[1], each successively smaller derived data product is replicated more widely, and the tags are no exception: at a scale of only a few terabytes, every Tier 1 and Tier 2 center is expected to have sufficient storage capacity to enable it to host a copy of the tag database. The technology used to host a tag database may vary with tier and institutional capabilities.

EVENT TAGGING INFRASTRUCTURE

The ATLAS I/O framework and event store architectures consistently distinguish writing from registration. When a data object is written, a reference to its location in the persistent data store is returned. This reference may in turn be recorded, along with identifying information and optional metadata, in one or more object registries by means of a registration service. Within an event, a DataHeader object serves as the registry for references to constituent objects such as track or particle collections. For the events themselves, ATLAS uses POOL collections [2] as its event object registries, storing a reference to the event's DataHeader as an entry point into the event, along with event identification values and attributes of the event upon which future event selections might reasonably be based. (Detector calibrations and conditions data follow this architectural pattern as well: for objects used to convey time-varying information, a temporal database (interval of validity database) serves as the object registry.)

POOL collections support both file-based and relational database incarnations, and relational implementations have been delivered for a variety of database technologies, including Oracle and MySQL. ATLAS uses all of these implementations, writing tags into files, importing them into relational databases, and replicating tag databases between sites, sometimes changing database technology in the process.

TAG CONTENT

Tags contain event identification and global event quantities, trigger information, "quality" information, and some high-level physics object data. The guiding principle is to support efficient and sensible selections,

not to support direct analysis on tags. The tag storage budget is another factor in deciding which data do and do not appear in the tags. Tags also contain sufficient navigational information to allow retrieval of the corresponding events.

Standard POOL collections support the association of attributes to one specific object or object reference. The ATLAS event store team has extended the model to support multiple object references per tag, and, in particular, to associate event-level metadata with pointers to event data at all processing stages. The current deployment includes references only to AOD and ESD, because RAW data are not necessarily written in an object format, but this reference implementation limitation will be lifted in coming releases.

POPULATING THE METADATA SYSTEM: OPERATIONAL MODEL

ATLAS will write event tags into ROOT files at the time that AOD are written (or when small AOD files are merged into larger ones). These tag files will later be used as input to a bulk load operation to populate a relational-database-resident tag database at the Tier 0 center. The initial motivation for this strategy was to avoid the contention associated with on the order of a thousand processors all trying to add rows to the same tables. The file-based tags have since proven useful in their own right, as described in a later section. While the baseline plan is to update Tier 1 databases from the Tier 0 master tag database, distribution of the tag files along with AOD to the Tier 1 centers provides a potential alternate source for updates to Tier 1 tag databases; indeed, it is possible that the files themselves might serve as the tag database at Tier 2 sites lacking the resources to host a database service. Use of POOL collections allows us flexibility in defining ATLAS policies.

STREAMS, SKIMS, AND COLLECTIONS

High energy physics collaborations often need to confront the issue of defining their event data streams, and ATLAS is no exception. Decisions must be made regarding the handling of events that are, for example, of interest to more than one physics working group, or that satisfy more than one trigger: should one write disjoint or overlapping streams, should one write many streams, or consolidate and write only a few, and at what processing stage should streaming be done?

Event collections provide a potential tool for dealing with such issues. It is possible, for example, to write each event exactly once, but to register the event in many collections—one for each physics and detector working group to which the event is of interest. This approach may be used in conjunction with a streaming strategy, so that the events of interest to group A are written contiguously into a sequence of files, and the events of interest to group B that were not already claimed by group A are written contiguously into a separate sequence of files. In this case, group B's files are missing the events already claimed by group A, but because a reference to every qualifying event is written into group B's event collection, independent of the stream to which the event may have been written, group B's event collection is complete.

It is natural to ask what the cost of such an approach might be. Studies were undertaken to explore this question on behalf of the ATLAS Computing Model group. Because all Tier 1 and Tier 2 sites are expected to host the complete ATLAS AOD, an analysis job iterating over group B's sample will run successfully on such sites, whether or not all of group B's events have been stored contiguously in a set of files. In exchange for the storage savings accrued by writing each event exactly once, the cost is that additional files need to be opened to read the events scattered across group A's files, events that might alternatively have been replicated in group B's file set. To test this, a uniformly distributed random selection of events distributed throughout several hundred event data files was extracted into a single file. Instrumentation was inserted into several analysis jobs, and the jobs were configured to use, alternatively, either the event collection pointing to the uniformly distributed events or the replicas of those events that had been gathered into a single file.

The additional cost of reading events scattered over N files rather than reading them consecutively from a single file was approximately the cost of opening N-1 additional files (0.5-1.0 seconds/file from a Castor disk pool, less from AFS)—navigational overhead was too small to be measurable.

Alternatively, one may actually use the tags to extract a subsample based on stream or metadata selection. In several previous experiments this has been called 'skimming'. Tags act as a natural tool for this activity. In the ATLAS framework, direct navigation to events that satisfy a query on tag attributes is faster than reading each event and deciding whether it is of interest. One limitation of our current implementation is that tags cannot yet be extended. This means that the tags used to do the skim would not know anything about the resulting skim data files. Development work may eliminate this problem in the future.

QUERYING THE TAG DATABASE

All implementations of the ATLAS tag database support SQL-style queries (SELECT … FROM … WHERE …). While one can select any or all of the associated event attributes, the most central of the selected fields are the returned references, which are the components that allow navigation to the events themselves. Just as the return value of an SQL query is another table, the output of a query to a collection is another collection. Any collection may be used as input to an ATLAS analysis job: it simply acts as a list of pointers to events over which the job should iterate, with optional associated metadata attributes.

When a query result is returned, the default practice is to group the output by unique file id (GUID) of the files containing the selected events. This allows an analysis task to be split into multiple jobs, each using a disjoint subset of the needed files, and the corresponding subset of the event list, as input. Some ATLAS distributed analysis prototypes are already capable of splitting jobs accordingly. The strategy also allows a single job to avoid thrashing, reading some events from one file, moving to another file, and then returning later in the input event list to the first file. Utilities are provided to return the list of unique file GUIDs of qualifying events (without the event references themselves), for use as input to resource brokers that need to decide where to run the job or which files to prefetch.

DEPLOYMENT EXPERIENCE

An ATLAS Physics Workshop in Rome in June 2005 provided the first opportunity to put event-level metadata into the hands of physicists. While production and reconstruction of simulated data for the workshop was a globally distributed undertaking involving several computational grids, Analysis Object Data for just under three million events were hosted by a grid Storage Element at CERN. With tag content defined by the ATLAS Physics Analysis Tools group, tags were produced for the CERN-resident events. Tags were written into files when the many small AOD files were merged into larger ones. A bulk load utility was employed to copy the contents of the tag files into a relational database. To support controlled random sampling in scalability tests of the tag database infrastructure, a uniformly distributed random variable was added to the tag. The tag database was instantiated multiple times, in both MySQL and in Oracle, with a variety of indexing strategies. A number of timing studies were conducted, comparing load times, query times, cross-technology replication times, and more. For further information, see the Twiki page [3].

Several physicists employed the full tag database to make event selections. More common was the use of the tag files, each of which was isomorphic to a merged AOD file. At the level of jobs requiring one or a few event data files, the file-resident tags proved useful for selection or omission of specific events, with each tag file serving as an index to allow direct navigational access to the corresponding events, in lieu of pure sequential iteration through the files. Most user queries returned a fraction of the tag database in the range 0.001-0.1.

The Rome tag database prototype was sufficiently well received that it triggered initiation within the collaboration of a tag content review working group [4], to adjudicate what should and should not appear in ATLAS tags in light of space budgets and endorsed selection use cases. The recommendations of the working group will appear in March 2006.

REPLICAS AND REPLICATION

As part of the Rome Physics Workshop deployment exercise, the ability to support heterogeneous replication, in this case from Oracle at CERN to MySQL at Brookhaven, was successfully demonstrated. Octopus-based tools [5] were used for cross-technology replication. For replication of data from a Tier 0 “master” database to Tier 1 sites, CERN and the LCG Project’s distributed database deployment project are recommending use of Oracle Streams [6], an Oracle-provided tool for distributing updates and ensuring that database replicas are synchronized. ATLAS plans to test this approach to Oracle-to-Oracle replication in its 2006 Service Challenges.

Even when Tier-0-to-Tier-1 replication is an Oracle-to-Oracle operation, replication from Tier 1 centers to Tier 2 centers is anticipated to require a technology change: Tier 2 centers are not, in general, expected to maintain Oracle servers. ATLAS experience with MySQL servers has been quite promising: cookbook methods for installation of MySQL servers and MySQL-based database replicas have been employed at many scales, from Tier 1 centers to individual laptops, for ATLAS data challenges and personal use, with little or no maintenance burden. It is nonetheless possible that some Tier 2 sites may support only file-based access to event-level metadata.

ON SCALABILITY

ATLAS deployment experience provides some grounds for optimism and some grounds for concern regarding scalability. The infrastructure, while satisfactory for a few million events, is not yet ready for an increase of three orders of magnitude in data volume: simply dumping event-level metadata records into a giant table will not suffice. Experimentation with divide-and-conquer strategies has, on the other hand, been encouraging. One can partition horizontally, grouping events by dataset, for example. One can also partition vertically, so that attributes irrelevant to one’s query have little or no effect on performance. Indexing strategies have a pronounced effect, principally beneficial, though it is demonstrably easy to degrade performance by use of an index when a significant fraction of events satisfies a selection predicate applied to indexed attributes. Replication of the tag database will also improve scalability with respect to the number of supported clients by providing a means to reduce the query workload of any single database instance.

ONGOING AND FUTURE WORK

The ATLAS event-level metadata system has been demonstrably successful to date at modest scales, but a great deal of work remains to ensure scalable deployment when LHC data-taking begins.

A number of functional extensions are underway. It is understood, for example, that tags must be extensible—that some attributes (detector status and quality bits, for example) may not be known at the time the tag is first written. Work is underway in the context of the LCG POOL project to support such extensions.

Not every query is easy to express in SQL, and some queries may benefit from the ability to invoke procedural code (e.g., to decode a trigger signature, particularly when trigger menus may vary with time). ATLAS has already demonstrated the use of stored procedures written in Java for such purposes when the tag database implementation is Oracle-based. The controlled use of stored procedures, and a means to invoke pre-formulated queries hand-coded by experts, are both subjects of ongoing work.

Much work in the coming year will be devoted to addressing the scalability issues cited earlier. While a number of technology-specific optimizations are possible, such optimizations are currently *ad hoc* and experimental. It must be possible to refactor the persistent implementation of event collections to improve performance without disrupting client code. Current POOL collection implementations make natural assumptions about a simple one-to-one mapping of attributes to columns in a single database table (modulo some optimizations regarding the representation of object references), but work is underway to eliminate this implicit exposure of implementation details to clients.

ACKNOWLEDGEMENTS

The submitted manuscript has been created by The University of Chicago as Operator of Argonne National Laboratory ("Argonne"). Argonne, a U.S. Department of Energy Office of Science laboratory, is operated under Contract No. W-31-109-Eng-38. 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]ATLAS Computing Technical Design Report – TDR (CERN-LHCC-2005-022), July 2005.

[2]http://lcgapp.cern.ch/project/persist/metadata/index.html

[3]https://uimon.cern.ch/twiki/bin/view/Atlas/CollectionPerformanceTests

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

[5]https://uimon.cern.ch/twiki/bin/view/Atlas/DatabaseReplication

[6]http://www.oracle.com/technology/products/dataint/htdocs/streams_fo.html