The ATLAS Metadata Interface

S. Albrand, T. Doherty, J. Fulachier, F. Lambert

The ATLAS METADATA INTERFACE

Solveig Albrand[1], Thomas Doherty[2], Jerome Fulachier[1] and Fabian Lambert[1]

[1] Laboratoire de Physique Subatomique et Corpusculaire, Université Joseph Fourier Grenoble 1, CNRS/IN2P3, INPG, 53 avenue des Martyrs, 38026, Grenoble, FRANCE

[2] Department of Physics and Astronomy, University of Glasgow, Glasgow G12 8QQ, Scotland. UK

Email: solveig.albrand@lpsc.in2p3.fr

Abstract. AMI was chosen as the ATLAS dataset selection interface in July 2006. It is the main interface for searching for ATLAS data using physics metadata criteria. AMI has been implemented as a generic database management framework which allows parallel searching over many catalogues, which may have differing schema. The main features of the web interface will be described; in particular the powerful graphic query builder. The use of XML/XLST technology ensures that all commands can be used either on the web or from a command line interface via a web service. We also describe the overall architecture of ATLAS metadata and the different actors and granularity involved, and the place of AMI within this architecture. We discuss the problems involved in the correlation of metadata of differing granularity, and propose a solution for information mediation.

1. Introduction

This article describes the ATLAS metadata interface (AMI) [1] which was chosen to be the official ATLAS tool for dataset selection in July 2006, following a review. The first part of this article discusses the ATLAS metadata context, giving an introduction to the different metadata granularities, and the different actors involved. In section 3 some technical details of AMI itself are given and we describe the various dataset selection tools now available for ATLAS physicists. AMI has been designed to be a highly generic framework; it is in fact a very flexible general catalogue system. We will show how we have been able to use these generic features to provide other ATLAS metadata services such as the Tag Collector, a tool for release management. Finally some areas for future development are discussed.

2. Metadata in ATLAS

A large amount of event data will be produced by the ATLAS experiment, during online data taking, and offline processing of the data. To be able to analyse the data it is necessary to have a cataloguing system which will allow the retrieval of data by members of the collaboration. To achieve this, the data must be described by "metadata" which we define as "data about data". Metadata may be classified in several ways. A recent ATLAS report [2] attempts to provide a taxonomy for metadata in the experiment. It provides the basis for future metadata developments; in particular the integration of the various metadata applications within ATLAS.

2.1. Granularities of metadata

Metadata must be created at different levels (Table 1). The smallest granularity is the physics event, but other levels are immediately apparent because they are directly linked to the hierarchical way in which ATLAS will organise its data taking. Events will be organised in runs, but each run may contain many luminosity blocks A luminosity block is a number of events acquired over a short time interval, for which the integrated, deadtime and pre-scale-corrected luminosity can be determined. [3]

Table 1. The granularities of ATLAS data and their order of magnitude expressed as number of events[1]

Granularity	Order of Magnitude
Event	1
File	10^2-10^4
Luminosity Block	10^4
Run	10^5-10^6
Datastream/run	10^4-10^6
Dataset	10^5-10^9

ATLAS event data will be sorted into physics streams (about 10 streams will be used), and then written into files. The scale of the ATLAS data, by both the quantity of events and the event size, means that it is impossible to get all the event data which should be grouped together into only one physical file. ATLAS therefore manages sets of files, or "datasets". An ATLAS physicist should not be concerned about individual files, only with datasets. The bulk file handling will be managed by the ATLAS distributed data management (DDM) tools [4]

In some cases the dataset composition will be determined as a function of the lower hierarchies of data organization. For instance files of RAW data will contain events from only one luminosity block, hence these files will belong to only one particular run and one particular stream. A RAW dataset will be made for each run and for each stream, but could contain several luminosity blocks. In general however, dataset composition can be considered to be arbitrary. Super datasets will be formed grouping selected events from several runs from one or more streams.

2.2. Metadata actors

Although many cataloguing applications are necessary to manage ATLAS metadata some of them are very specialized, and will not be needed directly by the average physicist. The Detector Control System is a good example of this type of application. Other applications are used directly by all physicists, but the metadata which they manage should be transparent for the user. For instance whereas the DDM accesses the physical files, the user is only concerned by the logical dataset and file names. Lastly come the applications where the metadata is actually used by the physics community to identify data of interest.

Table 2 The principle ATLAS metadata applications used for data selection.

Metadata Destination	Granularity
COOL (Run Conditions DB)	Run, Event, LB
Event TAG database	Event
AMI (Dataset selection)	Dataset, File

The metadata task force considered which subset of metadata parameters would be needed by physicists for data selection and identified the "front-line" applications (see Table 2) which should manage these parameters. Hence the value of each one of the designated parameters is held in at least one of these applications.

2.3. The role of AMI in ATLAS

It can be seen from Table 1 that the largest granularity in ATLAS data is the dataset. In the majority of searches for data the first step for a physicist will be determining datasets of interest. So the AMI application has a special responsibility for guiding ATLAS users down through the metadata hierarchy. Some metadata use cases require the means to navigate from one metadata granularity to another, or to put it another way, to navigate from one metadata catalogue to another. It may even be necessary to provide the possibility to combine the results of queries to more than one metadata catalogue. In consequence AMI is required to be a portal to all metadata within ATLAS by providing the appropriate links to other metadata applications.

3. The ATLAS Dataset Search Catalogue

3.1. Main features of the AMI framework

AMI is a framework consisting of relational schema which contain their own description, a software layer to exploit this auto-description, and a set of generic interfaces for the usual database functions (insert, update, delete and select). AMI can manage different schema deployed on geographically distributed servers with different relational database management systems (RDBMS) simultaneously. It must be pointed out that this enables AMI to support schema evolution in a seamless way.

AMI currently uses two RDBMS backends; mySQL (at the LPSC, Grenoble) and ORACLE (at CCIN2P3, Lyon). All new dataset catalogues will be implemented in ORACLE. However the generic mechanism hides details of database implementation from clients. A client who queries an AMI database is not expected to have knowledge of the name or type of the database, or the name of any database tables, or the relation between the database tables. The architecture allows clients to express queries in terms of the application semantics. Thus a user of the AMI dataset selection application should be able to work with a schema of datasets and files, whereas a client of another AMI based application, such as Tag Collector, works with a different semantic.

The AMI server code is written in Java and uses Java Database Connections (JDBC). In consequence, it is independent of platform, operating system and database technology. The AMI software is built in three levels above JDBC. The two lower level packages wrap the connections to the databases, the pooling of connections, the transmission of SQL commands, and the recuperation of query results. The top layer of the software contains application specific packages with knowledge of non generic features of the database tables. Those which are for dataset searching contain in particular the software to manage all the tasks which pull data from the various sources, and the dataset provenance tracking.

AMI is essentially a web application but also provides a SOAP Web service for clients. The python client is part of the ATLAS software distribution.

A role based system of user rights has been implemented. It is possible to grant any user the rights to update information in one or any number of AMI dataset catalogues. Authentification is either by password or Grid certificate. We have also implemented VOMS functions which allow us to recognize roles and physics group membership. AMI can also pass a VOMS proxy to a third party application.[5]

3.2. Overview of the dataset search

The vast majority of dataset search use cases require a powerful web interface, and this is where the most of our effort has been focussed.

As explained above, AMI uses self-describing relational schema. The generic AMI commands make no assumptions about the schema; they use the internal description to discover it. On the other hand, the top layer software contains detail of the particular application's semantics. In the case of the dataset catalogues a few semantic constraints are imposed. All schema must have a database table called "dataset" and this table contains a key field which holds the unique dataset name. Another imposed field is an internal status field, which enables the search to hide datasets which are known to

be bad. Apart from these constraints, the dataset catalogue schema can differ from one another. A catalogue of Monte-Carlo simulated datasets may have a different set of attributes than a catalogue of RAW TDAQ datasets.

Some global catalogues are maintained. The largest one is a list of all datasets known to AMI. It guarantees that the dataset names are unique across all catalogues. The dataset provenance information is also managed globally by using a database representation of a direct acyclic graph. The third type of global table is for use in the manner of reference tables. These tables limit the values which can be used for certain parameters. For example a dataset may be marked as belonging to a particular physics group. The name given to the physics group cannot be arbitrary; it must be part of the ATLAS VOMS physics groups.

Several web interfaces for searching are provided. The simplest search is to type part of a dataset name; an advanced search allows users to select using more detailed criteria. These interfaces are described in more detail in section 3.4.

Behind the scenes, the user request is transformed to a request expressed in the EGEE gLite grammar[6]. This grammar, which is also implemented by the native EGEE metadata application [7] looks similar to SQL but makes no assumptions about schema, so is ideally suited to the AMI philosophy. The gLite query is then sent out in parallel to all known dataset catalogues. They respond if they can reply to the query. (A query which asks for information about a particular parameter can only be satisfied by a catalogue which contains information on this parameter). A second request to the catalogue gets the data, which is then displayed for the user.

3.3. Data sources

The aim of AMI is to extract and correlate the information which users need for dataset selection from the primary sources of that information. This extraction – or data pulling – is done by a special AMI task server. Tasks are run as java threads, and scheduled with variable frequency using a task control database table.

The current sources for dataset information stored in AMI are the ATLAS production database[8], the production Task Request system [9], the ATLAS CVS repository, and two other AMI applications: Tag Collector [10] and the Monte Carlo Dataset Number Broker (described in section 4.3).

In the very near future we will be able to correlate information obtained from the DDM catalogues.

Naturally it is also possible for authorized physicists to update information in the dataset catalogues using either the web interface or a web service client.

3.4. The AMI web interface

The AMI web interface gives users access to the dataset search, to their personal page, to related AMI applications such as the Monte Carlo dataset number broker, and it is also possible simply to browse all the data in AMI.

The AMI dataset search web interface implements several types of search:

- A simple search which performs a partial string search on the dataset name
- A simple search which performs a keyword string search on a selection of fields.
- A more complex search which allows the user to specify the required values for a number of key parameters. This interface was defined with the help of one of the ATLAS user groups.
- A “hierarchical” search for datasets, where the user selects one parameter at a time.

The results are displayed for all catalogues. The user can modify the appearance of the results by selecting the fields to display, or the order in which they are displayed. Pop-up windows give additional information about the meaning of the fields. Where it is appropriate, the user can request a list of the values present in the catalogue for a field.

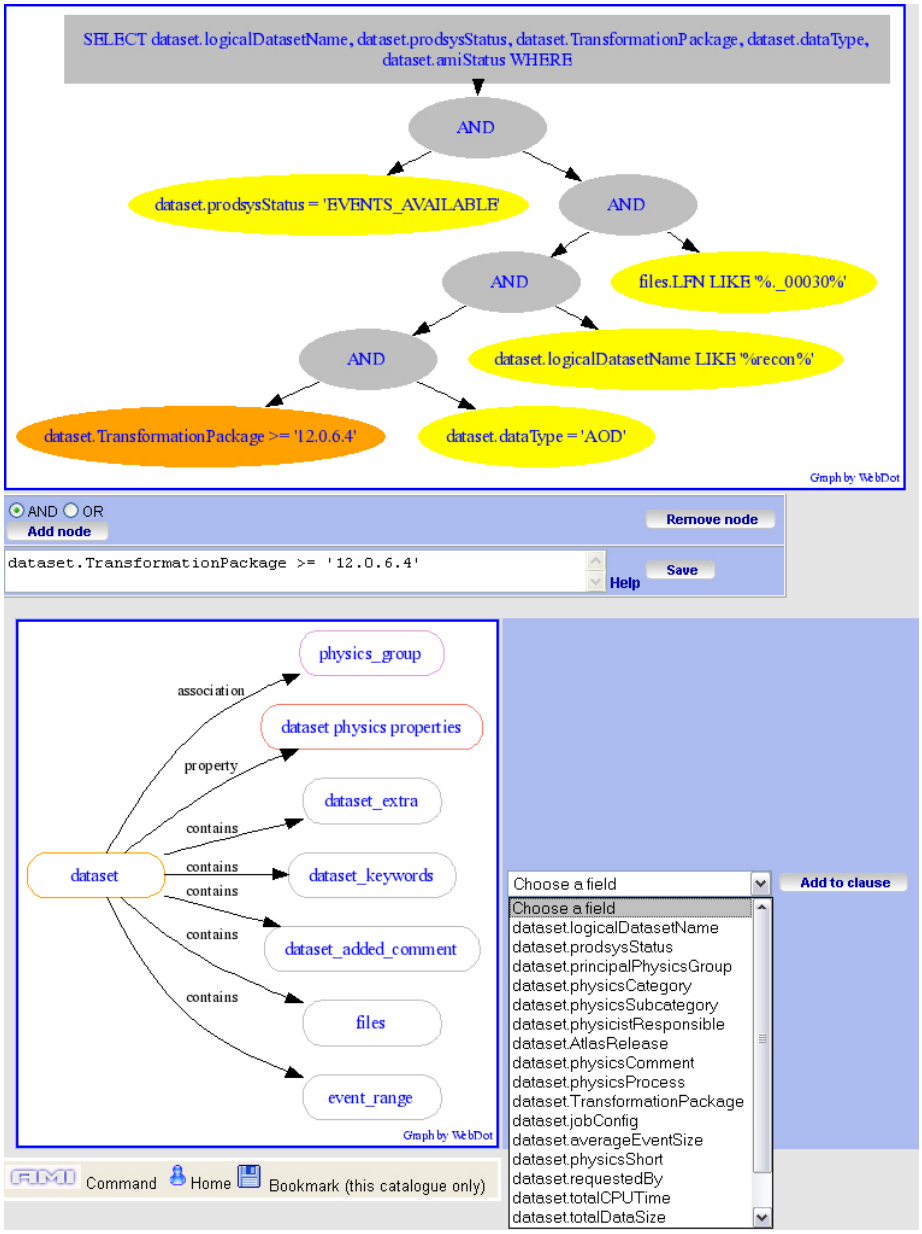

Figure 1 : A screen shot of the AMI interface for refining a standard query. The user sees a graphic representation of the gLite query, which he or she can modify by adding a condition on any field in the schema.

Because of the generic, multi-catalogue structure of AMI the parameters which the user can define in the "fixed" query interfaces are those which are present in the majority of catalogues. But we also provide a means to build an arbitrary SQL request on a selected catalogue. A graphic interface shows the user the complete schema for the selected catalogue, and guides the selection of the fields. As an example, one set of ATLAS users is interested only in those datasets of a catalogue which have the AOD format and which have exactly 30 files. A bespoke query constructed using the "Refine Query" function permits this list to be provided. This interface is shown in Figure 1; it is constructed dynamically by introspection of the schema.

When a user logs on to AMI, all the web pages are modified according to the rights the user has in a particular catalogue. For example, a user who has the rights to update dataset information will see a button which will enable this to be done. Other users will not see the button. Each user has access to a personalized home page. The home page is to list the user's bookmarks, and the roles assigned to the user.

Since we have used a “POST” technology, where parameters passed to the server are not visible in the URL, bookmarking AMI pages in the usual way was not possible. Therefore we have implemented an internal bookmarking system. On every AMI search result page there is a button which saves the query for the user. Several physics groups have used this feature to link the datasets of particular interest to their members. Results are always recalculated dynamically.

The AMI server exploits the bookmarking system and the AMI task running mechanism in order to refresh the database cache with frequently used pages which are resource hungry. This results in a considerable reduction in the response time, whilst ensuring that the user always gets the latest information.

As mentioned above (section 3.1), AMI now supports the passing of a VOMS proxy to a third party application, and this feature will be used to allow ATLAS members to register datasets both in the DDM catalogues and in AMI in an integrated way, using a web interface.

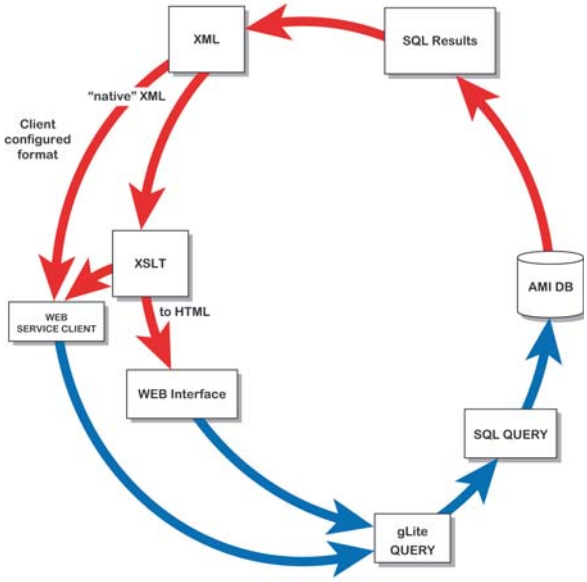

Figure 2: The XML/XSLT mechanism and the use of gLite grammar.

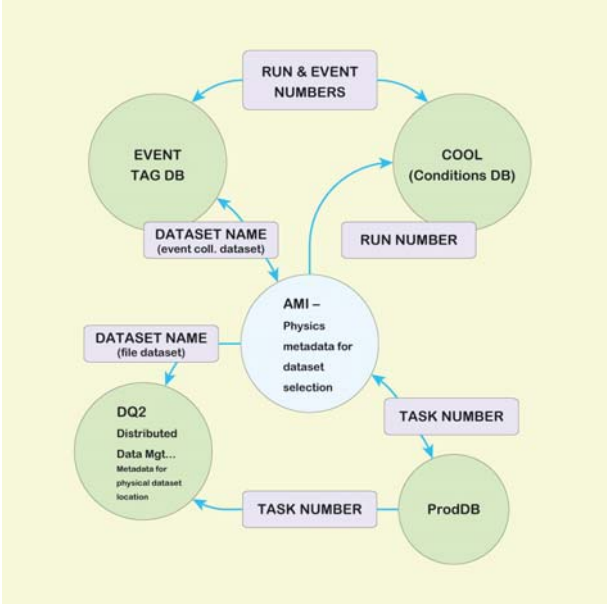

Figure 3: A “map” showing the parameters which enable navigation around different metadata actors in ATLAS

3.5. Web Service

AMI has provided a classic “SOAP” web interface since 2003. The AMI API now contains over 50 different commands for different actions on the dataset catalogues, although in many cases they are just wrappers to generic commands. We decided not to represent these commands individually in the web service interface (the “WSDL” file) but to keep things very simple. The client process can be summarized as “send us AMI a command; get the reply”. The XML of the reply can be used directly by the client, or transformed using one of the XSLT files we provide. The advantage of this approach is that we do not have to redistribute clients if a new command is added. The disadvantage is that clients must rely on independent documentation to know which commands are available.

The web interface implementation uses the same set of commands as those which are available for the web service, (see Figure 2) but in this context the XML output of the command is transformed to HTML. The web interface provides a means to show the AMI command which was actually used. So it is possible to extract the command and to paste it into a script.

The web interface also provides a special “web service” page where users can try out web service commands, without having to install a client.

A SOAP interface is not well adapted for all client needs, and so we have also provided a Representational State Transfer (REST) type interface for some special functions.

3.6. The Python interface

One of the advantages of a web service is that in theory a client can be generated for any modern computing language. But we do not expect the average physicist to be able or willing to do this. An AMI client written in Python (pyAMI) is available in the ATLAS software release. It will return results either as a Python object or in text. One of the main users of pyAMI will be the ATLAS distributed analysis tool GANGA [11]

4. Other AMI applications in ATLAS

The generic framework of AMI has been exploited to provide several other applications. Below we list the main ATLAS applications; however the AMI generic framework has also been used for non ATLAS applications. Each application uses the same generic code in its lower layers, as explained in paragraph 3.1., and each has a specific top layer of code.

4.1. Tag Collector

The Tag Collector tool is a central part of the release management of ATLAS [10], and as such holds all the information on the package versions which were included in each ATLAS offline software release. It allows releases to be built in a highly controlled way.

The versions of software which make up the release are part of the physics metadata; in consequence the dataset catalogues need only record the overall ATLAS release number to gain access to all the other software package versions.

4.2. The ATLAS Metadata Dictionary

The ATLAS metadata task force report [1] lists a number of essential metadata parameters. It is certain that this list will grow with time. No attempt was made to harmonize the nomenclature of parameters. Since we have several applications which hold this essential metadata, we need to make sure that there are no name clashes.

The purpose of the metadata dictionary is to catalogue the parameters, and to record a unique mnemonic name for each parameter. The dictionary can then be used by physicists who want to know which catalogues hold the value for a particular parameter.

Internally AMI should be able to use the metadata dictionary as one of the components of the planned mediator interface (see section 5).

4.3. The Monte-Carlo Dataset Number Broker

The ATLAS convention for dataset nomenclature requires that each Monte Carlo dataset be tagged by a number, which is similar to the run numbers used for real data. Monte Carlo datasets numbers are associated with a particular physics process. Once assigned, a given number should not be used for any other process. Each dataset number is said to belong to one particular physics or detector working group.

The dataset number broker, implemented in AMI, catalogues the association of numbers and physics processes. It allows physics groups to give additional information about the number like the name of a particular physicist who is responsible for the production of the data, or a link to a web page which contains more complete information. Physics groups may reserve a range of dataset numbers for future use. The information in the dataset number catalogue is linked to the Monte Carlo datasets which are produced.

4.4. Other ATLAS metadata reference tables

Many metadata parameters have a restriction on the values which they can take. Some examples are the data type (RAW, AOD, ESD…) and the tag of the geometry database which was used in the data

processing. Such reference tables are essential for keeping metadata coherent throughout the collaboration. AMI is well adapted as a tool to hold many of these reference tables for all the collaboration, and has already been used in this capacity, as explained in the previous sections.

5. AMI as a metadata mediator.

ATLAS metadata information is available from many sources, with different granularities. Users need to be able to navigate from one granularity to another in order to find and combine the data from these different sources. This is not a trivial task since the information is distributed across many applications, and each source application presents a different interface and exports the data in a different format. Figure 3 shows the essential parameters for navigation around ATLAS metadata.

A mediator interface is defined as "a software module that exploits encoded knowledge about some sets or subsets of data to create information for a higher layer of applications" [12]. To put it another way, a mediator is an application which puts some domain expertise between the user and a group of data sources, so that information coming from these different sources can be aggregated. The metadata task force identified the need for some types of user queries which would span more than one of the metadata actors. For example:

1. Select datasets from AMI where runs in (select runs from COOL where Conditions)
2. Select datasets from AMI where (AMI Condition) **or** (runs in (select runs from COOL where Conditions))
3. Select datasets from AMI where (AMI Condition) **and** (runs in (select runs from COOL where Conditions))

The job of the mediator level is to analyze the cross-queries and divide them into sub-queries which are sent to the appropriate data holder. The results are then translated into some common language, combined and returned to the user. In some cases the most efficient way to proceed is to wait for the result of one sub-query before sending another; in others it is possible to execute sub-queries in parallel. The combined results can be UNIONS or INTERSECTIONS of sub-query results.

The first step in a mediator implementation is to put all the expert knowledge needed into the AMI server software. The ATLAS metadata dictionary (section 4.2) is just the first small step towards mediation, but it has helped to establish a list of "intersection" parameters which must be shared between applications to allow for navigation (see Figure 3). This architecture implies that AMI needs to contain adaptors for all the metadata actors, which is a limiting factor in terms of scalability.

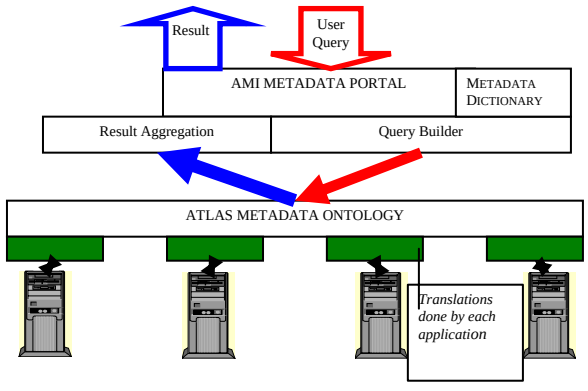

Figure 4 : A scalable architecture for mediation. Each application implements an adaptor to the common language.

A more scalable architecture would be based on the agreement of an "ontology" for ATLAS metadata. "Ontology" in this context can be defined as a common vocabulary for sharing information about a domain. It includes definitions of the basic concepts in the domain, and the relations among them. Each actor would then implement its own adaptor to translate to and from the common language (Figure 4). But for the moment, such implementation plans are very long term in the ATLAS environment!

6. Current status and prospectives

The number of users of AMI is expected to increase rapidly with the advent of real data. The flexibility of the AMI framework will allow us to meet the new use cases for analysis and new functions will be added as required.

The previous section described our intention to work towards the establishment of AMI as a metadata portal for ATLAS which will allow navigation to other metadata holders, and also cross queries. In the following sections we discuss some other specific aspects of our preparation for the exciting period ahead of us.

6.1. The Tier 0 test: File catalogues in AMI

The first version of the ATLAS distributed data management architecture assumed that users would never need direct access to files, and so the physics bookkeeping would not need to have a file catalogue. All data access would be by using datasets, or sets of files. No physics metadata would be recorded at the file level. However the luminosity task force [3] expressed concern about this decision, and it was later decided that a file physics metadata catalogue should be implemented.

In 2006-2007 ATLAS has performed an exercise called the Tier 0 test, which is a prototype for the handling of real data. The main purpose was to test the feasibility of the computing model which requires the distribution of AOD to all ATLAS Tier 1 grid sites. We were asked to test the capacity of AMI to catalogue files as a peripheral part of the exercise.

The registration of files in a database needs an insertion frequency an order of magnitude greater than the registration of datasets. The AMI software was not designed for rapid insertion, so in order to demonstrate that AMI could be used for file metadata a dedicated task was written. It bypasses almost all the schema discovery layers of AMI, and exploits a multi-threading technique. Data is read from the Tier 0 test database at CERN and written in parallel to an AMI database at the CCIN2P3, Lyon. The required data insertion rate was specified as 1 Hz. This task has been running without problem for 4 months, and has attained an average insertion rate of 8 Hz. We were able to demonstrate that if the CCIN2P3 database is unavailable for as long as three days, the AMI insertion task can catch up in a matter of hours.

6.2. Quality of service

As ATLAS moves towards real data we expect a large increase in the number of physicists using AMI. It is therefore one of our primary concerns to ensure that the quality of service remains good. Two approaches are followed. Firstly each component of AMI must be as solid as possible. Secondly we must remove single points of failure.

We are slowly moving away from mySQL; all new dataset catalogues are implemented on ORACLE at the CCIN2P3. We benefit not only from the power of ORACLE, but also from the support available at a national Tier 1 grid site. We have participated in a test of replication [14] which demonstrated that it would be relatively easy to have a read-only replica of AMI at CERN. We anticipate that this will be useful in a time scale of about a year. The ORACLE resource requirements for AMI are fairly modest in the global ATLAS database context.

We must also duplicate our web server, and introduce a mechanism for “failover” and load balancing. A solution such as that proposed by the Sequoia project [15] is envisaged.

6.3. Conclusions

AMI has shown itself to be a flexible tool which can be used for the physics metadata catalogues of ATLAS. Both datasets and files will be catalogued, from real and from simulated data. The AMI interface will be one of the main points of entry for all ATLAS metadata and so it should become a portal for all metadata sources. Metadata is correlated from several input sources and held in the AMI tables. In addition we will provide mechanisms for navigation towards other metadata actors within ATLAS, and if required, we will implement ways of addressing a query to more than one actor.

Acknowledgements

We would like to thank Chun Lik Tan of the School of Physics and Astronomy, University of Birmingham, UK, for providing the first version of the Python AMI client, and Claudia Roncancio, Maître de Conférences at the Institut National Polytechnique de Grenoble, France, for many useful discussions in particular concerning database mediator architecture.

References

[1] http://ami.in2p3.fr
[2] Giovanna Lehmann Miotto et al, https://edms.cern.ch/file/833723/1/Metadata.pdf
[3] Stefan Ask, David Malon, Thilo Pauly and Marjorie Shapiro https://edms.cern.ch/file/755341/1/lumi_report_v1_1_final.pdf
[4] https://twiki.cern.ch/twiki/bin/view/Atlas/DistributedDataManagement
[5] Thomas Doherty, http://ppewww.physics.gla.ac.uk/preprints/2007/01/
[6] Peter Kunszt and Ricardo Rocha. https://edms.cern.ch/file/573725/1.2/EGEE-TECH-573725-v1.2.pdf
[7] Birger Koblitz and Nuno Santos The gLite AMGA Metadata Catalogue 2006 *Proc. Int. Conf. on Computing in High Energy and Nuclear Physics (CHEP-06)* vol II (Macmillan India) p 820
[8] https://twiki.cern.ch/twiki/bin/view/Atlas/ProdSys
[9] Alexei Klimentov and Pavel Nevski, AKTR Task Request, Private Communication
[10] Emil Obreshkov et al. Organization and Management of ATLAS Software Releases. *Submitted to Nuclear Inst. and Methods in Physics Research, A* . NIMA-D-06-00734
[11] Jakub Moscicki, The Ganga job management system, *XI International Workshop on Advanced Computing and Analysis Techniques in Physics Research (ACAT2007)*, Amsterdam, 23-27 April 2007
[12] Gio Wiederhold, Mediators in the architecture of future information systems, *IEEE Computer Magazine*, Vol 25, No3, p38-49 March 1993
[13] Armin Nairz and Luc Goossens, The ATLAS T0 software suite, *(this conference)*
[14] Dirk Duellmann et al. LCG 3D Project Status and Production Plans. *Proc. Int. Conf. on Computing in High Energy and Nuclear Physics (CHEP-06)* vol II (Macmillan India) p 994
[15] http://sequoia.continuent.org/HomePage