An integrated overview of metadata in ATLAS

E J Gallas[1], D Malon[2], R J Hawkings[3], S Albrand[4] and E Torrence[5]

[1]Department of Physics, Oxford University, Denys Wilkinson Building, Keble Road, Oxford OX1 3RH, United Kingdom

[2]Argonne National Laboratory, High Energy Physics Division, Building 362, 9700 S. Cass Avenue, Argonne - IL 60439, United States of America

[3]CERN, CH - 1211 Geneva 23, Switzerland

[4]Laboratoire de Physique Subatomique et de Cosmologie, CNRS-IN2P3, Universit Joseph Fourier, INPG, 53 avenue des Martyrs, FR - 38026 Grenoble Cedex, France

[5]University of Oregon, Eugene, OR 97403-1274, United States of America

E-mail: gallas@cern.ch

Abstract. Metadata–data about data–arise in many contexts, from many diverse sources, and at many levels in ATLAS. Familiar examples include run-level, luminosity-block-level, and event-level metadata, and, related to processing and organization, dataset-level and file-level metadata, but these categories are neither exhaustive nor orthogonal. Some metadata are known a priori, in advance of data taking or simulation; other metadata are known only after processing–and occasionally, quite late (e.g., detector status or quality updates that may appear after initial reconstruction is complete). Metadata that may seem relevant only internally to the distributed computing infrastructure under ordinary conditions may become relevant to physics analysis under error conditions ("What can I discover about data I failed to process?"). This talk provides an overview of metadata and metadata handling in ATLAS, and describes ongoing work to deliver integrated metadata services in support of physics analysis.

1. Introduction

Metadata – data about data – arise in many contexts, from many diverse sources, and at many levels in ATLAS. This presentation focuses specifically on the usage of metadata along the data recording and processing chain from

- online data taking
- through various stages of processing
- to physics analysis.

Metadata is recorded at each phase. Parallel chains of data processing are used to improve the understanding of detector calibration, alignment and data quality. Metadata connections between these chains make it possible to

- know which data does not have sufficient quality to be included in the final analysis,
- improve and/or understand the resolution of various measurements and their efficiencies,
- follow the data through various stages of processing (and know when processing failure occurs), and

- make the connection between the data in the final physics analysis to the conditions existing when those events were recorded online.

This ultimately enables the calculation of the probability of particular particle production mechanisms in a range of beam conditions.

The ATLAS Luminosity Task Force [1] and the Metadata Task Force [2] inventoried the Metadata required for analysis connecting these stages. This presentation provides an overview of the implementation of this metadata and its handling in ATLAS and describes ongoing work to deliver integrated metadata services in support of physics analysis.

2. ATLAS Data and its Processing

The ATLAS experiment is one of two large multi-purpose detectors designed to study the proton-proton collisions of the LHC [3]. An overview of the detector, a description of its subsystems and its trigger system are described in the ATLAS Technical Paper [4]. See Ref. [5] and references therein for an overview of the ATLAS Computing Model and how distributed analysis tools have evolved for data processing and analysis on the world-wide Computing Grid.

Data from ATLAS is recorded in various forms and processed using task definitions described in the subsections below, noting the quantities which form the metadata that is used to combine information in later stages of processing and analysis. These metadata quantities will be denoted in this text by typesetting them in a BOX.

2.1. Data Path for Events

At LHC design luminosity, the ATLAS detector will have the opportunity to look at nearly 10^9 beam crossings per second, each of which may contain zero, one, or more proton-proton collisions (an **event**) of interest. The ATLAS multilevel trigger system is designed to quickly evaluate which crossings have characteristics meeting criteria defined in a prearranged menu (called a trigger configuration). Items of the menu are called trigger **chains**, each of which can be thought of as a logically independent set of criteria which is imposed on each event producing one of two outcomes: pass or fail.

If an event passes one or more chains as well as the corresponding prescale for that chain, that event is written out for further offline analysis. Included in the event output is the **trigger decision** for each event: a map of pass/fail bits, one bit for each chain in the trigger configuration. Prescales may be adjusted during data taking to regulate the output event rate – the software model offline processing/analysis constraints suggest an output event rate of about 200 events/second is acceptable. The menu is optimized to search for a range of topological signatures so that the recorded events satisfy a broad range of analysis topics.

When the LHC reaches proton-proton "collision mode", ATLAS starts a Run which is designated by a unique integral RUN_NUMBER. Runs are expected to last on the order of hours. Because detector and beam conditions can vary a great deal on that timescale, we choose to divide Runs into Luminosity Blocks which last no more than a couple of minutes. Luminosity Blocks (LB) are denoted with a unique integer in the Run, called the Luminosity_Block_Number (or LBN) which starts with 0 or 1 at the beginning of the Run, increasing incrementally by a count of 1 for the duration of the Run.

When an event satisfies the criteria of one or more trigger chains (and its prescale) the event is written to one or more streams ("inclusive" streaming) each designated by a STREAM_NAME. The mapping of chains to stream name is defined in the trigger configuration. ATLAS has chosen an inclusive streaming model [6] for all physics data to facilitate physics processing and analysis on the offline side as well as to control accounting complexities through the processing chain. The RAW event contains its RUN and LBN designation, its unique EVENT_NUMBER within the

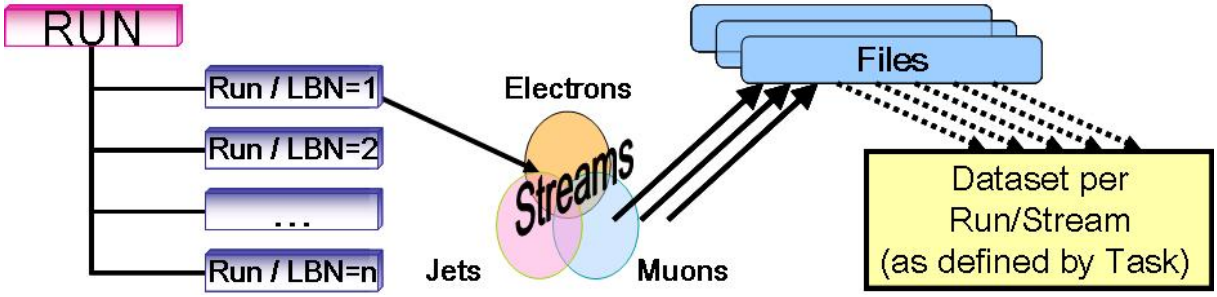

Figure 1. Runs consist of sequential Luminosity Blocks. Events satisfying any chain are written to that chain's designated streams into files opened at LB boundaries. Valid datasets are formed from the merging of complete LB for each Run and Stream designation.

Run, data sampled from detector elements active in the event [1], and a select set of derived quantities from the trigger system along with the trigger decision bit mask mentioned earlier. RAW events are combined in files designated by unique FILE_NAME respecting LBN boundaries for Physics analysis streams [2]. RAW files are combined into "valid" datasets designated by unique DATASET_NAME. These datasets generally correspond to all valid data from a Run and Stream assignment, removing luminosity blocks where events or files were lost.

It is these datasets which continue into the offline official processing chain of reconstruction and many stages of reprocessing described in subsection 2.3. ATLAS physicists analyzing data generally use the valid datasets described here or use data processed into files by tasks which insure they are derived from complete datasets. Components of datasets designated as "invalid" are also processed for the purpose of evaluation, but are not included in physics analysis.

Before tasks can be described, it is necessary to describe the "conditions data" associated with the data being processed with each task.

2.2. Data Path for Conditions

A wide variety of information is needed in online data taking and offline analysis which is not "event-wise": the information represents "conditions" or states of systems associated with an interval ranging from very short to infinity. Such information, called "conditions data," is stored in the ATLAS Conditions Database (based on LCG Conditions Database infrastructure) and is accessed using the COOL (Conditions database Of Objects for LHC) API. See Ref [7] and references therein for details.

All conditions data is characterized by an IOV (an Interval of Validity) which is an interval in time (UTC timestamp) or an interval in RUN_NUMBER/LBN range.

Conditions data is organized by subsystem each with a designated SCHEMA_NAME. Within each subsystem, folders (specified with FOLDER_NAME) contain a set of information (the payload) which share the same granularity. Folders containing repeated sets of information for many identical components (such as high voltage channels) have an additional index enumerating/naming CHANNEL numbers/strings.

Folders which may contain independent sets of information (versions) are defined to have an

[1] A single event is not expected to register a signal in all 100M detector elements so only the signals above a minimum threshold are stored in the RAW event.

[2] Streams in ATLAS fall into two distinct categories: Physics streams (events recorded for physics analysis) have extensive bookkeeping to insure events and files are not lost at any stage of merging, reconstruction or reprocessing. Non-Physics streams (such as calibration, debug or express streams) do not adhere to strict accounting rules so that processing is not inhibited by intermittent failures in the system.

additional index called a COOL_TAG. Thus many sets of calibrations can be stored within the Conditions Database – a system of hierarchical tags [8] designate a distinct set of calibrations to be used in processing tasks over many IOVs.

A wide variety of storage options is available within the infrastructure for the payload – the storage option is optimized in each case for the type and/or volume and/or structure of the payload and how it is expected to be used in processing tasks. As a rule, any conditions data needed for offline processing is stored in the ATLAS Conditions Database (conditions needed online comes from a combination of the conditions database and other sources).

Itemized here is a sampling of the (notably wide) scope of conditions information stored within the ATLAS Conditions Database which is used offline:

- LHC beam conditions,
- online configuration and operation,
- calibration,
- alignment,
- data quality,
- luminosity normalization,
- object reconstruction efficiency and
- bookkeeping data for various cross checks of data completeness/integrity.

Conditions data spans online and offline and in some cases bridges the gap between them. A good example is trigger configuration information which is recorded online but is used potentially through the final stages of offline analysis wherever trigger decisions are decoded. Each of these sets of conditions has metadata characterized by SCHEMA_NAME, FOLDER_NAME, and IOV along with the optional CHANNEL and COOL_TAG forming a complete set of the metadata pointers needed to access distinct sets of conditions for that system over a range of IOVs. Exploiting higher-level hierarchical tags, COOL_TOP_LEVEL_TAGs can include many subsystem's COOL_TAGs characterizing conditions across many subsystems over a range of IOVs.

It is quite advantageous to store this wide variety of data under the same infrastructure. While the conditions database may be humongous, uniformity in the architecture makes access to all conditions from processing and analysis jobs more methodical. It makes extraction and partial distribution of conditions data from many subsystems relatively unburdened by the diversity which would inevitably result from conditions stored in a variety of frameworks which would arise by optimizing conditions storage for each subsystem.

The complete and authoritative repository for ATLAS Conditions data is in an Oracle instance at CERN. In the ATLAS Computing Model, though, much of the processing and all the analysis of data takes place on the grid. The model for distribution of conditions data on the grid remains relatively unchanged from that reported in the previous CHEP 2007 [9], (using the Oracle Streams technology to replicate conditions data to Tier 1 Oracle sites) but the implementation continues to be tuned as additional subsystems with new use cases exercise the system. SQLite file replicas of slices of the conditions database and FronNTier [10] / Squid (web) caching have been further explored and utilized. Recent results from conditions data distributed operations scalability testing have been presented in this meeting [11].

2.3. Managing Data and Tasks

In this report, the term **task** is used to generically refer to the execution of a program on input event data using conditions data (not required for all tasks) producing an output. If the program designation or the input data or conditions are not precisely defined, the task will fail in the sense that it is not reproducible. To make it reproducible, the code being executed and its configuration, the input data and the input conditions must be specified precisely – which

is accomplished using the appropriate metadata pointers in the input specification to the task. Of course the other requirement for success is that the metadata pointers point to data within systems which is consistent with the other references (i.e. there must be Conditions data in the requested folders during the intervals of the input dataset).

The most commonly discussed and most strictly defined tasks in ATLAS are the staged processing of RAW datasets into data formats suitable for reprocessing and analysis described in the Computing Model:

- RAW (RAW datasets)
- ESD (Event Summary Data) - full reconstructed data,
- AOD (Analysis Event Data) - physics analysis data,
- DPD (Derived Physics Data) - reduced AOD information in more compact form, and
- TAG (Event Metadata files)

Critical in the management and description of "valid" Datasets for physics analysis is

- **AMI (the ATLAS Metadata Interface) [12]**
  AMI is the dataset selection application cataloging all official datasets for the experiment. Physical metadata (dataset location) is tracked by DQ2[13].
  To note a couple of AMI features:
  (i) Datasets registered in AMI follow nomenclature rules [14] which continue to evolve as new use cases arise. Dataset names are a concatenation of strings separated by "." (full stop). For example, a dataset name might have the form:

  Project.runNumber.streamName.prodStep.dataType.ConfigTag

  These rules allow dataset name component strings to be mnemonic (hint toward dataset content or origin), while they are actually metadata pointers with precise meaning being tracked deeper within AMI.
  For example, the ConfigTag in the above convention is actually a shorthand pointing to a list of task configurations used for the production of the dataset.
  (ii) Datasets registered in AMI have provenance, tracing parentage of the dataset back to its origin and the processing configurations along the way.

Critical in the management and description of Tasks is

- **ATLAS Tag Collector [15]**
  The Tag Collector application is a repository (with a database back end) and an associated interface (both web and command line based) for management of the official software releases. ATLAS software is stored in a code repository, with versions tagged within that repository. Software managers control which tagged software is allowed into each ATLAS software release, tagging it at a higher level for the precise building of a complete set of code for official execution on ATLAS datasets.

Both AMI and the Tag Collector have web interfaces providing entry and reporting interfaces for the information behind the metadata and links to other ATLAS systems where further information can be found. Progress and status of these applications has been presented at this [16] and previous [17] CHEP conferences and work is ongoing to meet the needs of a growing information base and expanding the application for new dimensions of metadata.

The PanDA [18] system actually manages the ATLAS distributed production processing tasks. PanDA shares information about these tasks and the datasets they produce with AMI. AMI combines this information with other information on validation, to present a coherent view of valid ATLAS datasets to users. To summarize, metadata is used extensively to create and

track official ATLAS datasets and configure data processing tasks referring precisely to input configuration and conditions and the software executed in the task.

A wide variety of data processing is executed by subsystems which may not be completely encompassed by the above. An example of this parallel processing for subsystem calibration and alignment was presented previously at CHEP [20]. The result of jobs such as these may be written to the Conditions Database (with a new COOL_TAG) and is thereby available[3] for use by the online system or at any stage in the offline processing.

3. Metadata and Distributed Computing

Many technical innovations in the last 30 years have made it possible for us to execute increasingly complex computational tasks: reconstruction, reprocessing and analysis in a distributed environment. Remaining a fundamental challenge is that of data access latency, which occurs at many levels. This is illustrated in Figure 2 at a fundamental scale. Square data points show the improvement in processing speed, while starred points show the improvement in the DRAM transfer rate over the same period; Memory latency has not kept pace with the improvements in processing speed. Bruce Elmegreen (IBM) denotes the gap which continues

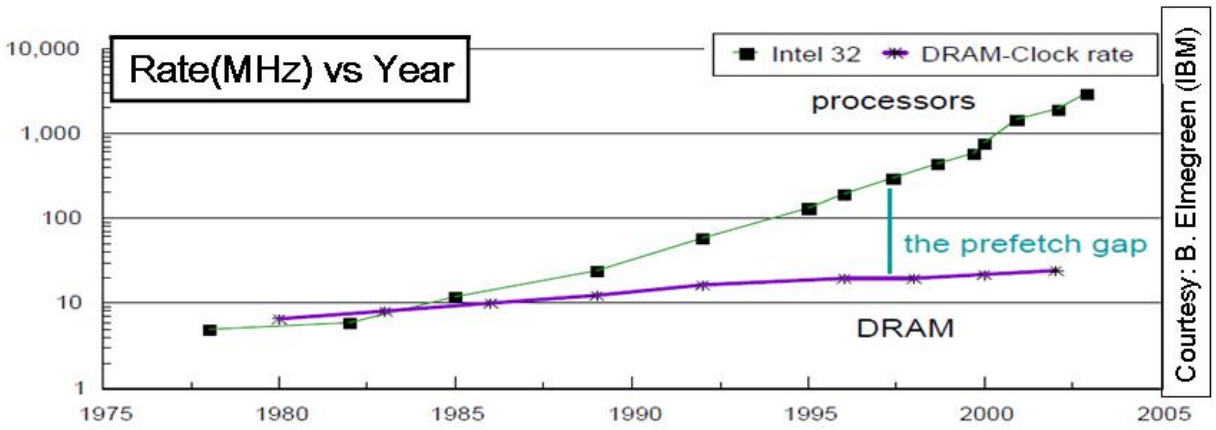

Figure 2. Processing speed and DRAM access speed over the last 30 years shows a growing gap (courtesy of Bruce Elmegreen, IBM Research Division, 2009).

to widen between these curves as "the prefetch gap" – while CPU has improved considerably, access to memory increasingly limits overall performance. For our tasks, this kind of prefetch gap for data access is magnified at each level—memory to CPU, disk to memory, and storage element or remote database to local disk. How does metadata help ?

In ATLAS tasks execution, data from two sources is generally required: the input datasets (a set of files) and the conditions data required by the task (in the Oracle database). There are many ways in which metadata helps bridge the gap: putting the specific data required in proximity of the grid computing elements that will execute the task.

For the conditions data, obviously, this is easier when we have pre-knowledge of the input data required. This is the case for major reprocessing campaigns, which are executed on datasets with known RUN_NUMBER ranges and distinct sets of conditions data (known IOV ranges of data tagged with COOL_TOP_LEVEL_TAGs) The slice of conditions data needed from Oracle can be extracted into SQLite files and included in the release of the code to the sites which will execute the tasks.

[3] Admittedly, recombining conditions from many sources into the database can be a knotty operation, the details of which cannot for lack of space be elaborated here.

For other tasks where the conditions data required is difficult to predict, we can learn by use cases where database intensive tasks tend to be run and what data they generally require. Examples include calibration, alignment, monitoring and user analysis code. In the first three cases, these tasks will be run generally at the same sites, so for those cases, we can extract the COOL_TAG-ged data for the subsystems required into local files and configure those sites to look for those files (site-specific configuration is a feature of the database connections made through the COOL interfaces). This is already being put into place for muon calibration, which runs at various Tier 2 sites.

If FroNTier technology is deployed, conditions database access in recent releases of conditions access code has been modified so that IOV requests are made more uniform. This helps FroNTier cache information that will satisfy a number of similar queries in the same time range.

Streamlining conditions database access for user analysis is less well understood because use cases have not been fleshed out and we don't know when user analysis will move from AOD based analysis with customized code to analysis based on official releases of code. When ATLAS analysis code reaches the latter stage, we can narrow the range of conditions data that will be required: data quality, luminosity and beam conditions, and object efficiency. This knowledge in itself makes it possible to imagine creating replicas with these folders for the latest COOL_TAGs (which will have the latest quality, luminosity etc). The size of these replicas and extent of distribution requirements are a matter of fine tuning, file management and/or optimization of FroNTier or squid caches to handle the required queries.

4. Is MIF evil ?

In one of the meetings on ATLAS metadata coordination, a fellow collaborator [21] enumerated the reasons why "MIF (Metadata in Filenames) is evil". To paraphrase, it is:

- Fragile: under changing standards, spelling/case variations, or moving files,
- Closed: Addition of metadata is not possible and it is
- Limiting: It probably doesn't contain everything you want.

But we're sure if we looked in our fellow collaborator's userdisk area, we would find nondescript filenames like "test.py", loads of files without a clear naming convention or an index describing the function of each file. We all do it whenever we create any file or new program – and we should continue to do so because it is useful. A tipping point is reached when we recognize the stage of development where formalized naming rules are important and useful, such as when files enter code repositories ... creating naming rules with a view to the long term. Dataset names in ATLAS went through this transition, and the rules on ATLAS dataset nomenclature mentioned earlier [14] emerged, then grew to be composed of strings which are metadata pointers to information at a deeper level. Dataset nomenclature continues to evolve and is a regular point of discussion.

Metadata in files (another MIF) is another slippery slope which can lead to unmanageable consequences if not carefully considered and controlled. Up to this point, we have stated that event records are stored in files. The records themselves contain, obviously, lots of event-wise information and they know just enough to connect to data at a higher level (i.e. RUN_NUMBER, LBN, EVENT_NUMBER, and timestamp); but event records have a limited world view. For example, they knows which trigger chains were satisfied (by bit number), but they don't know the trigger requirements behind those bits which is critical to analyze them. The conditions database makes it possible to get this information, but if some conditions could judiciously be added to files, it can speed the types of analysis using it.

At later stages of processing, collections of events are produced for specific analyses based on event selection predicates forming new files. A file containing such culled events would be meaningless for a complete physics analysis without saving, in some form, the requirements

imposed on the sample (for efficiency evaluation) and pointers to the datasets from which the sample was derived (for cross section calculations or scaling relative to background processes).

We concluded some time ago that metadata in files is a necessity [22] and have continued to explore its utility and and its consequences. In fact, at various stages of reprocessing, metadata is added to files and in general, the more data is processed and reduced, the more metadata is helpful at the file level. In-file metadata is used to track processing and provenance (e.g., the samples from which this sample was derived), to assist in robust auto-configuration of jobs to read data correctly, and as a cache for frequently-accessed metadata authoritatively managed elsewhere. File volume concerns keep the level of in-file metadata in check and the balance between content, volume, and backward compatibility is evolving. In this conference, ATLAS presented a new usage of file-based metadata [23] which facilitates event based selection without reading event data.

5. Metadata for People

Physicists have broad interests and responsibilities in ATLAS. They need to find and analyze events offline which suit the task at hand. For example:

(i) A subsystem expert would like to find some data taken in the online cosmic ray running period about 6 months ago... he/she want Runs that were at least an hour long where subsystems X, Y, and Z were included in the detector readout.

(ii) A member of physics Group A wants to find events with at least 2 reconstructed jets with $E_t > 5$ GeV in all Runs when a particular trigger chain was configured and during Run periods which Group A has assessed the data quality to be "good."

ATLAS has a growing base of event samples: simulated event datasets, a growing repository of datasets composed of "mixed" simulation events (samples of events mixed in proportion to their expected cross sections), and online commissioning and cosmic ray datasets. To efficiently analyze these and in anticipation of real collision data, the experiment has developed increasingly realistic infrastructure to allow us to efficiently find runs and event samples of interest.

The two subsections below describe two systems which satisfy use cases such as those above using dynamic queries to metadata stored in databases.

5.1. Finding Runs of interest

The ATLAS Conditions Database houses a considerable expanse of conditions data as described in previous sections. This data is not only useful for processing and analysis tasks – it includes a good deal of information which is of interest to physicists seeking to dynamically assess aspects of the data.

An application called CoolCherryPy [24] makes dynamic read and write access to the conditions database possible via dynamic web interfaces which can be customized to include a variety of queries of interest using conditions predicates. A number of such web interfaces have emerged, one of which is the ATLAS Run Query interface [25] which physicists can use to select Runs by one or more criteria available in the conditions database run and data quality related folders. Also in this conference, the author of CoolCherryPy explores recent web technologies for database information retrieval and interactive web display[26], useful for further explorations of conditions as well databases with a more traditional relational structure.

5.2. Finding Events of interest

Production of ATLAS TAG files was listed in 2.3 as the final stage of formal reconstruction and processing. TAG datasets contain not only event-level Metadata (in a format readable using ROOT[27]) but also information needed to navigate back to those events in earlier stages of processing.

To enhance the metadata selection available to the user, TAGs are uploaded to a relational database and select conditions data is added into relational tables which can be updated as improved conditions data becomes available. A web-based tool is being developed [28] which combines the power of event-level and conditions metadata with other ATLAS and grid infrastructure tools to further enhance the selection and retrieval process. The technical challenges involved in the maintenance and distribution of the database and interfaces should not be underestimated, and is also described in this conference[29].

6. Status

We have alluded to, but have insufficient space in this article to fully describe, many other uses of Metadata in ATLAS:

- Using metadata to get statistics without reading large amounts of data
- Determining the status of various processes
- Accounting systems used to track event and/or file losses
- Checking data integrity
- Managing tasks in a grid environment [18]
- Managing files and datasets in a grid environment [13]
- Tracking and use of dataset provenance (history of processing) [12]

It should be mentioned that many pitfalls exist in the dependence on Metadata: Metadata sources may not be currently or consistently filled so inputs must be continually cross checked. Metadata is generally an amalgam of information from various systems, each of which becomes a point of failure for the metadata integrity. Difference sources of the same metadata field may contain conflicting values (no one likes digging through someone else's dirty laundry). Sources of information may change with or without notice. The scope of Metadata usage must be well defined, as well as the business logic of each application using it.

Work is ongoing

- to deliver integrated metadata services in support of physics analysis,
- on infrastructure to connect seamlessly the Events to the Condition intervals,
- to incorporate Good Run List definition and management, and associated tools and
- to address identification and consequences of processing failures.

7. Conclusions

We have used metadata to facilitate many aspects of data handling and analysis in the ATLAS experiment. Metadata not only helps us meet technical challenges of data processing, it also facilitates physics analysis:

- bringing validated data samples to collaboration physicists and
- allowing an unprecedented possibility for the customization of analysis in a distributed computing environment.

Our metadata is not just a continuous thread tying together online data taking conditions to events in an offline ROOT file. It is a mesh of pathways connecting events by the metadata which they have in common, creating multidimensional views of the data by which we can optimize selection criteria to facilitate the type of calibration or analysis in our task list.

Our model for metadata usage is currently being exercised with online detector commissioning, online cosmic ray and offline simulation samples. We are eager to put the system to the trial of reality (pp collisions are currently expected in late 2009) and look forward to the challenges of beyond 2010, when the volume of the metadata itself becomes so large, we need metametadata.

References

[1] Ask S, Malon D, Pauly T and Shapiro M (July 2006) "Report from the Luminosity Task Force" https://twiki.cern.ch/twiki/pub/Atlas/DataStreamingReport/lumi_report_v1_1_final.pdf.

[2] Costanzo D, et al (April 2007) "ATLAS Metadata Task Force" ATL-GEN-PUB-2007-001 http://cdsweb.cern.ch/record/1026734/files/gen-pub-2007-001.pdf.

[3] LHC Home Page: http://lhc.web.cern.ch/lhc/

[4] ATLAS Collaboration, "The ATLAS Experiment at the CERN Large Hadron Collider", https://twiki.cern.ch/twiki/pub/Atlas/AtlasTechnicalPaper/Main_jinst_submission.pdf.

[5] Jones RWL and Barberis D (2008) The ATLAS Computing Model *J. Phys.: Conf. Series* **119** 072020 http://www.iop.org/EJ/article/1742-6596/119/7/072020/jpconf8_119_072020.pdf.

[6] Von der schmitt H, et al, "Report on Data Streaming in ATLAS" https://twiki.cern.ch/twiki/bin/view/Atlas/DataStreamingReport.

[7] ATLAS Conditions Database: https://twiki.cern.ch/twiki/bin/view/Atlas/ConditionsDB

[8] Working with COOL Tags: https://twiki.cern.ch/twiki/bin/view/Atlas/CoolTagging.

[9] Vaniachine A V, Von der schmitt J G, "Development, Deployment and Operations of ATLAS Databases"(2008) *J. Phys.: Conf. Series* **119** 072031 http://www.iop.org/EJ/article/1742-6596/119/7/072031/jpconf8_119_072031.pdf.

[10] CERN FronNTier Home Page: http://frontier.cern.ch/

[11] Walker R, et al, (2009) "Advanced Technologies for Scalable ATLAS Conditions Database Access on the Grid", these proceedings http://indico.cern.ch/contributionDisplay.py?contribId=81&sessionId=60&confId=35523

[12] The Atlas Metadata Interface (AMI): https://twiki.cern.ch/twiki/bin/view/Atlas/AtlasMetadataInterface.

[13] ATLAS Distributed Data Management: https://twiki.cern.ch/twiki/bin/view/Atlas/DistributedDataManagement.

[14] Albrand S, et al, (evolving) "ATLAS Dataset Nomenclature" ATL-COM-GEN-2007-003 http://cdsweb.cern.ch/record-restricted/1070318/.

[15] The Tag Collector: A Tool for ATLAS code release management https://twiki.cern.ch/twiki/bin/view/Atlas/TagCollectorInAtlas.

[16] Albrand S, Fulachier J, Lambert F, "The ATLAS Metadata Interface", this conference. http://indico.cern.ch/contributionDisplay.py?contribId=212&confId=35523.

[17] Albrand S, Doherty T, Fulachier J and Lambert F "The ATLAS METADATA INTERFACE" (2008) *J. Phys.: Conf. Series* **119** 072003 http://www.iop.org/EJ/article/1742-6596/119/7/072003/jpconf8_119_072003.pdf.

[18] The PanDA Production ANd Distributed Analysis system https://twiki.cern.ch/twiki/bin/view/Atlas/Panda.

[19] Nilsson P, et al, "Experience from a pilot based system for ATLAS" (2008) *J. Phys.: Conf. Series* **119** 062038 http://www.iop.org/EJ/article/1742-6596/119/6/062038/jpconf8_119_062038.pdf.

[20] Verducci Monica, "ATLAS Conditions Database Experience with the LCG COOL Conditions Database Project" (2008) *J. Phys.: Conf. Series* **119** 042031 http://www.iop.org/EJ/article/1742-6596/119/4/042031/jpconf8_119_042031.pdf.

[21] Shaun Roe, quote and paraphrase from an ATLAS meeting (2007).

[22] Malon D, Van gemmeren P, Hawkings R and Schaffer A, "An inconvenient truth: file-level metadata and in-file metadata caching in the (file-agnostic) ATLAS event store" (2008) *J. Phys.: Conf. Series* **119** 042022 http://www.iop.org/EJ/article/1742-6596/119/4/042022/jpconf8_119_042022.pdf.

[23] Van gemmeren P, Malon D and Nowak M, "New Developments in File-based Infrastructure for ATLAS Event Selection", this conference, http://indico.cern.ch/contributionDisplay.py?contribId=97&confId=35523.

[24] Roe SA, "A RESTful web service interface to the ATLAS COOL database", this conference http://indico.cern.ch/contributionDisplay.py?contribId=170&sessionId=60&confId=35523.

[25] ATLAS Run Query: http://atlas-runquery.cern.ch/.

[26] Roe SA, "Ajax, XSLT and SVG: Displaying ATLAS conditions data with new web", this conference http://indico.cern.ch/contributionDisplay.py?contribId=168&sessionId=60&confId=35523.

[27] ROOT: http://root.cern.ch.

[28] Cranshaw J, et al, "Event Selection Services at ATLAS", this conference. http://indico.cern.ch/contributionDisplay.py?contribId=191&confId=35523.

[29] Viegas F, et al, "The ATLAS TAGS Database distribution and management - Operational challenges of a multi-terabyte distributed database", this conference. http://indico.cern.ch/contributionDisplay.py?contribId=172&sessionId=28&confId=35523.