EPJ Web of Conferences **214**, 04051 (2019) https://doi.org/10.1051/epjconf/201921404051
CHEP 2018

Optimizing access to conditions data in ATLAS event data processing

Lorenzo Rinaldi[1,*], *Elizabeth J* Gallas[2,**], and *Andrea* Formica[3],
on behalf of the ATLAS Collaboration [***]

[1]Dipartimento di Fisica e Astronomia, Università di Bologna and INFN Sezione di Bologna, Via Irnerio 46, I-40126 Bologna, *IT*
[2]Department of Physics, University of Oxford, Denys Wilkinson Bldg, Keble Rd, Oxford OX1 3RH, *UK*
[3]Université Paris-Saclay, CEA/Saclay IRFU, 91191 Gif-sur-Yvette, IRFU/CEA, *FR*

Abstract. The processing of ATLAS event data requires access to conditions data which are stored in database systems. This data includes, for example alignment, calibration, and configuration information which may be characterized by large volumes, diverse content, and/or information which evolves over time as refinements are made in those conditions. Additional layers of complexity are added by the need to provide this information across the worldwide ATLAS computing grid and the sheer number of simultaneously executing processes on the grid, each demanding a unique set of conditions to proceed. Distributing this data to all the processes that require it in an efficient manner has proven to be an increasing challenge with the growing needs and numbers of event-wise tasks. In this presentation, we briefly describe the systems in which we have collected information about the database content and the use of conditions in event data processing. We then proceed to explain how this information has been used not only to refine reconstruction software and job configuration but also to guide modifications of underlying conditions data configuration and in some cases, rewrites of the data in the database into a more harmonious form for offline usage in the processing of both real and simulated data.

1 The ATLAS Conditions Database

All processing of event data collected by the ATLAS [1] experiment at the CERN LHC is performed using a complementary set of data in the ATLAS Conditions Database (DB) which describes the configuration, state, calibration and alignment of the detector sub-components during the time interval in which the event data was taken. The database design is based on the LCG Conditions Database and the database is accessed by clients using the COOL API, both of which were developed by the CERN IT department for the LCG [2]. The cardinal repositories of the data reside in two Oracle DBMS instances at CERN; one for online data taking and another for offline processing. The offline instance is replicated to three Tier 1 sites (TRIUMF in Canada, IN2P3 in France, and RAL in the UK) to provide global access to the data on the WLCG.

[*]e-mail: rinaldi@bo.infn.it
[**]e-mail: gallas@cern.ch

Conditions data are in general not event-wise, but extend over longer time intervals from minutes to hours in duration. The data collectively characterizes the states of all ATLAS subsystems, organized by "Intervals of Validity" (IOV), which is the span in time over which that data is valid. The data is collected from many sub-systems such as the Detector Control System (DCS), trigger and DAQ, Data Quality, ATLAS sub-detectors, and the LHC accelerator complex.

Conditions data are stored in several Oracle schemas according to subsystem and whether the data is used online or exclusively offline. This division is historic, dating from before the start of LHC Run 1 data taking, in order to cleanly divide the data in the database for the different subsystems and its online usage. Within each schema, the data is logically grouped into "folders" so that clients may selectively access that folder's data when needed.

Folders are classified as single-version (SV) when the nature of the data is such that it has a fixed value in each IOV (e.g. a configuration setting). A multi-version (MV) folder, on the other hand, is used for folders where more than one set of data can apply in a given IOV (e.g. a calibration or alignment). Online applications save conditions which have been observed during a run such as configuration, run parameters, measured physical parameters, etc. This type of data is generally stored in SV folders, while data stored from the offline (such as calibrations resulting from offline studies) are stored in MV folders (leaving the possibility to refine them in future). MV folder data is always associated with a version, called a "folder tag". When event data processing requires MV folders, the folder tags for each folder requested by the client must be specified: sets of folder tags are put into a group called a "global tag" (a collection of folder tags) which can more succinctly be cited in client task configuration. At the present time, all such data is stored in the ATLAS Conditions DB, which currently contains data in over 2300 folders, distributed over 32 schemas. Conditions data are accessed in all ATLAS event data processing activities, in many phases of analysis, as well as in monitoring and calibration studies by subsystems.

A folder's "payload" is the set of values stored in the relational database columns of that folder. The LCG Conditions database software leaves a lot of freedom in the choice of the payload structure and on the column data types. Allowed data types include the standard numeric and string types, along with LOB (Character or Binary Large Object types, i.e. CLOBs and BLOBs). In addition, special string-type columns are used to designate references to external data files (file GUIDs and internal files pointers) registered in the ATLAS DDM/Rucio system [3].

In offline distributed computing, Athena [4] jobs are efficiently provided with concurrent conditions data access from the ATLAS Conditions DB via a caching system which mediates client requests: the Frontier [5] system. If a client queries data from a folder with the special external references, Athena methods provide the data from a distributed CVMFS [6] file system. For the case of clients operating in environments without network access to Frontier, the required conditions are extracted into SQLite files which can be provided to the processing server so that jobs can access the data file locally.

In this proceedings, we describe how we have built upon existing tools and resources to gain an improved understanding of database usage by clients and of the structure and content of the database itself. We further describe how we used this knowledge to

1. direct subsystems toward more efficient utilization of the Conditions DB infrastructure for their data storage and
2. develop improved tools for building collections of Conditions data into SQLite files for use by specific data processing tasks.

This knowledge also puts us in a better position for further improvements in how we store and use this data.

2 Conditions DB Content

The LGC Conditions database and COOL API were designed and implemented in the years before the start of LHC Run 1 for generic use by experiments. The design factored in the experience from previous experiments as well as from feedback from the experiments who planned to use it (including ATLAS). Knowing that some database tables may be large and that all queries via the COOL API would be qualified with filters on key columns such as IOV and tag names, the database design includes the indexing of those columns and the COOL API includes SQL hints with every query: these measures considerably improve the read performance of the database for clients by instructing the database how best to execute each query (e.g. whether to use the index or not).

The above design considerations served the experiments using COOL well throughout Run 1 as long as, it was thought, the table volumes were well regulated. In order to manage volumes, ATLAS deployed an "COOL Monitor" application which stores daily a set of metrics associated with each folder and generates a simple HTML report based on that data. The report is intended for subsystem and database experts to use to view folder-wise insert rates and volume. A snapshot of this report is shown in Figure 1.

Figure 1. Snapshot of the list of ATLAS COOL production database instances

Specifically, this web page shows, for each COOL DB schema and instance, the folder name and type (SV or MV), the average row length of the payload tables, the number of rows, channels and tags in the last day, week and month. This interface worked well enough in the Run 1 start-up to recognize and reduce unnecessarily high rates, but the interface gradually fell into disuse once data entry procedures stabilized.

With the start of Run 2, the application was updated to include new folders as they were created over time. There was a slight degradation in read performance which was not immediately recognized, obscured by Oracle storage capacity improvements. As Run 2 evolved however, demands for conditions increased with an ever increasing number of jobs on the grid, creating a significant degradation in conditions delivery per job. The COOL Monitor itself was found to have a number of limitations in its misleading assumptions in the use of its input data and limited scope for improvement in its current implementation. We realized that we needed better tools to understand not only job requests, but also to understand the data content in a more thorough and systematic way, not only for current operations, but to design a better system for Run 3[7] as well.

The COMA application, which had already aggregated metadata about Conditions DB folder and tags [8] was a natural place to add more detailed information about each folder.

The COMA schema extensions for adding these metrics are shown in the red font in Figure 2.

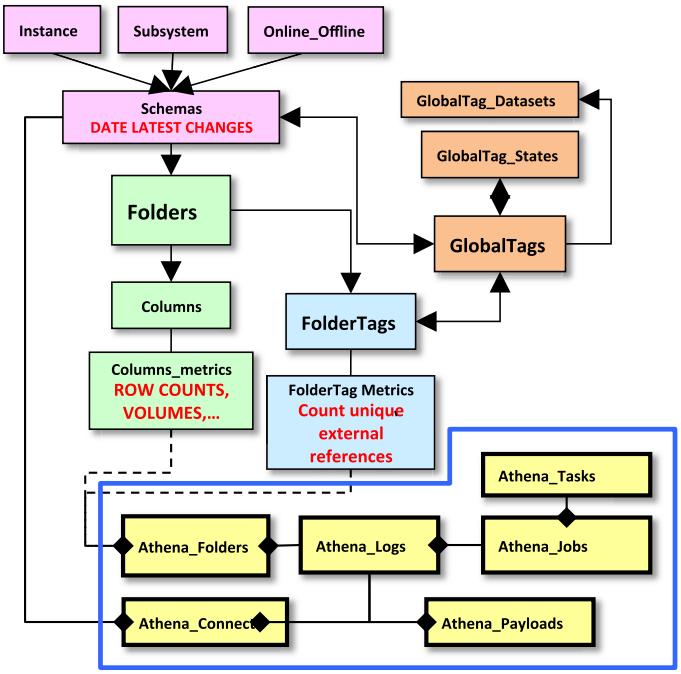
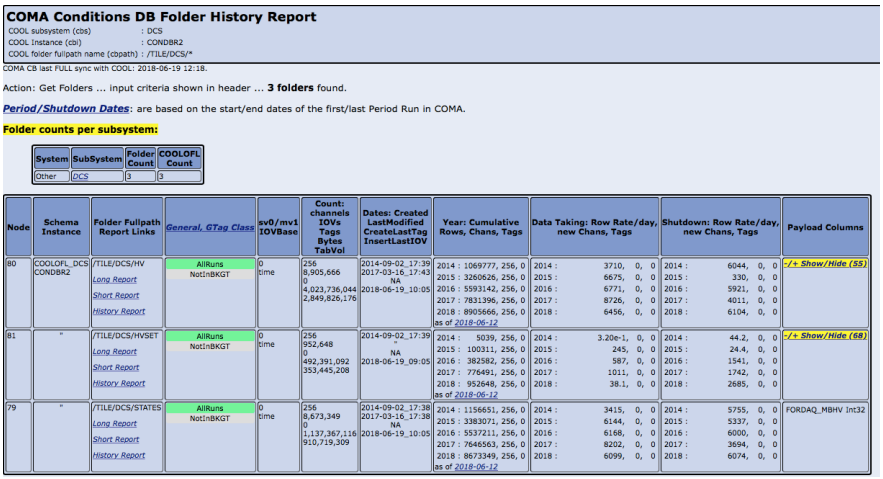


Figure 2. The COMA Conditions DB metadata schematic. Metrics associated with each folder are added to tables with the red font. Additional yellow tables show the extension of the schema to collect conditions data usage from Athena jobs.

The COMA metadata now includes information such as row counts and average row volume (for folders as well as in the underlying database tables), the counts of distinct references to external data by folder and folder tag, counts of channels and payload columns along with details about the payload. A variety of time-stamps were added, which are used to synchronize the application with its data sources and is useful to determine when the latest data was added to each folder or folder tag (tells us which parts of the Conditions DB are growing, or not). These enhanced metrics have been added to existing COMA reports and command line access to this information has also been made available.

In addition, a new relational table was added to COMA to store an aggregation of the history of row, channel, and tag counts and volume per folder from the COOL Monitor schema (which contained both duplicated rows and rows with stale data) in order to provide a more succinct but complete history of these metrics over time. This information is used to populate new COMA Folder History Reports, such as the one shown in Figure 3, which can be dynamically generated for any folders. This report shows a yearly summary of cumulative rows, channels and tags as well as an estimate of the yearly folder row insertion rate during data-taking and shutdown periods. A more detailed report is available per folder, showing the complete history in weekly intervals (which is in sync with the real collection of these rates from Oracle Statistics, which is only generated on a weekly basis). Average daily rates are particularly important not necessarily from the storage point of view but because high rate translates to higher load offline when the data is accessed by grid jobs. These reports are



COMA Conditions DB Folder History Report

COOL subsystem (cbs) : DCS
COOL Instance (cbi) : CONDBR2
COOL folder fullpath name (cbpath) : /TILE/DCS/*

COMA CB last FULL sync with COOL: 2018-06-19 12:18.

Action: Get Folders ... input criteria shown in header ... **3 folders** found.

Period/Shutdown Dates: are based on the start/end dates of the first/last Period Run in COMA.

Folder counts per subsystem:

System	SubSystem	Folder Count	COOLOFL Count
Other	DCS	3	3

Node	Schema Instance	Folder Fullpath Report Links	General, GTag Class	sv0/mv1 IOVBase	Count: channels IOVs Tags Bytes TabVol	Dates: Created LastModified CreateLastTag InsertLastIOV	Year: Cumulative Rows, Chans, Tags	Data Taking: Row Rate/day, new Chans, Tags	Shutdown: Row Rate/day, new Chans, Tags	Payload Columns
80	COOLOFL_DCS CONDBR2	/TILE/DCS/HV *Long Report* *Short Report* *History Report*	AllRuns NotInBKGT	0 time	256 8,905,666 0 4,023,736,044 2,849,826,176	2014-09-02_17:39 2017-03-16_17:43 NA 2018-06-19_10:05	2014 : 1069777, 256, 0 2015 : 3260626, 256, 0 2016 : 5593142, 256, 0 2017 : 7831396, 256, 0 2018 : 8905666, 256, 0 as of *2018-06-12*	2014 : 3710, 0, 0 2015 : 6675, 0, 0 2016 : 6771, 0, 0 2017 : 8726, 0, 0 2018 : 6456, 0, 0	2014 : 6044, 0, 0 2015 : 330, 0, 0 2016 : 5921, 0, 0 2017 : 4011, 0, 0 2018 : 6104, 0, 0	*-/+ Show/Hide (55)*
81	"	/TILE/DCS/HVSET *Long Report* *Short Report* *History Report*	AllRuns NotInBKGT	0 time	256 952,648 0 492,391,092 353,445,208	2014-09-02_17:39 " NA 2018-06-19_09:05	2014 : 5039, 256, 0 2015 : 100311, 256, 0 2016 : 382582, 256, 0 2017 : 776491, 256, 0 2018 : 952648, 256, 0 as of *2018-06-12*	2014 : 3.20e-1, 0, 0 2015 : 245, 0, 0 2016 : 587, 0, 0 2017 : 1011, 0, 0 2018 : 38.1, 0, 0	2014 : 44.2, 0, 0 2015 : 24.4, 0, 0 2016 : 1541, 0, 0 2017 : 1742, 0, 0 2018 : 2685, 0, 0	*-/+ Show/Hide (68)*
79	"	/TILE/DCS/STATES *Long Report* *Short Report* *History Report*	AllRuns NotInBKGT	0 time	256 8,673,349 0 1,137,367,116 910,719,309	2014-09-02_17:38 2017-03-16_17:38 NA 2018-06-19_10:05	2014 : 1156651, 256, 0 2015 : 3383071, 256, 0 2016 : 5537211, 256, 0 2017 : 7646563, 256, 0 2018 : 8673349, 256, 0 as of *2018-06-12*	2014 : 3415, 0, 0 2015 : 6144, 0, 0 2016 : 6168, 0, 0 2017 : 8202, 0, 0 2018 : 6099, 0, 0	2014 : 5755, 0, 0 2015 : 5337, 0, 0 2016 : 6000, 0, 0 2017 : 3694, 0, 0 2018 : 6074, 0, 0	FORDAQ_MBHV Int32

Figure 3. Snapshot of the COMA Folder History Report showing yearly cumulative rows, channels, and tags and a yearly estimate of row insertion rates during data-taking and shutdown periods.

currently being used by subsystems to view their relative rates and in cases where the rate is high, to consider how they may better structure their data or smooth existing data to obtain the needed level of precision for use by offline event data processing.

3 Conditions DB usage

Conditions data stored in the database are accessed via COOL methods by either real users using CLI utilities or Athena jobs executing on distributed computing nodes. Since Conditions DB usage globally is dominated by Athena job access, we chose to focus our analysis on the use of the data from Athena jobs, specifically by studying Athena logs from jobs of particular interest. Understanding usage generally and how usage varies with different job types is crucial to identify areas where modifications have the potential for the greatest improvement in overall efficiency. In order to gain a clearer picture of conditions data usage, job logs can be parsed to populate additional tables shown in yellow in Figure 2. The available information includes which global tag(s) were used by the job and, independently, as folders are accessed, which folder tag was associated with that access. Also, the logs contain a job-wise summary of all folders requested by the job along with the related total number of rows, data volume, channels, wall time required to retrieve this information, and whether (or not) the job actually used the conditions retrieved.

The fact that we have data retrieved which is unused by jobs is an obvious inefficiency which is being addressed on a case by case basis. Furthermore, by studying Athena logs against collected information about database structure and content, we have found several additional sources of inefficiency. Fundamental issues include how global tags are implemented within the Conditions DB infrastructure, how jobs use not only the conditions in a global tag but also request additional conditions outside of the global tag, and which folders should be targeted for potential refinement and reduction.

To further elucidate some of these limitations, we describe in the next section the improved process in producing customized SQLite replicas for jobs which do not have access to Frontier.

4 The Custom DB Release use case

SQLite replicas of portions of the Conditions DB content have always been a part of the ATLAS model for database distribution. In this model, ATLAS DB Releases are composed of file-based 'static' SQLite replicas of data selected from the Conditions, Geometry, and Trigger databases packaged with the POOL ROOT conditions data files pointed to by reference in some Conditions DB folders. DB Release tarballs are distributed on the grid and on shared file systems. These DB Releases allow jobs executing in environments with no Frontier access (such as HPC centres and home/small clusters) to have access to all the database-hosted data that they require.

Early in Run 1, DB Releases were used for real data reprocessing on the grid (before the full Frontier deployment). Since that time, they have been used solely for Monte Carlo simulation (MC) event processing. Adding real data conditions to DB Releases is possible, but the larger data volumes involved make this more difficult to manage and the addition would be needed only for processing real data on HPC clusters.

In the previous CHEP Conference, we described [9] general findings that simulation jobs require only a subset of all MC conditions. We have, since then, proceeded in developing more automated tools for determining more precisely which data we should include in DB Releases both inside and outside of the latest global tags. The goal is to produce slimmed replicas which contain sufficient conditions for current simulated event processing while minimizing the DB Release file size.

Initially, DB Releases containing all MC conditions were in the order of 10 GB in size. With the use of the latest tools for customized condition extraction, based on metadata collected from Athena logs from archetype test jobs and the use of collected folder and tag metadata, we have successfully moved to a model of production-on-demand customized DB Releases. These contain only the most recent best-knowledge conditions which are needed in all current simulation processing. When new MC conditions are added to the database, a revised DB Release can be produced including those latest conditions.

With this new machinery in place, the former routine production of DB Releases containing all MC conditions has been entirely dropped. We are now building and deploying these customized DB Releases on demand with size now on the order of 100 MB.

Studies are currently under way to build Custom DB Releases for some specific tasks requiring real data conditions, such as "MC Overlay" event processing. In this work-flow, real data zero bias events are inlaid with MC events to simulate multiple interactions of beam crossings so these jobs require both MC as well as real data conditions. The challenge in producing slimmed replicas including real data conditions is the large volume of data stored in these folders, as elucidated by the collected metrics. This is an ongoing effort which will likely involve condensing data in these folders without sacrificing the precision needed by offline event processing.

Custom DB Releases are also used by the benchmarking software tools. In those cases, the software should not overload the hosting system, so having light components could enhance the precision of benchmarking measurement. Recently, custom light DB Release have been implemented into the ATLAS KitValidation benchmarking tool [10], which is used inside the CERN Cloud Benchmarking suite.

5 Conclusions

Considerable progress has been made in the understanding of the content and usage of data stored in the ATLAS Conditions Database. While the database superficially may seem homogeneous since it is accessed by a fixed set of access utilities designed for an IOV-based

database, internally, the content is quite heterogeneous, with individual folder design varying widely (taking advantage of the wide variety of customization available within the LCG Conditions Database infrastructure). In ATLAS, we have elucidated this diverse content by collecting an expanded set of metadata for all folders and folder tags, which enables quick access to this information in reports and dynamic tools. We are also in the process of collecting more information about conditions data usage by clients alongside the content-related metadata. This combined knowledge enables us to better advise subsystems toward more efficient utilization of the infrastructure for their data storage and which is more amenable to conditions usage work-flows. It has also been used to further refine our production of customized DB Releases for use by jobs which do not have operational access to Frontier, such as HPC centres.

In addition to facilitating improvements in ATLAS tools for conditions handling and distribution, this knowledge also puts us in a better position for further improvements in how we store and use this data and how we can better model it for future needs.

References

[1] The ATLAS Collaboration, JINST **3** S08003 (2008); http://cern.ch/atlas/

[2] R. Trentadue et al., *LCG Persistency Framework (CORAL,COOL, POOL): status and outlook in 2012*, J. Phys.: Conf. Series **396** 053067 (2012) http://iopscience.iop.org/article/10.1088/1742-6596/331/4/042043

[3] V. Garonne et al., *Rucio – The next generation of large scale distributed system for ATLAS data management*, J. Phys. Conf. Ser. **513** 042021 (2014) http://iopscience.iop.org/article/10.1088/1742-6596/513/4/042021

[4] P. Calafiura et al., *Running ATLAS workloads within massively parallel distributed applications using Athena multi-process framework (AthenaMP)*, J. Phys.: Conf. Series **664** 072050 (2015) http://iopscience.iop.org/article/10.1088/1742-6596/664/7/072050

[5] D. Barberis et al., *Evolution of grid-wide access to database resident information in ATLAS using Frontier*, J. Phys.: Conf. Series **396** 052025 (2012) http://iopscience.iop.org/1742-6596/396/5/052025

[6] P. Buncic et al., *CernVM - a virtual software appliance for LHC applications*, J. Phys.: Conf. Ser. **219** 042003 (2010) http://iopscience.iop.org/article/10.1088/1742-6596/219/4/042003/meta

[7] L. Rinaldi, A. Formica, E.J. Gallas, N. Ozturk, S. Roe, *Conditions evolution of an experiment in mid-life, without the crisis (in ATLAS)*, this conference. https://indico.cern.ch/event/587955/contributions/2936820/

[8] E.J. Gallas ,S. Albrand, M. Borodin, A. Formica, *Utility of collecting metadata to manage a large scale conditions database in ATLAS*, J. Phys.: Conf. Series **513** 042020 (2014) http://iopscience.iop.org/article/10.1088/1742-6596/513/4/042020

[9] L. Rinaldi et al., *Collecting conditions usage metadata to optimize current and future ATLAS software and processing*, J. Phys.: Conf. Series **898** 042028 (2017) http://iopscience.iop.org/article/10.1088/1742-6596/898/4/042028

[10] M. Alef et al., *Benchmarking Cloud Resources for HEP*, J. Phys.: Conf. Ser. **898** 092056 (2017) http://iopscience.iop.org/article/10.1088/1742-6596/898/9/092056