A Tool for Conditions Tag Management in ATLAS

A. Sharmazanashvili[1], G. Batiashvili[1], G. Gvaberidze[1], L. Shekriladze[1], A. Formica[2] on behalf of ATLAS collaboration

[1] Georgian CADCAM Engineering Center
52, Rustaveli Ave., 0108 Tbilisi, Georgia

[2] CEA-IRFU, Saclay, Gif-sur-Yvette, France

E-mail: Lasha.Sharmazanashvili@cern.ch

Abstract. ATLAS Conditions data include about 2 TB in a relational database and 400 GB of files referenced from the database. Conditions data is entered and retrieved using COOL, the API for accessing data in the LCG Conditions Database infrastructure. It is managed using an ATLAS-customized python based tool set. Conditions data are required for every reconstruction and simulation job, so access to them is crucial for all aspects of ATLAS data taking and analysis, as well as by preceding tasks to derive optimal corrections to reconstruction. Optimized sets of conditions for processing are accomplished using strict version control on those conditions: a process which assigns COOL Tags to sets of conditions, and then unifies those conditions over data-taking intervals into a COOL Global Tag. This Global Tag identifies the set of conditions used to process data so that the underlying conditions can be uniquely identified with 100% reproducibility should the processing be executed again. Understanding shifts in the underlying conditions from one tag to another and ensuring interval completeness for all detectors for a set of runs to be processed is a complex task, requiring tools beyond the above mentioned python utilities. Therefore, a JavaScript /PHP based utility called the Conditions Tag Browser (CTB) has been developed. CTB gives detector and conditions experts the possibility to navigate through the different databases and COOL folders; explore the content of given tags and the differences between them, as well as their extent in time; visualize the content of channels associated with leaf tags. This report describes the structure and PHP/ JavaScript classes of functions of the CTB.

1. The COOL Database

ATLAS conditions database (also known as COOL) is implemented using LCG Condition DB infrastructure, with Oracle DB as backend and COOL as the API to enter and retrieve data [1]. Conditions data are associated with various ATLAS activities like detector commissioning, Monte-Carlo simulation, reconstruction and calibration. The condition database consists of several Oracle database schemas for each subsystem (e.g. online and offline dedicated schemas), and different COOL instances (sets of tables) within every schema, to separate reconstruction from simulation conditions. Within a COOL instance, the conditions data are organized in nodes (folders), every node corresponding to a specific set of data (payload), using a hierarchical tree structure (a parent folder will have one or many leaf folders). Inside a leaf folder, the data are stored using Intervals Of Validity (IOVs) to determine a range in time for a set of conditions. The COOL API allows to use as time interval either a real date-time in nanoseconds (since 1st January 1970), or a number associated to run/event number (or run/luminosity block) pair. Each folder uses only one of the above definitions. Several folder types are defined in COOL, allowing different levels of versioning and tagging for a set of IOVs. For example, in the case of multi version folders, the experts can tag a set of IOVs by using

an identification string (tag) that will be associated to every single IOV. Another level of tagging is called global tagging, where a user can associate a set of tags from different leaf nodes to a parent tag (called Global Tag or GTag) which will be defined at the root level of all folders for a given schema and COOL instance. Schema in Fig.1 illustrates the hierarchical structure of COOL nodes.

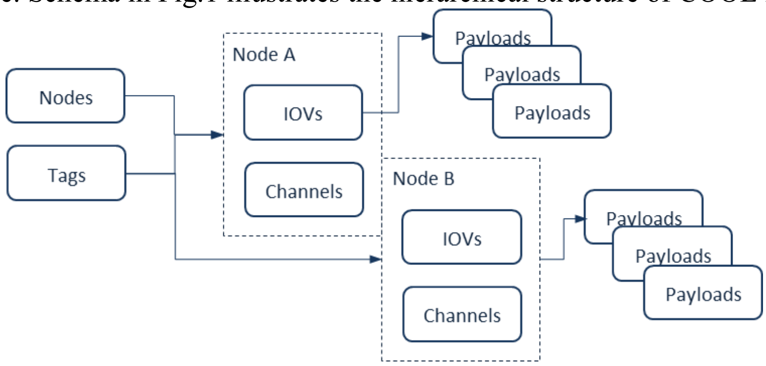

Fig.1: COOL structure

2. Data methods and sources

Several python/Java/JavaScript-based tools are implemented to explore COOL database content (Fig.2). The Cool Tag Browser uses various methods and sources in order to gather the relevant data for visualization of conditions database content. These data sources are either directly connected to COOL schemas and tables, or to other databases in which we can find part of the COOL DB content, and which are optimized for accessing a sub set of the COOL information. They can be accessed in general via web services, or using direct connections to Oracle DB (via SQL queries).

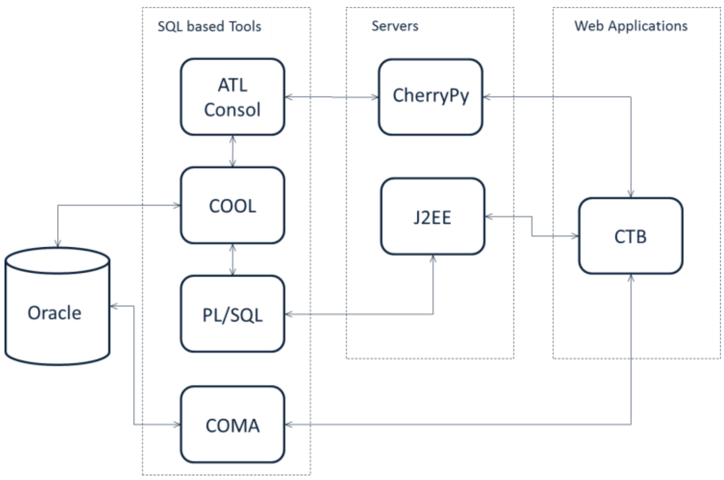

Fig.2: Database browsing tools

2.1. *CherryPyCool*

CherryPyCool [2] is a python based web application (developed under CherryPy server); it provides a RESTful web service [3] accessing COOL data via the COOL API, and making the data available via standard GET, PUT and POST HTTP methods. The URLs are interpreted to select specific resources as we can see in the following:

```
http://voatlas207.cern.ch:8080/ATLAS_COOLPROD/ATLASOFL_CALO/COMP200/tags
```

1 2 3 4 5

1/ hostname and port number of the CherryPyCool application;
2/ the COOL DB server to be accessed;
3/ the COOL Schema to be browsed;
4/ COOL DB instance name;
5/ the keyword delimiter of the method (in this example to retrieve tags).

The server makes available many important functionalities of the COOL API. The XML format is used for input and output data.

2.2. PL/SQL API

A PL/SQL API [4] has been developed mainly to overcome some limitations in the extraction of meta-data information from COOL; in particular the API delivers information on nodes, tags, number of channels, number of IOVs for a given tag etc., from several COOL schemas at the same time. Some special functions have been created to gather statistics related to the IOV range of a folder tag. The API is accessible from the database as a package, and it is used in read-only mode.

A Java application deployed in a J2EE server was developed in order to access the PL/SQL functions via a RESTful web service. A URL example to for this application is the following:

```
http://<HostName>:8080/JBRestCool/rest/{schema}/{db}/.../<KeyWord>
                1                 2                  3            4
```

1/ contains server references – address and port
2/ RESTful service references
3/ COOL schema, instance (nodes…) description
4/ special words act as a data delimiter, like: ‘nodes’, ‘tags’, ‘iovs’, ‘list’, ‘data’, etc.

The output formats are XML and JSON.

2.3. COMA Database

In a recent extension of the COMA database [5] a set of tables has been added in order to gather relevant meta-data from COOL schemas, to provide a fast method to look at folders, global tags and tags associations over all schemas. A typical SQL query to get COOL folder description from COMA, looks like:

```
$query="select NODE_DESCRIPTION, NODE_NAME,NODE_FULLPATH , CBO_NAME, CBS_NAME,CBI_NAME
        from ATLAS_TAGS_METADATA.COMA_CB_NODES inner
        join ATLAS_TAGS_METADATA.COMA_CB_OWNER_INSTANCES on
        ATLAS_TAGS_METADATA.COMA_CB_OWNER_INSTANCES.CBOI_INDEX
        =ATLAS_TAGS_METADATA.COMA_CB_NODES.CBOI_INDEX
        where CBO_NAME='".$database."' and CBS_NAME='".$schema."' and CBI_NAME='".$instance."'"." and
        NODE_FULLPATH='".$folder."'"
```

3. Conditions Tag Browser

The Conditions Tag Browser (CTB) is an application developed by a team of researchers from Georgia. It is a user interface framework providing coherent access to the above sources related to conditions data storage. CTB is a PHP/JavaScript application based on an Apache web server and using CherryPyCool, PL/SQL API and COMA DB tables for navigation through the COOL nodes, data retrieval and visualization. CTB implements also some higher level functions to compare global tags in order to spot differences in the associations with folder tags and check IOVs statistics on a given folder and tag by means of the PL/SQL API among others. CTB was developed for experts coordinating conditions data activities in ATLAS, needing to have an efficient tool to explore tag content at the level of the whole set of schemas and DB instances. CTB is a read-only service. Initially it was built on ASP and AJAX platforms. ASP configuration avoids 8080 port security restrictions on the server and is verified to work on popular browsers (Firefox, Opera, Safari, and IE). However this system was found to be slow, because JavaScript functions are executing on an intermediate server that send data to a local PC (Fig.3). The newer AJAX/JavaScript configuration (Fig.4) has faster performance because JavaScript functions execute inside the client browser.

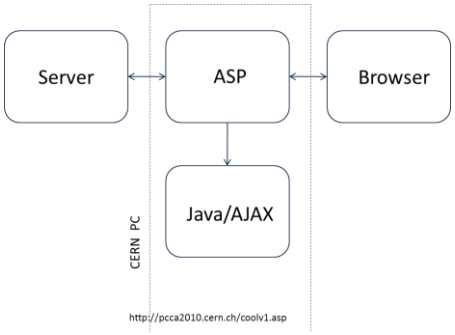

Fig.3: ASP based Browser

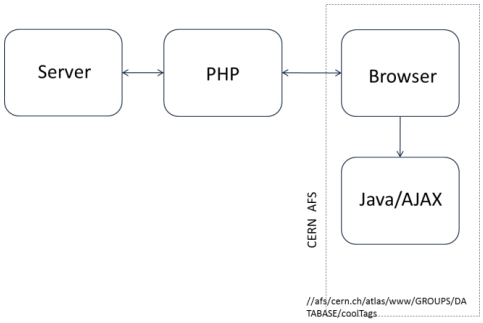

Fig.4: AJAX based Browser

As a result CTB architecture was built as a set of 5 PHP modules executing on the server and 3 JavaScript modules executing locally. PHP modules deliver two main categories of functions: Navigation and Search. JavaScript modules are instead used for data visualization and user interface. Following the COOL structure (Fig.1) four hierarchical levels of navigation have been identified for retrieving data from COOL:

1st level: for navigation through the system, instance and folders

SCHEMA
- ATLAS Subsystems
  - {CALO, CSC, DCS, GLOBAL, INDET, LAR, MDT, MUONALIGN, PIXEL, RPC, SCT, TILE, TRIGGER, TRT}
- COOL Databases
  - COMP200
    - {Folders}
      - {Description, Channel}

2nd level: for navigation through the global Tags

Global Tag
- {Description, Status, Hierarchy}

3rd level: for navigation through the Leaf Tags (folder tags)

Leaf Tag
- {Description, Status, Channels}

4th level: for navigation through the Channels

Tag Channel
- {ID, Insertion Date, IOV, Time Span, Data}

Search functions enable retrieval of all tag related information from COOL matching the users selection criteria. Depending on the input search functions are divided into two categories.

Selectable input functions include:

1. *Trace* function: returns list of all leaf tags associated to selected tag. Trace is most useful from the top-level folder to see the tags defined hierarchically for any leaf folder
2. *Back_Trace* function: returns a list of all schemas and folders of the selected tag
3. *Diff* function: enables comparison of two selected tags for the given schema and instance
4. *Compare* function: does the same but inside of a given folder
5. *Channel_Search*: retrieves list of channels with IOVs for the selected leaf tag.

Editable input functions include:

1. *Global_Search* function: enables navigation to start from the schema-DB-Folder set corresponding to the tag name of the reference string
2. *Folder_Search* function: retrieves the tag corresponding to the name selected inside the given schema-DB-Folder selection
3. *IOV's_Search* function: retrieves IOVs within a given schema-DB-Folder-tag selection
4. *Payload_Search* function: retrieves data for the given IOV and channel.

For each exploring method, the CTB chooses the data source which is known to give the best performance to the users (Fig.5).

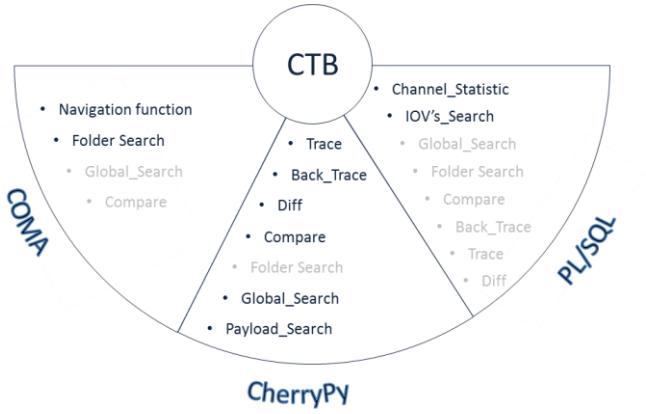

Fig.5 Distribution of functions by subsystems

COMA DB gives the fastest performance when retrieving meta-data information from COOL related to schemas, folders and tags content. According to our investigations query execution time, for CherryPyCool is in the range of 250-10'000ms, for COMA 1-3ms and for PL/SQL 50-400ms. So COMA was chosen for navigation and folder search functions. Since CherryPyCool access COOL tables directly, it gives more detailed information not present in COMA, like IOVs related information, and guarantees that the client retrieves the most up-to-date information (since COMA DB is synchronized daily). The main part of the search functions are realized on CherryPyCool – Trace, Back Trace, Diff, Compare, Global Search and Payload Search. The most interesting features of PL/SQL API is related to tag coverage checks, to verify the tag content by using special queries which are not implemented in COOL API. So PL/SQL was chosen as a base for the Channel statistic and IOVs search functions.
User interface functions based on JavaScript enable functionalities for:

1. Bookmarking current navigation scenario into a URL
2. Showing navigation path in a status string
3. Displaying job execution time to estimate system performance
4. Filtering of large tag list in a given folder
5. Displaying names of special Global Tags – Current, CurrentES, Next and NextES.

CTB is used for regular Conditions Data coordination tasks in order to prepare the Global Tags for the official reprocessing of physics data in ATLAS.

References

[1] A. Valassi, R. Basset, M. Clemencic, G. Pucciani, S. A. Schmidt, and M. Wache. Cool, LCG conditions database for the LHC experiments: Development and deployment status. In *Nuclear Science Symposium Conference Record, 2008. NSS '08. IEEE*, pages 3021–3028, 2008

[2] S. Roe et al. A restful web service interface to the atlas cool database. In *Journal of Physics: Conference Series*, volume 219, page 042021. IOP Publishing, 2010

[3] Fielding, R.T. Architectural Styles and the Design of Network-based Software Architectures, PhD dissertation, University of California, Irvine, 2000

[4] Formica A, private communication on PL/SQL API for COOL Metadata

[5] Gallas E, Albrand S, Borodin M and Formica A. "Utility of collecting metadata to manage a large scale conditions database in ATLAS", CHEP 2013, https://indico.cern.ch/contributionDisplay.py?contribId=251&confId=214784