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 Java/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/Java 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 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 (set 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 data (in general an IOV can contain conditions data for one or more channels, depending on the folder structure): 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, a set of IOVs can be tagged by the experts using an identification string 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. COOL API has been developed in C++ and python. Hierarchical structure inside COOL presented on Fig.1

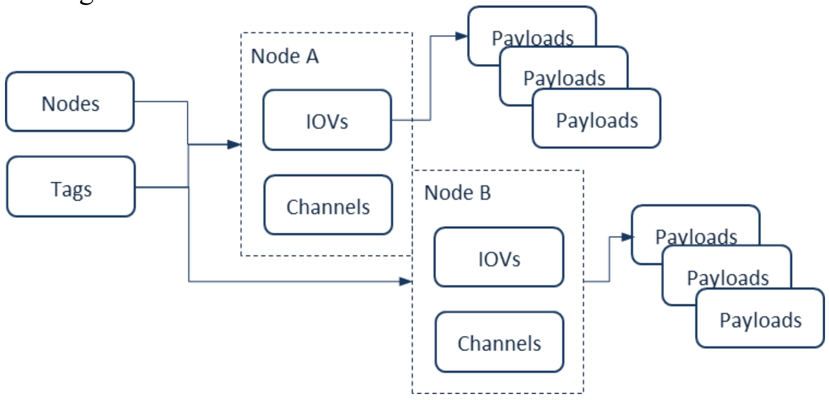

Fig.1 COOL structure

2. Database Browsing

Several python/Java/Javascript based tools are implemented to explore COOL database content (Fig.2). The Cool Tag Browser is using existing tools and databases 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 (SQL libraries).

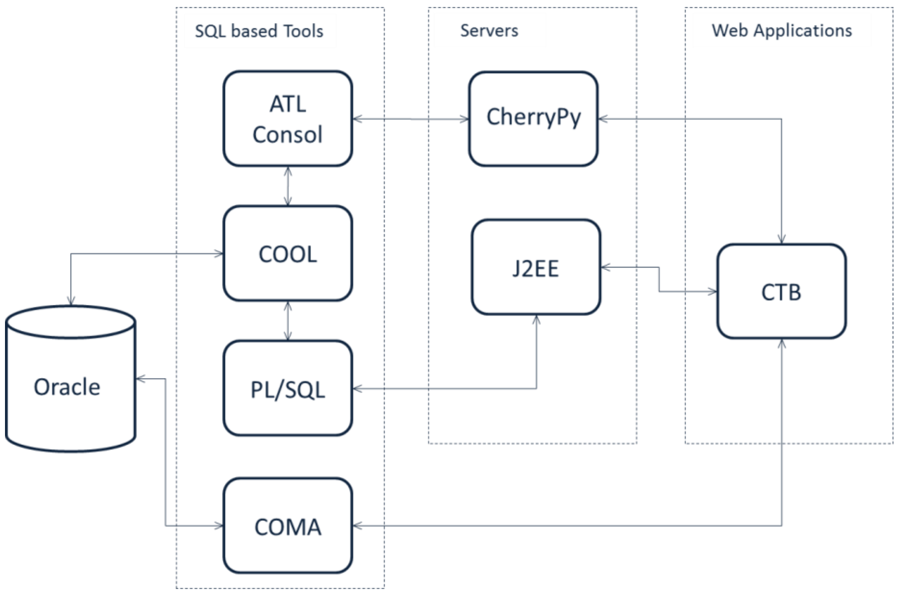

Fig.2 Database browsing tools

2.1. CherryPyCool

CherryPyCool [2] is a python based web application (developed under CherryPy server): provides RESTful web service [4] accessing COOL data via the COOL API, 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:

$$\texttt{http://}\underbrace{\texttt{voatlas207.cern.ch:8080}}_{1}\texttt{/}\underbrace{\texttt{ATLAS\_COOLPROD}}_{2}\texttt{/}\underbrace{\texttt{ATLASOFL\_CALO}}_{3}\texttt{/}\underbrace{\texttt{COMP200}}_{4}\texttt{/}\underbrace{\texttt{tags}}_{5}$$

1/ - hostname and port number of the CherryPyCool application; 2/ - COOL DB server to be accessed; 3/ - the COOL Schema to be browsed; 4/ - COOL DB instance name; 5/ the data delimiter of the method (which in this example retrieves all the tags for that schema and instance).
The server is delivering the most important functionalities of COOL API itself, allowing a client (with the appropriate privileges) to also create tables in Oracle (adding a new folder) and to insert data in a given schema. The XML format is used for input and output data.

2.2. PL/SQL API

PL/SQL API has been developed mainly to overcome some limitations in the extraction of meta-data information from COOL; in particular the API allows to collect 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 IOVs folder content. The API is accessible from Oracle, and it is used in read-only mode.
A Java application deployed in a J2EE server has been also developed in order to access the PL/SQL package functions via a RESTful web service. An example of URL for this application is the following:

$$\texttt{http://}\underbrace{\texttt{<HostName>:8080}}_{1}\texttt{/}\underbrace{\texttt{JBRestCool/rest}}_{2}\texttt{/}\underbrace{\texttt{\{schema\}/\{db\}/...}}_{3}\texttt{/}\underbrace{\texttt{<KeyWord>}}_{4}$$

1/ contains server references – address and port to be served
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 from this application are XML and JSON.

2.3. COMA Database

This database has been developed to collect metadata for the management of the ATLAS Conditions Database. In a recent extension, a set of tables has been added in order to gather all relevant meta-data from COOL schemas, to provide a fast method to look at folders, global tags and tags associations for every schema at the same time. A complete description of the system is given in [5]. A typical 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

Conditions Tag Browser (CTB) is an application developed by the Georgian Team for the organization of a user interface framework providing a coherent access to the different sources related to conditions data storage. CTB is a Php/Javascript application based on 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 taking into account needs of experts who are coordinating conditions data activities in ATLAS, and want to have an efficient way to explore tags content at the level of the whole set of schemas and DB instances. CTB supports read-only account. Initially it was built on ASP and AJAX platforms. ASP configuration permits to avoid 8080 port security restrictions on server and works fine for all browsers – Firefox, Opera, Safari, and Explorer. However this system has low performances, while Javascript functions are executing on intermediate server which then send data to local PC (Fig.3).

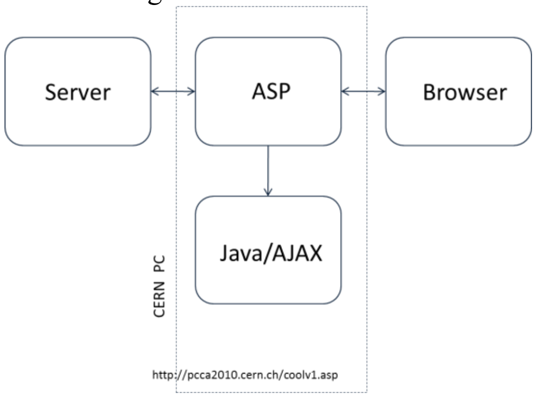

Fig.3 ASP based Browser

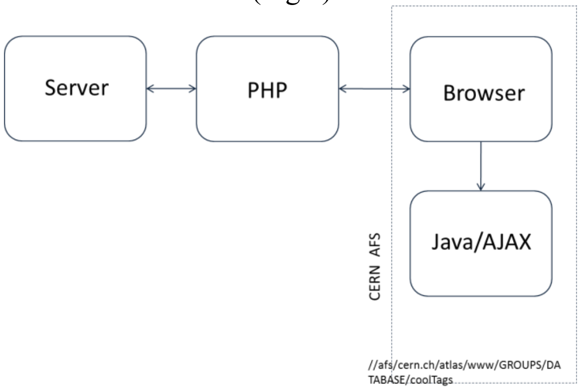

Fig.4 AJAX based Browser

AJAX/Javascript version (Fig.4) has high performance due to the execution of Javascript functions inside the client browser. As a result CTB architecture was built as a set of 5 Php modules executing on 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 [6]. According to COOL structure (Fig.1) four hierarchical levels of navigation have been separated for retrieving data from COOL:

1st level: for navigation through the schema, DB 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 to retrieve all tag related information from COOL according to input reference string. Depending on weather the reference string is selectable or editable, search functions divided into two categories. For selectable string several search functions are available for user:

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 folders
2. *Back_Trace* function returns list of all schemas, DB's and folders where selected tag is situated
3. *Diff* function enables comparison of two selected tags for the given instance and DB
4. *Compare* function doing the same but just inside of given folder
5. *Channel_Search* – backs list of channels with IOV for the selected leaf tag.

For editable type of reference string following functions are available:

1. *Global_Search* function enables to start navigation from the schema-DB-Folder set which is corresponds to the tag name entered in the reference string
2. *Folder_Search* function retrieves tag with entered name 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.

Every data source is chosen for an appropriate set of exploring methods, on the basis of the best performances that the system can rich [Fig.5].

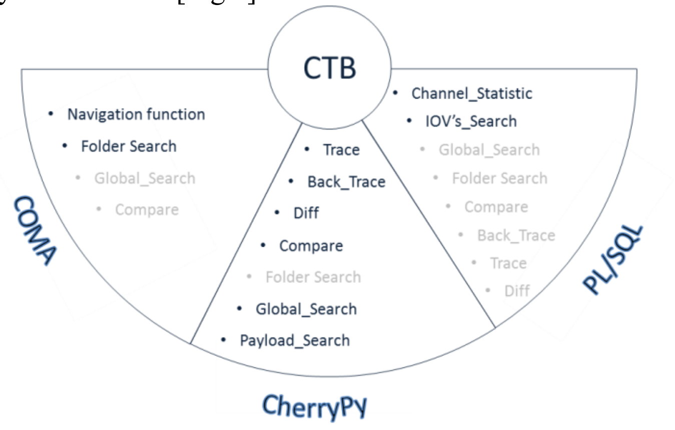

Fig.5 Distribution of functions by subsystems

COMA DB allows the highest performance when retrieving meta-data information from COOL, related to schemas, folders and tags content. So COMA was chosen for navigation functions and folder search function. Since CherryPyCool access COOL tables directly it gives more detailed information not present in COMA, like IOVs related information and also guarantee that the client retrieves the most up-to-date information, while COMA DB is synchronized daily. The main part of 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 Java script and enable functionalities for:

1. Bookmarking current navigation scenario into URL
2. Showing navigation path in status string
3. Displaying job execution time. So current performance of system can be estimated
4. Filtration of large tag list in folder
5. Displaying names of special Global tags – Current, CurrentES, Next and NextES.

References

[1] LCG Conditions Database Project. http://legapp.cern.ch/project/CondDB
[2] Shaun Roe "A RESTful Web Service Interface to the ATLAS COOL Database"/IOP Science, http://iopscience.iop.org/1742-6596/219/4/042021
[3] Andrea Formica "PL/SQL API for COOL Metadata", https://indico.cern.ch/ConferenceDisplay.py?ConfId=252773
[4] Fielding, R.T. "Architectural Styles and the Design of Network-based Software Architectures", Ph.D. dissertation to the University of California, Irvine, 2000
[5] Gallas E, Albrand S, Borodin M and Formica A "Utility of collecting metadata to manage a large scale conditions database in atlas"/ Tech. Rep. ATL-COM-SOFT-2013-084 CERN Geneva, 2013 https://cds.cern.ch/record/1602305
[6] Alexander Sharmazanashvili "New COOL Tag Browser"/ATLAS SW Week, CERN 15 July, 2010, https://indico.cern.ch/ConferenceDisplay.py?ConfId=214713