Time-critical Database Condition Data Handling in the CMS Experiment During the First Data Taking Period

Francesca Cavallari 1, Michele de Gruttola 2,3, Salvatore Di Guida 3, Giacomo Govi 4, Vincenzo Innocente 3, Andreas Pfeiffer 3, Antonio Pierro 5

1 INFN Sezione di Roma and Università degli Studi di Roma I, “La Sapienza”, Piazzale Aldo Moro, 2, I-00185 Roma, Italy
2 INFN Sezione di Napoli and Università degli Studi di Napoli “Federico II”, Complesso Universitario Monte Sant’Angelo, Edificio 6, I-80126 Napoli, Italy
3 CERN, CH-1211 Genève 23, Switzerland
4 Fermi National Accelerator Laboratory, Batavia, IL-60510, USA
5 INFN-Bari - Bari University, Via Orabona 4, Bari 70126, Italy

E-mail: salvatore.di.guida@cern.ch

Abstract. Automatic, synchronous and reliable population of the condition databases is critical for the correct operation of the online selection as well as of the offline reconstruction and analysis of data. In this complex infrastructure, monitoring and fast detection of errors is a very challenging task.

In this paper, we describe the CMS experiment system to process and populate the Condition Databases and make condition data promptly available both online for the high-level trigger and offline for reconstruction. The data are automatically collected using centralized jobs or are “dropped” by the users in dedicated services (offline and online drop-box), which synchronize them and take care of writing them into the online database. Then they are automatically streamed to the offline database, and thus are immediately accessible offline worldwide. The condition data are managed by different users using a wide range of applications. In normal operation the database monitor is used to provide simple timing information and the history of all transactions for all database accounts, and in the case of faults it is used to return simple error messages and more complete debugging information.

1. Introduction

In March 2010 the LHC started to accelerate and collide proton-proton beams at the center-of-mass energy of 7 TeV, and since that time the total luminosity delivered to the experiments has continuously grown up to \( \sim 50 \text{ pb}^{-1} \). In November 2010 the LHC accelerated also Pb-Pb ion beams, with a total luminosity of \( \sim 10 \mu \text{b}^{-1} \). In this new era for High Energy Physics, the large amount of data collected by the LHC experiments, which are produced by different sources, must be safely stored and consistently retrieved. A large part of this information is actually stored in databases, which have therefore become an essential component for correct

6 Present address: University of Florida, Gainesville, USA.
operation of the experiments’ data-taking and data-processing systems. This implies that their status and availability must be constantly monitored by robust and stable applications. After many years of data taking with cosmic rays that allowed for the development of a complex data handling infrastructure and a robust monitoring system, the CMS experiment[1] is now taking collision data and performing a large number of physics analyses with them.

Condition data change with time and require frequent lookups by time. A Relational Data Base Management System (RDBMS) is suitable for handling concurrent random access to a centralized storage. Therefore, CMS online and offline systems use RDBMS for condition data persistency. Oracle[2] Real Application Clusters (RAC) on Linux has been chosen as the main platform to deploy the database services for physics. Three different levels of services are deployed to support Oracle database applications: a development, a validation/pre-production and a production service. Applications at different stages of their development move from the first to the last depending on their maturity and needs. In addition, the validation/pre-production services are used also to test new Oracle server software releases and security update patches. The database architecture for CMS envisages two production RACs, and two validation RACs, one in the online private network, the other in the CERN network, and one development RAC, accessible from both the online and the CERN networks. Safe and stable data storage operation on the production database is performed using the Offline Drop-box service. The data stored using this infrastructure are made available to users for a variety of different applications, in particular the calibration of the various subdetector components and the reconstruction of collision events.

In this complex environment, the monitoring system must check not only the database infrastructure, but also all the applications performing database transactions. If errors occur, the system should detect them promptly, raise an alert to the relevant operators and developers, and, if possible, recover the infrastructure and put it in a safe state. In addition, one of the main challenges for CMS users is to monitor not only their own database transactions, but also the handling of the data that are relevant for the specific sub-detector or task. Therefore, different types of users need different data aggregation views depending on their role.

Figure 1. CMS condition Database Architecture.
2. Condition Database Architecture

A proper reconstruction of collision events makes use of “secondary data”, such as alignment and calibration constants, that are stored in ORACLE Databases and are accessed during the execution of the reconstruction application. These set of data, referred to as *Condition Data* that are stored in the *CMS Condition Databases*, can be classified in different groups: *configuration data*, needed to bring the detector into any running mode, *detector state data* that describe the current state of any sub-detector support system (gas pressures, high and low voltages, magnetic field, currents etc.), and *calibration constants* of single CMS sub-detector devices that are used in the offline event reconstruction (pedestals, offsets, noise, alignment constants).

The database architecture for the persistent storage of condition data in CMS experiment is the following (see Figure 1): on the online Oracle RAC in the experimental area network, two logical databases are deployed: a pure relational database, namely Online Master Database System (OMDS), and an Object Database (i.e. a database in which relational structures are correlated to objects), labelled as Offline Reconstruction Condition Database Online System (ORCON). On the other hand, the offline Oracle RAC in the CERN network hosts one logical database, namely Offline Reconstruction Condition Database Offline System (ORCOFF), which is the master database for the persistent storage of conditions, and is accessible in read-only mode. This distributed database network requires synchronization between the master databases at CMS Interaction Point 5 and the read-only database at CERN. In order to allow CMS data to flow through this distributed infrastructure, an asynchronous replication technology (Oracle Streams) is used to form a database backbone between online at Point 5 and offline site at CERN. An efficient condition data reading operation is assured from the *FrontTier* system. This system translates database queries into HTTP and caches the results in a proxy cache server called *squid*.

2.1. The Online Master Database

Online Master Database System (OMDS) is the online master database located at the experimental area at LHC Point 5 (P5) on the CMS online network. It stores the configuration of the detector and the detector state data produced by the sub-systems like slow control, electronics, data acquisition (DAQ) and trigger. The online database must allow for accessing individual, ideally self explaining data items: hence a pure relational access and manipulation structure has been chosen for OMDS, using ORACLE technology. The data size is expected to become very large (several TBs), and, since condition data will constantly flow into the database, the time needed to store these data in OMDS is a critical issue. In order to fulfill these requirements, each sub-detector has designed its own database schema, reflecting as far as possible the detector structure[6]. The total amount of condition data stored in the OMDS schemas is 1.7 TB: they were recorded in the 2008 and 2009 cosmic data challenges, and during the 2010 collision runs. During collision data taking, 150 GB of condition data per month are stored.

2.2. The Object Databases

The CMS database infrastructure envisages two object databases intended for condition data. In order to encapsulate conditions and deliver them as objects, an Object Relational Access layer (ORA) was designed inside the CMS software framework (CMSSW) on top of the RDBMS persistency backend. It has the capability of mapping objects to relational structures, and also the required relational data abstraction through the CORAL[5] framework providing a database abstraction layer. Moreover, it also implements some data compression algorithms, such as storing large sets of values through LOB data formats. ORCON is located at the CMS experimental area in the CMS online network on the same Oracle RAC as OMDS. The condition data can be both stored in and retrieved from it. ORCON stores all the condition
data required by the High Level Trigger (HLT) and for offline reconstruction. It also contains conditions needed offline for data quality indicator and for more detailed offline analysis. The data contained in it are written using the ORA layer in CMSSW by many applications following an API implemented by a library in the framework; since the ORA layer is also able to read data from the database and deliver them as objects to the framework, they are retrieved by the HLT programs as C++ objects for the offline software.

ORCOFF is located at CERN, and accessible in read-only mode. This allows prompt access to conditions data for the reconstruction jobs run at Tier0, and for alignment and calibration algorithms run at the CMS Central Analysis Facility (CAF). ORCON and ORCOFF are synchronized using Oracle streams, thus preventing each application from writing straight to ORCOFF, and providing consistent data for offline reconstruction. The conditions stored in object databases during both cosmic and collision data taking so far are 50 GB, and they increase with a rate of 1.5 GB per month.

2.3. SQLite

The CMS database infrastructure allows to store and retrieve data on both ORACLE and SQLite files, intended for testing and development. In the context of the CMS community, these SQLite files are simple files that the users can create locally on a personal computer and exchange with others or use to load conditions on the Oracle RDBMS’s. Thanks to the CORAL abstraction layer, handling data across different database back-ends is fully supported.

2.4. The Condition Data Flow

The general condition data flow can be described as follows: every subproject calculates and measures all the parameters needed to setup its hardware devices, mainly related to the detector, DAQ and trigger. Hence, configuration data are prepared for both hardware and software, and stored in the configuration database for different setups. Before the run starts, a given hardware and software setup is loaded. During data taking, the detector produces many kinds of conditions, which are stored in OMDS. Part of the data in OMDS are needed by HLT and offline analysis. PopCon (Population of Condition Objects) tool[3] is an application package fully integrated in the overall CMS framework intended to store, transfer and retrieve these data. PopCon encapsulates these data as C++ objects and adds meta-data information to them, so that they can be retrieved by both the HLT and the offline software. PopCon operates the transfer of the condition data from the OMDS and other online sources to ORCON; it encapsulates these data as C++ objects and adds meta-data information to them, so that they can be retrieved by both the HLT and the offline software. All the database transactions performed by PopCon generate logs that are stored in tables of a dedicated account in the CMS databases, so that every transaction is traceable to a single user on a single host. Finally, data are automatically streamed to the offline database (ORCOFF) and become accessible for offline applications as C++ objects. Calibration data will be written back to ORCON, using the Offline Drop-box service, and are propagated to ORCOFF using ORACLE streaming. In this way, collision data are processed using the offline condition data. The amount of collision data generated by the CMS detector is so huge that it is impossible to process all of them at one site, so the processing is distributed around the world. The over 100,000 computer cores at approximately 80 worldwide sites need access to data describing the alignments and calibrations of the detector at the time of the collisions. Very many cores need to access the same data, so it is well-suited for caching. The amount of the data can vary according to the analysis to be performed, but is on the order of 100 Megabytes per processing job. The caching subsystem that CMS uses for conditions data is

---

7 SQLite is a C library that provides a lightweight disk-based database that does not require a separate server process and allows accessing the database using a nonstandard variant of the SQL query language.
called **FroNTier**: it translates database queries into networking protocol requests (HTTP), looks up the results in a central database at CERN, and caches the results in an industry-standard HTTP proxy/caching server called **squid**.

### 2.5. The Condition Metadata

As discussed before, a subset of configuration data and detector state data, as well as calibration data, must be stored in the object database. All these data need a versioning and a time validity range for describing their time information, as meta-data. This interval of validity (IOV) has a start time and an end time. According to the use-case, time can be expressed in terms either of universal timestamps (mainly for detector state data), run number and luminosity section ID (usually for calibrations). When measuring physics observables coming from the collision data, the IOV provides the set of events for which condition data must be taken into account in the reconstruction algorithms. Some of the condition data may then be recalculated to take advantage of a wider statistics, thus getting a more precise picture of the detector performance, or by using a different software release: therefore, the condition data may exist in more than one version. Each version of the conditions can be identified by a tag, a unique human readable label. The matching with the data from the collision events is indeed possible via these meta-data: since the reconstruction algorithms for the analysis of a given run should retrieve a large set of consistent conditions corresponding to the same run, condition meta-data must be grouped through a collection of tags, called **global tag**. A global tag, therefore, identifies uniquely a consistent set of conditions to be used together in the reconstruction of collision events.

### 3. The Offline Drop-box

The **Offline Drop-box service** is implemented using web applications and supports the automatic export of calibration constants that are produced offline into the Offline Condition Database accounts, synchronizing them to HLT processing or to reconstruction applications that run at Tier0[8]. The population of condition databases with calibration constants must be fast and reliable, since these data are used both in the online context, for HLT algorithms, and in the offline context, for the reconstruction chain and for physics analysis. In order to guarantee HLT and/or offline reconstruction reproducibility, we must synchronize the interval of validity of the new calibration sets to the next run that will be processed by the HLT, or to the next run that will reconstructed by “Prompt Reconstruction” jobs at Tier0. This synchronization allows also to run HLT processing and prompt reconstruction with a version of calibration data that best represents the true performance of the detector, as required for the most accurate reconstruction of the recorded collision event data.

The design of the online cluster does not envisage data transfer from the outside world, neither from the CERN General Public Network (GPN), nor external networks. A strict security policy is applied to the administration of the online system such that only users deeply involved in the detector construction and commissioning can have access to it. Therefore, many physicists involved in the CMS offline calibration cannot populate the accounts in ORCON. As a result, data produced by offline analysis must be “pulled” into systems running on the internal CMS network by querying servers and repositories located inside the CERN GPN. Moreover, many users are not yet familiar with the Condition Database workflow. They do not know the software infrastructure in detail, and they are not aware of the network communication between the CMS internal network and the CERN GPN. Hence, the **Offline Drop-box service** simplifies tasks for end-users in case they need to upload calibrations resulting from an offline analysis into the condition databases. In addition, this new service is used to facilitate the automation of alignment and calibration workflows.

---

5 A luminosity section is a short section of a run where the luminosity can be considered as stable.
A web repository, hosted the SQLite files uploaded by the end-users, was deployed on a machine provided by the CERN IT department and located on the CERN GPN. A daemon was also installed on a machine in the CMS internal network that regularly checks the web repository for new files through an HTTP proxy. The daemon is also able to retrieve the number of the ongoing run at the CMS detector by querying the condition database account hosting run data, and can interrogate the smallest run number released for prompt reconstruction by querying the Tier0 Data Service (T0DAS), a web server providing information on the status of all jobs running at Tier0. The python script then checks the metadata values filled by the users and validates the data format, in particular the mapping between C++ objects’ data members and relational structures; if correct, it uploads the calibration constants in the condition database accounts and the transaction is logged and traced in the dedicated PopCon database account; this is accomplished using an application provided by the CMSSW framework and based on ORA libraries.

4. PopCon Monitoring Architecture and Features

PopCon monitoring (Populator of Condition Objects monitoring) is an open source web-based application implemented in python, which monitors all data transfers towards central database servers in a heterogeneous software environment. It makes use of standard web technologies, service interfaces, protocols and data models. In order to meet end-users requirements, PopCon monitoring provides also a group level data aggregation, based on use case models, provided by a recorded user interaction sequence.

The PopCon Monitoring architecture is based on a three tier infrastructure:

- The presentation tier (frontend server) supports user interaction and data presentation;
- The logic tier (backend server) handles information exchange between the database and the user interface;
- The data tier supports the access to the data stored in the database.

The three-tier architecture enables to encapsulate the functionality processing related to user interaction in an application separated from the client application. It also provides the possibility to program efficiently connection strategies (such as clients requests queuing, database access control). Finally all the code related to database connection can be totally separated from the client application. Moreover, a firewall protection assures security of both backend server and databases system on the CERN network, since the logic and data tiers are not accessible from outside CERN. The front-end part is therefore responsible for authenticating users, using tools such as Single Sign On (SSO).

The presentation tier, acting as a frontend server, translates the results yielded by the backend. The frontend application runs on top of an Apache server: its components are built using the jQuery libraries and are written in CSS and JavaScript. They allow to create alerts if a server or network device is not reachable or to prevent and correct possible mistakes made by the users, and to manipulate or get information in HTML, so that it can be displayed in a standard web browser. In details, when a user selects a particular method, the AJAX (Asynchronous JavaScript and XML) engine is called: it sends the request to the backend server using XML HTTP Request (XHR), parses the response, and updates the relevant portion of the HTML page. The logic tier processes user requests coming from the presentation tier, retrieves data from the data tier, makes logical decisions and evaluations, and performs calculations using them, and sends back the results encoded in JSON format to the frontend server. The backend

---

9 XML HTTP request (XHR) is an Application Programming Interface (API) available in web browser scripting languages such as JavaScript. It is used to send HTTP or HTTPS requests directly to a web server and load the server response data directly back into the script.
application server is implemented in python using the CherryPy\textsuperscript{10} module. It communicates with the frontend Apache server through the Web Server Gateway Interface (WSGI) module. The PopCon\textsuperscript{10} monitoring information stored in the database are retrieved by the backend application using python modules which conform to DB API interface specification, such as cx\textunderscore Oracle and SQLAlchemy. This architecture ensures at the same time scalability, e.g. allowing to run on many load-balanced servers, security, such as avoiding direct database access and respecting the “rule of least privilege”, and good performance, such as fast web page rendering.

5. Concluding Remarks

A complex system was set up by the CMS experiment in order to handle consistently and reliably condition data produced by the detector, which are needed by HLT, DQM and offline reconstruction. It relies on Oracle RDBMS as a persistency backend for data storage, on a pure relational data access for the online system, and on an object relational access layer in the offline system, which delivers data as objects to the offline software framework. The object database stores only a subset of the conditions produced in the online system, and exploits data compression techniques in order to reduce further the data size. The PopCon mini-framework, fully integrated in the CMS software framework CMSSW, allows to retrieve online data and store them in the object database very efficiently: this is especially important for time critical data, such as trigger and run information, which must be used by online processing. PopCon has some additional features, such as transaction logging, that helps identifying faulty situations. Besides this, PopCon allows to automate the jobs storing conditions changing frequently in time. The offline drop-box allows to store safely and reliably calibrations coming from offline analysis. The PopCon web based monitoring allows to check the status of the CMS database infrastructure and spot hardware and software faults very quickly.

The infrastructure and workflow for CMS conditions have been running smoothly since November 2009, both for 900 GeV and 7 TeV data taking. Up to now, 50 GB of conditions are stored by the tools described above in the object database, with an increase rate of 1.5 GB per month during data taking, and are accessed by reconstruction applications.

References

[1] The CMS Collaboration, CMS Physics TDR, Volume I: Detector Performance and Software. Technical Report CERN-LHCC-2006-001; CMSSTD-008-1, CERN, Geneva, 2006.
[2] Loney, K. Oracle Database 11g: the Complete Reference, Oracle, Release 1 (11.1), 2008.
[3] De Gruttola, M. et al., Persistent Storage of non-event Data in the CMS Databases. JINST, 5 P04003, 2010. doi: 10.1088/1748-0221/5/04/P04003. arXiv:1001.1674 [physics.ins-det].
[4] Dave Dykstra and Lee Lueking, Greatly improved cache update times for conditions data with FroNTier/Squid, International Conference on Computing in High-Energy Physics Conference (CHEP 2009), Distributed Processing and Analysis track, 23–27 March 2009, Prague, Czech Republic.
[5] Papadopoulos, I. et al, CORAL: a Common Relational Abstraction Layer, 9th ICATPP Conference on Astroparticle, Particle, SpacePhysics, Detectors and Medical Physics Applications, 17–21 October 2005, Villa Erba, Como, Italy.
[6] Janulis, M. et al., CMS Online Database Experience with First Data, International Conference on Computing in High-Energy Physics Conference (CHEP 2010), Software Engineering, Data Stores, and Databases track, 18–22 October 2010, Taipei, Taiwan.
[7] Brun, R. and Rademakers, F., ROOT - An Object Oriented Data Analysis Framework, Proceedings AIHENP’96 Workshop, Lausanne, Sep. 1996, Nucl. Inst. & Meth. in Phys. Res. A 389 (1997) 81–86. See also http://root.cern.ch/.
[8] Hufnagel, D. et al, The Architecture and Operation of the CMS Tier-0, emphInternational Conference on Computing in High-Energy Physics Conference (CHEP 2010), Software Engineering, Data Stores, and Databases track, 18–22 October 2010, Taipei, Taiwan.

\textsuperscript{10}CherryPy: a pythonic, object-oriented HTTP framework. http://www.cherrypy.org/.