

UNIVERSITÀ DEGLI STUDI DI BARI ALDO MORO

Dipartimento Interateneo di Fisica

DOTTORATO DI RICERCA IN FISICA

CICLO XXIII

Settore Scientifico Disciplinare FIS/01

**DEVELOPMENT OF TOOLS FOR HANDLING AND
MONITORING THE CONDITION DATA FOR THE
CMS EXPERIMENT**

Dottorando: Dott. Giuseppe Antonio Pierro

Coordinatore: Ch.mo Prof. Francesco Romano

Supervisori: Ch.mo Prof. Giorgio Pietro Maggi
Dott. Vincenzo Innocente
Dott. Domenico Giordano

ESAME FINALE 2011

Abstract

The Standard Model is a relativistic quantum field theory that encapsulates current knowledge of elementary particles and their interactions. The predictions of this theory coincide with observations in experiments with an astonishing precision. Still, the Standard Model falls short of being a complete theory of particle physics. A fair amount of theoretical and experimental research attempts to extend the Standard Model into a theory of everything, all foreseeing new physics at the TeV scale. Inadequacies of the Standard Model that motivate such research include: it does not incorporate gravity, it does not contain a viable candidate for dark matter, it requires a large number of constants whose values are unrelated and arbitrary and it gives rise to the hierarchy problem, namely why the electro-weak scale and Planck scale are so disparate. Furthermore, the mechanism to give mass to particles is introduced ad-hoc and requires the existence of a scalar boson, the Higgs boson, which is currently the only unobserved Standard Model particle.

Numerous extensions, revisions, replacements, and reorganizations of the Standard Model exist in attempt to correct for these and other issues. Unfortunately, there is, at present, no experimental evidence supporting one theory over the others. The Large Hadron Collider (LHC), the world's largest and highest-energy particle accelerator, was built with the intention of testing these various hypotheses and searching for the hypothesized Higgs boson in the entire allowed mass range.

The Compact Muon Solenoid (CMS) detector is one of the two multipurpose experiments at the LHC: it is a sophisticated and massive system: 7 different detector technologies, a high number of front-end electronic channels ($\sim 10^8$), an input rate of 10^9 interactions per second, a trigger able to reduce the frequency by a factor 10^5 , and

online computer farms with a storage capability rates of $\sim 10^2$ MB/s. CMS is foreseen to take data, with high efficiency, for over 15 years. It is, therefore, necessary to assure a perfect and stable behavior of each of its components.

To exploit its full physics potential, the control and the management of a large amount of calibration and alignment data of the various components (crystals, drift tubes, silicon devices) and their attached electronics, is absolutely necessary. This enormous amount of data needed to configure detector and being produced by it is stored in a database system, which has become an essential service for data taking like electricity or cooling. To handle this task, the *CMS Core Database Group* was formed by the *CMS collaboration*.

In this thesis, we describe the needs for the operation of a Database Management System (DBMS) to store and retrieve non-physics data, i.e. those data used for support, configuration, maintenance and monitoring of the detector. The quality and the amount of this kind of data is, by nature, mutable in time. The database services for them must be designed to be flexible enough to accommodate changes in rates and types of data, but overall the database has to be efficient. Data retrieval must not add a consistent amount of time to computing processing (like reconstruction and simulation) which has to be as fast as possible. Given the above requirements, in this doctoral thesis, we will present the contribution given to the development of tools and services for the *CMS condition database* using the CMS Software framework (CMSSW).

The thesis is structured as follows. Chapter 1 describes the LHC and gives a comprehensive description of the CMS detector, accounting for details of the sub-detectors. A special attention is paid to each sub-detector performance because the issues related to calibration and alignment constants are closely related to each of them. Besides, experience with CMS detector and computing starting from the detector commissioning phase to the physics are given in order to show how difficult was to deploy a high energy experiment for which no commercial device meets the requirements.

In Chapter 2, the computation of alignment and calibration constants as well as the computing resources used for these purpose are described. This chapter explains the complex environment which I had to take into account before starting to plan and carry out the implementation of tools for the certification of data, relevant for physics

analyses, and the monitoring of the calibration and alignment activity.

In Chapter 3, the data model and base technology choices for the Condition Data of CMS Experiment are described. This survey will give me the opportunity to describe the CMS Database architecture as well as the key elements of my work.

In Chapter 4, the tools for controlling the calibration and alignment activity of the CMS experiment and to perform the population work-flow of CMS database are described. The complexity of these services and their relevance for the physics analysis drive a need for continuous monitoring of these activities, because even a short failure could result in a non negligible loss of the physics analysis data.

In Chapter 5, we shall detail about operational challenges and technical solutions employed to develop and enhance the tool presented in the Chapter 4. Results and conclusions are given in Chapter 6.

Contents

1	The LHC and the CMS experiment from the detector commissioning phase to the physics collisions	1
1.1	Introduction	1
1.2	The Large Hadron Collider	2
1.2.1	Design and Operation	3
1.3	The CMS Detector	6
1.3.1	The CMS Trigger System	9
1.3.2	Silicon Strip Tracker	11
1.3.3	Calorimeter system	13
1.3.4	The Muon System	16
1.4	CMS computing campaigns from 2006 to 2010	17
1.4.1	Magnet Test and Cosmic Challenge	18
1.4.2	Cosmic Run At Four Tesla 2008	19
1.4.3	Computing campaigns	20
1.4.4	Proton-Proton collisions data taking	22
1.4.5	Pb-Pb collisions data taking at 2.76 TeV	25
1.5	Conclusion	25
2	Computation of Alignment and Calibration Constants in the CMS Experiment	27
2.1	Introduction	27
2.2	CMS CERN Analysis Facility and Tier-0	28
2.2.1	The CMS CERN Analysis Facility	28

2.2.2	Tier-0	30
2.3	Calibration and alignment work-flow	32
2.4	Transfer of data streams for calibration and alignment from CMS to the Tier-0	34
2.4.1	Offline processing at the Tier-0: production of AlCaReco skims	35
2.4.2	The prompt calibration loop	36
2.5	Performance and experience	37
2.5.1	LHC data-taking	37
2.6	Conclusion	40
3	Persistent storage of condition data in the CMS experiment	43
3.1	Introduction	43
3.2	The Condition Data description	44
3.2.1	Introduction	44
3.2.2	Reconstruction chain for the SST	47
3.2.3	Metadata of CMS conditions data	48
3.3	The CMS database	51
3.3.1	Networking Infrastructure	52
3.3.2	The CMS database - Design Requirements	53
3.3.3	Online Master Data Storage	54
3.3.4	Off-line Database	55
3.3.5	FroNTier	58
3.3.6	Service Architecture	58
3.3.7	Service levels and release cycles	59
3.3.8	SQLite	60
3.4	Data sources and consumers	62
3.4.1	Data sources	62
3.4.2	Data consumers	64
3.5	Condition Database Data model	66
3.5.1	Introduction	66
3.5.2	Persistency framework	66

3.5.3	POOL Object Relational Database Access: POOL-ORA . . .	67
3.5.4	Data model and base technology choices for the CMS condition data	68
3.5.5	Core Software and Services	69
3.6	Performance	72
3.6.1	Introduction	72
3.6.2	Performance Considerations	73
3.6.3	Data Archiving strategy	74
3.6.4	ECAL use case - Database performance optimization	75
3.6.5	The Silicon Strip Tracker Detector use case	76
3.7	Conclusion	78
4	Tools for handling and monitoring the Condition Data	79
4.1	Introduction	79
4.2	Tool for handling the Condition Database - PopCon	80
4.2.1	Introduction	80
4.2.2	PopCon Tool	82
4.3	Tool for handling Condition Database: off-line DropBox service . . .	84
4.3.1	Introduction	84
4.3.2	Off-line DropBox service implementation	86
4.4	Population of the conditions database monitoring	88
4.4.1	Introduction	88
4.4.2	PopCon monitoring architecture and features	88
4.4.3	PopCon monitoring from the different users' perspectives . .	94
4.4.4	Results	96
4.5	Payload Inspector and Global TAG Browser	96
4.5.1	Introduction	96
4.5.2	Payload Inspector	97
4.6	Conclusion	103

5	Operational Challenges and Technical Solutions for CMS Web Condition	105
	database	105
5.1	Introduction	105
5.2	Why not adopting a commercial software	106
5.2.1	Introduction	106
5.3	Desktop vs Web Application	108
5.3.1	Advantages and disadvantages of different applications	108
5.4	n-Tier Architecture	109
5.4.1	Introduction	109
5.4.2	Two-Tiers Architecture	110
5.4.3	Three-Tiers Architecture	111
5.4.4	Multi-Tiers Architecture	113
5.4.5	CMS condition DB web application architecture	114
5.5	HTML version 5 vs HTML version 4	120
5.5.1	Web Workers	120
5.5.2	The Web Sockets API	123
5.5.3	Web Security	125
5.6	Open Issues	125
5.6.1	Display Data Updates in Real-Time	125
5.7	Results and conclusions	126
6	Conclusion	129
A	Riassunto	133
A.1	Introduzione	133
A.2	Strumenti per l'archiviazione e il monitoraggio delle costanti di calibrazione e allineamento	134
A.2.1	Ambiente di archiviazione dei dati di condizione	134
A.2.2	PopCon	135
A.2.3	DropBox offline	136
A.2.4	PopCon monitoring	136
A.2.5	Payload Inspector	137

A.3 Conclusion	138
B Acronyms and Abbreviations	139
References	146
Acknowledgments	i

Chapter 1

The LHC and the CMS experiment from the detector commissioning phase to the physics collisions

1.1 Introduction

The Large Hadron Collider (LHC) [1] and the Compact Muon Solenoid (CMS) experiment [2] constitute an exceptional challenge both for physics, engineering, computing and software issue, with unique operational issues on account like the amount of energy stored in the magnets and the beams and the huge amount of data to be collected, managed, and monitored. In particular ensuring functionality of the computing and software components is a significant challenge that can only be achieved pushing the limits of the hardware and software components.

In order to setup the CMS detector and to reconstruct physics events, physicists use a large amount of non-event data, also called condition data that describe the behaviour of the various CMS detector components. The work of this thesis concerns the development of software components to control and handle these condition data which are stored in different databases.

Firstly, this chapter describes the LHC particle accelerator and the CMS Experiment. Detailed knowledge of the CMS detector is important in order to have a full

understanding of the experiment needs. Then, the CMS detector commissioning and the computing campaigns will be described. The LHC operation since 2006 till today had a huge impact on the CMS operation's schedule: in particular, the incidents during LHC commissioning can help to understand the challenges CMS is facing. Besides, a detailed overview of the CMS physics results, both for cosmic rays and for collision events, will be given. These outstanding results were reached also thanks to the work presented in this thesis.

1.2 The Large Hadron Collider

The LHC, at the CERN laboratories outside Geneva (Switzerland), is the largest and highest-energy particle accelerator and collider ever built. Designed to collide opposing particle beams of protons (p) at a center of mass energy of 14 TeV and a luminosity¹ of $L = 10^{34} \text{cm}^{-2} \text{s}^{-1}$, it is located in a 27 km circular tunnel about 100 meters beneath the French-Swiss border. In addition, during some dedicated runs, the LHC collides heavy ion ($^{208}\text{Pb}^{82+}$) beams at energies of 2.76 TeV/nucleon, yielding a total center of mass energy of 1.148 PeV and a nominal luminosity of $L = 10^{27} \text{cm}^{-2} \text{s}^{-1}$.

Four main experiments take place at each LHC interaction point: two with general purpose detectors, ATLAS [3] and CMS, and two with dedicated detectors, ALICE [4] and LHC-b [5] which will study heavy ion physics and B-physics respectively. Figure 1.1 shows the four experimental sites along the LHC ring; the CMS experiment is highlighted.

¹Luminosity is the number of collisions per unit time and cross-sectional area of the beams. It depends only on collider parameters. For circular accelerators, colliding bunches of n_1 and n_2 particles at a frequency f , the luminosity reads:

$$L = f \frac{n_1 n_2}{4\pi\sigma_x\sigma_y} \quad (1.1)$$

where σ_x and σ_y characterize the Gaussian transverse beam profiles in the horizontal and vertical directions.

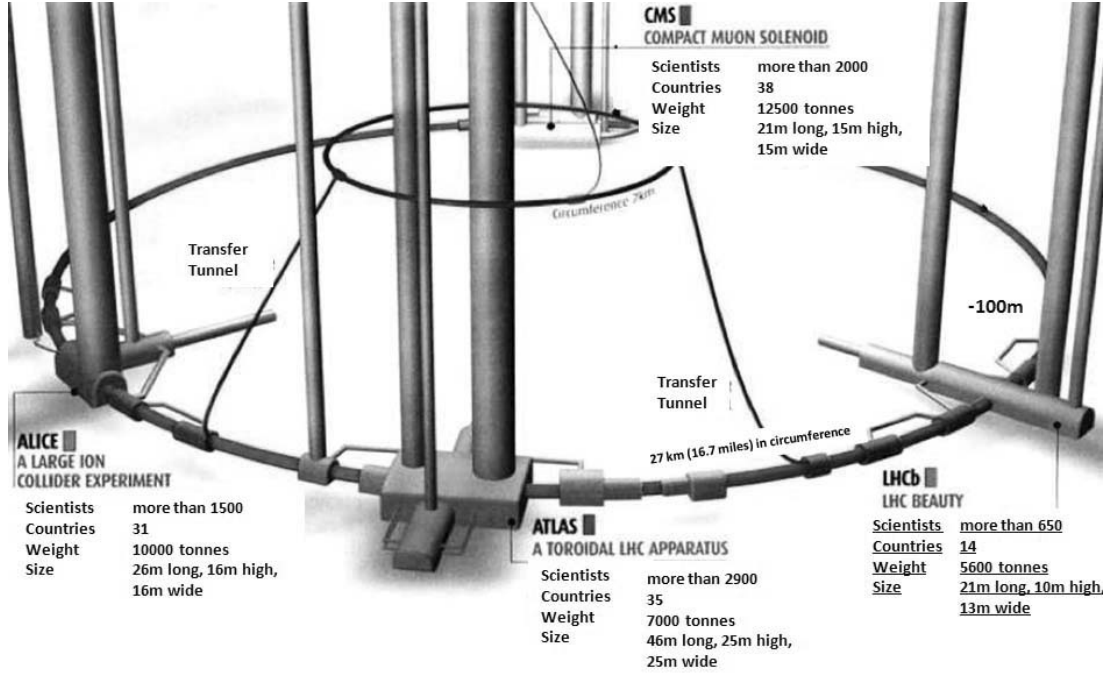


Figure 1.1: Schematic view of the LHC accelerator including the location of the four experiments.

1.2.1 Design and Operation

The design and measured parameters for LHC are given in Table 1.1 for both proton and lead beams.

		design values		2009 collisions runs		2010 collisions runs	
		pp	HI	pp	HI	pp	HI
Energy per nucleon (TeV)	E	7	2.76	1.18	-	3.5	1.38
Dipole field at 7 TeV	B	8.33	8.33	-	-	-	-
Luminosity ($cm^{-2}s^{-1}$)	L	10^{34}	-	-	-	$2.07 * 10^{32}$	$2.88 * 10^{25}$
Bunch separation (ns)		25	100	25	-	25	100
No. of bunches	k_B	2808	592	-	-	368	121
No. particles per bunch	N_p	$1.5 * 10^{11}$	$7 * 10^7$	-	-	10^{11}	-

Table 1.1: Operation parameters of the LHC machine.

The CERN accelerator complex is a network of particle accelerators that includes: the *Linear Accelerator* (LINAC2), the *Proton Synchrotron Booster* (PSB), the *Proton Synchrotron* (PS) and the *Super Proton Synchrotron* (SPS). All these particle accelerators prepare beams before their injection in the LHC, as shown in Figure 1.2.

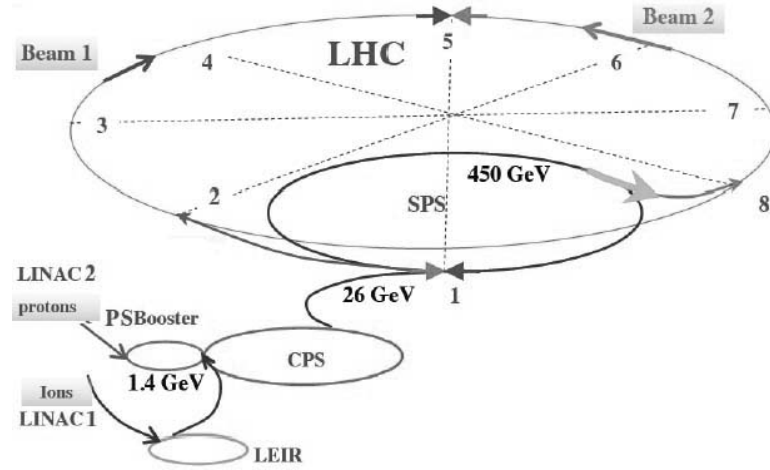


Figure 1.2: Schema of the CERN accelerator complex.

The protons produced at 92 KeV are collected as an input to LINAC2, which increases their energy up to 50 MeV. In the next step, the PSB increases beam energy to 1.4 GeV. Protons are then accelerated up to 25 GeV by the PS, which builds up the proton bunches with a $\sim 25\text{ ns}$ separation and less than 4 ns time extension. Subsequently the SPS pushes particle energy up to 450 GeV and injects the beam into the LHC pipes. Inside the LHC accelerator, particles circulate in opposite directions in two separate beam pipes. 1232 superconducting dipoles and more than 2500 other magnets guide and squeeze the beams to a diameter of $\sim 16\mu\text{m}$. By design, bunches of 10^{11} protons will collide every 25 ns with an instantaneous Luminosity $L = 10^{34}\text{ cm}^{-2}\text{ s}^{-1}$ and a center of mass energy of 14 TeV.

Lead ions, instead, start from a source of vaporized lead and enter LINAC3 before being collected and accelerated in the Low Energy Ion Ring (LEIR). They then follow the same route to maximum acceleration as the protons.

In December 2009, the LHC started to produce the first *proton-proton* (p-p) collisions. The energy of the beams was fixed to 450 GeV, yielding to a center of mass

energy of $\sqrt{s} = 0.9$ TeV. The Luminosity for these first runs was kept several orders of magnitude below the nominal value $10^{34} \text{cm}^{-2} \text{s}^{-1}$. The energy of the beams was increased along December until a center of mass energy of $\sqrt{s} = 2.2$ TeV was reached, making LHC the most powerful hadron accelerator ever built. The four particle detectors installed at the LHC were able to record the products of the collisions. Figure 1.3 shows the first p-p collision recorded by the CMS detector.

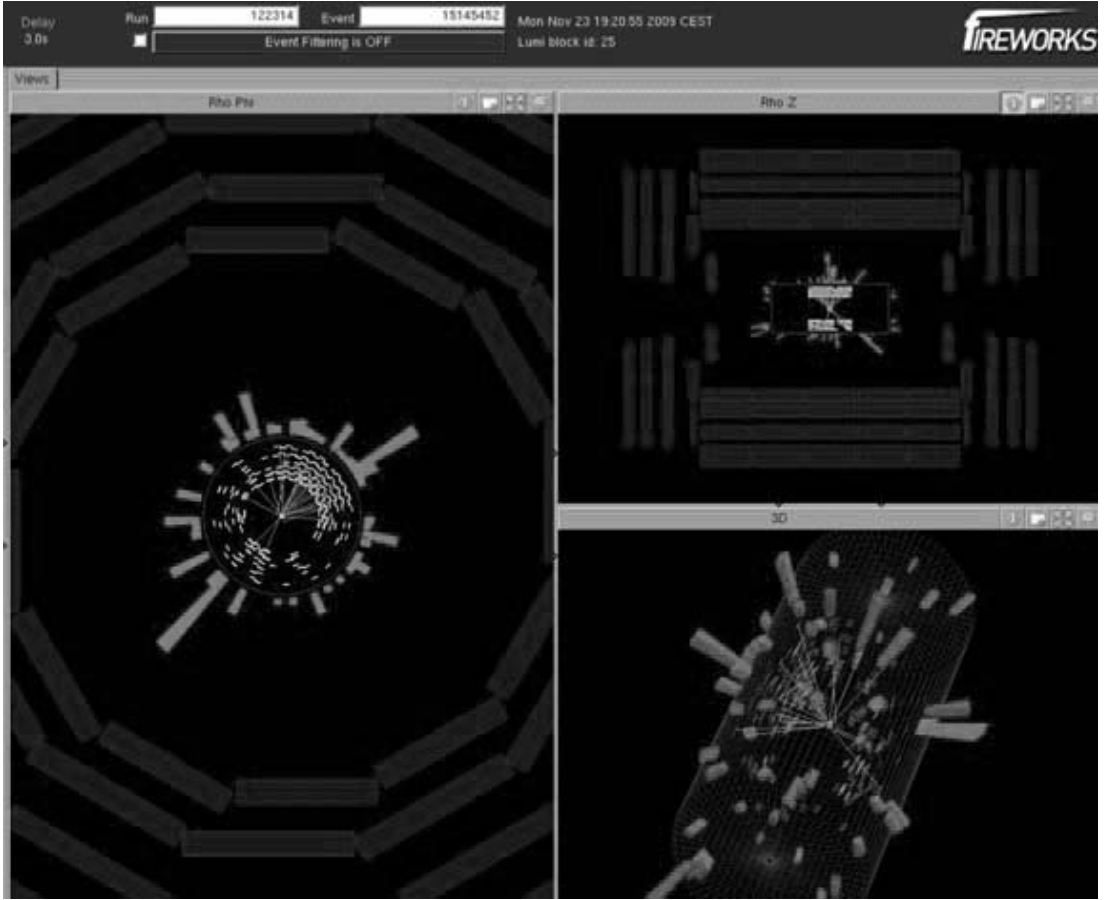


Figure 1.3: Minimum bias event from the first p-p collision recorded in CMS. The points represent hits in the central silicon tracker, while the bars represent the energy deposited in the hadronic and electromagnetic calorimeters respectively.

During the last data-taking at 7 TeV between February and November 2010, an integrated Luminosity of almost 45 pb^{-1} has been delivered (see Figure 1.4).

Table 1.2 shows the timeline of the LHC Operation.

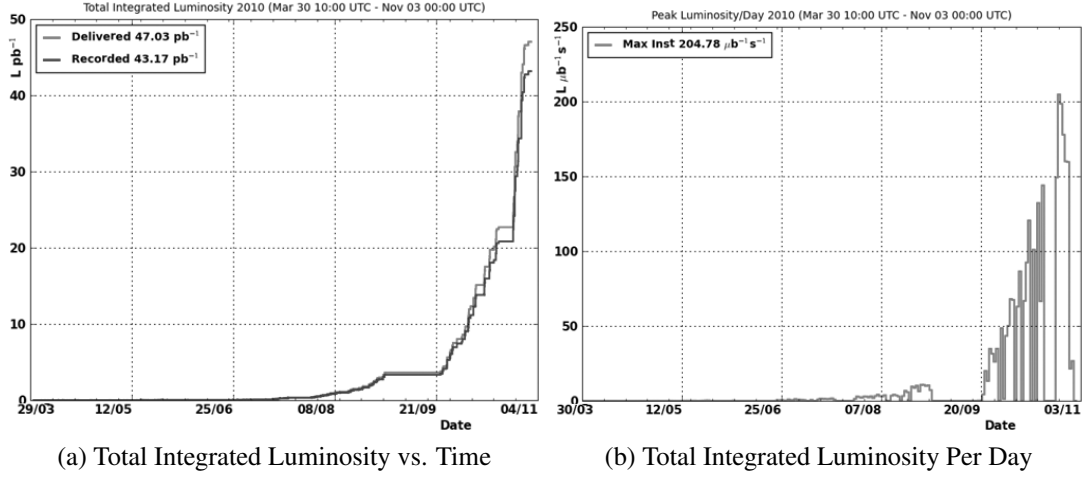


Figure 1.4: 1.4a (left): Integrated luminosity versus time delivered to (lighter color), and recorded by CMS (darker color) during stable beams at 7 TeV centre of mass energy. 1.4b (right): Maximum Instantaneous luminosity per day delivered to (red) CMS during stable beams at 7 TeV centre of mass energy.

Date	Bunches/beam	Colliding bunches	Luminosity ($\text{cm}^{-2} \text{s}^{-1}$)
15 th November	121	113	$2.88 * 10^{25}$
9 th November	17	16	$3.5 * 10^{24}$
4 th November	LHC switched to heavy ions (fully stripped lead)		
25 th October	368	348	$2.07 * 10^{32}$
16 th October	312	295	$1.35 * 10^{32}$
14 th October	248	233	$1 * 10^{32}$
8 th October	248	233	$8.8 * 10^{31}$
4 th October	204	186	$7 * 10^{31}$
29 th September	152	140	$5 * 10^{31}$
25 th September	104	93	$3.5 * 10^{31}$
23 rd September	56	47	$2 * 10^{31}$
22 nd September	24	16	$4.6 * 10^{30}$
1 st - 22 nd September	Bunch train commissioning		
29 th August	50	35	$1 * 10^{31}$

Table 1.2: LHC Operational timeline between August and November 2010

1.3 The CMS Detector

The Compact Muon Solenoid (CMS), is one of the two “general-purpose” detectors at the Large Hadron Collider (LHC). It is located at the Interaction Point 5 of the LHC

(IP5), exactly opposite to ATLAS detector, at Point 1. The main distinguishing feature of CMS is its large superconducting solenoid magnet, which creates a strong field of 3.8 Tesla and allows CMS to be more “compact” (smaller in size) than ATLAS while reaching similar momentum resolution. Figure 1.5 shows a schematic view of the CMS Detector. Close to the interaction point is an all silicon Tracker, that is surrounded by the Electromagnetic Calorimeter (ECAL) and the Hadronic Calorimeter (HCAL). All these systems are contained inside the superconducting solenoid. The detectors of the muon system: Drift Tubes (DT), Resistive Plate Chambers (RPC) and Cathode Strip Chambers (CSC) are embedded in the iron return yoke of the solenoid.

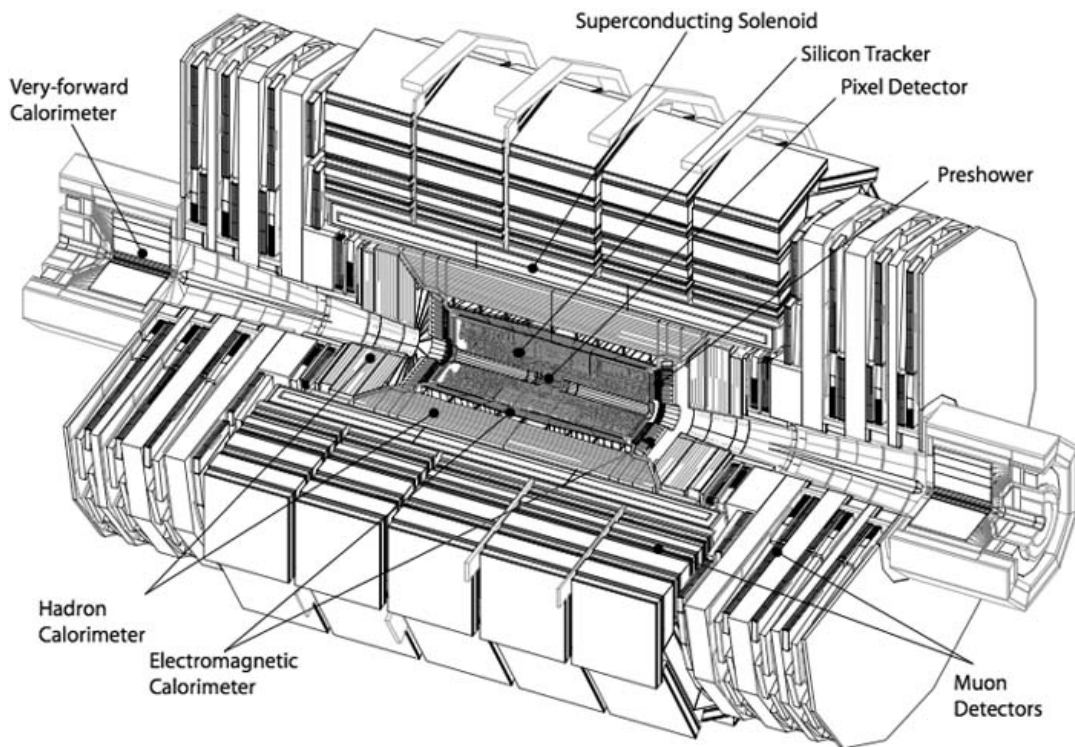


Figure 1.5: Schematic view of the CMS Detector.

After a short initial period of LHC beams towards the end of 2009, the commissioning of the CMS detector has been taking place mostly in 2010.

In the spring of 2010, LHC started running at low luminosity, with only one pair of colliding bunches. The nominal bunch size of about 10^{11} protons per bunch was

soon reached and the increase in beam intensity and also in instantaneous luminosity was then mostly due to the increase in the number of bunches. This is reflected in the steep rise in integrated luminosity, which was about one quarter of an inverse picobarn (pb^{-1}) when first physics results were reported at the International Conference on High-Energy Physics (ICHEP) in Paris in July 2010 and reached almost $45 pb^{-1}$ by the time of this thesis. when LHC already supplied over 300 bunches colliding in CMS. During the last collision runs, CMS was running the detector without any major problem. Over 2010, all subsystems of CMS have been functional to at least 98%, as shown in Figure 1.6.

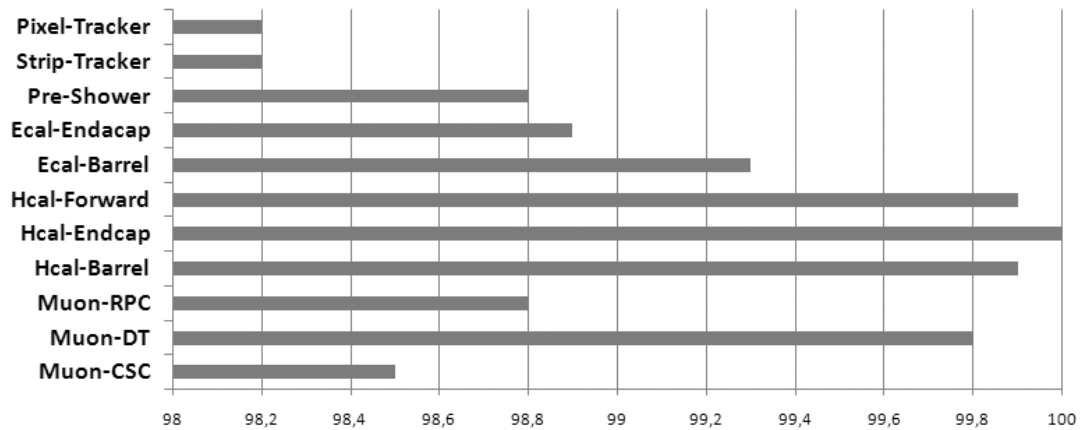


Figure 1.6: Percentage of availability of the various sub-detectors of CMS during the 2010 run.

A more detailed description of the CMS apparatus can be found in [6]. The physics potential of the CMS experiment is described in [2].

A large part of the commissioning activities of the individual sub-detector systems was also performed during the construction and assembly in various test beams, integration facilities and in situ in the surface hall and the experimental cavern. The following section concentrates on CMS-wide global commissioning activities outlined in the Section 1.4.

1.3.1 The CMS Trigger System

The triggering system for CMS had to be adapted to the steady increase in LHC luminosity: while at the beginning of operation in 2010, it was possible to trigger on any passing pair of colliding bunches (detected by beam pickup detectors: “zero-bias trigger”) or on any activity in the detector (detected by beam scintillation detectors (BSC): “minimum-bias trigger”), over time triggering had to become more and more selective, by triggering mostly on “physics objects” (such as muons, electrons, jets etc.) and their coincidences.

CMS uses a two-stage trigger. The Level 1 is implemented in electronics and reduces the data rate to a maximum of 100 kHz. Trigger signals from the electromagnetic and hadron calorimeters are collected by the Global Calorimeter Trigger, those from the various muon detectors are collected in the Global Muon Trigger. Trigger requests from these two systems and from a few additional sources are fed into the Level-1 Global Trigger, which takes the Level-1 decision based on pre-defined criteria programmed in firmware and distributes the readout request (“Level-1 Accept”) to all sub-detectors.

In case of a Level-1 accept decision, all parts of the CMS detector are read out in full resolution (also sub-detectors that do not participate in the Level-1 decision, such as the silicon tracker) and the full data are made available to the computer farm of the High-Level Trigger (HLT) which carries out the complete reconstruction of all events and selects those which are sent to the CERN computing center on the Meyrin site for permanent storage at the Computing Grid’s Tier-0. The CMS HLT, differently from ATLAS, does not use “regions of interest”, The HLT output stream size was originally planned to be 100 Hz. Due to the good performance of the computing system, CMS has been taking data at a stable output rate of several times this rate.

As the Level-1 trigger uses data of reduced resolution, the parameters on which its decision is based (such as “transverse energy”) will not match exactly the values obtained during final reconstruction, and in order to reach clean and efficient data samples the off-line analysis thresholds must be chosen somewhat higher than the on-line trigger thresholds. Figure 1.7 shows the “turn-on curves” of Level-1 trigger

efficiency against the transverse energy reconstructed off line for various calorimeter trigger objects.

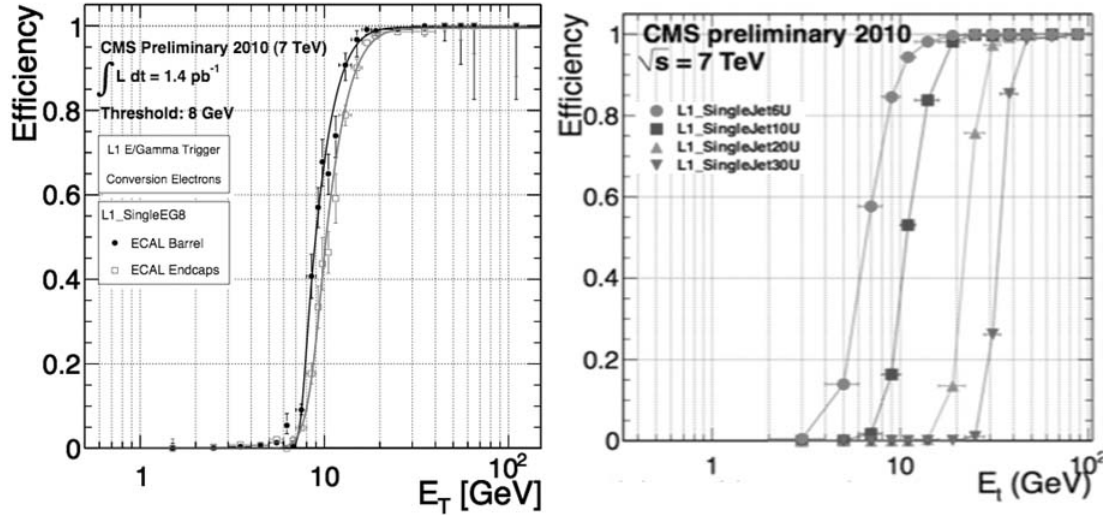


Figure 1.7: Turn-on curves of the Level-1 trigger efficiency plotted against the reconstructed transverse energy for the electron/gamma trigger with 8 GeV nominal threshold (left) and for jet triggers with several different thresholds (right) [7].

Obviously, taking the physics data is only part of the experiment's job and it is no less important to make sure that these data are of good quality. After many years of detailed Monte-Carlo studies it is important to see that the CMS detector is really understood and that resolutions and other features as obtained from the analysis of accelerator data match the values derived from simulations. Figure 1.8 shows the primary vertex resolution obtained by the CMS detector versus the number of reconstructed tracks along the LHC beam direction (i.e. in Z, left) and perpendicular to it in the horizontal plane (in X, right) for several different ranges in transverse momentum. Data (full symbols) and Monte Carlo simulation (open symbols) show good agreement.

The known members of the “particle zoo” up to the W and Z boson mass have been clearly identified in the data (see Figure 1.9) and work is ongoing to obtain a clear top-quark signal [8]. The CMS detector has been calibrated against known physics channels and is delivering first results on new effects never seen before.

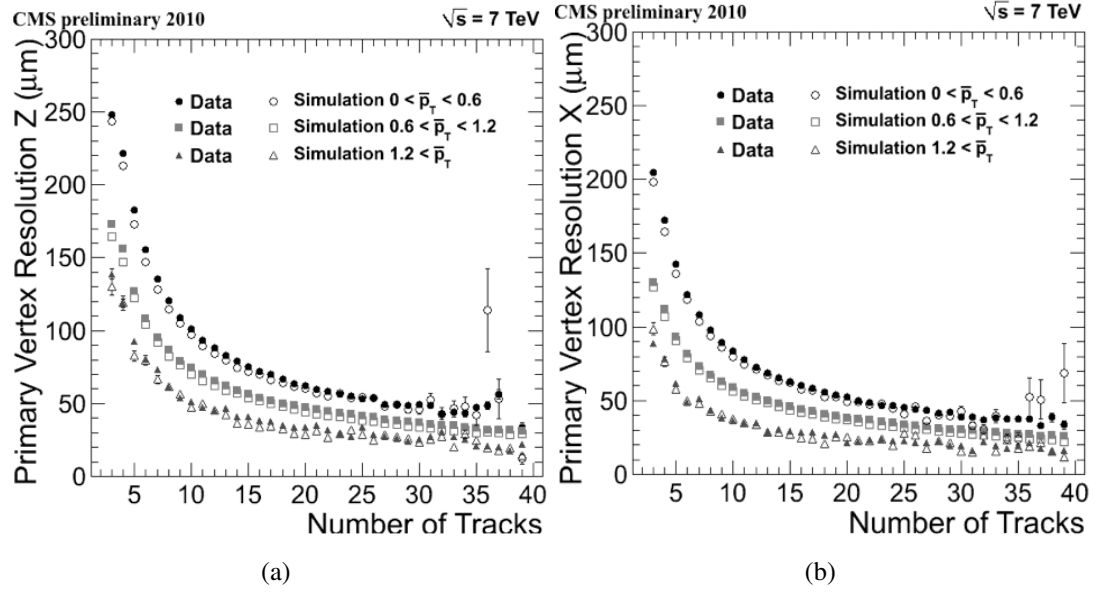


Figure 1.8: Primary vertex resolution versus the number of reconstructed tracks along the LHC (Z 1.8a, X 1.8b).

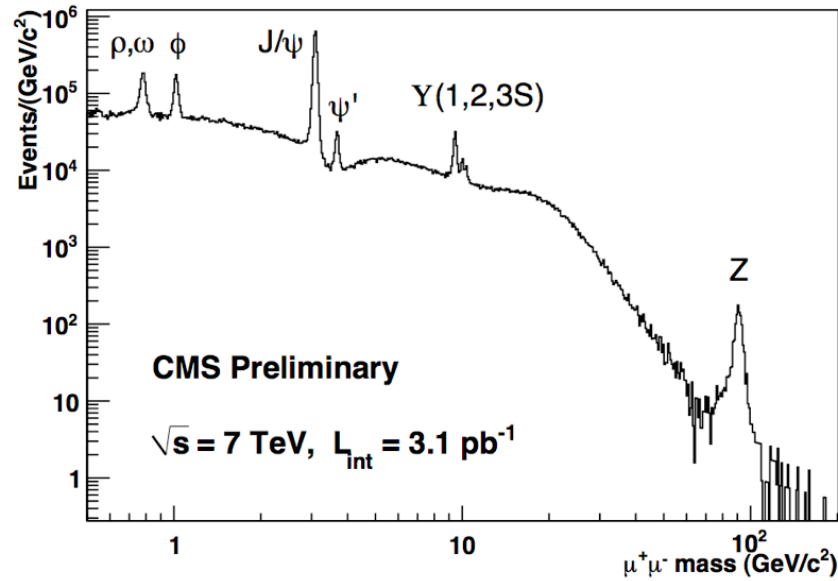


Figure 1.9: Invariant-mass spectrum of $\mu + \mu^-$ events seen in the CMS detector.

1.3.2 Silicon Strip Tracker

The CMS Silicon Strip Tracker (SST) is the largest silicon detector ever built with almost 10 million readout channels and an active area of close to 200m^2 . Its more than

15,000 individual silicon modules are powered by almost 1000 power supply modules and produce more than 60 kW of power while operating at low temperatures [9]. The SST detector performance is excellent, manifested in a nearly 100% functional tracker with high single hit efficiency, high quality track resolution and a good *Signal To Noise Ratio* (S/N) performance [10].

Figure 1.10 shows the S/N distribution of the *Tracker Inner Barrel* (TIB) and the *Tracker Outer Barrel* (TOB). The solid line represents the best fit of the experimental data (step line). The S/N is of the order of 20 in the whole SiStrip Tracker.

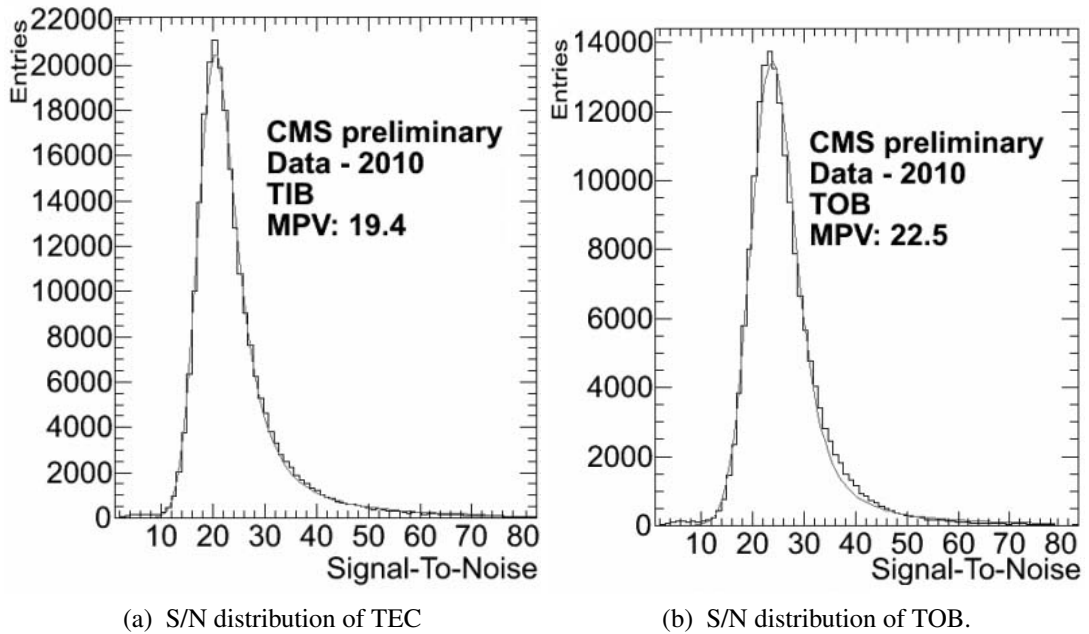


Figure 1.10: Signal to Noise ratio in the Silicon Strip Tracker.

This is made possible by a fine-grained calibration process and a monitoring of all important quantities for the detector performance during different steps of the operation. As an indication of the excellent performance of the Silicon Strip Detector, the overall mass distribution of the reconstructed $K_S \rightarrow \pi^+ \pi^-$ is shown in Figure 1.11.

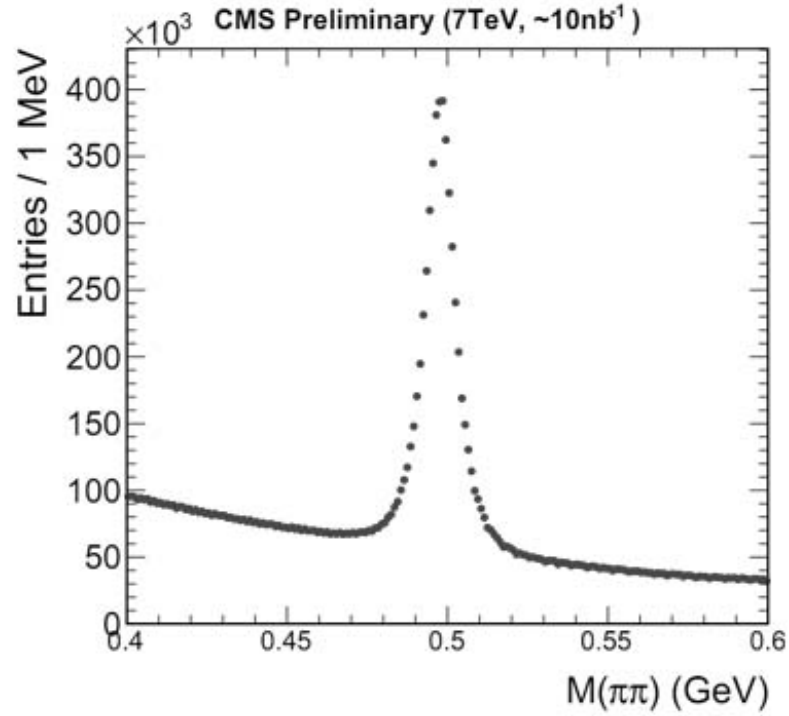


Figure 1.11: The dipion invariant mass for $K_S \rightarrow \pi^+\pi^-$ decay. A clear K_S peak can be seen.

1.3.3 Calorimeter system

The CMS calorimeters have distinct hadronic (HCAL) and electromagnetic (ECAL) systems. The central HCAL is made of brass and scintillators [11] while the ECAL comprises lead tungstate crystals ($PbWO_4$) [12].

Both calorimeters, ECAL and HCAL, were pre-calibrated before their installation in CMS. The 10 year long campaign of calibration and test beam allowed starting in excellent conditions and with a very good comprehension of the apparatus [13].

ECAL

The design goal of the ECAL channel-to-channel calibration is 0.3% and will be achieved using electro-magnetic decays of W and Z bosons. The off-site pre-calibration conducted using cosmic muons provided a precision at start-up of 1.5% to 2.2% in

the barrel depending on the pseudorapidity for every channel and 5% in the End-cap. Nine of the 36 modules constituting the barrel were independently pre-calibrated with an electron beam at a precision better than 0.5% [14]. A strategy was developed to select π^0 and η mesons with a dedicated trigger and use their decay in two photons for calibration in the limited luminosity start-up conditions. The Figure 1.12 shows the π^0 invariant mass reconstructed from photon pairs accepted by off-line selection obtained with $0.31nb^{-1}$. The black dot line represents the experimental data, the blue line is a fitting curve of experimental data and the red dot line represents the background.

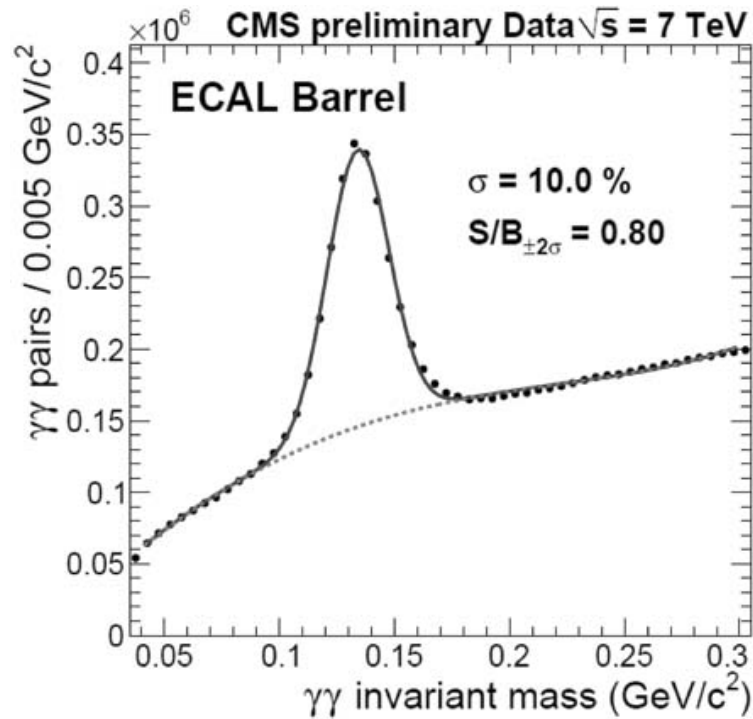


Figure 1.12: π^0 invariant mass reconstructed from photon pairs accepted by off-line selection obtained with $0.31nb^{-1}$.

With the first $250 nb^{-1}$ the π^0 in-situ channel-to-channel calibration, combined with a φ -symmetry calibration, reached a precision of 0.6% in the central part of the barrel ($|\eta| < 0.785$) [15]. This measurement asserted the validity of the precalibration obtained off-site for its in-situ use. Its accuracy is limited by statistical uncertainties and is getting improved while acquiring new data. It is expected to reach 0.5% in the barrel

and 1 to 2% in the End-caps. The preshower was calibrated with a precision of 2.2% already achieving the design goal. The first data showed a very good matching with the Monte-Carlo simulation without any tuning, demonstrating a good understanding of the detector. The Figure 1.13 illustrates the pseudorapidity distributions of the ECAL barrel channel with the highest reconstructed energy in 7 TeV minimum bias collision events. The black line represents the results of Montecarlo simulations, the dotted line shows the experimental results.

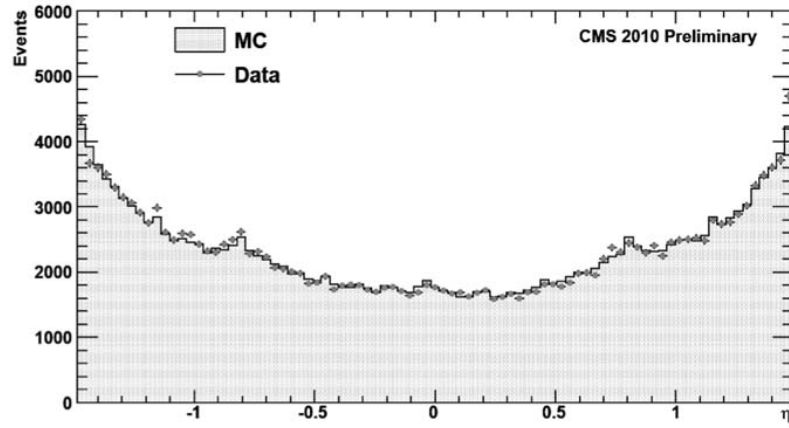


Figure 1.13: Pseudorapidity distributions of the ECAL barrel channel with the highest reconstructed energy in 7 TeV minimum bias collision events.

HCAL

The pre-calibration of HCAL combined test beam calibration, in-situ channel-to-channel calibration with a ^{60}Co source and in-situ cosmic ray events and “splash” events (single beam shots sent to closed collimator near the CMS experiment resulting in a large flux of muon traversing the detector). The channel-to-channel pre-calibration has a precision of 5% to 12% for 85% of the barrel channels, 10% for End-cap channels, 12% for the forward detector and 5% in the tail catcher [16].

The pre-calibration and the good understanding of the calorimeter response, allowed a start up with an energy scale calibration obtained from Monte-Carlo with a fair precision of $10\% + 2\% |\eta|$ for jet reconstruction using only the calorimeter information and $5\% + 2\% |\eta|$ for algorithms combining calorimeter and tracker information. The

first 71nb^{-1} of data were used to assert the validity of this precision. They support the quoted uncertainties as conservative numbers [17]. Figure 1.14 shows the resolution of the jet transverse momentum as measured with the dijet asymmetry method [18]. The jet resolution in pseudorapidity region $0 < |\eta| < 1.4$ is determined with the asymmetry method from *Quantum Chromodynamics* (QCD) simulation and compared with the result from data using the same procedure. The jets are reconstructed with the particle flow algorithm.

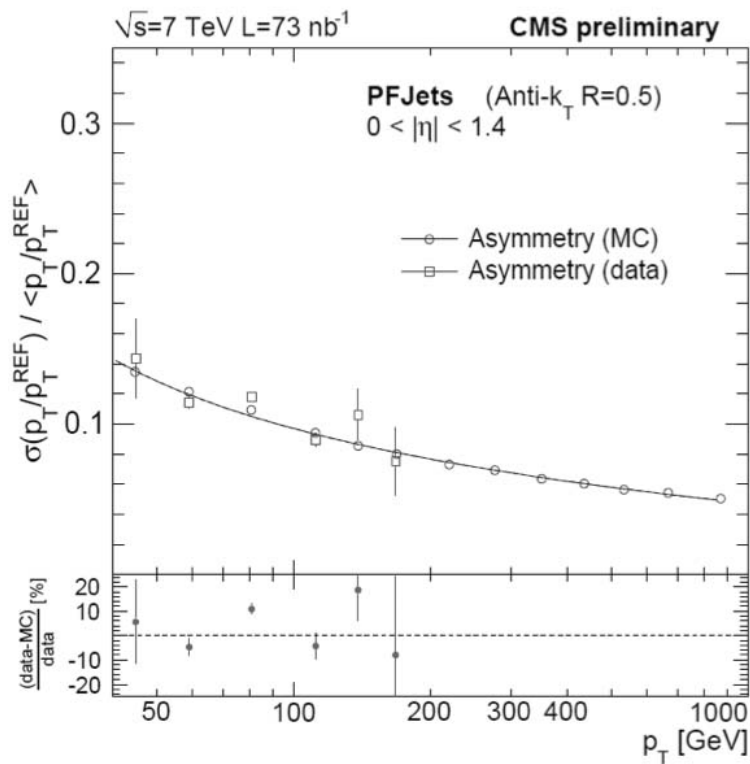


Figure 1.14: Jet resolution in pseudorapidity region $0 < |\eta| < 1.4$.

1.3.4 The Muon System

Final states with muons are a signature for important processes produced at LHC. The Muon System is the outermost detector in CMS and consists of three independent subsystems: the Drift Tubes (DT) [19], the Cathode Strips Chambers (CSC) [20] and

the Resistive Plate Chambers (RPC) [21]. The muon system has to ensure an efficient muon triggering as well as an accurate reconstruction of particle trajectories. Results on the efficiency of the system, using known physics processes, especially in the di-muon channel, are presented in the following Figure 1.15. Figure 1.15a shows the efficiency (per muon) as a function of the isolation cut for muons from $W \rightarrow \mu\nu$ in simulated events, obtained from the Monte Carlo truth (solid line); isolation cut efficiency obtained by applying the *Lepton Kinematic Template Cones* (LKTC) method for $W \rightarrow \mu\nu$ events to simulated $W \rightarrow \mu\nu$ events (dashed line), and to candidate $W \rightarrow \mu\nu$ events in data (markers). Figure 1.15b shows the ratio of the efficiencies measured in data and simulation using the LKTC method.

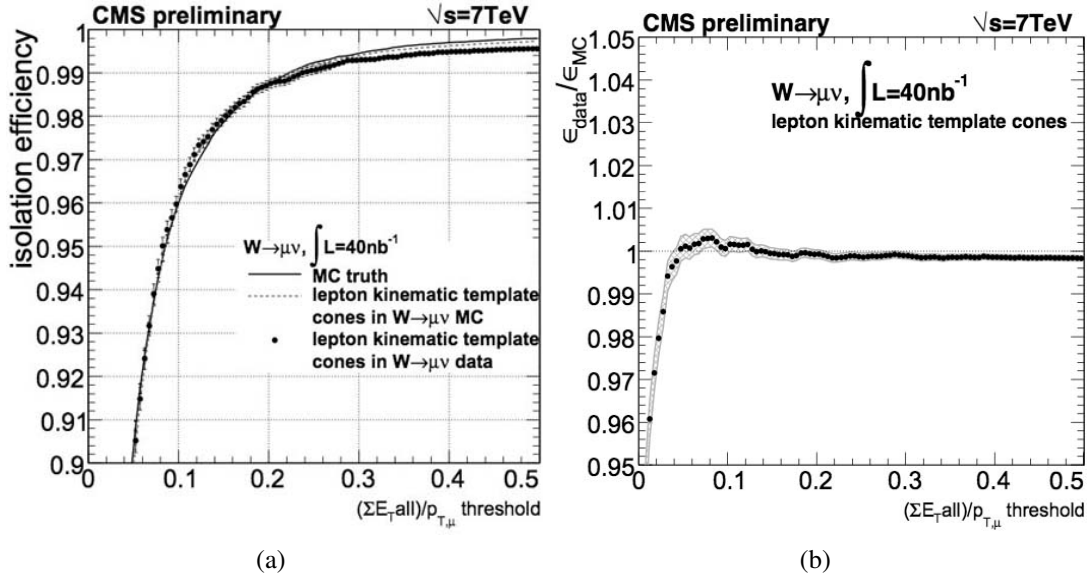


Figure 1.15: Isolation cut efficiency for prompt muons [22].

1.4 CMS computing campaigns from 2006 to 2010

Since the year 2006, the CMS experiment has been instrumented and commissioned in different campaigns using cosmic muons for testing the detector response and data taking and producing the first calibrations of the detector. Additionally the computing and software infrastructure of CMS has been tested in large simulation exercises

involving all the stages of the computing challenge, ranging from the reconstruction process to the high level physics analysis.

The Figure 1.16 shows the timeline of the major commissioning phases of CMS from 2006 to 2011. Muon alignment workflows and CMS software release were tested in most of these stages and up to the level of development achieved at the moment of the campaign.

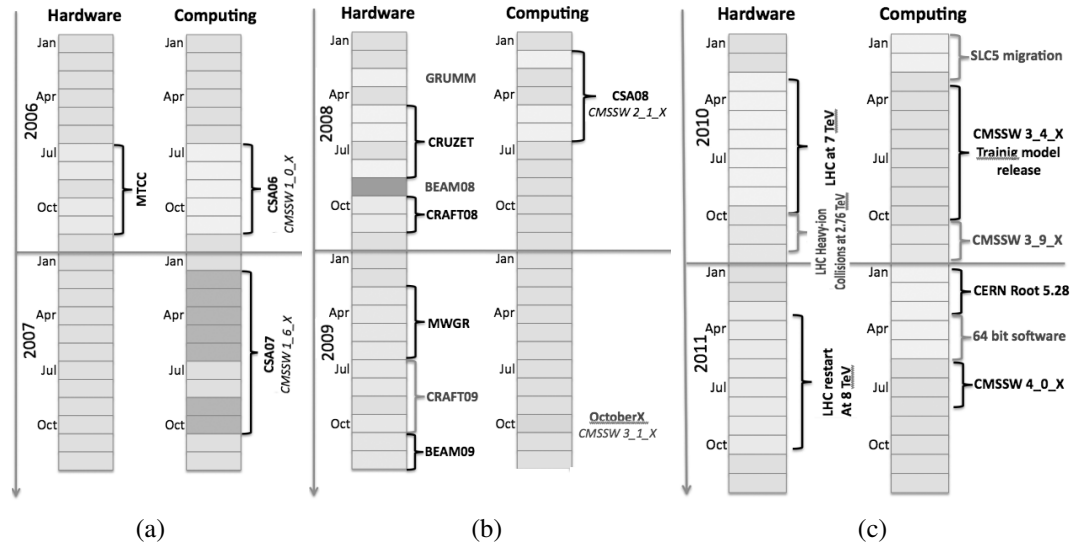


Figure 1.16: The Figures 1.16a, 1.16b refer, respectively, to time distribution of the commissioning phases for detector and computing exercises from 2006 to 2007 and from 2008 to 2009. The computing exercises during the 2010 and the expectations for year 2011 are given in Figure 1.16c.

1.4.1 Magnet Test and Cosmic Challenge

From the point of view of the detector commissioning, the first large-scale campaign was the *Magnet Test and Cosmic Challenge* (MTCC), that took place in the year 2006 when the detector was still installed at the surface of Intersection Point 5 (IP5). The main results of this campaign are:

- The successfully ramping of the magnet to several field strengths from 2 Tesla to nominal 4 Tesla including tests of fast discharges.

- The collection of about 200 million cosmic muon events.
- The measure of the field maps for different field values with a precision of 10^{-4} .

These results confirmed that the large CMS solenoid was fully operational. In this sense, the MTCC represented the first global commissioning test of the CMS at all levels, from the data taking up to the physics analysis.

After the MTCC, CMS was lowered to the cavern, and fully instrumented again. In the beginning of 2008 the first global commissioning runs were produced under the so called Global Run in Middle-March (GRUMM) [23] of 2008. After GRUMM, the Cosmic RUn at Zero Tesla (CRUZET [24]) took place until the end of August, in a series of time-separated cosmic runs. At this point the detector was prepared for the first collision data during September 2008.

1.4.2 Cosmic Run At Four Tesla 2008

After a short period of beam operation in 2008, which stopped when the LHC suffered damage, the collaboration continued a month-long data taking exercise known as the Cosmic Run At Four Tesla (CRAFT) [25]. In addition to commissioning the experiment operationally for an extended period, the cosmic muon dataset collected during CRAFT has proven to be extremely useful for understanding the performance of the CMS experiment as a whole.

The objectives of the CRAFT exercise were the following:

- Test the solenoid magnet at its operating field (3.8 T), with the CMS experiment in its final configuration underground.
- Gain experience operating CMS continuously for one month.
- Collect approximately 300 million cosmic triggers for performance studies of the CMS sub-detectors.

The detector was fully instrumented and all the sub-systems were involved in the data taking. The magnet was commissioned again during this period in its final configuration at the underground cavern, giving the opportunity to measure the momentum

of the cosmic muons. One of the drawbacks of the previous global runs GRUMM and CRUZET was due to the fact that no momentum measurement was possible because the magnet was not connected. This is important, as many calibration work-flows have a strong dependency on the momentum of the muons.

Although the CRAFT exercise took place from October 13th until November 11th, 2008, the analysis of CRAFT data extended during the following months, being extremely useful from the point of view of the calibration and alignment procedures of all the subsystems.

1.4.3 Computing campaigns

Several computing exercises were carried out in parallel with the commissioning of the detector. During the years 2006, 2007 and 2008 three exercises called Computing, Software and Analysis challenge (CSA06, CSA07 and CSA08) took place testing the response of the computing infrastructure at different scenarios of data taking rates. An analysis-oriented exercise was carried out as well in October 2009, and called the October Exercise in preparation for the imminent collision data.

CSA06

The combined Computing, Software, and Analysis challenge of 2006 [26] was an O(50) million event exercise to test the work-flow and data-flow associated with the data handling model of CMS. It was designed to be a 25% capacity test of what was expected for operations in 2008. The main results of this exercise are:

- the preparation of large simulated datasets (some with High Level Trigger tags)
- the prompt reconstruction at the Tier-0
- the distribution of all the data all participating Tier-1s (as well as to some Tier-2s)
- the calibration jobs at Tier-1, Tier- 2 and *CMS Analysis Facility* (CAF)

- re-reconstructions performed at Tier-1 sites, skim jobs at some Tier-1s with data propagated to Tier-2s
- physics analysis jobs at Tier-2s.

CSA07

The Computing Software and Analysis Challenge 2007 (CSA07) [27] has been an opportunity to exercise as many elements as possible of the CMS Computing and Analysis models simultaneously at a defined scale. The results of this campaign can be enclosed in the following points:

- CSA07 succeed to emulate the 50% of the computing load expected for the data taking. The exercise was focused in the use of computing and off-line tools with a diverse and active user community, using similar data streams as the expected from the data taking, and focusing in the balance of the simulation and analysis activities.
- The total number of generated events was $O(130 \text{ MEvents})$.
- One of the improvements in CSA08 is the inclusion of HLT bits in the samples at the Tier-0 level.
- On the other hand, data transfers from Tier-0, Tier-1s and Tier-2s were extensively commissioned through load tests.

CSA08

The CMS Computing Software and Analysis challenge (CSA08), constitutes the first full-scale challenge with large statistics under the conditions expected at LHC startup: initial mis-alignments and mis-calibrations as expected before collisions, and event signatures and rates typical for low instantaneous luminosity. The focus for this exercise was on the off-line detector commissioning, with calibration and alignment constituting the central components. The scope of the prompt alignment and calibration

activities conducted during CSA08 demonstrate the complete work-flow. The results obtained are as follows:

- successfully reconstruction of the data at the Tier-0, performing the skims by selecting the information for the alignment and calibration algorithms on the basis of HLT bits, including necessary background rejection and running these algorithms at the CAF with a low turn-around latency in the generation of new constants.
- satisfactory monitoring of all these steps wherever possible.
- be able to run concurrently the full complexity of almost 20 calibration and alignment work-flows intended for startup, with interdependencies taken into account.
- the schedule was matched to the pace of the reconstruction. Alignment and calibration were performed in real-time, with resulting constants used to re-reconstruct the data.

1.4.4 Proton-Proton collisions data taking

On 20th November of 2009, the LHC circulated its first beams after the one-year interruption following the incident of September 2008, initiating a remarkably rapid beam-commissioning phase.

On 11th December the first “physics fill” of the LHC began with multiple-bunch stable beams at 450 GeV each with a long lifetime. CMS quickly saw some jet events (see Figure 1.17), single muons (see Figure 1.18) and even a candidate decay of a K-short (see Figure 1.19).

A couple of days later the LHC accelerated, for a couple of minutes, two counter-rotating beams of protons to 1.18 TeV each, corresponding to 2kA in the superconducting dipole.

On 18th December 2009, the LHC ended its first full period of operation, following collisions at a total energy of 2.36 TeV a world record. Following these milestones, a

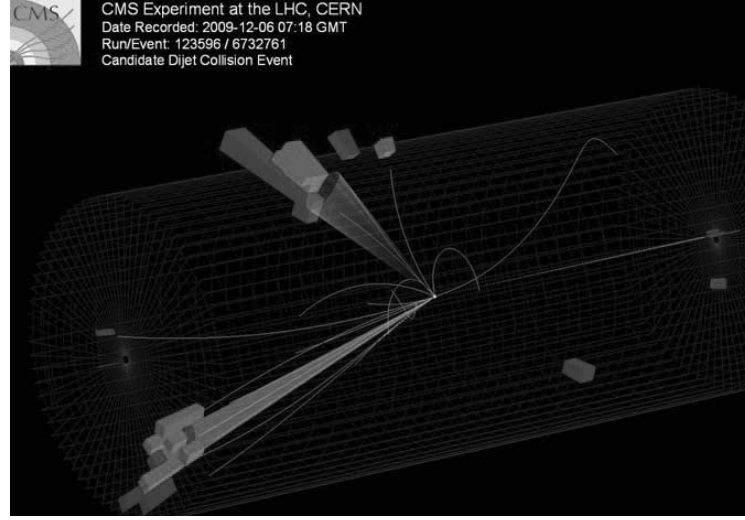


Figure 1.17: Example 2-jet event from a 900 GeV collision in CMS.

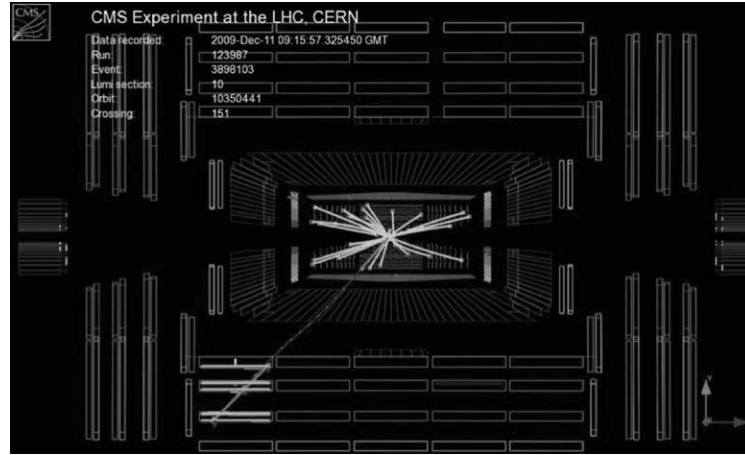


Figure 1.18: Example single muon event produced in a 900 GeV collision in CMS.

systematic phase of commissioning led to a period in which the CMS experiments recorded more than a million collision events, which were distributed for analysis around the world on the LHC Computing Grid. At the end of this first period of running, the LHC went into standby mode for a short technical stop to allow preparations for higher energy running after a restart scheduled for February 2010.

In the early hours of 28th February, beam was circulating again in the LHC. During the following weeks of data taking the consistency of the CMS computing model has

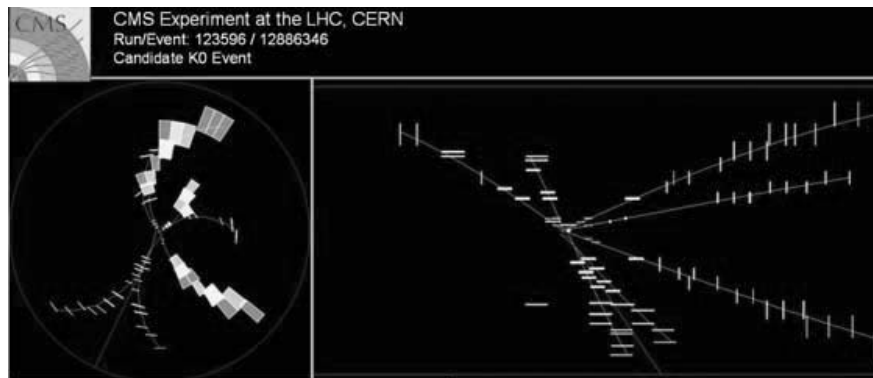


Figure 1.19: A close-up view of a candidate decay of a K-short to two pions in the central tracker of CMS, produced in a 900 GeV collision

been confirmed. This model is based on a hierarchy of use-cases deployed between the different tiers and, in particular, the distribution of RECO data to Tier-1s, who then serve data on request to T2s.

On 30th March, CMS control echoed with cheers as the machine produced the first collisions at a record energy of 7 TeV in the centre of mass. Over the following days, the LHC experiments started to collect millions of events during long periods of running with stable beams, thus beginning an extended journey of exploration at a new energy frontier.

Computing activities at the CAF have been marked by a good response time for a load almost evenly shared between **ALCA** (Alignment and Calibration tasks - highest priority), commissioning and physics analyses. Latencies, in particular at **Tier-0** and **CAF**, were well within design goals, allowing prompt reconstruction to be performed and calibration constants to be produced promptly.

Since March 30th there has been a continuous export of data from CERN, with high peaks during the first LHC “squeeze” fills (increasing the density of the protons in the bunches) at the end of April, tripling the initial transfer rate without any difficulties. Aggregated transfer rates of processed data from CERN towards all sites of the LHC Computing Grid were well in the range of a few GB/sec and the system showed flexibility in dealing with occasional backlogs. The observed quality of service at Tier-1s for prompt skimming (selecting samples of data for particular analyses)

and reprocessing was satisfactory.

Following the first 7 TeV collisions, the intensity of the LHC beams has rapidly increased, allowing the “re-discovery” of all known Standard Model particles from the neutral pion through to J/ψ and Upsilon mesons and culminating in the detection of thousands of W and Z bosons, as well as many hundreds of Top quarks. In addition, searches for new physics phenomena such as leptoquarks, quark-compositeness (internal structure of quarks) and Supersymmetry have begun, with CMS setting new limits on these and more.

1.4.5 Pb-Pb collisions data taking at 2.76 TeV

On 8th November 2010, the CMS experiment, has recorded its first Lead-Lead collisions at a centre of mass energy of 2.76 TeV per nucleon pair, marking the start of its heavy ion research programme. CMS immediately detected the first collisions, each producing thousands of particles whose trajectories were reconstructed in the CMS silicon detectors and whose energies were measured in the calorimeters. Moments later, the data were analysed and the first images of these events were produced.

The LHC collided heavy-ions until December 6th, delivering enough data for CMS to exploit new probes such as jets of hadrons, Upsilon mesons and Z bosons. Every collision was recorded, thanks to the large bandwidth of the CMS data acquisition system. The upcoming results will shed light on the nature of the strong interaction as well as on the state of the Universe in the first microseconds after the Big-Bang

1.5 Conclusion

The Large Hadron Collider (LHC) started operations with first proton-proton collisions in November 2009. This marked the transition of the CMS computing system from preparation to operations. Detailed preparations of the system and the experience gained in numerous computing challenges paid dividends. Computing and off-line projects have proven to be flexible enough to cope with the rapidly changing environment and demands. The 2010 data taking campaign has been completely successful

and data was delivered with short latencies for first physics analysis.

The year 2011 promises even higher intensity beams from the LHC, allowing CMS to push the boundaries of particle physics even further and, perhaps, discover something completely new.

Chapter 2

Computation of Alignment and Calibration Constants in the CMS Experiment

2.1 Introduction

The CMS experiment has developed an appropriate monitoring system to ensure the precise and prompt alignment and calibration of its components, which is a major prerequisite to achieve the optimal performance for physics analysis.

The prompt alignment and calibration activities exploits computing resources both at the *LHC Computing Grid Project* (LCG) Tier-0 and at the *CMS CERN Analysis Facility* (CMS-CAF) to ensure fast turnaround for updating the corresponding constants in the *Condition databases*. An essential element is the creation of dedicated data streams containing only the specific event information required by the various alignment and calibration work-flows (see Section 2.4). A low latency is required for feeding the resulting constants into the prompt reconstruction (PrompReco) process, which is essential for achieving swift physics analysis of the LHC data.

This chapter, briefly describes the CMS-CAF and the Tier-0. Next, an overview of calibration and alignment work-flow is given. Then, the various data stream transfers from CMS to Tier-0 are described. Finally some physics results depending on the

alignment and calibration are shown.

2.2 CMS CERN Analysis Facility and Tier-0

At CERN, the Tier-0 and the CMS-CAF provide complementary functions for CMS computing, being different “logical” entities within a single large computing farm. Together do they provide the required CMS computing at CERN.

The CMS detector is a very complex apparatus with more than 70 million acquisition channels. To exploit its full physics potential, a very careful calibration of the various components (crystals, drift tubes, silicon devices) and their attached electronics, is absolutely needed. Table 2.1 can be interesting to have an idea of the alignment and calibration challenge.

CMS sub-detector	Acquisition channels/components
Tracker	76M channels 16588 modules
ECAL	76K crystal
HCAL	10K channels
Muon	840K channels 1600 chambers

Table 2.1: Number of components used for each CMS sub-detector.

2.2.1 The CMS CERN Analysis Facility

The design of the CMS-CAF is based on the assumption of routine data taking mode. Under such regime, while most CMS physics analysis is distributed around the world as described in the CMS computing model [43], the CMS-CAF plays a unique role for latency-critical functions at the source of the analysis chain, with no dependence on the GRID infrastructure. Such work-flows are based on a highly selected subset of the full production data: the so called Calibration and Express Streams (ES), and are made available to restricted analysis groups soon after data taking occurred ($\sim 1h - 2h$). They take precedence over all other activities on the CMS-CAF and can be classified as follows in order of priority:

- alignment and calibration
- trigger and detector diagnostics, monitoring and performance analysis
- physics monitoring, analysis of *Express stream*, fast-turnaround high-priority analysis

The overall CMS data flow at CERN including these three main CMS-CAF activities are shown in Figure 2.1. The data is sent from the online computing centre, near the CMS detector Interaction Point 5 (IP5), to the Tier-0 centre at the Meyrin CERN site, where dedicated calibration and *Express streams* are processed and sent to the CMS-CAF within a very short latency ($\sim 1 - 2h$). After new calibration and alignment constants have been computed on the CMS-CAF, the condition databases are updated and the full event reconstruction can start at the Tier-0 within 24 hours. Other performance or physics analysis activities on the CMS-CAF might be carried out on a much longer timescale and with different priorities, hence requiring careful resource management due to the large variety of work-flows.

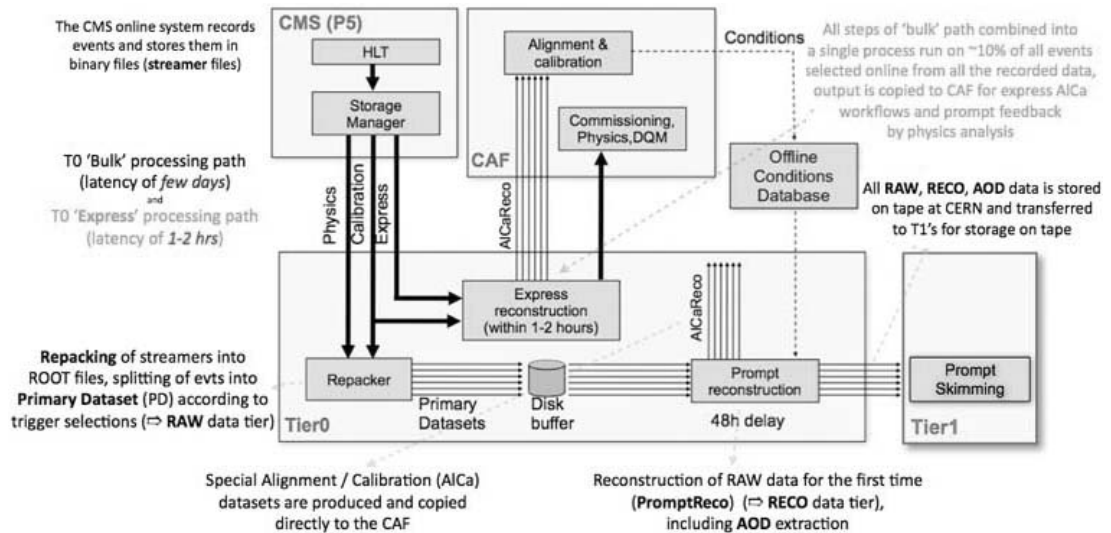


Figure 2.1: Alignment and calibration data paths and work-flows within the overall CMS off-line processing scheme.

The key hardware components for addressing the low-latency and high-priority requirements are a large dedicated processing farm and a highly accessible disk storage system. The limitation in available resources demands selectiveness in the jobs to be run and in the data to be stored, hence the need of a careful CPU and data management planning and prioritisation. It also involves rigorous user management, by restricting access to persons performing the corresponding priority tasks and by closely monitoring their activities. A detailed description of these monitoring tools will be given in Chapter 4.

CMS-CAF Resources

A dedicated processing farm has been installed in several steps during 2008, reaching a total of ~ 100 worker nodes and ~ 700 cores¹. Access, scheduling and sharing is handled by the CERN *Load Sharing Facility* (LSF) system [44]. The purpose of the LSF software is to dispatch the user jobs to the CPUs available, to control the jobs during their runtime and to collect and report their output. The CMS-CAF worker nodes can receive 1 job slot per core², leading to nearly 700 available job slots as shown on the left of Figure 2.2. The batch system, computer software that performs job scheduling, comprises one main queue (“cmscaf” ~ 630 job slots) available to all CMS-CAF users and a special queue (“cmsexpress” ~ 50 reserved job slots) dedicated to very high-priority alignment and calibration work-flows.

A large disk storage system based on CASTOR technology [45] has been installed in various steps during 2008, reaching a total of 215 disk servers and a capacity of ~ 1.3 PB on December 2008.

2.2.2 Tier-0

The Tier-0 centre is hosted at CERN and its primary functions are to:

- Accept data from the online system with guaranteed integrity and latency, and

¹Some worker nodes contain 8 cores, some 4 cores

²2 worker nodes were reserved for memory intensive alignment activities (2 cores - 4GB RAM - per job slot).

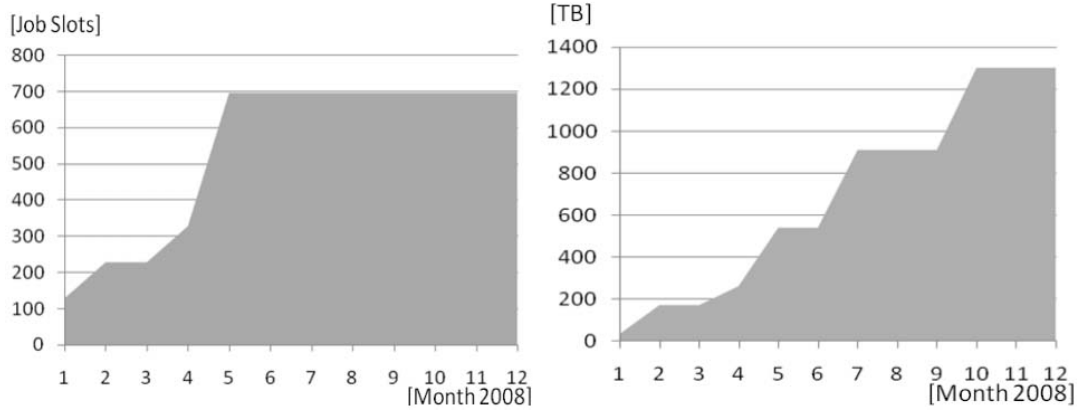


Figure 2.2: CMS-CAF CPU and Disk Resource ramp up during 2008.

copy it to permanent mass storage;

- Carry out prompt reconstruction of the RAW data to produce first-pass RECO (Reconstructed events) datasets. The centre must keep pace with the average rate of data recording, and must provide sufficient input buffering to absorb fluctuations in data rate;
- Reliably export a copy of RAW and RECO data to Tier-1 centres. Data is not considered “safe” for deletion from Tier-0 buffers until it is held at at least two independent sites.

The resources deployed for CMS are listed in Table 2.2. KSI2K (Kilo Specmarks Integer year 2000) is the benchmark used to measure the computing power of all the HEP experiments.

	CPU (KSI2K)	Disk (Tbytes)	Tape (Tbytes)
Tier-0	19100	400	11100
CMS-CAF	9700	3400	3200

Table 2.2: Tier-0/CMS-CAF computing resources on December 2010 [46]. The disk space of Tier-0 refers to the internal data buffer of the tape drive. 400 TBs correspond to \sim one hour of data taking.

The Tier-0 centre generally performed well during the 2010 proton-proton running. The CMS data acquisition rate was higher than the design during much of the proton-proton running as CMS developed a better understanding of the trigger and selection methods. The lower level of pile-up in these events means the reconstruction time is also shorter than the design parameters, which allowed the Tier-0 to accommodate the higher rates. The trigger group was conscientious to stay below rates that would overwhelm the off-line system. Toward the end of the October the machine achieved a higher average time in stable beams and the average utilization of the Tier-0 processors increases accordingly. Even with higher instantaneous luminosity and a higher trigger rate CMS maintained close to the design rate into the *Express stream* (see Section 2.4). In CMS the *Express stream* is expected to be fully reconstructed within an hour of data collection. The Tier-0 successfully maintain this rate even with higher luminosity and progressively more interesting events.

2.3 Calibration and alignment work-flow

The accurate and timely determination of alignment and calibration constants is essential in order to achieve the optimal performance of the CMS detector for physics analysis, and has played a central role in the commissioning of the detector during early LHC data taking. Calibration and alignment work-flows are categorized by their required latency [47]. There are three main categories:

- Detector-near calibrations: Fast changing quantities such as electronics pedestals and gains, which can vary from one run to the next, are in general determined on-line, run by run, at the site of the experiment, either during collision data taking, or in dedicated pedestal or pulser runs.
- Prompt calibration and alignment: Constants which can be determined off-line within 24 hours of data taking and are required to be updated regularly are fed back into the conditions database to be used in first full reconstruction (prompt reconstruction) of the RAW data arriving from the detector. This is referred to as the prompt calibration loop, and is described in Section 2.4.2.

- Long-range calibration and alignment: Constants determined using a data sample accumulated over a longer period of time, ranging from a few days to a few months, are typically applied to the data in re-reconstruction campaigns.

With the exception of detector-near calibrations which are carried out online, alignment and calibration algorithms are run at the CMS-CAF at CERN. An overview of the work-flow for prompt calibration is illustrated in Figure 2.1. The work-flow for long range calibration is illustrated in Figure 2.3.

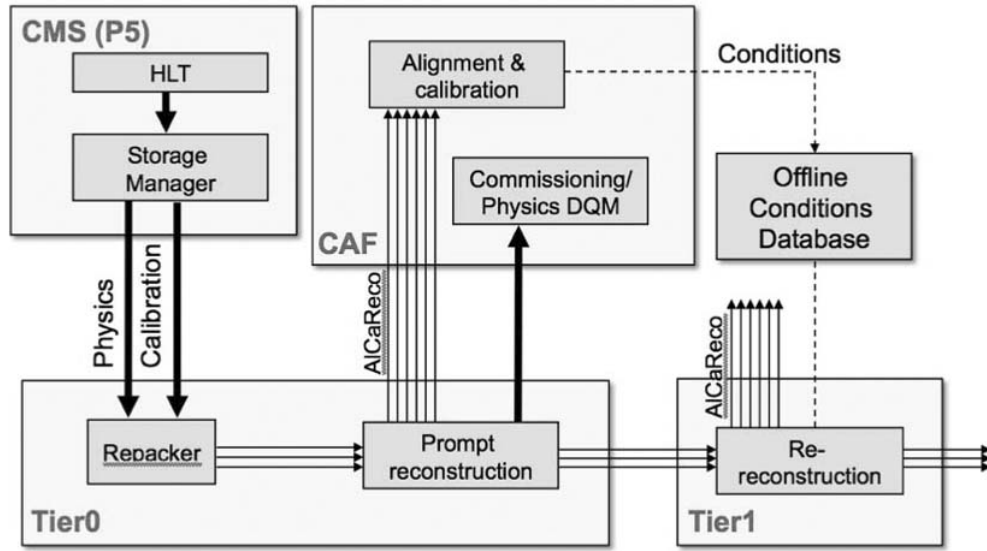


Figure 2.3: Sketch of the CMS alignment and calibration work-flows.

The main components, discussed in the sections which follow, consist of dedicated data streams from LHC Point 5 to the Tier-0, the transfer of dedicated skims of the reconstructed data from the Tier-0 to the CMS-CAF, the operation of the calibration and alignment algorithms at the CMS-CAF, and the uploading of the constants produced by these algorithms to the conditions database, ready for use in subsequent (re-)reconstruction.

A description of the CMS detector, and of the various alignment and calibration procedures for each sub-detector can be found in [48].

2.4 Transfer of data streams for calibration and alignment from CMS to the Tier-0

Data is transferred from LHC Point 5 to the Tier-0 in four principal streams:

- *Physics stream.* The main data stream intended for physics analysis.
- *Express stream.* A specially selected subset of the physics stream (approximately 10%, initially more, of the total data bandwidth), chosen according to High Level Trigger (HLT) bits, intended for prompt physics analysis, calibration and alignment, detector and trigger commissioning, and fast physics validation.
- *Calibration Streams.* Dedicated streams are set up for hardware calibration and alignment using special triggers during the *LHC orbit gap* (approximately 10% of the total data bandwidth). The LHC orbit gap is the time interval when no bunch passes. Orbit gap data is important for pedestal and noise calibration, which needs to be carried out when no signal is present. Orbit gap triggers are also used to transfer laser data used for tracker alignment and for monitoring of electromagnetic calorimeter crystal transparency.
- *AlCaRaw streams.* In addition to hardware calibration streams, there are dedicated collision event data stream for calibration with special reduced event content. These are referred to as “AlCaRaw” streams. Three calibration procedures require a very high event rate, of order 1kHz each, in order to perform the calibration within a reasonable time, which would saturate the affordable data bandwidth (approximately 300MB/s) if the full event content were transferred [49]. The procedures are as follows.
 - Calibration of the hadron calorimeter using φ – *symmetry*.
 - Calibration of the electromagnetic calorimeter using φ – *symmetry*. Both calibrations, HCAL and ECAL φ – *symmetry*, rely on the φ invariance of the energy deposition in physics events.

- Calibration of the electromagnetic calorimeter using neutral pions and eta mesons.

Instead of transferring the full RAW event content in these cases, only the minimal information needed for the corresponding calibration algorithms is transferred. Special high rate triggers are processed on the HLT farm at IP5, producing dedicated output streams with event content reduced by factor of around 100 with respect to the full RAW event.

2.4.1 Offline processing at the Tier-0: production of *AlCaReco* skims

In some cases, the calibration and alignment algorithms need to run over very large datasets, with many iterations, within a short period of time. *AlCaReco* datasets are skims of reconstructed data (both event selection and event content selection), containing the minimal information required as input to a given calibration or alignment algorithm. All *AlCaReco* skims perform an initial filtering of events based on a set of HLT bits. In some cases, additional processing is performed to produce specialized data collections needed as input to the algorithm. In general *AlCaReco* skims take reconstructed datasets as input. Some also require RAW information and these *AlCaReco* skims must be run in parallel with the reconstruction. A subset of *AlCaReco* skims is run on the *Express stream* as needed for the prompt calibration loop, as described in section 2.4.2.

An example of online calibration stream is the one selecting events containing π^0 and η candidates detected in ECAL and used for the inter-calibration of the $PbWO_4$ scintillating crystals [50]. The calibration performance depends on the number of selected π^0 candidates per crystal and on the signal to background ratio. The candidate di-photon decays are selected at HLT level from events passing single $-e/\gamma$ and single-jet Level-1 triggers. After selection, only information about a limited region of ECAL (energy deposits in 20 to 40 individual crystals) near the π^0 candidates is stored for the actual calibration. This allows to save bandwidth and CPU time.

The effectiveness of this calibration stream is demonstrated by the plots in Fig-

ure 2.4, which show the measured signal rate from the online stream and from generic minimum bias event stream as a function of the instantaneous luminosity and the invariant mass of the candidates in the central region of the detector. While the need of prescale on the minimum bias triggers suppresses the yield of π^0 events, the rate keeps increasing for the dedicated ECAL stream.

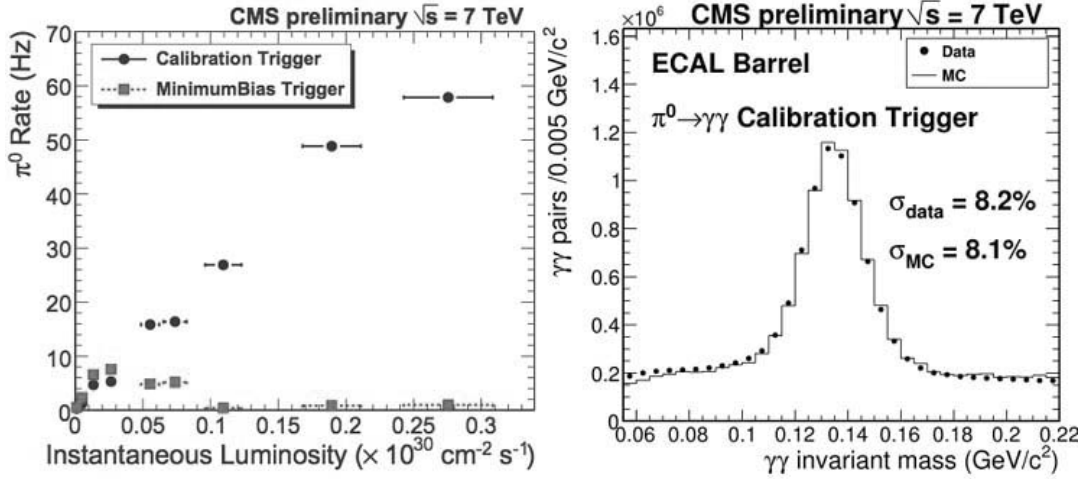


Figure 2.4: Rate of events containing π^0 candidates in the dedicated calibration skim and in events accepted by the minimum bias trigger (left). The drop in the minimum bias events visible around $L > 5 \times 10^{28} \text{ cm}^{-2} \text{ s}^{-1}$ is due to trigger prescaling. Invariant mass distributions for photon pairs accepted by the π^0 stream for the data and simulations in the central region of the detector (right).

2.4.2 The prompt calibration loop

Several types of conditions, for example the beam line parameters, can change at short time scales. The prompt calibration concept relies on an intentional delay of prompt reconstruction during which raw data are buffered on disk (see Figure 2.1), while prompt calibration work-flows determine the corresponding constants on the fly from the *Express stream* data, such that prompt reconstruction will proceed with the updated constants. Intensive tests have been performed with the beam line calibration work-flow. In the first step of this work-flow, the beam line fit is performed once per luminosity section using *express stream AICaReco* data. In a second step, ranges with stable

constants are “collapsed” into larger intervals of validity to increase the statistical precision and reduce the database storage size. Finally, the calibration object is validated and uploaded to the database. Figure 2.5 shows the latency of the first two steps observed during constant monitoring; the constants payloads with multiple intervals of validity are available typically within 1-2 hours after the end of data-taking for a CMS run.

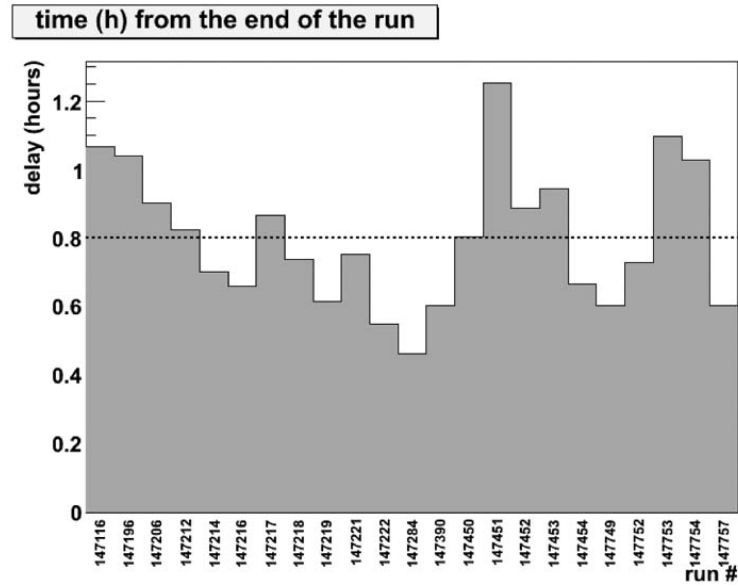


Figure 2.5: Time between the end of the run and the completion of the corresponding beam line parameter objects within the prompt calibration loop, for a series of CMS runs in October 2010.

2.5 Performance and experience

2.5.1 LHC data-taking

This section discusses performance and experience for the tracker alignment workflows. More details regarding calorimeter and tracker calibration can be found in [51], [52].

Tracker alignment

As said in Chapter 1, with about $200m^2$ of surface area and in total 16,588 modules, the CMS tracker is the largest silicon tracker ever built. CMS uses mainly two methods to align this complex system with tracks:

- the “local” method (named HIP) fits the alignment parameters of each module individually from set of the residuals in the given dataset. All tracks are then refitted with the alignment corrections applied. This procedure is repeated until convergence is reached.
- The “global” method `Millepede-II` performs a simultaneous fit of all alignment and track parameters.

The virtues of both methods can be combined for best results. Alignment has been performed combining cosmic ray data taken in early 2010 with the first nb^{-1} of 7 TeV collisions data. The goodness of fit (see Figure 2.6) is found to be significantly improved beyond the level of cosmic ray-only alignment (labeled STARTUP). The addition of tracks from collisions has the largest impact in the pixel detector, in particular the pixel end caps which were poorly illuminated with cosmics alone.

The methodology of alignment is further being improved. Residual curvature of silicon sensors, if not accounted for, may lead to a hit bias depending of position and slope of the track. In addition, some modules in the CMS tracker are composed of several sensors which may slightly deviate from complanarity, a feature described as “kink”. The optimal solution is to determine these features module by module as additional alignment parameters, which, however, leads to an increase in the overall number of parameters to about 200,000.

The successful determination of such a large set of alignment parameters has been achieved using an extended version of the `Millepede-II` program. Figure 2.7 shows how a systematic residual deviation, whose dependence on the position on the sensor reflects the sensor curvature, disappears after the appropriate curvature constants, determined module by module, have been taken into account.

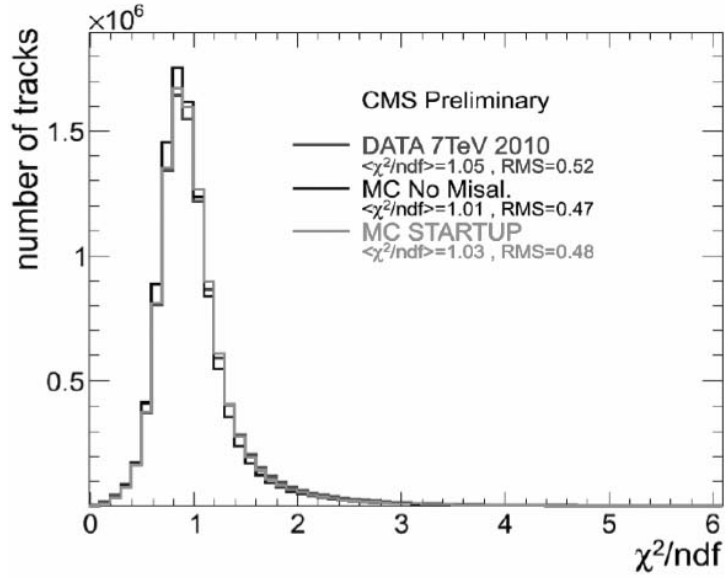


Figure 2.6: Goodness of fit for tracks in the CMS tracker after alignment with 7 TeV collision data combined with cosmic muon events, in comparison with simulated data at the level of alignment only with cosmic rays (“STARTUP”), and without any misalignment [53].

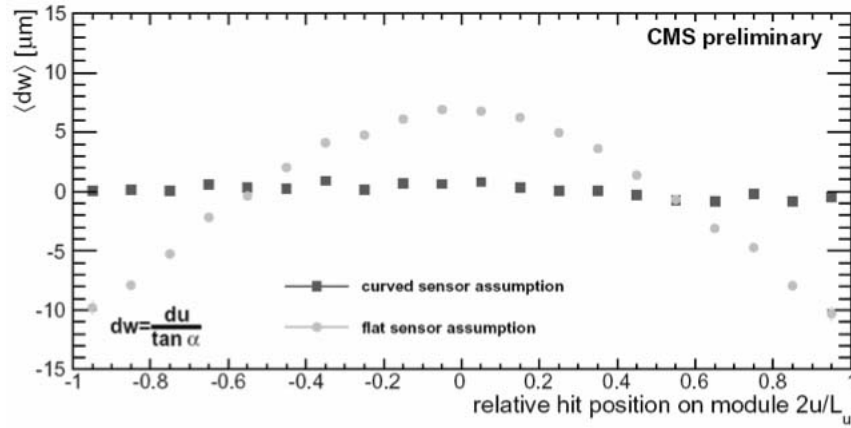


Figure 2.7: Mean residual bias translated to deviation vertical to the module plane as a function of the hit position on the module in units of its overall size. The graphs show the result assuming flat sensors, and the one with the determined curvature parameters taken into account.

Computational aspects of the alignment workflow

The Millepede alignment work-flow (see Figure 2.8) is structured into two steps.

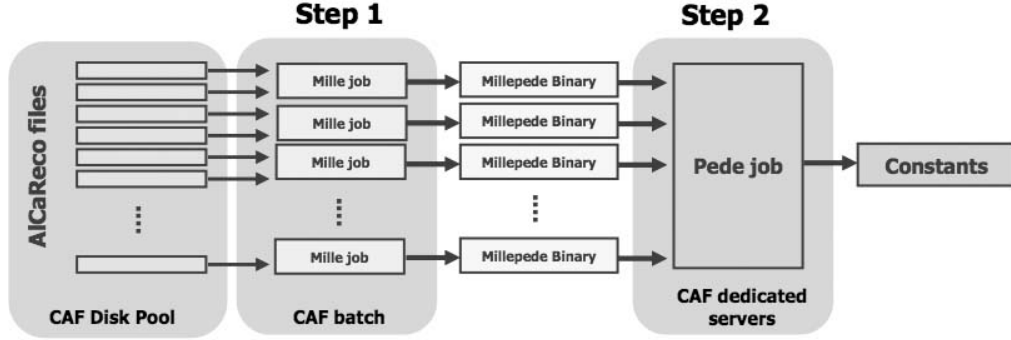


Figure 2.8: Illustration of the Millepede alignment computing work-flow.

- The first step includes all computations that proceed event by event, namely the selection of tracks and computation of residuals and their derivatives with respect to the parameters to be fitted. This step is performed in parallel in typically 100-200 jobs at the CMS-CAF.
- The second step consists of the global fit of all track and alignment parameters and is very CPU and memory intensive. Two dedicated CMS-CAF servers each with 48 GB of memory are available for this task. In order to optimize the turnaround for this work-flow, multi-threading has been introduced for the most CPU-intensive parts using the *OpenMP interface* [54]. On seven CPU cores, the overall execution time of a test job with 200,000 alignment parameters and about a million of tracks has been reduced from 9 hours to about 1.5 hours.

2.6 Conclusion

A powerful framework has been set up to ensure the precise and prompt alignment and calibration of the various components of the CMS detector. In order to ensure a fast turnaround for updating calibration and alignment constants, the strategy includes the transfer from CMS to the Tier-0 of dedicated high rate data streams for calibration,

and the production at the Tier-0 of dedicated skims of reconstructed data, containing only the information needed as input to calibration and alignment algorithms, which are run at the CMS-CAF. The resulting low latency is required for feeding the updated constants into the prompt reconstruction process, in order to swiftly provide high quality data for physics analysis. The framework is successfully employed in the LHC high energy data-taking period. A large set of work-flows are operating successfully on dedicated alignment and calibration skims [55] (see Table 2.3).

Job Type	Total Jobs	Failures	Success Rate
Express	132314	94	99.93%
Repack	7262	6	99.92%
PromptReco	36538	0	100%
AlCaSkim	21336	0	100%

Table 2.3: Example from early 7 TeV Running at the Tier-0 System.

Constants are being delivered in time and physics results of high quality have been derived based on these constants. The overall CMS alignment and calibration concept proves to be sound and efficient. Future steps will introduce additional work-flows that become feasible only with mounting luminosity, resulting in improved statistical and systematic accuracy, and further extension of the set of prompt calibration work-flows.

Chapter 3

Persistent storage of condition data in the CMS experiment

3.1 Introduction

The storage and the ways to provide access to the Condition data are key issues in the CMS computing model. The access to the condition data is necessary to the event processing applications, to guarantee the best precision of the reconstructed data, and to the monitoring tools used to check the quality of the data taking. The management of the Condition Data and in particular of the alignment and calibration constants, together with their correctness, is essential in order to achieve the optimal performance of the CMS detector for physics analysis. The main tasks addressed by the CMS condition database services consist in providing high-standard for quality, scalability, and maintainability for several applications, such as Data Quality Monitoring (DQM), alignment and calibration work-flows, web services, physics analysis.

During the detector construction phase, different technologies and software solution were developed in order to store the detector and equipment data. Finally, Oracle Real Application Clusters (RAC) and Automatic Storage Manager (ASM) on Linux has been chosen as the main platform to deploy the database services for physics.

The database services play a crucial role in the CMS experiment and also in the operation of the Worldwide LHC Computing Grid (WLCG) services. The CMS databases

are part of a distributed database network which require synchronization between the master database at CMS Interaction Point 5 (IP5) and the read-only Tier-0 databases. In order to allow CMS data to flow through this distributed infrastructure, an asynchronous replication technique (Oracle Streams) is used to form a database backbone between online at *IP5* and off-line site at *Tier-0*. An efficient condition data reading operation is assured from the FroNTier system. This system translates database queries into HTTP and caches the results in a proxy cache server called Squid. Moreover, secure and stable condition data storage operation from the off-line to the on-line database is performed using the **Off-line DropBox Service**. This tool is one of the main component developed in this thesis and is presented in Chapter 4.

Solid procedures for backup and recovery also need to be in place for such a service, which is required for its importance to be operated with a 24 hours per day, seven days per week (24/7) schedule. The data which was necessary to set up the detector (tens of GBytes) and the data produced by the experiment (few TBs per year), necessity to guarantee the access to this data to a large number of physicists worldwide distributed makes the database system an essential and delicate service for CMS to maintain and run.

In this chapter details on Condition data descriptions will be given. Finally, the CMS off-line database performance will be described and will be evaluate using both the traditional real-time database performance metrics (e.g., the number of jobs serviced on time, average tardiness) and the economic benefits (e.g., benefits to the CMS organization).

3.2 The Condition Data description

3.2.1 Introduction

In order to reconstruct the physics events of the CMS experiment[28], we use quantities which do not change from event to event and for this reason they are called *Condition data* or *Not-Event Data*. The *Condition Data*, produced by CMS detector, are stored in the CMS condition databases and can be classified in different groups,

according to their needs: Construction data, Equipment management data, Configuration data, Detector state data, Calibration data. Table 3.1 shows the condition data for the Silicon Strip Tracker (SST) detector. The construction data and equipment management data are not involved during the calibration and the alignment work-flow.

A unique classification of data items into the above classes is not possible. The *configuration data* for instance becomes a *detector state data*, once it has been used in the detector.

Configuration data	Fed Cabling, Threshold, APV Gain, Latency
Detector state data	Channel status (Bad Channel, Bad Fiber, Bad Module, high voltage, low voltage), Gain calibration
Calibration data	Noise, Pedestal, Lorentz angle, Bad Component, Cluster Threshold

Table 3.1: SST Detecotor condition data.

Construction data

Construction data includes all information about the sub detector construction up to the start of integration. The collection of such information was started at the beginning of the CMS detector construction and will be maintained for the full lifetime of the experiment. During the construction of the detector, data gathered from both the production process and the produced items. The different CMS sub-detectors agreed to keep their construction data available for the lifetime of the experiment in order to be able to trace back production errors. The construction data are stored in construction database based on Oracle technology. An example of construction data for the *Silicon Strip Tracker* (SST) is the number of dead strips in each module.

Equipment management data

Detector items need be tracked in order to log their history of placements and repairs. The classification of CERN as INB (Installation Nucleaire de Base) requires,

in addition, to keep a continuous trace of the location of irradiated items. Equipment management data contain, therefore, the location history of all items being installed at the experiment, in particular detector parts as well as off detector electronics (racks, crates, boards, cables, channels, etc.). This data is stored in the on-line database (OMDS). Thanks to this data is possible to make a 2D or 3D representation of each sub-detector, showing, for example, the number of dead channels obtained from construction database (see Chapter 4).

Configuration data

The data needed to bring the detector into any running mode are classified as configuration data. They comprise voltage settings as well as programmable parameters for front-end electronics. Configuration data require a version and time validity information as meta-data. This data is stored in OMDS.

Detector State Data

The data describing the state of any detector subsystem are defined as *Detector State Data*. These conditions are measured online and are stored in OMDS. They include data quality indicators (such as bad channel lists) and settings of the detectors needed off-line (such as pedestals). Condition data in OMDS are used in the online system for post mortem analysis of detector errors. Condition data needed for HLT and off-line reconstruction are uploaded in the condition database, ORCON (Off-line Reconstruction Conditions database ON-line subset), located in the CMS-TN. The *Detector State Data* meta data are described by a version and the time validity information corresponding to the set of data for which they are measured.

Calibration data

The data describing the calibration and the alignment of the individual components of the different sub-detectors are labeled as calibration data. These quantities (such as drift velocities, alignments constants, etc.) are evaluated by running off-line dedicated algorithms. Since they are needed by HLT and for off-line reconstruction, they appear

only in the off-line databases. Calibrations must match the corresponding raw data coming from the collision events revealed by the detector. Calibration data can be grouped by the version and the time range in which they are valid.

3.2.2 Reconstruction chain for the SST

The low level reconstruction chain for the SST is schematically represented in Figure 3.1. In the first step, it digitizes the raw data, after cabling calibration applied.

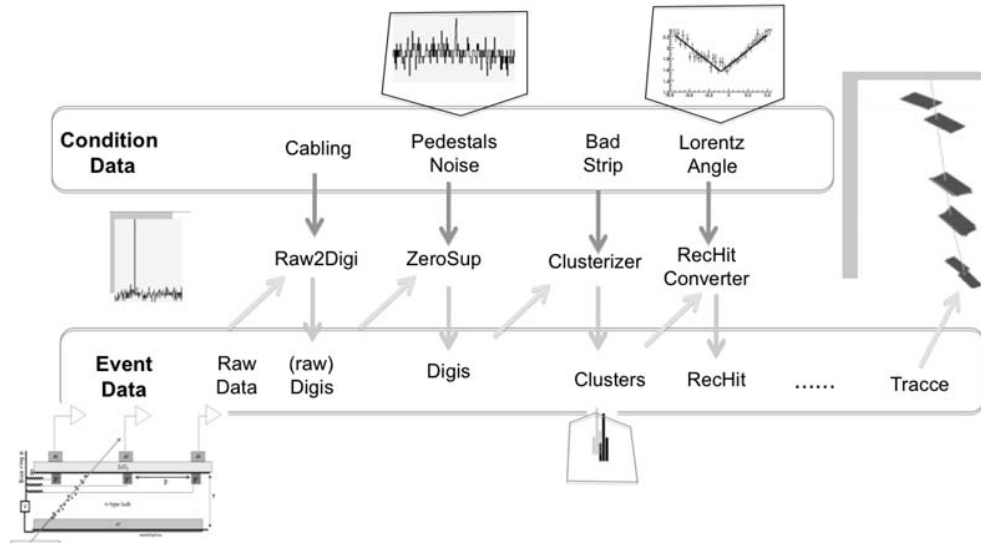


Figure 3.1: Block diagram of the low level reconstruction in the SST.

Then it proceeds with the clusterization, in which the clusters are reconstructed looking for those strips where the collected charge, opportunely calibrated by using the gain information, is higher than some definite thresholds of the signal-to-noise ratio. The clusters related to modules, fibres or APVs identified as bad (i.e. noisy or disconnected from readout) are removed from the reconstruction chain. Finally the *hits* are produced calculating the centroid of the clusters in terms of the local module coordinates. In this last step of the low level reconstruction the Lorentz angle condition data are needed to correct opportunely the hit position. The hit coordinates can then be converted in CMS global coordinates and used for the subsequent tracking procedure [29].

3.2.3 Metadata of CMS conditions data

A “conditions data object” is the smallest atomic entity of conditions data that can be manipulated individually. Every condition data object contain many different specific data items, such as mapping to electronic channels of the RPC, alignment constants of the Silicon Strip detector, and calibration of electronic circuits of the ECAL. For this reason each condition *Data Item* is identified by:

- the name of the CMS sub-detector element such as: ECAL, RPC, DT, SiStrip for the Silicon Strip Tracker and SiPixel for the Silicon Pixel Tracker,
- the type or nature of the data such as calibration, geometry, temperature.

The combination of both names, CMS sub-detector and data type, uniquely identifies each *Data Item*.

Since the condition data may vary over time each condition data item has a validity time range, *Interval of Validity* (IOV), with a start time (since) and an end time (till). Time can be expressed by means of either a time-stamp or run number or luminosity id. Some of the condition data may be recalculated to take advantage of a wider statistics to get a more precise picture of the detector behaviour or by using a different release of the CMS Software Framework CMSSW: therefore, the condition data may exist in more than one *Version*.

Data Item, IOV and Version are referred to as the three axes of “metadata” of conditions data in the CMS condition data model. The combination of *Version* and *Data Item* labels, as a unique human-readable name, is defined as TAG. Examples of such TAG names are reported in Table 3.2.

The actual values of the physics observables associated to a conditions data object (e.g. the actual values of the physics parameters describing the alignment of a tracking device in a given IOV and for a given version of the alignment algorithm) are referred to as the *payload* of that conditions data object (see Figure 3.2).

From the CMSSW point of view, a payload object is stored using the POOL-ORA technology; it might be rewritten as new objects after processing, but never updating or overwriting. The IOV index, implemented as a IOVSequence class, can be deleted

TAG name	Meaning
SiStripLorentzAngle_22X_express	Measured Lorentz angle values for the <i>Silicon Strip Tracker</i> (SST) running CMSSW version 2.2.X using the express stream.
SiStripLorentzAngle_31X_express	Measured Lorentz angle values for the SST running CMSSW version 3.1.X.
SiStripGain_22X_v1_offline_express	Gain values of the optical readout components for the SST, computed in the off-line calibration.
SiStripGain_22X_v2_offline_express	Gain values of the optical readout components for the SST. Differently from the previous version (v1), the second one (v2) has rescaled values for the temperature changes.

Table 3.2: Some TAGs name for the Tracker Calibrations

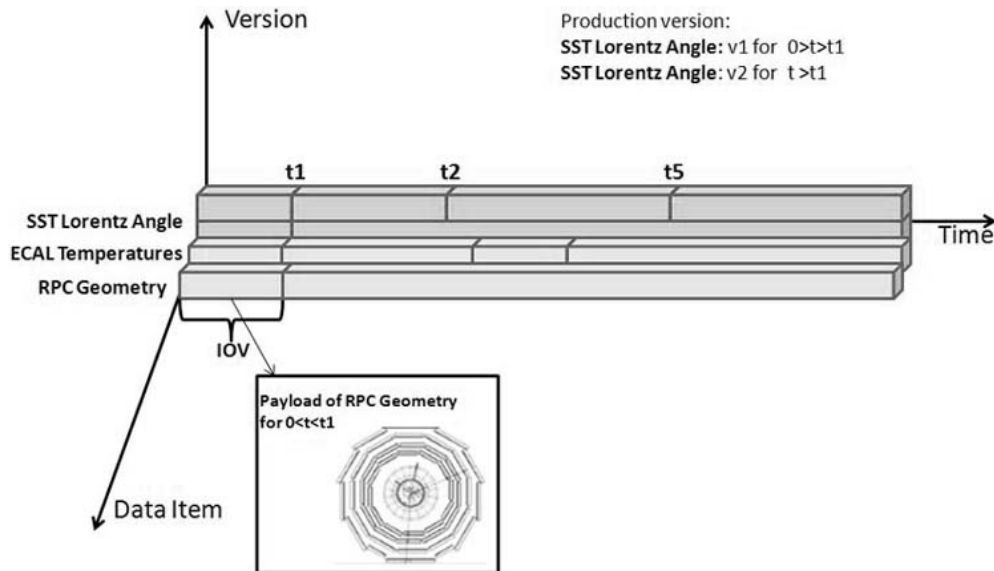


Figure 3.2: The three-axes for uniquely identifying each payload in the conditions database.

and recreated independent of the data object it points to. The IOV index (IOVSequence) is annotated with metadata which are called the IOV metadata. Table 3.3 shows an example layout of an IOVSequence. In this example, the IOVSequence ob-

Since	Till	Payload object token	Tag Name
1	3	DA11-ABCE-00A0C9DA776A	SiStripPedestals_31_v1_offline
4	7	DA11-99EF-00A0C9DA776A	SiStripPedestals_31_v2_offline
8	∞	DA11-67EF-00A0C9DA776A	SiStripPedestals_31_v3_offline

Table 3.3: IOVSequence example.

ject is an index on three payload objects. The first object, which is identified by its unique POOL object token DA11-ABCE-00A0C9DA776A, is valid from run 1 to run 3 (the first IOV of the IOVSequence); the second object DA11-99EF-00A0C9DA776A is valid from run 4 to run 7; the third object is valid from run 8 on.

Global TAG

When processing large quantity of calibration/alignment data, the production manager needs to organize collections of IOVSequence TAGs coming from different data sources. The Global TAG is designed to ease such efforts by organising TAGs into hierarchy similar to organising files in folders in the file system. The Global TAGs are high level metadata which do not interfere with the IOV information and basic TAGs stored together with the data.

The Global TAGs can be and usually are decoupled from the data and are used as the entry point to the underlying IOV TAGs. Each Global TAG is represented as a node of the tree. The tree topology requires that each node name (Global TAG) must be unique within the same tree.

Figure 3.3 shows a typical use case of Global TAG during a data taking. Starting from the time instant t_2 , because of a temperature change in SST, the gain values of the optical readout components change. In order to keep pace with temperature variation, new conditions for the gain values of the optical readout components are generated and labeled with a new TAG, SiStripGain_v3. Starting from the run number 10000, to perform the data taking with new condition, a new Global Tag, GR10_H_V3 is

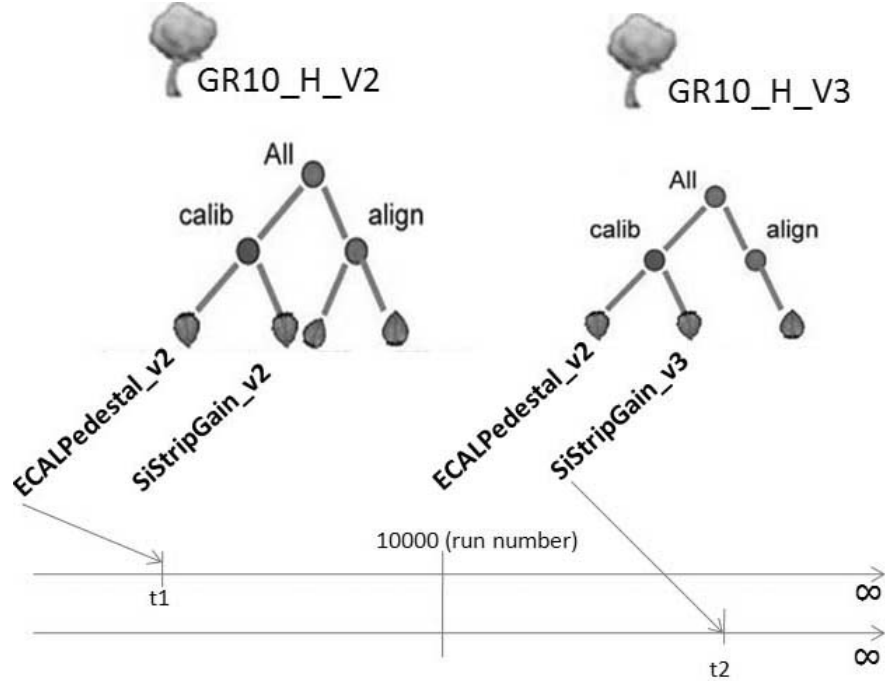


Figure 3.3: Conditions database data model.

generated including the new conditions together to the old conditions until to the time $t_2 - 1$.

At present, the Global TAGs used for data taking are organized in four separate groups:

- GR10_H_V* for the *High Level Trigger* (HLT),
- GR10_E_V* for the express reconstruction,
- GR10_P_V* for the prompt reconstruction,
- MC_V* for Montecarlo.

3.3 The CMS database

Before describing the CMS database design, we will describe the networking infrastructure of the CMS experiment where the database and web service (see Chapter 4,

5) are delivered.

3.3.1 Networking Infrastructure

The general networking infrastructure of the experiment is based on Ethernet and is separated into:

- CERN general purpose network (CERN-GPN)
- CMS Technical Network (CMS-TN)
- Connection to the Central Data Recording (CDR)
- LHC technical network (LHC-TN)

The CERN-GPN is the CERN-site network. It is used at all places to connect visitors (wired or wireless), who will use the application gateways to connect to the CMS-TN. This network typically provides 1 Gbit/s connections to the GPN backbone.

The CMS-TN is a general purpose network connecting all machines directly related to the operation of the experiment. It is used for system administration of all computers and for configuration, control and monitoring of all online applications. This network is not accessible directly from outside in order to protect the CMS site from threats from the internet. It can be reached through dual-homed Application Gateway (AG) machines, which have one connection to CMS-TN and one to CERN-GPN switches.

The CMS-TN is implemented as a distributed backbone with two routers located in a building in the surface (SCX5), and two in the underground electronics room in a cavern (USC) approximately 80 meters below the ground level. Typically all computers in a rack are served by a switch in that rack. Each switch has two Gigabit Ethernet uplinks to the backbone routers [30]. The desktop computers in the control room are also connected to the CMS-TN.

The CDR connects to the Tier-0 system at the CERN Meyrin site using a minimum of 10 Gbit/s. A set of 8 triple-homed servers (one connection on the *Data Acquisition system* (DAQ), one to CERN-TN, one on the CDR switches) are dedicated to this task. These are the Storage Manager nodes.

The LHC-TN allows data exchange between some of the CMS equipment and the LHC controls. The CMS-TN interconnects with the CERN-GPN and the LHC-TN using a filter implemented in the backbone routers.

3.3.2 The CMS database - Design Requirements

During the construction phase, many databases using different technologies and software were developed by the sub-projects in order to store all the detector and equipment data. In 2007, the CMS collaboration decided to start a common and central project, called the *CMS Database Project*, in order to converge towards an unique database technology and a set of central software tools supported by the experiment for all data taking. The current layout of the database model and data-flow was developed after two workshops and one review and close interaction with the CMS subsystems. The three most important requirements identified by CMS are:

- CMS has always to be able to operate without network connection to the outside world. Therefore, CMS must possess an independent DBMS infrastructure based at LHC Interaction Point 5 (IP5) to ensure CMS operation.
- Access to condition data required for off-line reconstruction shall hide the underlying database technology to the user. The access mechanism should be steered with a simple, C++ based, interface to facilitate the usage of the off-line condition service (i.e. no specific DB knowledge is required)
- The off-line condition data work-flow has to fit a multi-tier distributed structure as used for event data.

In order to comply with these requirements, the *CMS Database Project group*, with the help of CERN-IT Division and in collaboration with all the CMS sub-projects, designed a system based on 3 different databases, all based on the ORACLE technology (see Figure 3.4).

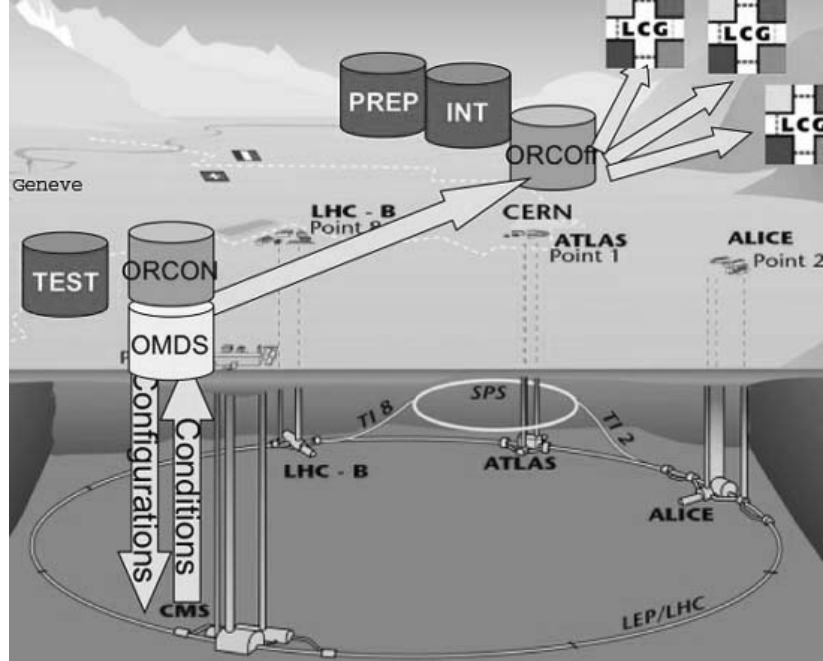


Figure 3.4: Condition databases architecture.

3.3.3 Online Master Data Storage

The Online Master Database System (OMDS) is the online database located at IP5 on the CMS online network (CMS-TN). It stores the configuration of the detector and the non event data produced by the sub-systems like slow control, electronics, DAQ and trigger data. A subset of the online data is streamed into the off-line database for event reconstruction.

Figure 3.5 shows the data flow in the CMS databases. It is a purely relational database. The data size is becoming 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. To fulfill these requirements, each sub-detector has designed its own database schema, reflecting as far as possible the detector structure.

The average amount of data stored in OMDS during 30 days data taking amounts to about 500 GB (see Figure 3.6). In particular, for example the Condition data coming from the *Electromagnetic Calorimeter* (ECAL), amounts to about 5 GB of data per day.

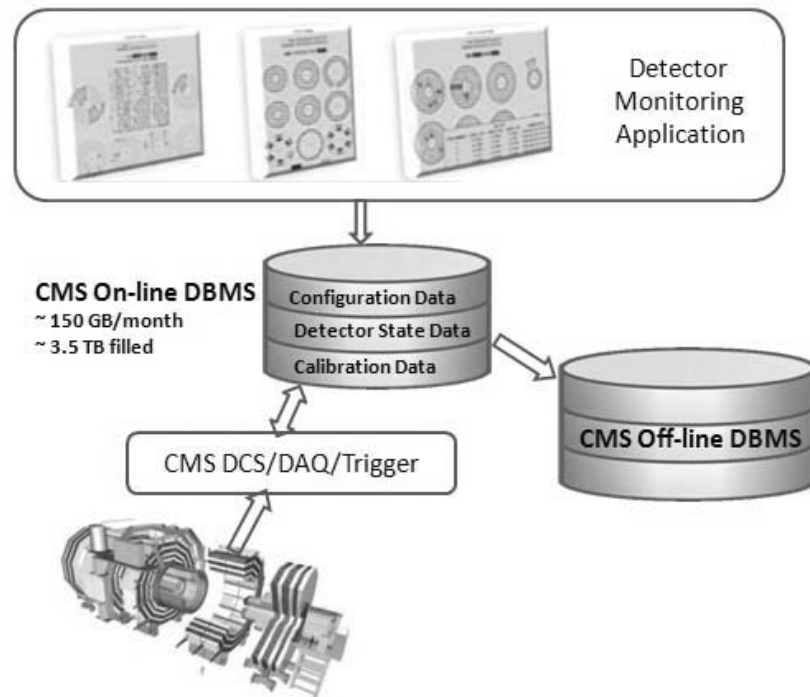


Figure 3.5: Data flow in the CMS databases.

3.3.4 Off-line Database

The off-line database is represented by the Off-line Reconstruction Conditions DBMS and it consists of two subsets: ORCON and ORCOFF (Off-line Reconstruction Conditions database OFF-line subset). The *Off-line subset* is located in the CERN-IT department facilities building in direct proximity to the *CMS Analysis Facilities* and the *Tier 0*; this allows prompt access to conditions data at the reconstruction step for data arriving from CMS. The *On-line subset* is physically located in the CMS underground cavern allowing a fast link to the OMDS. It has exactly the same internal structure as ORCOFF; thus from the software point of view, writing to ORCON is equivalent to writing directly to ORCOFF.

ORCON and ORCOFF are centrally synchronized, thus preventing each separate subsystem from writing straight to ORCOFF, and providing consistent data for off-line reconstruction. To fill ORCON/ORCOFF we use the *Pool software* (see Section 3.5.2) which fills the database directly from the c++ objects. Each entity of the object has an

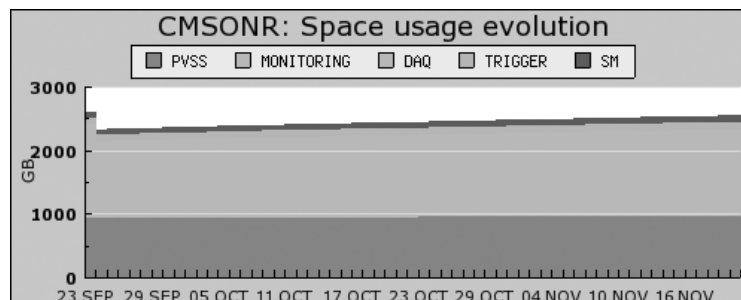


Figure 3.6: Space usage evolution in OMDS from 23/09/2010 to 22/11/2010.

IOV assigned to it. Access to the data is organized so that only the objects whose IOV contains the current time can be retrieved, thus providing conditions which were valid for that specific moment in detector history.

Reconstruction Condition DB - Online subset

The *Reconstruction Condition database for ON-line use* (ORCON) is placed on the online network (CMS-TN). It stores all the condition data required by the HLT and for off-line reconstruction. It also contains conditions needed off-line for data quality indicator and for more detailed off-line analysis. The data contained in it are written using the POOL [56] technology (see Section 3.5.2) and are retrieved by the HLT programs as C++ objects for the off-line software. Quota usage for Condition database user's account are shown in the Table 3.4.

Reconstruction Condition DB - off-line subset

The *Reconstruction Condition database for off-line use* (ORCOFF) is the master off-line database located at the Tier-0 site (CERN Meyrin). It contains a copy of ORCON made through ORACLE streaming. Data contained in it are retrieved by the reconstruction algorithms as C++ objects for the off-line software.

Account	Quota (MB)
CMS_COND_31X_ALIGNMENT	192
CMS_COND_31X_BEAMSPOT	117
CMS_COND_31X_CSC	683
CMS_COND_31X_DT	2400
CMS_COND_31X_ECAL	16956
CMS_COND_31X_FROM21X	7900
CMS_COND_31X_FRONTIER	417
CMS_COND_31X_GLOBALTAG	155
CMS_COND_31X_HCAL	1998
CMS_COND_31X_L1T	1432
CMS_COND_31X_PIXEL	1481
CMS_COND_31X_PRESHOWER	106
CMS_COND_31X_RPC	5623
CMS_COND_31X_RUN_INFO	727
CMS_COND_31X_STRIP	3473
Others	46062
Total	89722.0

Table 3.4: This table displays the storage used per CMS condition accounts currently in production on CMS off-line rack on 23th November, 2010.

Data Flow in the CMS database

The general non-event data flow (see Figure 3.4) can be described as follows: every subproject calculates and measures in advance 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. Different hardware setups can be stored at the same time in the configuration database, but only one will be loaded before the run starts. During data taking, the detector produces many kind of conditions, to be stored in the online database called OMDS, from the slow control (PVSS ¹) and from other tools like DAQ (XDAQ²), Run Control and DQM. Part of OMDS data, needed by the HLT and off-line reconstruction, will be transferred to the

¹PVSS is the Slow Control And Data Acquisition system (SCADA) adopted at CERN for the controls of the four experiments at LHC.

²XDAQ is a generic data acquisition software environment.

database called ORCON. The software application named PopCon (Populator of Condition Objects) operates the transfer of the condition data from the OMDS to ORCON database and encapsulates the data stored in relational databases as C++ objects. PopCon adds meta-data information to non-event data, so that they can be retrieved by both the HLT and the off-line software. In addition to condition data transferred from OMDS to ORCON by the *O2O* procedure, calibration and alignment data determined off-line are also written to ORCON using PopCon. Finally, data are transferred to off-line database named ORCOFF, which is the main condition database for the CMS Tier-0, using ORACLE streaming.

3.3.5 FroNTier

The CMS detector generates too much particle collision data to be processed all at one site, so the processing is distributed around the world. All of 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 varies but is on the order of 100 Megabytes per processing job. The caching subsystem that CMS uses for conditions data is called FroNTier [31]. FroNTier 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*. One of the most challenging aspects of any cache system is coherency, that is, ensuring that changes made to the underlying data get propagated out to all clients in a timely manner.

3.3.6 Service Architecture

Oracle 10g Real Application Clusters (RAC) and Automatic Storage Manager (ASM) on Linux has been chosen as the main platform to deploy the database services for physics. Each database cluster consists of several processing nodes accessing data shared in a storage area network (SAN). The cluster nodes use a dedicated network to share cached data blocks to minimize the number of disk operations. A public net-

work connects the database cluster to client applications, which may execute queries in parallel on several nodes. The setup provides important flexibility to expand the database server resources (CPU and storage independently) according to user needs. This has been particularly important during the early phases of LHC operation since several applications are still under development and data volume and access patterns may still change.

3.3.7 Service levels and release cycles

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. This structure is in place since a few years and has proven to be very effective for assuring that only good quality applications be deployed in production, with a clear benefit for the stability of the production services. The validation/pre-production and the production services are deployed on Oracle 10g Real Application Clusters (RAC) on Linux. Both the validation and production services are deployed on similar hardware, with the same configuration, OS and Oracle server versions.

The development service

The development service is a public service, meant for pure software development. A limited database space is available and no backups are taken for the pre-production service. Once the application code is stable enough and needs larger-scale tests, it moves to the next service level as described below. In the off-line CMS experiment the database used for this purpose is called ORCOFF_PREP (off-line database for preparation).

The validation/pre-production service

We refer to this service as the pre-production service. It should be noted that no backups are taken for the pre-production service. In the off-line CMS experiment the database used for this purpose is called ORCOFF_INT (off-line database for integration). It should be used if all the tests on ORCOFF_PREP have been successful.

The production service

The production service is characterized by full monitoring, availability and backups. For deploying an application in production, it is necessary to have validated it or define the use of reader/writer accounts/roles, the expected data volume, number of simultaneous connections and total number of connections (see Figure 3.7), in the case it did not go through the validation procedure.

Best practices, tips and guidelines are provided by the *Database Administrator* (DBA) team to help database developers. In addition, the experiments database coordinators receive a weekly report that summarizes the database activities in terms of sessions, CPU, physical and logical reads and writes (see Figure 3.8). The production database for the CMS experiment is called ORCOFF.

3.3.8 SQLite

As said in previous sections, the system provides three levels of Oracle services: production, integration and development. Additionally, users can also use SQLite [36] databases for testing and development. SQLite is a C library that provides a lightweight disk-based database that doesn't require a separate server process and allows accessing the database using a nonstandard variant of the SQL query language. In the context of 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 real DBMS. Thanks to the CORAL abstraction layer, importing/exporting data across different database back-ends is fully supported. One can also read a mix of data from any of these sources in the same CMSSW.

USERNAME	OS_USERNAME	USERHOST	SESSIONS
CMS_COND_31X_POPCONLOG			8511
CMS_TRANSFERMGMT_WRITER	cmssgm		9020
CMS_COND_FRONTIER_LUMI_R			9292
CMS_SITEDB_WRITER	_frontend		9348
CMS_TRANSFERMGMT_SC_WRITER	_phedex		9623
CMS_COND_FRONTIER_R			9734
CMS_CENTRE			9861
CMS_CENTRE		cmscentre	9861
CMS_CENTRE			9862
CMS_SITEDB_WRITER	_sitedb	vocms106.cern.ch	10228
CMS_TRANSFERMGMT_SC_WRITER	phedex		11737
CMS_TRANSFERMGMT_TEST_WRITER	phedex		12381
CMS_COND_FRONTIER_LUMI_R			13634
CMS_TOAST_2_READER	_t0datasvc	vocms106.cern.ch	14529
CMS_SITEDB_WRITER	_phedex	vocms50.cern.ch	16196
CMS_SITEDB_WRITER	_sitedb		17249
CMS_SITEDB_WRITER	_phedex		18088
CMS_TOAST_2_READER	_t0datasvc		19494
CMS_TOAST_2_READER			20458
CMS_TRANSFERMGMT_WRITER	_phedex	vocms50.cern.ch	21108
CMS_TRANSFERMGMT_WRITER	_phedex		26692
CMS_TRANSFERMGMT_TEST_WRITER			28831
CMS_TRANSFERMGMT_WRITER	phedex		36034
CMS_COND_GENERAL_R	pierro	popcon2vm	36118
CMS_TRANSFERMGMT_SC_WRITER			39533
CMS_COND_GENERAL_R	pierro		44357
CMS_SITEDB_WRITER			45288
CMS_COND_GENERAL_R			46550
CMS_TRANSFERMGMT_WRITER			106168
TOTAL			340495

Figure 3.7: Showing only the 20 pairs user/host with most sessions between 06-Dec-2010 00:00 and 13-Dec-2010 00:00.

Top CPU usage (cpu hours)				Top physical reads (GiB)			
OS_USER	HOST	USERNAME	CPUH	OS_USER	HOST	USERNAME	PREADS
cmsprod	vocms69.cern.ch	CMS_TOAST_2_WRITER	263	cmsdbs	vocms74.cern.ch	CMS_DBS_PROD_GLOBAL_READER	19776
cmsdbs	vocms74.cern.ch	CMS_DBS_PROD_GLOBAL_READER	204	phedex	vocms02.cern.ch	CMS_TRANSFERMGMT_WRITER	14562
phedex	vocms02.cern.ch	CMS_TRANSFERMGMT_WRITER	179	cmsdbs	vocms73.cern.ch	CMS_DBS_PROD_GLOBAL_READER	11622
cmsdbs	vocms73.cern.ch	CMS_DBS_PROD_GLOBAL_READER	102	phedex	vocms03.cern.ch	CMS_TRANSFERMGMT_SC_WRITER	11208
phedex	vocms03.cern.ch	CMS_TRANSFERMGMT_SC_WRITER	66	cmsprod	vocms69.cern.ch	CMS_TOAST_2_WRITER	8300
Top writes (GiB)				Top logical reads (GiB)			
OS_USER	HOST	USERNAME	LWRITES	OS_USER	HOST	USERNAME	LREADS
phedex	vocms03.cern.ch	CMS_TRANSFERMGMT_TEST_WRITER	11369	cmsprod	vocms69.cern.ch	CMS_TOAST_2_WRITER	1483201
phedex	vocms03.cern.ch	CMS_TRANSFERMGMT_SC_WRITER	5456	phedex	vocms02.cern.ch	CMS_TRANSFERMGMT_WRITER	332824
phedex	vocms02.cern.ch	CMS_TRANSFERMGMT_WRITER	3920	_t0datasvc	vocms106.cern.ch	CMS_TOAST_2_READER	291753
?	dbsrvg3512.cern.ch	CMS_HLT_R	1071	phedex	vocms03.cern.ch	CMS_TRANSFERMGMT_SC_WRITER	133666
cmsdbs	vocms73.cern.ch	CMS_DBS_PH_ANALYSIS_01_WRITER	534	_t0datasvc	vocms107.cern.ch	CMS_TOAST_2_READER	74594

Figure 3.8: CPU usage, physical and logical reads and writes between 06-Dec-2010 00:00 and 13-Dec-2010 00:00.

3.4 Data sources and consumers

Sources and consumers of condition data in CMS are quite different, going from pure on-line monitoring to the off-line calibration processes. The information written and retrieved from the databases (both on-line and off-line) can be separated out in two main categories:

- Monitoring information needed to check the status of the various sub detector components, like temperatures, high voltage and low voltage status, etc.
- Production information needed for production environment, including pedestals, gains, hot/dead channels, calibration constants, etc.

As an example, few of those are spotted in the following.

3.4.1 Data sources

On line software - XDAQ

Under this heading sit all configuration data which are stored in the on-line database (and possibly exported to the off-line database) which comes from either global or local DAQ runs. Most of them come from dedicated “calibration” runs which are foreseen for every sub detectors, in particular for pedestals and gain calibrations. From the requirement collection two specific tasks, from the Pixel detector and from the ECAL respectively, are the most demanding in terms of data size and frequency:

- **Pixel gain calibration:** The data are produced by special detector calibration runs (charge injection), taken between LHC fills, approximately once per week. Processing of raw event data takes place off-line, as a prompt calibration. An option exists to perform some averaging of the raw data in real time on the filter farm during the calibration run. This reduces the data volume transferred from 50 GB (for 20 pulse heights x 10 events per pixel) to 10 GB. A special event data format is used in this case. The pixel gains are parameterized by a simple fit over the linear and saturated regions. The calibrations produced by off-line

analysis, using the *off-line DropBox service*, are then inserted into the ORCON and, then, transferred to ORCOFF.

- **Laser transparency monitoring:** The laser run take place during physics run using the abort gap. There is approximately 1 run every 30 minutes: the laser cycles through the super-modules and the End-Caps. Providing data for 1 super-module every 30 minutes. The Laser analysis task is conceptually divided in 2 sub-tasks: a first program analyzes the Laser events and calculates average quantities for a given laser run crystal by crystal to be stored in the DB. A second Laser analysis task uses the results of the first computation and their time evolution, plus other Detector Control System (DCS) information and the data from the CCS³ like Temperatures etc. and calculates the crystal inter-calibration corrections that are needed for the off-line analysis.

Run control and monitoring system

At the starting of a physics run, and during data taking, the status of the single sub detectors is monitored and recorded in the on-line database as conditions sets. Different monitoring mechanism are used, from pure DAQ on-line monitoring of the electronics to slow control following the DCS/PVSS path. An example is given by the ECAL case where the CCS supervisor accesses the front-end cards via the Token Ring to configure them and to read the temperatures, voltages and APD (Avalanche Photo Diode) dark currents. These values are then written in the Condition DB. In parallel these data are sent to a PVSS application that displays some averages of these readings for the shifters. All the data are available in the condition DB.

Data quality monitoring

Specific monitoring tasks for the quality of the data are also foreseen as sources for the on-line and off-line conditions databases. An example is given by the Tracker where there are specific DQM processes running on-line and computing the configuration parameters for the readout electronics to be stored in the configuration database. Other

³Clock and Controls Supervisor. Reads non event data in some sub detectors.

DQM processes are foreseen to monitor the calibration constants which are computed off-line and used for HLT operations (i.e. stored in HLT). The DQM processes running at the interaction point have the possibility to gather information not only from the dedicated data stream but also from DCS and on-line. Statistical, as well as historical information might be retrieved off site using the visualization method of the **Payload Inspector** (see Section 4.5.2). In some cases, see Figure 3.10, the off-line database is used as source of historic monitoring plots.

Off-line calibration software

Under this heading is considered all software which runs “off-line”, with no interaction with the on-line database. It can go from the complex pedestal calculation mechanism of the HCAL, which needs resources not available on-line, to the pure inter calibration of the crystal of the ECAL using physics events like $W \rightarrow e\nu$ or $Z \rightarrow e^+e^-$. In general the constants calculated off-line (either at IP5 or at Tier-0), are needed only for (prompt) reconstruction and as such need to be stored in the ORCOFF database (see Chapter 2).

3.4.2 Data consumers

Most of the above listed processes are naturally also consumers of database info, mainly for what concerns conditions data like pedestals, gains, calibrations and monitoring. In the following a different categorization is made based on specific use cases. Two main categories are identified: *production use cases* which are vital for the production environment, and *monitoring use cases* needed to perform detailed/complex monitoring of the sub detectors (see Chapter 4).

Production use cases

- **On line Pedestals/gains, hot/dead channels:** this information is clearly used both for on-line operation, for example in configuration of the electronics board, and off-line for a proper reconstruction of the data. As such this information,

whose source are diverse, are usually stored in the on-line database and propagated with the proper IOV to the off-line databases.

- **Off line event based calibration:** as consumers, the off-line (prompt) calibration processes use status info of the single sub detectors and previous versions of the same calibration constants as starting/monitoring points.
- **Special “event” samples:** specific cases where non-physics events, for example hardware alignment “runs” must be processed accessing at the same time event-like structures originated by local or global DAQ with pure monitoring information from DCS system (sensors, temperatures, etc.). A similar example, with the same requirements, can be individuated in the ECAL transparency monitoring. For this particular cases, the “consumer” job might sit directly in IP5, having access to all information from the databases, or off site. In the latter case, a transfer mechanism is put in place, allowing to expose the needed information in a transparent way to be analyzed in CMSSW using the *data analysis framework ROOT* (ROOT) format [32].

Monitoring use cases

- **Pure monitoring:** Many database info will be monitored to check the detector conditions using different mechanisms. The monitoring can be performed completely on-line, using the DQM system, and exporting final histograms/-value off site using a server-client approach. In other cases, the monitoring has to be performed on off-line calculated quantities like calibrations and “off-line” pedestals. In this case, the info to be monitored are stored in the off-line database and can be accessed through the Payload Inspector (see Section 4.5.2).
- **Correlation between real data and monitoring:** In some cases information from databases have to be correlated with real event data. Use cases are represented by complex monitoring, requested by Tracker, HCAL and ECAL sub detectors, where (slowly) changing physics observables need to be checked against low level monitoring quantities like high voltage, low voltage and temperatures.

The major concern is about cases where these correlations are unforeseen at the beginning of the data taking, so that no specific monitoring is in place. In order to fulfill the most general case, a web application has been put in place with root export mechanism, and will be discussed in Chapter 4.

3.5 Condition Database Data model

3.5.1 Introduction

This section gives an overview on the off-line software framework and services for conditions data management in the CMS experiment. The CMS model distinguishes two types of time variant condition data: those taken during the operation of the experiment for monitoring purposes and those used in off-line reconstruction and analysis or results from calibration and alignment algorithms. In this section we focus on the data of the latter type. The data model and the base technology choices will be discussed.

For the system to be most effective, different usages are identified and the core software components and services are designed to satisfy requirements from each different type of user activity. How such a design goal is achieved will be explained along with the description of the software components and services. The computing and deployment model of the system will be briefly described.

3.5.2 Persistency framework

The database play a central role, especially for conditions and event metadata, because they can provide consistent storage to many concurrent users.

Given the complexity of the current condition database systems, it is often difficult for users to exploit the underlying services in the most efficient way. However, many of the required optimizations can be delegated to an intermediate software layer if both the main physics use cases and service constraints are taken into account early on in its design.

The *persistency framework* for the LHC Computing Grid (LCG), which is being developed jointly between the LHC experiments and the *Physics Services Support*

group (IT/PSS), aims to provide such a software layer. Its purpose is to decouple the user code from the features of any particular database implementation.

The project started in 2002 in the LCG Applications Area. It produced the common persistency framework for the LHC experiments, namely POOL. This name is an acronym that stands for **P**ool⁴ **O**f *p*ersistent **O**bjects for **L**HC,

3.5.3 POOL Object Relational Database Access: POOL-ORA

The *Object-Relational database in POOL* (POOL-ORA) [33] is the baseline technology for the CMS off-line conditions database software. The POOL-ORA implementation sits on top of a generic relational database layer that hides the underlying relational database technology from the user. The POOL-ORA interface used for handling non-event data is identical to that of POOL-ROOT used for handling event data, and thus a change of one input parameter is all that is required to switch between the ORA and the ROOT implementation. The off-line database schema is automatically created from the definitions of the persistent objects by following the *Object Relational Mapping* (ORM) rules. The data are retrieved from the underlying relational database and materialized as C++ objects in memory.

POOL-ORA is the relational back-end of POOL and therefore is part of the general POOL environment. POOL-ORA consists of three domains:

- *COMmon Relational Access Layer* (CORAL): it defines a vendor independent *Application Programming Interface* (API) for relational database access, data and schema manipulation. Three technology plugins, `Oracle` [34], `MySQL` [35] and `SQLite` [36] are released together with the API.
- *Object Relational Access* (ORA): it implements the object relational mapping mechanism and is responsible for transforming C++ object definitions to relational structures and vice-versa.
- *Relational Storage Service*: it implements the POOL Storage Service using the Relational Access and the Object Relation Access components.

⁴A pool in computer science is a set of initialised resources that are kept ready to use, rather than allocated and destroyed on demand.

CMS contribute to development of the POOL-ORA infrastructure to ensure that it satisfies the requirements from CMS. For this reason CMS has established a close collaboration with corresponding POOL developer team.

3.5.4 Data model and base technology choices for the CMS condition data

As already pointed out, condition data change with time and this is described by the IOV. In order to access to the data of interest, the “time” need to be specified. Time in this context is represented by either *run number*, *universal timestamps* or *luminosity section id*. A Relational Data Base Management System (RDBMS) is suitable for handling concurrent random access to a centralized storage. CMS online system uses RDBMS as condition data store [37]. For these reasons RDBMS is the choice for conditions data persistency at off-line level as well. On the other hand, the Object-Oriented nature of the off-line software requires the data to be delivered as objects. The relational storage component of the LHC persistency framework POOL has not only the capability of mapping objects to relational tables, but also the required relational data abstraction through a database abstraction layer CORAL. Therefore POOL and CORAL constitute the basic components for CMS off-line handling of conditions data.

The bulk data, also called the payload, are C++/POOL objects and can be accessed via their IOV. In the CMS model, the IOV sequence is always consecutive with no holes nor overlap in time. An IOV sequence is a C++/POOL object which contains the mapping of time to externalized POOL tokens of the payload objects of the same type. In the more recent performance enhanced design, an IOV object can contain also the summary of the payload it refers to. IOV can be versioned and the name of the version is called the IOV TAG. IOV TAGs are the lowest level of access to the data. Knowing the TAG, one arrives at the valid payload object for a given point in time by finding the corresponding time period in the IOV and following the POOL token [38].

3.5.5 Core Software and Services

For the software to be effective and fit the genuine needs of users, the overall design of the system focuses differently on different types of user activities.

In CMS off-line processing, users access the detector conditions through the generic *EventSetup* part of the Event Data Model (EDM) framework [39]. The Event Data Model is the set of structures for raw and reconstructed data, typically stored within some larger shared data structure and the associated user interfaces and utilities needed to locate, manipulate and store the data. *EventSetup* provides a snapshot of the detector conditions at an instant in time. *PoolDBESSource* is a specific *EventSetup* data provider which delivers the data objects to the *EventSetup* and guarantees the time validity of the delivery. *PoolDBOutputService* is a load-on-demand EDM service that writes out conditions data and associates IOV to the data. As the names of these modules imply, they are built on top of POOL and CORAL libraries. POOL is responsible for the mapping of objects to tables as well as the object input/output while CORAL is the generic RDBMS abstraction layer. These components together, as shown in Figure 3.9, ensure the conditions data I/O is an integral part of the event processing and the user awareness of the difference in *condition data* and *event data* handling is minimal.

No database APIs nor queries are exposed and thanks to the *EventSetup* design, users do not need to learn any conditions or IOV management API. The tool in charge of managing data transformation from external to the POOL format and data transfer from external sources to the off-line databases is called PopCon (see Section 4.2). It has the integrated capability of data transfer logging and IOV consistency check.

Apart from the standard HEP event processing usage described above, we recognize the need of software utilities for data management users. Different from the event processing jobs, production managers do not deal with payload data, they organize production jobs via global TAG management and data transfer between different database instances and different back-end technologies. Since tools for the data management are independent of the event processing, Python is chosen as the base language because its simplicity and flexibility. It is interpreted, general-purpose high-level programming

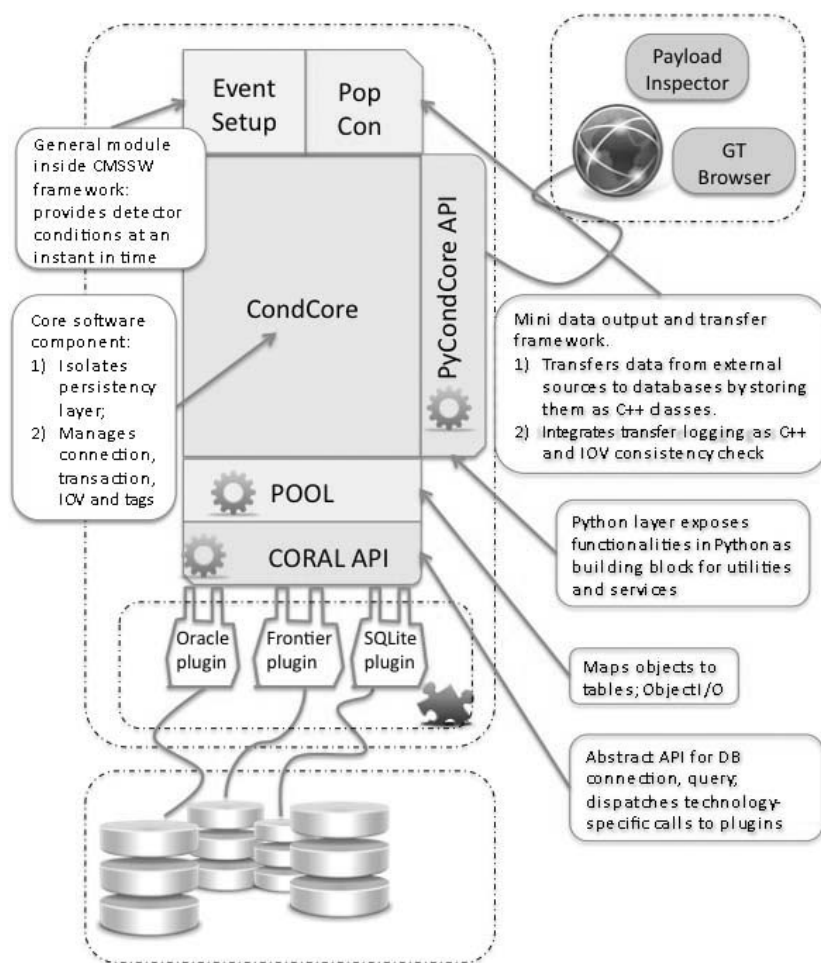


Figure 3.9: Software Components.

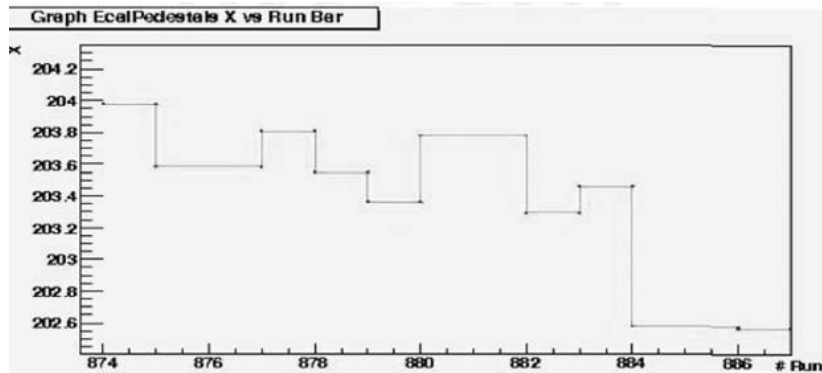


Figure 3.10: ROOT plot of Pedestals value versus run number.

language whose design philosophy emphasizes code readability.

All the global TAG management software are written directly in `Python` while at lower level, database connection and transaction management and query abstractions are exposed to `Python` via `PyCoral` which is the `Python` binding layer of `CORAL`.

Finally, the third kind of use case is similar to data mining where physicists browse TAGs, payloads and summary data searching for patterns and anomalies. We satisfy the requirements from such usages by providing various services. From the conditions database web service, one can browse the TAGs and make plots of the payload and the summary data. Similar to data management tools, the web service is written in `Python`. IOV, payload and summary data inspection capabilities are exposed to `Python` via a `C++/Python` binding layer of the core components. Given the nomadic nature of this kind of users, the web service is packaged in the way that it can be deployed centrally as well as on standalone personal computers. We also provide a conditions ROOT browser supporting interactive IOV manipulation, data inspection and display from the ROOT prompt. Figure 3.10 shows one example ROOT plot of ECAL pedestals value as a function of run number.

3.6 Performance

3.6.1 Introduction

The core component development for the Condition Data handling started in 2006. A prototype system was successfully tested for the first time in Computing-Software-Analysis challenge 2006 (see Section 1.4). While evolving, the system has been used routinely in all the global runs, cosmic run and Monte Carlo productions subsequently. By now the core components have reached complete functionalities. In order to improve performance and to increase the speed and improve the memory usage for interactive payload data inspection, some enhancements in the data model were introduced in recent releases together with corresponding modifications and new data types in POOL. The service development started in 2007.

Due to very different requirements for the various detectors, the collaboration agreed on the principle that each sub-detector group developed its own database schema and services for its control and monitoring. In this section we discuss:

- performance considerations.
- Data archiving strategy for the online database, in order to improve the data access performance of very large tables.
- The technique of using Binary Large Object (BLOB) for storing and accessing condition data.
- The ECAL Database schema, which, according the Table 3.4 is the most challenging sub-detector to be optimized in terms of performance because of amount of data.
- The *bit-packing optimization* applied to the data condition of Silicon Strip Detector.

3.6.2 Performance Considerations

During the design of the schema, performance and scale tests were done to ensure that the application would satisfy the long-term requirements for each sub-detector. For each type of data, growth rates of the data are constant and well-known, enabling us to simulate the growth of the database. It was assumed that the database would have to store data for one year worth of CMS running before older data would be archived. Knowing the limits to the database size allowed us to focus our performance testing.

Several different schemas were tested to learn how read and write performance would scale as a function of number of channels or number of IOVs in the database. These schemas differed in how the IOV timestamps were stored and indexed, and how the payload data would be stored. Options for IOV organization included inline with the data, or in a separate table with an integer ID (the winning method). Options for payload storage included in table columns with channels-in-rows, all channels together in CLOB (Character Large Object) or BLOB columns, and channels-in-columns format. CLOB and BLOB are Oracle data type that can hold up to 4 GB of data. Two access patterns were tested: selecting all channels for a type of data given a *UNIX timestamp* (T), and selecting a single channel at time T . The matrix of possibilities above was exercised and guided the design of the current schema.

These performance benchmarks were absolutely crucial to developing a schema that would work for our application monitoring and handling the condition data (see Chapter 4). As an example, the first iteration of our schema for storing the pedestals was found to take 20 seconds to query all sub-detector barrel channels (61200) at time T with only 10 days worth of sub-detector running stored. This result scales linearly with time, leading to the conclusion that if we had one year of data the query would take 12.2 minutes to execute. It may seem surprising that such a schema could even be considered, but at the time we were only working with small portions of the detector (3600 channels) and small databases, so the scaling nightmare was not immediately apparent. With the results of the performance testing, we arrived at a schema which could do the same query 55 times faster when executed on a database with 1,000 IOVs, and more importantly the scaling was constant with more IOVs. Calculations based

on the scaling factors obtained in testing showed that the same query on a one-year database could be done in 0.9 seconds, which is nearly 800 times faster than what we started with.

3.6.3 Data Archiving strategy

The main challenge for the CMS online database is to efficiently archive and provide fast access to large data volumes of parameters. In the context of CMS community, the most used are the most recent values. History data are accessed only occasionally. Since history tables are big, it would be difficult to serve the most recent values extracting them from the history table without facing performance issues. For this reason, we provide Last Value (LV) as separate tables, which contain only the most recent values. The history tables often contain millions of records whereas the LV tables mostly contain only thousands of rows [40].

In the case of OMDS database, a LV table is created automatically for a particular channel type (class of channel/device) and is updated every time new data are received from the Detector Control System (DCS). In addition to LV tables a redundant data filtering mechanism has been implemented. When the state of some parameter changes in the detector, DCS passes it to the PL/SQL procedures, which decide if and where the received data should be stored in the CMS DCS online database. The new value is compared with the previous one in the LV table. If they are equal (or the difference between old and new value is not greater than a predefined dead band), it is not stored. The LV mechanism prevents the database being polluted by redundant data from mis-configured applications. The data filtering procedures are part of the extended CMS DCS DB schema.

Even if the most requested data are provided by LV tables, all the historical data must be kept. The tables which hold these data can easily grow to millions of lines. Despite of that, a good query performance is required. But fast access to such tables is not trivial. Oracle partitioning allows to avoid unnecessary table/index scan operations by performing SQL operations only on a subset of the total data in a table. Thus partitioning increases the query performance. The queries to access historical data,

always include a time range predicate. Therefore time range partitions were created on the most crucial tables. When a new record is inserted, it is automatically assigned to a particular partition, depending on the partition key value. When the data is accessed, only partitions corresponding to the provided time period are scanned (see Figure 3.11).



Figure 3.11: Access to the partitioned table. The lighter color indicates the partitions which are scanned.

3.6.4 ECAL use case - Database performance optimization

Parameters connected with the Laser Monitoring System operation, referred to as Laser Primitives, represent a large amount of data stored for each individual ECAL crystal, leading to a considerable time consumption for data transfer. Access tests showed that using the standard upload procedure is not satisfactory for the required working conditions and a DB Access Optimization should be performed.

To improve performance, standard *Oracle C++ Call Interface* (OCCI) optimization techniques were used [41].

In one update, all 61200 values contained in an array are written to the database, and in one transaction 1000 lines are retrieved; this allows significant reduction of multiple network round trips to the server.

For testing purposes, we retrieved from and filled into the Online DB, data of a full transparency measurement cycle of the ECAL Barrel, i.e. 1700 x 36 (61200) channels.

For each channel we read/write the amplitude of the laser source used to monitor the stability of the crystals (APD, PN, APD/PN) from/in appropriate tables [42]. The results before and after optimization are summarized in Table 3.5.

	Before optimization	After optimization
Writing	75 sec	2 sec
Reading	22 sec	3 sec

Table 3.5: Online DB access optimization results.

3.6.5 The Silicon Strip Tracker Detector use case

The *Silicon Strip Tracker* (SST) condition data are subjected to a format manipulation to provide the best suitable structure for the reconstruction. Actually, the processing speed and the memory footprints are critical aspects of the reconstruction applications. The highly granular information available in the SST condition data can significantly contribute to the run-time performance of the process. In order to speed up the data loading from the databases and to reduce the memory allocation, the calibrations constants are structured in packed format (see Section 3.6.5) and transferred as Binary Large Objects (BLOB). For instance, this strategy reduces the size of the strip noise data from the 40 MB that would have been necessary with an uncompressed format to 11 MB, preserving the same measurement precision. The calibration constants evaluated in the SST commissioning phase and systematically transferred in the configuration databases are the strip pedestal and noise values, the set of disabled components, the thresholds of the Zero-Suppression.

Bit-packing - Motivation

Very often it happens that the information can be represented with a single bit or at most using few bits. However since microprocessor work with bytes (8 bits) or words (16 bits), even when the information can be stored in just one bit the full byte or, even

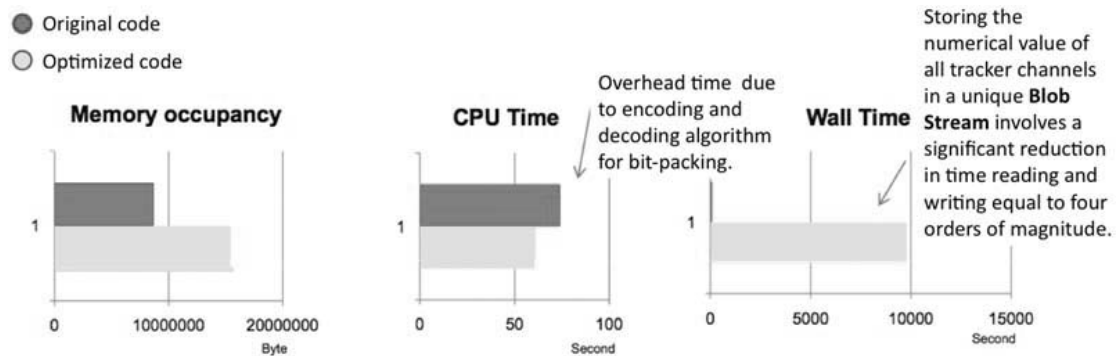


Figure 3.12: Off-line database performance metrics using *IgProf* and *profiler gprof*.

worse, the full word is used, filling the extra bits with zero. Clearly this corresponds to a big waste of computer resource and communication bandwidth, especially, when these kind of data need to be processed with very strict time-constraint as part of the calibration and alignment work-flow as described in Chapter 2.

To minimize such a problem, we proposed this bit packing optimization for SST data. The idea is very simple: we try to put many significant bits into an integer (16 bit) instead of just few of them, depending on a data quantity (pedestal, noise, etc.). This idea is especially feasible for SST data because of:

- fine granularity of the SST detector that implies a considerable volume of data,
- the maximum offset of pedestal and noise level of each individual electronic channel required just 9 and 10 bit respectively, which implies 7 and 6 wasted zero bits in each word transmitted.

This also addresses the communication problem since if we manage to pack an entire data set of conditions for all channels of the detector, then we can send the bit packed integers instead of individual bits as full words thus decreasing the communication requirements and especially increasing the performance for insert and read operations to off-line database. Off-line database performance metrics using *IgProf* and *profiler gprof*, two tools for measuring and analysing application memory and performance characteristics, are shown in Figure 3.12.

Bit-packing, example

If, for example, the maximum offset (or index) is 50 it can be represented by 6 bits and there will be 2 wasted zero bits in each byte transmitted. It would be more efficient to pack the offsets 6 bits at a time; 4 6-bit words would be packed into 3 8-bit bytes. Bit-packing also allows for the possibility of going beyond 8 bits. If, for example, the maximum offset (or index) is 500, as in the case of noise value in *Silicon Strip Tracker* (SST) readout it can be represented by 9 bits. It would be more efficient to pack the offsets 9 bits at a time; 8 9-bit words would be packed into 9 8-bit bytes, which is much better than 8 16-bit words (see Figure 3.13).

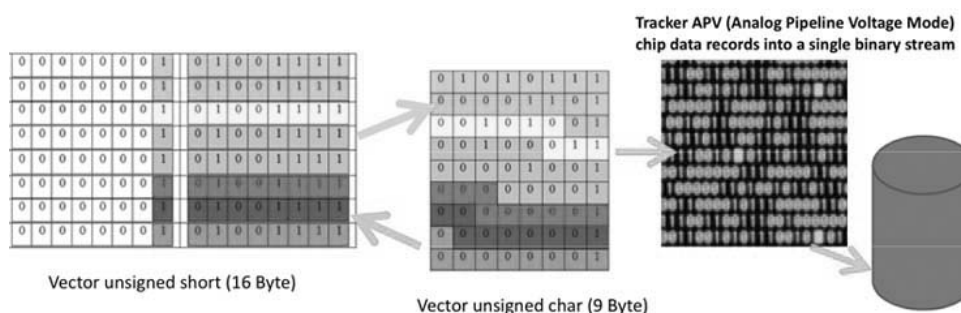


Figure 3.13: Bit-packing example.

3.7 Conclusion

CMS has developed a software system to manage conditions data as objects stored in a relational database taking advantage of the object-relational mapping capability of POOL. The core component plugs on one side into the CMS off-line framework and to the POOL/CORAL persistency layer on the other, yet to the end users the differences in event and condition data handling seem minimal. High level TAG collections are organized as trees with a light-weight relational design. A full set of tools are provided to the production managers for the global TAG management and data transfer.

Chapter 4

Tools for handling and monitoring the Condition Data

4.1 Introduction

As mentioned in the previous chapters, there are significant data storage requirements for the experiment Condition Data, which include detector databases (mechanical, cabling, geometric and calibration data from beams), LHC databases (bunch configuration, bunch structure information, beam intensity), monitoring and calibration data from online software, configuration databases for run conditions, trigger sets and voltage settings. Such data need to be accessible to the whole collaboration and updated in consistent way. In this context, the management and the monitoring of the Conditions Databases are be one of the most challenging applications for the CMS experiment, both in terms of data volumes and rates, but also in terms of the variety of data stored and their analysis.

In particular for the early data, the knowledge of the detector performance, the corrections in term of efficiency and calibration are extremely important for the correct reconstruction of the events. Moreover, in this context, monitoring and fast detecting errors is a very challenging task. To recover the system and to put it in a safe state requires spotting a faulty situation with strict timing constraints and a fast reaction.

To meet these needs, different tools have been development and put into produc-

tion.

In the first part of this chapter are described two tools, **PopCon** and the **off-line DropBox service**, used to store, transfer and retrieve Condition Data for the CMS experiment. The first tool, PopCon, is used to transfer the Condition Data, needed by *High-Level Trigger* (HLT) and off-line reconstruction, from the experiment at IP5 to the *Tier-0* computing site located at CERN. The second one, the off-line DropBox [62], is used to allow the automatic exportation of calibration constants, produced by running off-line analysis, into the off-line Condition Database to make them available to the HLT processing or to the reconstruction applications at Tier-0.

The second part of this chapter deals with two tools for monitoring the Condition Data, the **PopCon monitoring** [57] and the **Payload Inspector**. Both are used to visualize the status of the Condition Data transfer between the experiment hall and the Tier-0. The first tool, PopCon monitoring, is used for monitoring the activity on the Condition Databases from different user perspective based on the role in the experiment, Database administrator, CMS sub-detector manager or end-user performing a specific calibration or configuration of the CMS subsystem. The second one, Payload Inspector, is used by physicists for accessing, monitoring and handling the calibration and alignment data itself in different format including text, XML (Extensible Markup Language), JSON (JavaScript Object Notation), HTML table, trend plot, histogram all used to perform the certification of the Condition Data.

4.2 Tool for handling the Condition Database - PopCon

4.2.1 Introduction

The general condition data flow can be described as follows: every sub-project calculates and measures in advance all the parameters needed to setup its hardware devices, mainly related to the detector, DAQ and trigger, using the information provided by the equipment management data, in particular detector parts such as racks, crates, boards, cables and channels. Hence, configuration data are prepared for both hardware and software, and stored in the *Online Master Database System* (OMDS) for different se-

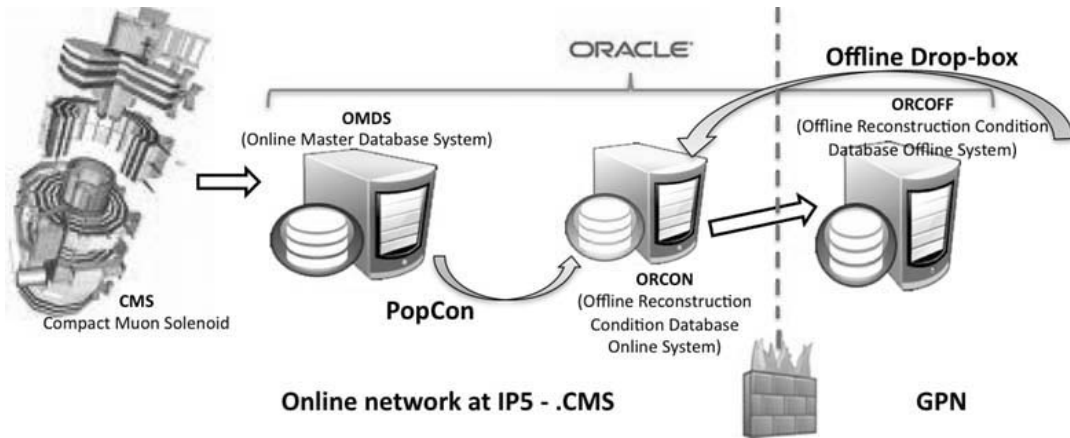


Figure 4.1: CMS Condition Database Architecture.

tups. The off-line conditions subset is extracted and sent to the off-line sites, which consists of two databases: the *Reconstruction Conditions Database On-line subset* (ORCON) and the *Reconstruction Conditions Database Off-line subset* (ORCOFF) as shown in Figure 4.1.

A software application named PopCon (Populator of Condition Objects) operates the online (OMDS) to off-line (ORCON) Condition Data transfer and encapsulates the relational data using the POOL-ORA technology. PopCon adds meta-data information, TAG and *Interval Of Validity* (IOV), to the Condition Data, so that they can be read by the off-line/HLT software.

Finally, data are transferred to the ORCOFF, which is the main Condition Database for the CMS Tier-0, using ORACLE streaming.

From ORCOFF data is then distributed to the other Tier centers, through FroNTier packages. Calibration data evaluated off-line will be written to ORCON, using PopCon. Collision event data are therefore processed using the off-line Condition Data. As data taking proceeds, we can understand better and better how the detector works; therefore, this new calibrations will require, hence new versions of Condition Data, identified by new TAGs.

4.2.2 PopCon Tool

The PopCon Tool is a package used for the population of `Oracle` databases in the CMS experiment at CERN and it is integrated in the *CMS Software Framework* (CMSSW), which relies on different underlying applications, like CORAL, POOL and Oracle. The CMSSW framework is a modular software built around the *Event Data Model* (EDM) and it runs with a single executable *cmsRun* [58] and many plug-in modules which run the different algorithms. This allows to use the same architecture to process different data source.

The CMS event data model, is based on the concept of an event as a C++ object container. The same concept is extended to the condition data, which become *condition object* (CondObjects): each piece of Condition Data (pedestals, Lorentz Angles, drift time, etc.) corresponds to a C++ object in the CMS software.

For each CondObjects a PopCon application is created in order to write the condition data into ORCON. Table 4.1 lists all the CondObjects currently used, grouped according to the subsystem they belong to. For each object the type, the approximate data size in ORCON and the upload frequency are also reported.

The core framework of the PopCon application consists of three parameterized classes, as can be seen in Figure 4.2:

- PopCon: writes all processed data and stores the data information to a destination (file or database).
- PopConSourceHandler: Read Condition Data and gives access the corresponding information. It provides the Condition Data to be processed.
- PopConAnalyzer: Analyze the properties of the Condition Data.

The “detector user” provides the code which handles the data source and specifies the destination for the data, writing a derived class of PopConSourceHandler, where all the online (source handling) code goes. The user instantiates his/her objects, provides the IOV information for such objects and configures the database output module. PopCon configuration file associates the *TAG Name* defined according to some

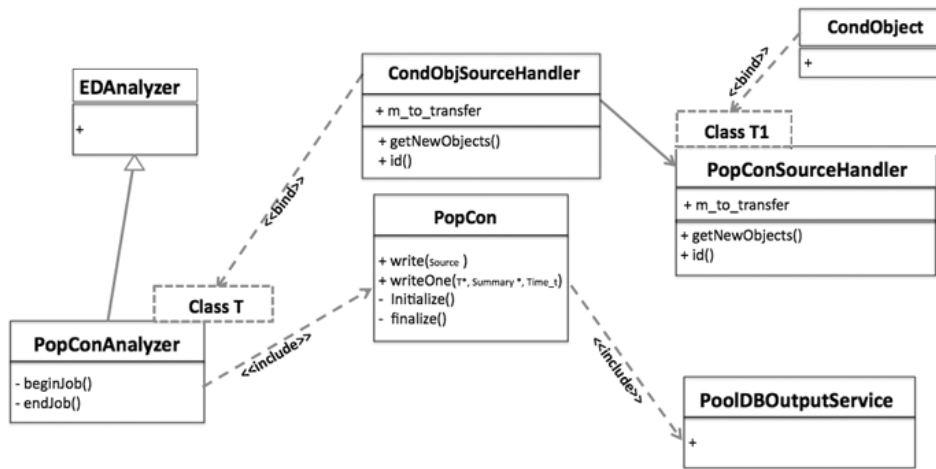


Figure 4.2: Schema of the classes for the PopCon package.

specific rules, to the *Payload object*. Once the object is built, the PopCon application writes the data to the specified database. Sub-detector code does not access the target output database: it only passes the objects to the output module. The PopConAnalyzer instantiates the PopConSourceHandler class, which is responsible for handling the data source. It also serves some additional functionality such as: locking mechanism, transfer logging, *Payload object* verification (IOV sequence), application state management, database output service.

The writer in PopCon iterates over the container of user objects and stores it in the user-configured data destination. Any transaction towards ORCON is logged by PopCon, and the process information is sent to a database account. A monitoring tool for this information was developed (see Section 4.4.2), in order to check the correctness of the various transactions, and to keep track of every upload for Condition Data.

Experience in operating the population of the condition databases

In the 2008-2009 global runs (with or without the magnetic field), the great majority of the Condition Data was transferred off-line using a PopCon application. Great effort was devoted by the CMS database project team in the integration of all the software and the infrastructural chain used to upload the calibration constants into the CMS Condition Databases. A central procedure, based on an automatic uploader into ORCON

on a dedicated machine in the online network, was successfully deployed during 2008, and, currently, is the recommended way to populate ORCON during the on-going data taking.

4.3 Tool for handling Condition Database: off-line Drop-Box service

4.3.1 Introduction

Prompt calibration and alignment work-flows produce, within 24 hours of data taking, the constants required for the *first full reconstruction* (Prompt Reconstruction) of the RAW data arriving from the detector.

In order to load these constants, produced off-line, in the condition database and make them available both to the HLT and to the off-line reconstructions and analysis, the *Off-line DropBox service* was deployed. It is an infrastructure that, using Web applications inside Virtual Machine technology, allows the automatic exportation of calibration constants, produced by running off-line analysis, into the off-line Condition Database (ORCON and ORCOFF), synchronizing them to the HLT processing or to the reconstruction applications running at Tier-0.

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 off-line context, for the reconstruction chain and for physics analysis.

Moreover, in order to guarantee the reconstruction reproducibility, we must synchronize the IOV of the new calibration sets to the next run that will be processed either by the HLT, namely the next run that the CMS detector will start for taking data, or to the next run that will be reconstructed by “Prompt Reconstruction” jobs at Tier-0.

This synchronization allows also to run HLT processing and Prompt Reconstruction with a version of calibration data that best fits the overall conditions of the detector and, hence, the produced collision event data.

Detector	Name	Type	Data size	Frequency
Pixel	SiPixelFedCablingMap	online configuration	1K	once (before the run)
	SiPixelLorentzAngle	offline calibration	1MB	each run (if different)
	SiPixelCalibConfigur.	online calibrations	5KB	each calibration run
Tracker	SiStripFedCabling	online configuration	1K	once
	SiStripBadStrip	online condition	1MB	each run (if different)
	SiStripTreshold	offline calibration	1MB	each run (if different)
	SiStripPedestals	offline calibration	20MB	each run (if different)
	SiStripNoise	offline calibration	12MB	each run (if different)
Ecal	EcalPedestals	online calibration	2MB	daily
	EcalLaserAPDPNRat.	online calibration	2MB	hourly
Hcal	HcalElectronicsMap	online configuration	1MB	once (before the run)
	HcalGains	offline calibration	1MB	each run
	HcalPedestals	offline calibration	1MB	each run
	HcalPedestalsWidths	offline calibration	1MB	each run
	HcalQIEData	online calibrations	1MB	each run
CSC	CSCChamberMap	online configuration	10KB	monthly
	CSCCrateMap	online configuration	10KB	monthly
	CSCDDUMap	online configuration	10KB	monthly
	CSCChamberIndex	online configuration	10KB	monthly
	CSCGains	offline calibration	2MB	each run
	CSCNoiseMatrix	offline calibration	2MB	each run
	CSCPedestals	offline calibration	2MB	each run
DT	DtReadOut	online configuration	10MB	once
	DtCCBConfig	online configuration	100KB	once (before the run)
	DtT0	offline calibration	10MB	rare
	DtTTrig	offline calibration	1MB	at trigger change
	DtMTime	offline calibration	1MB	daily
RPC	RPCMap	online configuration	10MB	once
	L1RPCConfig	online configuration	10MB	once
	RPCCond	online conditions	10MB	daily
DAQ	RunSummary	run conditions	10KB	run start/end

Table 4.1: CMS condition objects list.

4.3.2 Off-line DropBox service implementation

The design of the online cluster does not envisage data transfer from the outside world, neither *CERN General Public Network* (CERN-GPN), nor external networks, into it. A strict security policy is applied to the online system administration: only users deeply involved in the detector construction and commissioning can have access to it: therefore, many physicists involved in the CMS off-line calibration cannot populate the accounts in ORCON. As a result, data produced by off-line analysis cannot be “pushed” into the CMS internal network, but they must be “pulled” in it by querying servers and repositories inside the CERN-GPN. Moreover, many users are not yet familiar with the Condition Database work-flow, they do not know the software infrastructure in detail, and they are not aware of the network communication between *CMS Technical Network* (CMS-TN) and CERN-GPN: hence, the *Off-line DropBox service* simplify the end-user work in case he wants to upload calibration resulting from an off-line analysis in the Condition Databases. Finally, this new service allows the automation of alignment and calibration work-flows. The general work-flow of the *off-line DropBox service* can be described as follows (see Figure 4.3):

1. The user upload the `SQLite` file containing calibration data, together an `ASCII` file with metadata information into the web repository deployed on the CERN-GPN.
2. In the *CMS-TN*, on a machine dedicated to PopCon operation, a daemon (a `Python` script) checks regularly the web repository through a HTTP proxy. If a new file is found on the HTTP proxy, the daemon pulls the files from the repository in the CERN-GPN to the PopCon machine in the online network, and moves them in a web back-end, so that, if there are problems, the data can be easily retrieved. The daemon is also able to retrieve either the number of the ongoing run at the CMS detector, by querying the Condition Database account hosting run data, or the smallest run number ready for prompt reconstruction, by querying the Tier-0 Data Service (Tier-0 DAS), a web server providing information on the status of all jobs running at Tier-0.

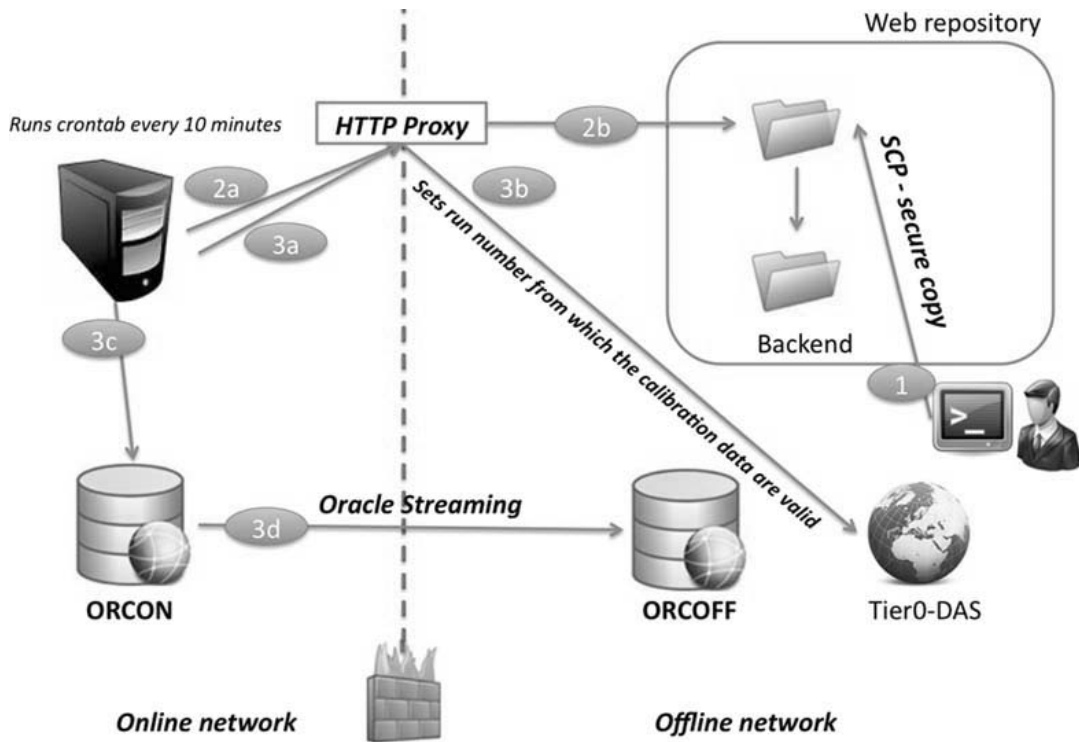


Figure 4.3: General hardware and software architecture for the automatic condition database population using the *off-line DropBox* service.

3. The `Python` script checks and, eventually, exports the data in the `SQLite` files:

- the script 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;
- the daemon sets the run number from which the calibration data are valid;
- using an application, provided by the `CMSSW` framework and based on *PopCon* libraries, which allows the exportation of Condition Data between different *relational database management system* (DBMS), the calibration constants in the local file are uploaded in the Condition Database accounts, and this transaction is logged and traced in the dedicated *PopCon* database account;
- if the user requests it, the data are appended to many IOV TAGs, thus

avoiding data duplication.

4.4 Population of the conditions database monitoring

4.4.1 Introduction

As said in Chapter 3, in the CMS experiment heterogeneous resources and data are put together in different Oracle-based databases, and made available to users for a variety of different applications, such as the calibration of the various sub-detector components and the reconstruction of all physic quantities.

In this complex environment it is absolutely necessary to monitor Database Resources and every application which performs database transactions in order to detect faulty situations, contract violations and user-defined events.

PopCon monitoring (Populator of Condition Objects monitoring) is an Open Source web based service implemented in `Python` designed for providing both fabric and application monitoring. In particular, PopCon monitoring allows the CMS users to monitor their own database transactions. Depending on the user's role in the experiment, presentation mechanism, based on the previous user interactions, have been introduced.

The organization of this section is the following: firstly we presents *PopCon monitoring* architecture and features. Following this, we explain how *PopCon monitoring* allows users, according to their previous record user interaction, to monitor their resources and applications, finally results are shown.

4.4.2 PopCon monitoring architecture and features

PopCon monitoring is structured in five main components (see Figure 4.4):

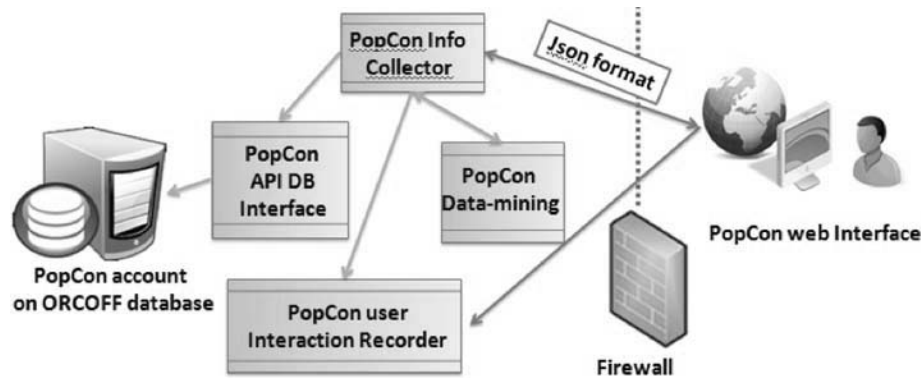


Figure 4.4: PopCon monitoring Architecture

PopCon API DB Interface

The *PopCon API DB Interface* is a Python script that gives access to the PopCon account on the Oracle database. This component uses the `cx_Oracle`¹ [67] Python module to connect to Oracle DBMS and call various PL/SQL (Procedural Language/Structured Query Language) package methods.

PopCon user Interaction Recorder

This component creates and makes accessible the records of activities made by each user. Collected records are used to implement and improve the web interface, which can be designed for information browsing for different users in different ways. This component interacts with, and receives information from the *PopCon Web Interface*.


PopCon info collector

The *PopCon info collector* aggregates the information produced by the different database transactions and the history of recorded user interactions, and encodes them in JSON format.

¹`cx_Oracle` is a Python extension module that allows access to Oracle databases and conforms to the Python database API specification.

PopCon Web Interface

The *PopCon Web Interface* displays the information about the database transactions from the different user perspectives, organizing data in tables and/or charts. Figure 4.5 shows the GUI of the PopCon monitoring. Using this GUI, a user can easily add or remove columns by clicking the check box and also columns can be sorted. Information could be grouped according to different filters.



The screenshot shows the PopCon Web Interface with a navigation bar at the top containing links: Home, PopConRecentActivityRecorded, PopConCronjobTallFetcher, popconActivityHisto, and Quota Info. Below the navigation bar is a table of database transactions. The table has columns: LOG ID, IOV TIME TYPE, EXEC TIME, IOV TAG, Payload container, Payload Name, DESTINATION DB, Exec Mess, and PAYLOAD INDEX. The table contains 10 rows of data. To the right of the table is a sidebar titled 'Set columns' with a list of columns and checkboxes to select or deselect them. The sidebar also includes a search bar and a 'Show 10 entries' button.

LOG ID	IOV TIME TYPE	EXEC TIME	IOV TAG	Payload container	Payload Name	DESTINATION DB	Exec Mess	PAYLOAD INDEX
100900	runnumber	20-11-10 16:21:35	runinfo_start_31X_hlt	RunInfo	RunInfo	oracle://cms_orcon_prod/CMS_COND_31X_RUN_INFO	OK	14398
100999	runnumber	20-11-10 16:21:33	runinfo_start_31X_hlt	RunInfo	RunInfo	oracle://cms_orcon_prod/CMS_COND_31X_RUN_INFO	OK	14397
100998	runnumber	20-11-10 16:15:02	runinfo_31X_hlt	RunInfo	RunInfo	oracle://cms_orcon_prod/CMS_COND_31X_RUN_INFO	OK	14105
100997	runnumber	20-11-10 16:15:00	runinfo_31X_hlt	RunInfo	RunInfo	oracle://cms_orcon_prod/CMS_COND_31X_RUN_INFO	OK	14104
100996	runnumber	20-11-10 16:13:42	runinfo_start_31X_hlt	RunInfo	RunInfo	oracle://cms_orcon_prod/CMS_COND_31X_RUN_INFO	OK	14396
100995	runnumber	20-11-10 16:13:40	runinfo_start_31X_hlt	RunInfo	RunInfo	oracle://cms_orcon_prod/CMS_COND_31X_RUN_INFO	OK	14395
100994	timestamp	20-11-10 15:02:36	SStripDetVOff_v2_offline	SStripDetVOff	SStripDetVOff	oracle://cms_orcon_prod/CMS_COND_31X_STRIP	OK	8946
100993	timestamp	20-11-10 15:02:35	SStripDetVOff_v2_offline	SStripDetVOff	SStripDetVOff	oracle://cms_orcon_prod/CMS_COND_31X_STRIP	OK	8945

Figure 4.5: The PopCon web interface represents information about database transactions in different types: both charts and tables.

Figure 4.6 shows the PopCon Activity History. Using this GUI, with the help of the mouse, the user can interact directly with the chart. He can point the cursor to the part of chart and see the information about transactions. Charts display the accounts on which transactions were done, date and time of it and the number of occurrences.

A set of reusable components, known as *widgets*, are being made available. These are usually built using *jQuery*, a cross-browser JavaScript library designed to simplify the client-side scripting of HTML, and *Cascading Style Sheets* (CSS) files, used to describe the presentation semantics (the look and formatting) of a document written in a markup language.

Where possible, these components are reused in order to provide identical functionality across various components, so that a user feels comfortable with a standard style for all web tools. The services run on a fairly standard configuration: a pair of Apache

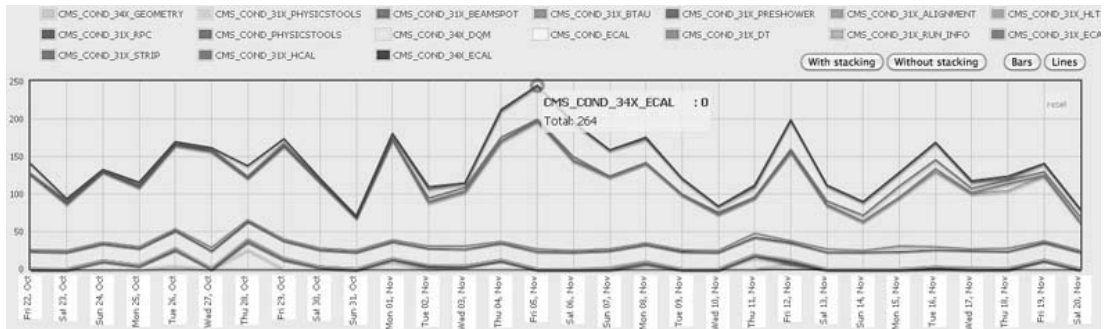


Figure 4.6: PopCon Activity History chart.

servers working as a load balanced proxy in front of many application servers. The front-end servers are accessible to the outside world, while the back end machines are firewalled off from remote access [60]. With this infrastructure we can minimize problems related with security issues: in particular, each user is unable to handle database objects. Thanks to AJAX (Asynchronous Javascript and JSON) we can provide real-time feedback to our users exploiting server-side validation scripts, and eliminate the need for redundant page reload that is necessary when the pages change. In fact, this component allows to send requests asynchronously and load data from the server. The *PopCon Web Interface* uses a programming model with display and events. These events are user actions: they call functions associated to elements of the web page and then actions are recorded by the *PopCon user Interaction Recorder*. The contents of pages coming from different parts of the application are extracted from JSON files provided by the *PopCon Info Collector*.

PopCon data-mining

This component can extract information from logs of database transactions (operator, data source, date and time, metadata) and the *PopCon User Interaction Recorder* (sequence of actions to get to the right contents, average time on each page to compute the attention applied by the visitor) finding existing patterns in data.

The first algorithm, used to scan the history of recorded user interactions, iterates two main steps.

The first step, called *harvesting user interaction statistics*, records the following list of measurements subdivided into two categories:

- tracks of the browsed page, like the most requested pages, the least requested pages, the most accessed directory, average Time on Page, average Time on Site, ordered sequence of visited pages, new versus returning visitors (by means of cookies) and the number of views per each page.
- tracks of user activity at the page level:
 - Changing attributes of graphical elements: (e.g. changing charts representation from line chart to pie chart or histogram chart, sorting and filtering data in a table)
 - Removing/adding object elements (e.g. remove/add columns to the table)

The second step, called *grouping attributes of user interaction with significant correlation*, gathers in different subgroups the *tracking user activity* and *tracking browsed page* that have similar attributes, like the most accessed directory and common graphics elements, in order to create mutually exclusive collections of user interactions sharing similar attributes.

To reach this goal, we use an algorithm handling mathematical and statistical calculations, such as probability and standard deviation, to uncover trends and correlations among the attributes of the user interaction.

For example, after scanning the history of recorded user interactions, an association rule “*the user that visits page one also visits page two and chooses to see histogram reports (90%)*” states that nine out of ten users that visit the page one also visit the page two and prefer to see the bar chart. We can build use case models, based on these statistics, in order to reflect the requirements and the needs of each user. As a result, the user, classified under this use case, will take advantage to see a web interface based on his perspective, helping him to find and manage the information he needs more quickly.

The second algorithm is used to scan the PopCon logs [61]. This application can be used in two different ways:

- since it is integrated in the CMS framework, users can write `Python` scripts which are executed by the framework executable *cmsRun*.
- the framework itself provides an application which, using *PopCon* libraries, allows the exportation of data into the off-line database.

These applications are responsible for maintaining and handling operations which are related to database transactions. In this scenario, it is very difficult to catch all error messages coming from different heterogeneous resources. Therefore, we follow this strategy: every application provides an error output consisting of three components: the name of application, the error code, that is unique for each tool, and the description of the error itself. So, *PopCon* developers can clearly understand what is wrong with their tool, while the end-user is able to check if the data exportation (database transaction) they want to perform was successful or not. Besides describing what the error is and how it occurred, most error messages provide advice about how to correct the error.

To help both users and developers to classify correctly the observed damage, the error messages are defined by the level of issue with a different colour. These levels are:

- Fatal. The program cannot continue (red colour).
- Major (Error). The program has suffered a loss of functionality, but it continues to run (orange colour).
- Minor (Warn). There is a malfunction that is a nuisance, but it does not interfere with the program's operation (deep green colour).
- Informational. Not an error, this is related information that may be useful for troubleshooting (green colour).

As further example, we describe another kind of error not depending on the particular application, but on Hardware/Software/Network problems. To discover this kind of error, we perform a time series analysis on database transactions associated with the discovery and use of patterns such as periodicity. Since dates and times of the database

transactions are recorded along with the users information, the data can be easily aggregated into various forms equally spaced in time. For example, for a specific account the granularity of database transactions could be hourly and for other account could be daily. This information allows to discover two main kinds of alarm:

- Scanning the entities monitored by *PopCon* (logs of database transactions), the association rule “*during a long period, a specific user performs a database transaction at regular time intervals*” states that, probably, if these regular intervals suddenly change without a monitored interaction by an administrator, and, for particular cases, by the user, there can be network connectivity problems, or machine failures on the network. In details, if the system finds an exception to this pattern in data, it triggers an action to inform a user about possible problems by email. Besides, the web user interface provides red/orange/green alarms, according to the seriousness of the problem, so that this exception is immediately visible by the user.
- Taking the size of data together with the periodicity of database data transactions we can forecast the rate at which disk capacity is being filled in order to prevent a disk becoming full, alerting the database manager and the administrators of the machines dedicated to the data exportation some days in advance.

4.4.3 PopCon monitoring from the different users’ perspectives

The design of the presentation of the data collected by *PopCon monitoring* is based on the requirements given by different types of users, each of them having to do with a different abstraction level of a Database administration issue: the ORACLE Database Administrator level, the central *CMS detector* level, the *CMS sub-detector* level and the End-User level (see Figure 4.7).

- The ORACLE Database Administrator may wish to face up to databases security issues for which he is responsible. Typical example of the information he need are:

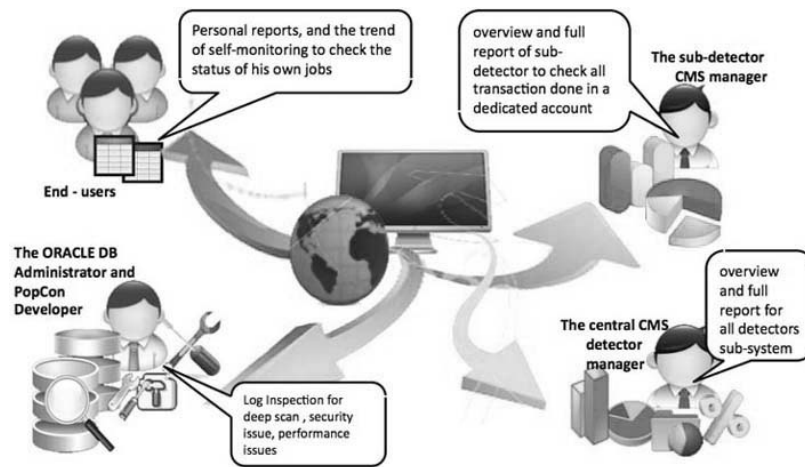


Figure 4.7: In the CMS experiment, many users have access to Condition Data with different needs: they exploit several applications with different roles.

- People on the inside (using *PopCon application*) and outside (using *PopCon Web Interface*) network accessing the database and their activity.
 - Programs accessing a database concurrently in order to control multiple access to the same account;
 - If all such processes leave the database or data store in a consistent state;
 - Illegal entries by hackers;
 - Malicious activities such as stealing content of databases;
 - Data corruption resulting from power loss or surge;
 - Physics damage to equipment;
- The central CMS detector manager and the *PopCon application* developer may require the possibility of analysing the behaviour of their applications for each CMS sub-detector.
 - The sub-detector CMS manager may require the possibility to analyse the behaviour of his transactions on his own sub-detector database account.
 - The End-User may require the possibility to analyse the behaviour of his own personal transaction such as size and rate/duration of the transactions, or detect

fault situations related to insufficient password strength or inappropriate access to critical data such as metadata.

To summarize, *PopCon monitoring* automatically detects the cookies installed in each user's browser and this information is used to match the user with a role (Oracle Database Administrator, *PopCon application* developer, sub-detector CMS manager, End-User) in order to provide a customized report that allows each user to have a customized printout of information depending on his needs.

The use of data mining techniques to extract patterns from logs of database transactions (operator, date and time) and the history of recorded user interactions has some general advantages. The storage of these patterns will help the user to read and understand quickly the current situation without going through several pages and use the search fields.

4.4.4 Results

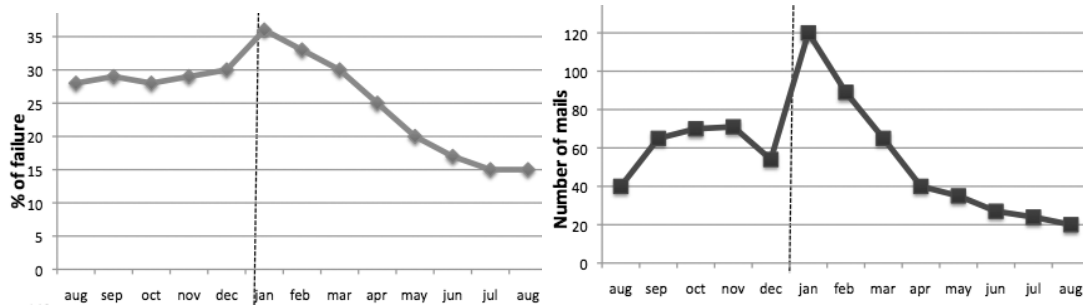
Since the *PopCon* web monitoring has been introduced the percentage of transactions failure has decreased from 28% to 10% (see Figure 4.8a), and the number of requests of assistance to the developer has decreased (see Figure 4.8b). The number of requests of assistance to the developer has been made counting the email to the *CMS HyperNews database support* ².

4.5 Payload Inspector and Global TAG Browser

4.5.1 Introduction

When processing collision events revealed by the CMS detector, Condition Data are needed for precisely reconstructing the physic quantities. Hence, huge collections of Condition Data coming from different sources must be properly organized.

²HyperNews is a discussion management system providing both the functionality of mailing list systems as well as the web interface of web forums.



(a) *Percentage of transactions failure performed during the period of August 2008 to August 2009.* (b) *Number of request of assistance to hypernews databases support during the period of August 2008 to August 2009.*

Figure 4.8: The peak in January is due to the introduction of the new tool for which many users were not yet familiar.

In order to help the user to manage these collections, the concept of *Global TAG* was introduced for Condition Data. It defines the scope of a collection of TAGs and IOVs.

An ensemble of Global TAGs can be organized in a hierarchy. In this scenario, a web application has been developed to deal with two different requirements: the Global TAG Browser, used by physicists for accessing, monitoring and handling these objects, and the Global TAG Collector, used by the CMS software management to organize collections of basic IOV and TAGs collected from different sources. This application takes advantage of the last web technologies, like jQuery library, to build a GUI (graphical user interface) which best fits to the user requirements.

4.5.2 Payload Inspector

Each conditions data, which is stored in the off-line database as POOL-ORA objects, is also called *Payload object*. Each *Payload object* is indexed by its IOV, while the *Payload objects* themselves do not contain any time validity related information.

The Payload Inspector Service allows to discover the version (TAG), the IOV of each set of conditions. In addition this web application allows to obtain information about the detector status in different formats from text-based format, to XML-based format, to chart format in 2-Dimensional Histogram or multi-dimensional histogram

using the technique capable of producing different pixel sizes and color depths. Figure 4.9 shows the GUI of the Payload Inspector where the physicists are able to:

- select the Database Service (e.g. Off-line, On-line, Production, Test) and the CMS sub-detector,
- search or select TAGs and IOVs,
- receive the CMS detector status information in different formats.

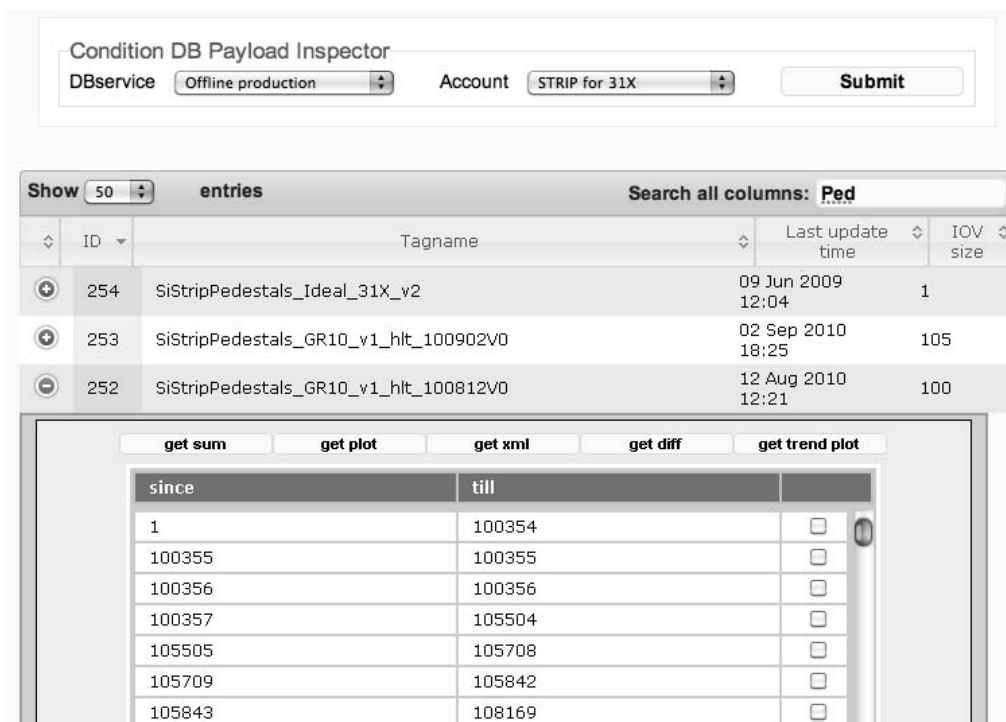


Figure 4.9: Payload Inspector GUI.

As matter of fact, multi-dimensional histogram is the most widely used format and at the same time is most hard to implement since it is different for each sub-detector and use case.

In the past, the main visualization tool for monitoring was a histogram presenter that would show a set of histograms and tables updated regularly. Each CMS sub-detector has millions channels organized in thousands of *sub-detector modules*, (for

example the CMS Tracker has more than 50 millions channels organized in 16540 *Silicon Strip Tracker* (SST) modules) each one being a complete detector: its monitoring requires many thousands of histograms to be computed every few minutes. An histogram presenter is not enough: people in charge for monitoring need a representation that would show all sub-detector modules at once in a single computer screen with single sub-detector module information coded in some way. This task has been accomplished using a 2D image representation called *sub-detector map*. This “map” is obtained in the following way: since the single sub-detector module is flat, we imagine to disassemble the whole sub-detector of CMS and to assemble it again on a flat surface putting the single sub-detector modules in positions which are connected to their spatial position. For example the single sub-detector module can show coded with a color the following quantities.

- The number of dead channels from construction database.
- The total number of RecHits³ hitting the module in the last 100 events readout.
- The Result of a comparison between the last histogram and a control histogram.

Holes or hot spots in the map can pinpoint detector problems.

We must consider the *sub-detector map* like an histogram: for each sub-detector module we have a count of calls to the fill method and a value. If the count is 0 then the sub-detector module is white. If the count is more than 0 and the value is the minimum than it gets a dark blue, the lower end of color scale. To the right of the Figure 4.10, 4.11 there is a vertical colour scale used to give a visual measure of the number of Bad Channel [63].

Tracker Alive Channels during the 7 TeV collision period are presented in Figure 4.12. In 2010, the CMS Tracker starts the 7 TeV collision runs with 98.1% working detector. White modules have been excluded because it is known to be bad.

³Raw data from the detector are unpacked off-line into integer-based objects called “digis”. There are digi collections for the strip signals, the wire signals, and the *local charged tracks* (LCTs). The information stored in the digis is processed to produce a collection of objects called “RecHit” with measured x and y coordinates at a known z coordinate. These represent the measurement of the intersection point between the track a detector.

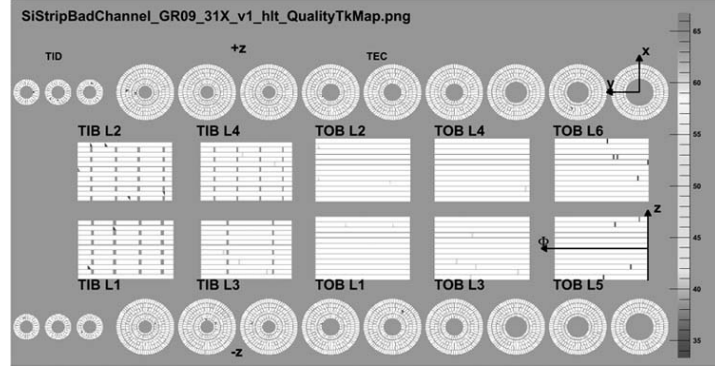


Figure 4.10: Measurement of the Bad Channel in all TIB and TOB layers for the first IOV.

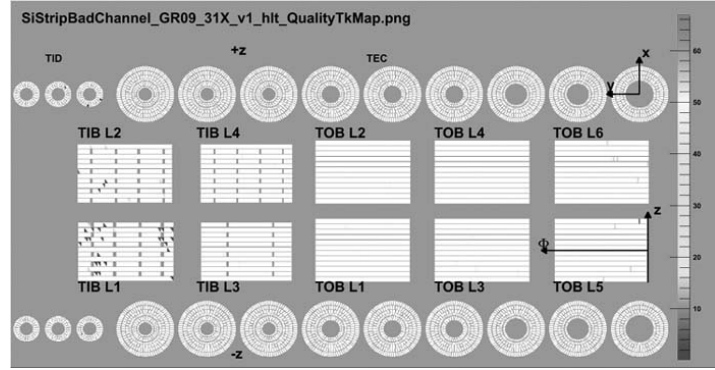


Figure 4.11: Measurement of the Bad Channel in all TIB and TOB layers for the second IOV.

An example of typical monitoring plot for the *CMS ECAL Laser Monitoring Run* is shown in Figure 4.13. The user can see the unfold of the ECAL Barrel 4.13a and two End-caps 4.13b. Each pixel represents a channel in ECAL, the color map corresponds to the laser light response of the crystals.

Figure 4.14 shows the *CMS RPC Electric Current Monitoring*. The distributions of mean and root mean square (RMS) for each chamber is shown in the upper plots. The mean-RMS correlation and time profiles of particular chambers is shown in the lower plots.

In this case since the algorithm calls the fill method only when it finds a RecHit in the channel, white indicates all channels that get no hit at all. Dark blue indicate

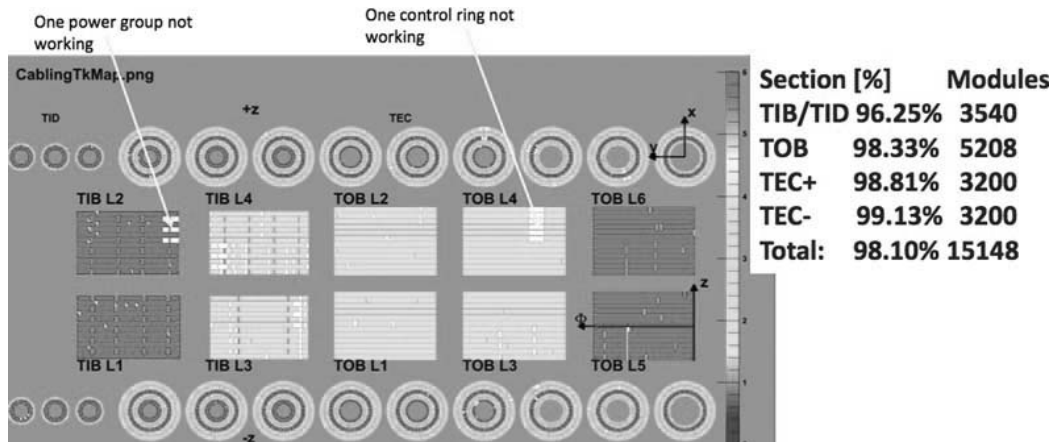


Figure 4.12: Tracker Alive Channels.

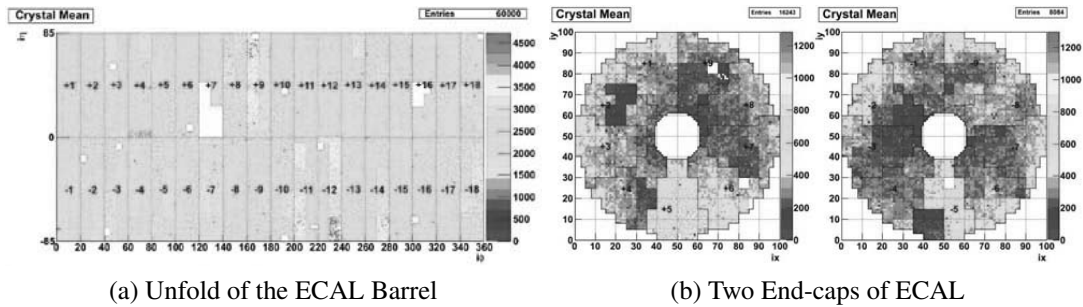


Figure 4.13: A typical Payload Inspector Web-based monitoring system output.

sub-detector modules that get the minimum hit number (this can be as low as 1 but never 0 and depends on statistics.)

Computing difference images using Python Image Library

Another important way to enhance the features the physicist is interested to see want to see is obtained with the difference images. They are images which are created by subtracting one image from another one. This feature is very important since it can be used in several use cases:

- in the case of Bad Channel Map it provides a quick glance useful to spot new bad components compared to previous interval of time (see Figure 4.15). The bluish colors refer to modules which recovered with respect to the previous mea-

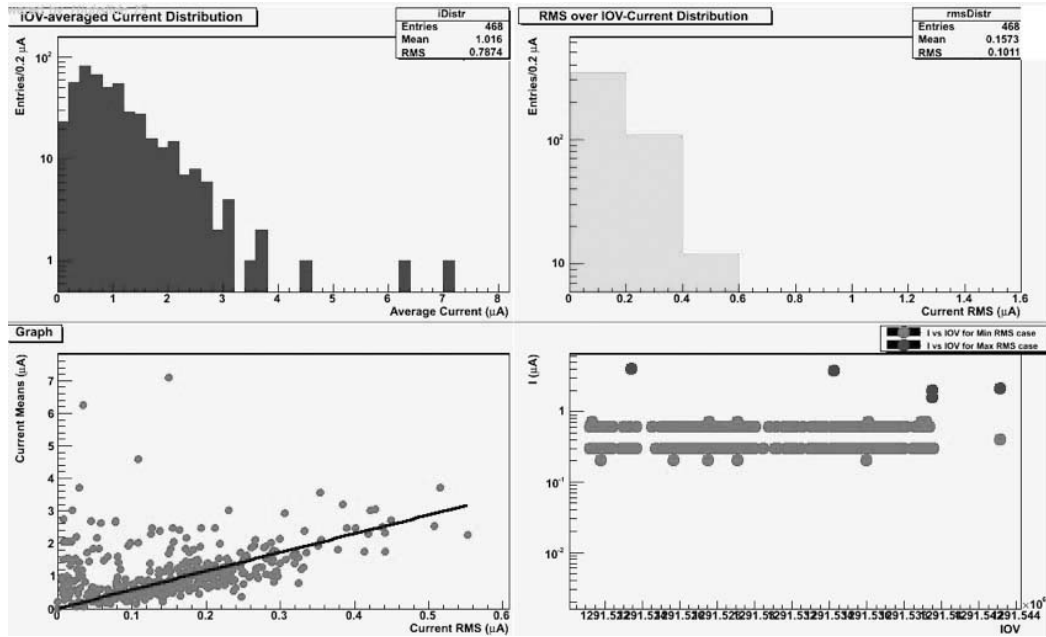


Figure 4.14: CMS RPC monitoring plot for the electric current.

surement. The reddish colors are associated to modules which shown a failure after been working in the previous measurement. The black color indicates no changes between the compared conditions.

- in the case of Lorentz Angle Map it can be used to certify the new condition have been uploaded successfully (see Figure 4.18). The physicists responsible to upload the Condition Data for the CMS-tracker sub-detector, tried to upload the new Lorentz angles condition relative to *Tracker Outer Barrel Layer 1* (TOBL1) component of SST and thanks to the image difference they can confirm if the change was effective. To see the details he needs to click on that part of map to inspect the detailed histograms with new conditions.

In detail, the module used to calculate difference images, *ImageChops* from *Python Imaging Library* (PIL), contains a number of arithmetical image operations, called channel operations (“chops”). These can be used for various purposes, including special effects, image compositions, algorithmic painting, and more. Each channel operation takes two image as argument and returns a new image. The result of a channel

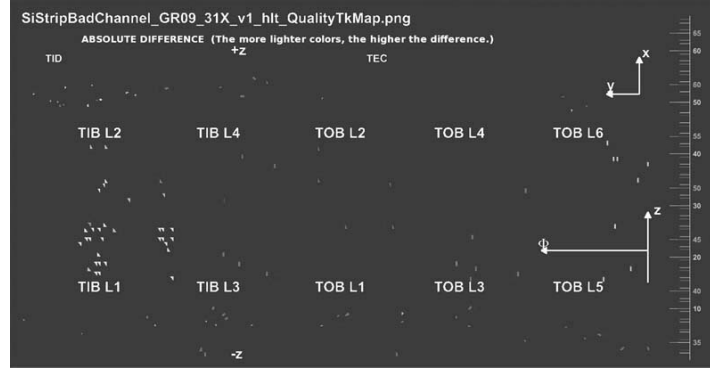


Figure 4.15: In the difference plots between Figure 4.10 and Figure 4.11, the plotted variable is the absolute value of the difference between the two chosen measurement of the (bad) channel.

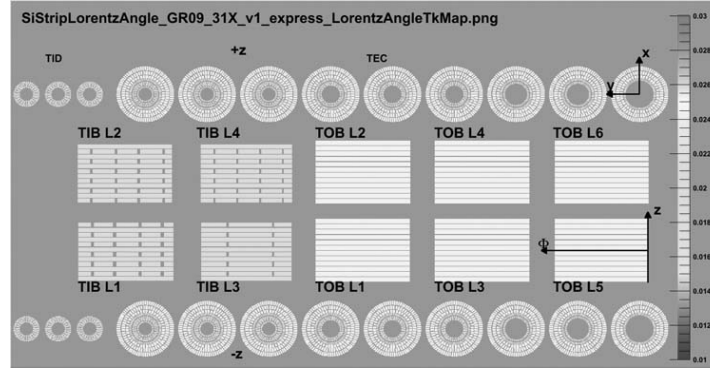


Figure 4.16: Measurement of the Lorentz angle in all TIB and TOB layers for the first IOV

operation is always clipped to the range 0 to MAX (which is 255 for all modes supported by the operations in this module). In our case the Algorithm returns the absolute value of the difference between the two images. $out = abs(image_1 - image_2)$

4.6 Conclusion

The monitoring system, *PopCon monitoring*, was successfully tested during the 2009 cosmic ray data taking and the early p-p and Pb-Pb collisions runs. The positive feedback of many users, and the prompt analysis of faulty situations discovered thanks to

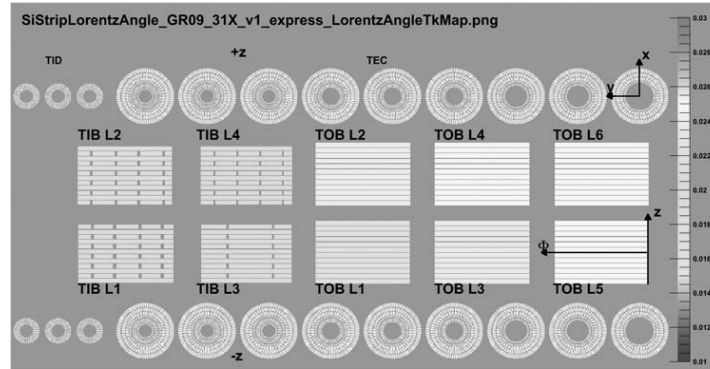


Figure 4.17: Measurement of the Lorentz angle in all TIB and TOB layers for the next IOV

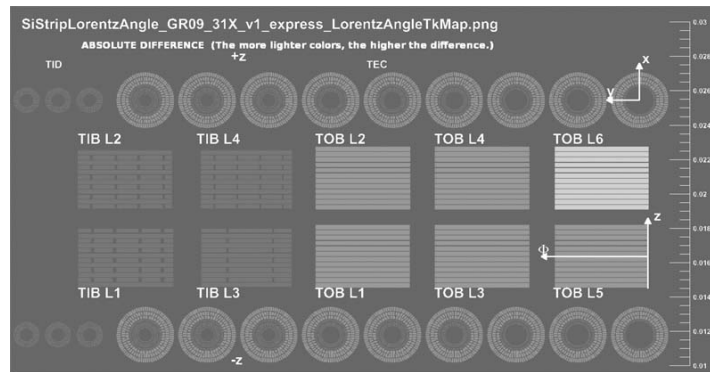


Figure 4.18: Absolute value of the difference between the two measurement of the Lorentz angle, Fig.4.16 and Fig.4.17.

this tool by ORACLE Database Administrators and PopCon developer team, allowed for further development.

The *Payload Inspector* is the newest application to be added to the Condition Data Tools. It was developed during the p-p collisions data taking at 7 TeV, and was successfully deployed and used during the Pb-Pb collisions runs.

The off-line DropBox service was intensively studied and tested during the 2009 p-p collisions data taking at 450 GeV and 1.18 TeV, and was successfully deployed for the 2010 collision data taking at 7 TeV. This tool is widely used by many physicists, and is one of the most stable parts of the automated calibration work-flows running at Tier-0.

Chapter 5

Operational Challenges and Technical Solutions for CMS Web Condition database

5.1 Introduction

In the Chapter 4, we have shown use case of web application, such as *PopCon Monitoring* and *Payload Inspector*, focused on the user's point of view, without giving details of the operational challenges and technical aspects.

The objective of these web projects, developed as part of the work for this thesis, consisted also, and above all, in delivering reliable and robust tools with high performance and availability. The accomplishment of such task resulted harder than originally foreseen. Several difficulties were encountered while making data available through the web application and the web service to the end users in a manageable way, namely:

- Uploading the data into GUI must keep up with the production rate.
- Accessing data can quickly impair the hardware resource of the condition database: careful resource usage control must be exercised.

- The potential total volume of data is large. An archiving policy needs to be enforced to manage disused data, not only because resources are limited but also to improve performance.
- Web applications are exposed to security risks. They are open to a large number of users in the Internet community thus widening the threat.

In order to find out the better solutions to tackle these aspects, we have been given no constraint for technical features to be chosen to match the project requirements.

In this chapter we first motivate why no market solution meet the CMS off-line project requirements adequately. Then, we will detail the operational challenges in terms of strengths, weakness and constraints (advantages and disadvantages) of using different technologies, such as *Desktop* vs *Web Application*, different tiers architecture, HTML version 4 (HTML4) vs HTML version 5 (HTML5). In particular we detail about the technical features chosen showing the comparison results between the performance of different implementation. Finally, the results are presented and discussed.

5.2 Why not adopting a commercial software

5.2.1 Introduction

Nowadays, the number of commercial tools available for accessing databases, built on Java or .Net, is increasing. Nevertheless, many of these applications have several drawbacks: usually they are not open-source, they provide interfaces only with a specific kind of database, they are platform-dependent and very *Central Processing Unit* (CPU) and memory consuming, and, above all, they don't envisage the access to object-oriented database such as CMS off-line condition data. For this reason we developed a free web based tool written using Python and JavaScript, called *jSPyDB*, whose software components were used to deploy the web application for CMS condition database.

The *jSPyDB* relies on jQuery and Python libraries, and is intended to provide

a simple handler to different database technologies inside a local web browser. Such a tool, exploiting fast access libraries such as `SQLAlchemy`, is easy to install, and to configure. The design of this tool envisages three layers. The front-end client side in the local web browser communicates with a back-end server. Only the server is able to connect to the different databases for the purposes of performing data definition and manipulation. The server makes the data available to the client, so that the user can display and handle them safely. Thanks to `jQuery` libraries, moreover, this tool allows data exportation to different formats, such as XML and JSON. Finally, by using a set of pre-defined functions, users are allowed to create their customized views for a better data visualization.

In this way, we optimize the performance of database servers by avoiding short connections and concurrent sessions. In addition, security is enforced since we do not provide users the possibility to directly execute any SQL statement.

Comparison with other Products on the market

First to deploy a new database browser application we conducted a market investigation of existing tools. This is a list of some of the most used database browsers that help users explore objects in a relational database:

- `phpPgAdmin` [64]: `phpPgAdmin` is a web-based administration tool for administering `PostgreSQL`. It is perfect for `PostgreSQL Data Base Administrators` (DBAs), beginners and hosting services.
- `phpMyAdmin` [65]: `phpMyAdmin` can manage a whole `MySQL` server (needs a super-user) as well as a single database. To accomplish the latter you'll need a properly set up `MySQL` user who can read/write only the desired database.
- `phpOracleAdmin`: it uses `PHP4` to offer simple `Oracle` object browsing, multiple database connections and easy viewing of `Oracle 8i` database structures. It needs *with-oci8 php* option which might not be installed by default. Unfortunately, development cycle is very slow, no evolution has been brought since January 2004.

- Oracle SQL Developer [66]: is a free and fully supported graphical tool for database development. With SQL Developer, the user can browse database objects, run SQL statements and SQL scripts, and edit and debug PL/SQL statements. You can also produce a large number of reports, as well as create and save your own.

The coverage of operations offered by these tools also implies some drawbacks:

- They permit execution of arbitrary SQL commands. Bad SQL queries can downgrade the performance of the whole system.
- They are designed to work only with a particular *DataBase Management System* (DBMS). Changing of the DBMS requires changing the application.
- They maintain open database connections for a long time: The problem with long-lived connections is that they can be left in unpredictable states by clients, and moreover it might not be entirely evident that the connection is still there. A network failure, server restart or *stateful firewall*¹ forgetting some of its state could all result in a *stale connection*² which looks open, but then gives an error when you try to use it.

5.3 Desktop vs Web Application

5.3.1 Advantages and disadvantages of different applications

There are two ways of implementing a user interface, either using a *desktop application* or a *web application*. Both methods offer specifically advantages and disadvantages. Following some of the ISO 9126 and most recent ISO 2500 software quality characteristics, we can compare the two approaches in terms of quality requirements:

¹In computing, a stateful firewall is a firewall that keeps track of the state of network connections traveling across it.

²From an application code perspective stale connections are connections which are no longer usable, for example, if the database server is shut down or the network is experiencing problems.

- Maintainability, Portability and Installability - Web based applications need to be installed only once on the server; while desktop applications need to be installed separately on each computer.
- Changeability - Updating applications would need to be done on every single computer, which is not the case with web applications.
- Usability and Functionality - Desktop applications are confined to a physical location and hence have usability constraints. Web applications development, on the other hand, makes it convenient for users to access the application from any location using the Internet.
- Interoperability - Using a web browser to access the application provides cross-platform compatibility.
- Security - Web applications are exposed to more security risks than desktop applications. You can have a total control over the standalone applications and protect it from various vulnerabilities. This may not be the case with web applications as they are open to a large number of users in the Internet community thus widening the threat.

5.4 n-Tier Architecture

5.4.1 Introduction

In web based applications, the internet is used to realize a communication between remote machines. Particularly, in the case of an application like *CMS web condition data*, a user runs a client application to retrieve data from a remote database. This kind of scenario is widely used in applications known as client/server applications.

In client/server applications based on a database access, three parts are generally defined in software architecture: the *User Interface*, the *Business Logic* and the *Data Management*. The User Interface contains the support for presentation and user interaction. The Business Logic handle information exchange between the Data Manage-

ment and the User Interface. Data Management supports the access to the database. Different software architectures are possible for approaching these three parts. The following sections present the existing ones: two-tiers, three-tiers and multi-tiers architectures. The dissertation will be useful to understand which architecture fits the specific requirements of the CMS condition database.

5.4.2 Two-Tiers Architecture

The two-tiers architecture involves two computers: a client machine and a server machine. These two remote tiers of the application communicate through the internet. The computing client talks directly to the server with no other intervening processes. There can be multiple clients using the same server. In the case of a database application, the client communicates with a data server running a DBMS that support the database access. The Figure 5.1 shows the two-tiers architecture.

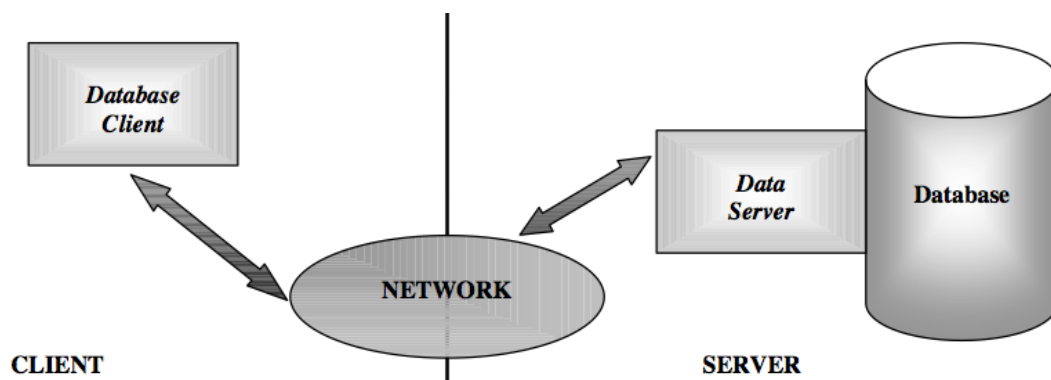


Figure 5.1: Two-tiers architecture for databases applications

The three parts of the software (user interface, business logic and data management) are divided between these two tiers:

- the client tier application covers the user interface tasks and a part of the business logic tasks
- the server tier application covers the other part of the business logic tasks and the data management tasks.

The client application enables the user interaction and composes the corresponding requests to the database (generally in a SQL-like language), and then transmit these requests to the server tier. When receiving the results from the server, the data returned can be manipulated by the client application for further sub-selection, business modeling, reporting, visualization.

The server application runs the DBMS in order to process the clients requests to retrieve the appropriate data from the database and to support database updating and integrity checking tasks. The communication between the client and the server is assured through the internet, and can be realized in different ways.

A major advantage of the two-tiers architecture software is a faster developing time compared with the more complex architecture software. This kind of architecture is perfectly suitable for applications devoted to a few number of users and when there is no need of elaborated processing between the user interactions activities and transmission of request to the server.

This kind of architecture is often not enough robust for elaborated database applications. First of all, the performance deteriorates with high number of users, because clients connect directly to the server. Secondly, the client application must handle all the processing corresponding to SQL preparation and SQL exploitation, and it can compromise flexibility since each client application need to use the correct requests corresponding to the server requests-processing management. Finally two tiers architecture are not suitable for elaborated applications when long processing tasks are required between user interaction and requests transmission to the server, since client applications would then be complex and maybe difficult to deploy.

5.4.3 Three-Tiers Architecture

Three-tiers architectures insert one middle-tier between the client and the database server. We will therefore have a client application running on a machine and communicating through the internet with one server application running on another machine.

This server will communicate with a database server, potentially running on another machine. This three-tiers architecture is depicted in the Figure 5.2 .

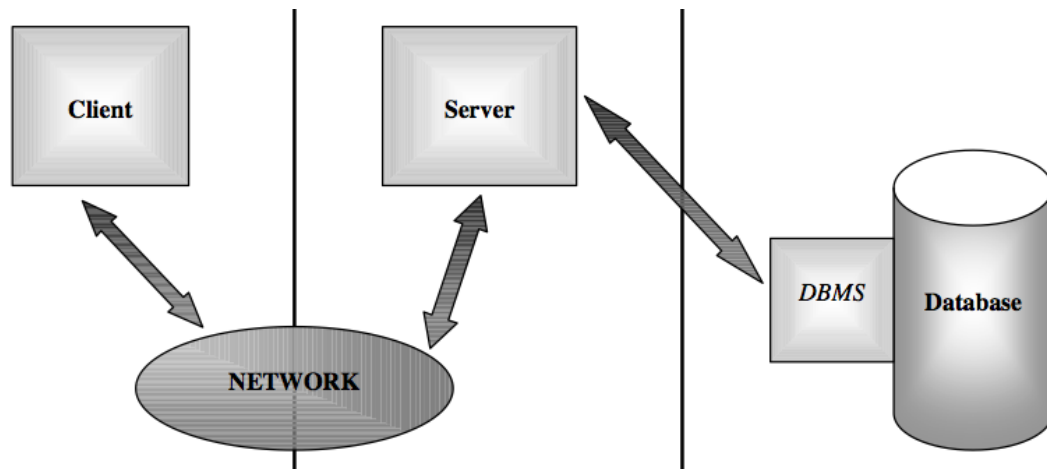


Figure 5.2: Three-tiers architecture for database applications.

The middle-tier in three-tiers architecture (that we call server) enables to totally separate the business logic part of the application from the client tier. In a three-tiers application, the different tasks are cleanly divided as following between the different tiers:

- The client tier covers the tasks related to user interface
- The server tier covers the tasks related to business logic
- The data server covers the tasks related to data management

Concretely the client application supports the user interaction and transmits the resulting requests to the server. The client application is then more specific than in a two tiers application since it does not process data or very little (or only few) when the user interacts, but just transmit the request to the server.

The server application manages the requests coming from the different clients and run the appropriate processing corresponding to each of them. The server is also responsible to provide to the clients the data they asked. Then the server codes the corresponding *SQL* queries and transmit them to the data server and collect the answers, in order to be able (eventually after processing the results in different ways) to send the appropriate answer to the client. The client is totally freed by the different tasks related to database access.

The data server is just running a DBMS and is able to receive the SQL queries of the server. It processes them and return the appropriate results.

Three tiers architectures provide many advantages compared to two tiers architecture. First of all, the clear functional separation between the three tiers can clarify the development process and the modularity, maintainability and flexibility of the entire system. The use of the middle-tier enables also to have a robust application since this tier can organize queuing mechanisms. It also increases considerably the efficiency of the entire system since the client can process other tasks once he had transmit its request to the server. The server will be in charge of processing the client request such as accessing a database, performing a calculation, or validating data, and then will re-contact the client giving the results back. Moreover the client is “lighter” than in two-tiers architecture since it does not deal with anything specific with database (for example the client never uses SQL, it just sends parameters to the server that creates the appropriate SQL queries). Finally introducing a middle-tier enables to process specific computations to react to user interaction, and not only access a database.

Finally, the three tiers architectures helps strategies and methodologies of computer security using, for example, industry standard firewall, password protection and access control list. More details will be given in the following sections.

5.4.4 Multi-Tiers Architecture

Multi-tiers architectures are a particular case of three-tiers architectures: the business logic tasks of the middle-tier can be realized by more than one server, and with connection to more than one database. Then we have a n-tiers architecture with multiple (n) servers and/or multiple database servers with their own DBMS. It can distribute clients requests across multiple servers and can access data in multiple databases. The Figure 5.3 presents this architecture.

This architecture derived from three-tiers architecture is used when several databases must be accessed, or when there must be different servers in order to deal with different kind of requests from clients. It is particularly interesting to use this kind of architecture to distribute the clients requests between multiple servers in order to divide the

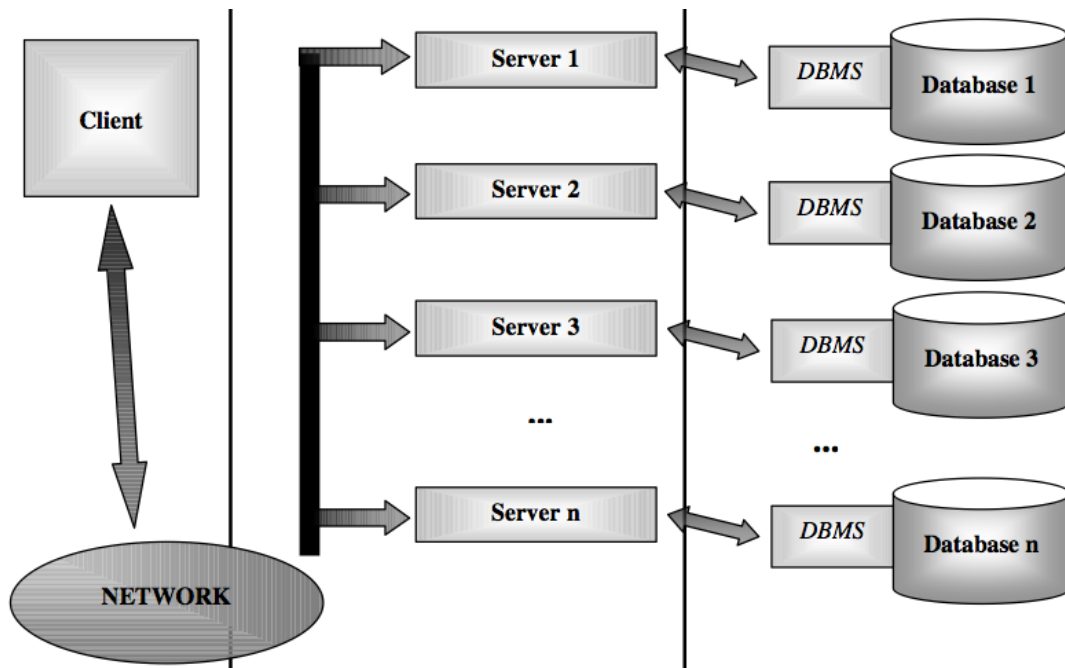


Figure 5.3: Multi-tiers architecture for database applications.

work between the different CPU of the servers (load-balancing).

5.4.5 CMS condition DB web application architecture

In the CMS condition DB web application case, it seems unsuitable to adopt a two-tiers architecture. Firstly, the interaction of the user requires some processing before being translated as database queries. This would require including this processing in the client application, that would increase its complexity and decrease its flexible deployability. Moreover, the connection to the off-line database (where are all the data requested by the client) can only be achieved through particular connections (by means of `CMSSW` or `cx_Oracle` plugin, and it is complex to encapsulate them in a client application distributed through the internet. The three-tiers architecture seems particularly better in the CMS condition database case. It 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 will be totally separated from the client application.

In the case of the condition database, we will use an ameliorated multi-tiers architecture with one server and two databases: `Oracle DBMS` and `FroNTier`. The idea is to use a second database, `FroNTier`, as a cache of the main `Oracle DBMS`. Figure 5.4 shows this special multi-tiers architecture. Reading the Figure from top to bottom we see:

- a firewall protection to assure optimal safety of both back-end server and databases system on CERN-GPN network.
- different programs to accommodate the different consumers: from the system developer to the end-users.
- different machines/servers to balance the computational load, the usage of resources and ensure system stability.

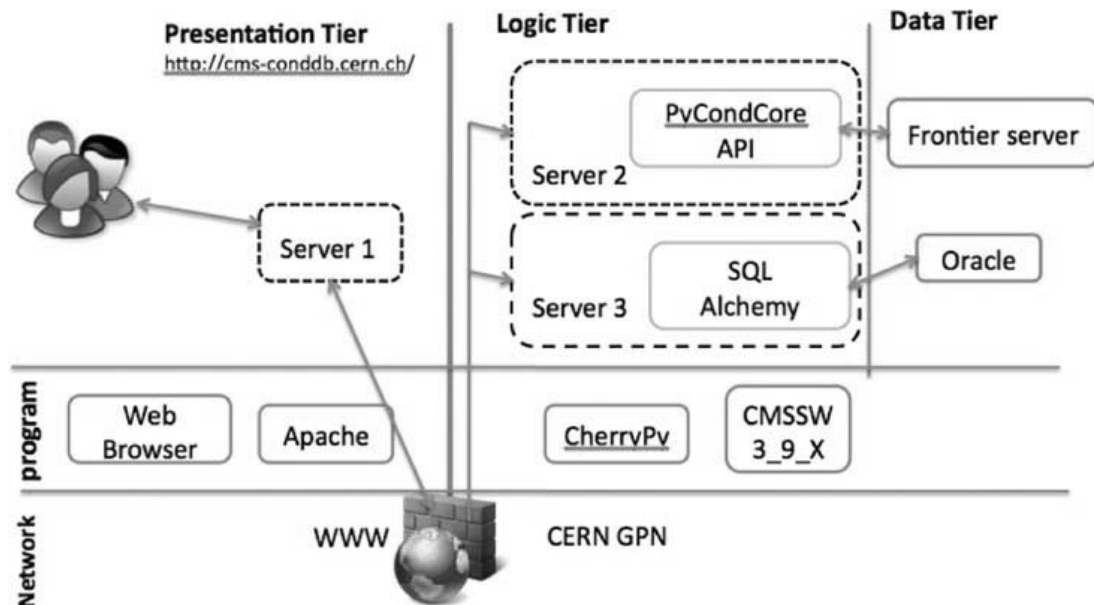


Figure 5.4: The *CMS condition DB web application* is designed as three-tier architecture that draws a clear distinction among information, logic and presentation.

The introduction of the second database, the FroNTier one, is motivated by efficiency reasons. First, because we decrease the number of connections to the main database (this is particularly interesting when this main database is located on another host than the server), replacing them with local connection to the FroNTier database. Moreover the local database is smaller than the main one, so request proceeding in this database is faster than on the main one. This approach is particularly interesting when the main database may suffer from multiple access from many users and different applications. It is of course interesting if multiple database request will require data cached parts of the users requests. This approach is subject to several restrictions and can be applied only in specific conditions: as there are two databases that are supposed to share the same information, the coherence between the two must be maintained.

The condition DB web application project fulfills all the specific conditions for using such an architecture. First, the data entries of the main database are only necessary in particular case when the FroNTier server does not respond properly, because it fails to deliver updated data or is not responsive. Secondly it must be remarked that the off-line DBMS system is a crucial element for the success of the CMS experiment due to the huge amount of detector data which needs to be handled. So in this context we try to limit the load produced by the users requests on the main database to the minimum, moving the majority of them to the FroNTier server. It seems then possible to use this ameliorated three-tiers architecture since the coherence between the two databases can be guaranteed from FroNTier services.

In addition, having the logic and data tier behind the firewall enhance security, such as using *Single Sign-On support* (SSO) for front-end part, and assuring inaccessibility for logic and data tier from outside CERN network. The policy used to deploy each tier is that the changes of a specific Tier don't affect the others Tiers. This results in high functionality, high reliability, easy debugging, low maintenance and ease of management.

This improved infrastructure was successfully deployed for 2009 collision data taking, and, after further hardware upgrades, it is now used by the whole CMS collaboration in order to monitor the correct population of the off-line condition database accounts with data coming from the ongoing 2.76 TeV Pb-Pb collisions.

CMS condition DB web application - Presentation Tier

The top-most level of the web server is the *CMS condition DB web application* front-end. The main function of this module is to translate tasks and results to something the user can understand. We use *jQuery* in order to:

- create alerts if a server or network device is off-line or to prevent and correct possible mistakes made by the users (for example inserting of wrong expressions in filter search field).
- manipulate or get information about objects in the HTML DOM. Objects in an HTML page have methods (actions, such as opening a new window or submitting a form) and properties (attributes or qualities, such as color and size).
- make a request to the server, interpret the results and update the current screen. In details, when a user selects a particular table/method the AJAX (Asynchronous JavaScript and XML) engine is called, which sends the request to the server using *XML HTTP Request*³ (XHR) object, parses the response, and updates the relevant portion of the page; therefore the whole page is not reloaded.
- finally, to create the interface table where data can be sorted, searched, and filtered [68] (see Figure 5.5).

CMS condition DB web application - Logic Tier

The *Logic Tier* is the jSpyDB back-end. This layer coordinates the application, processes commands, makes logical decisions and evaluations and performs calculations. It also moves and processes data between the two other levels, *Presentation Tier* and *Data Tier* [69].

In the bottom-most level the information is stored and retrieved from a database. The information is then passed back to the *Logic Tier* for processing, and then eventually back to the user. We use *SQLAlchemy* [70] to access the back-end database.

³XML 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.

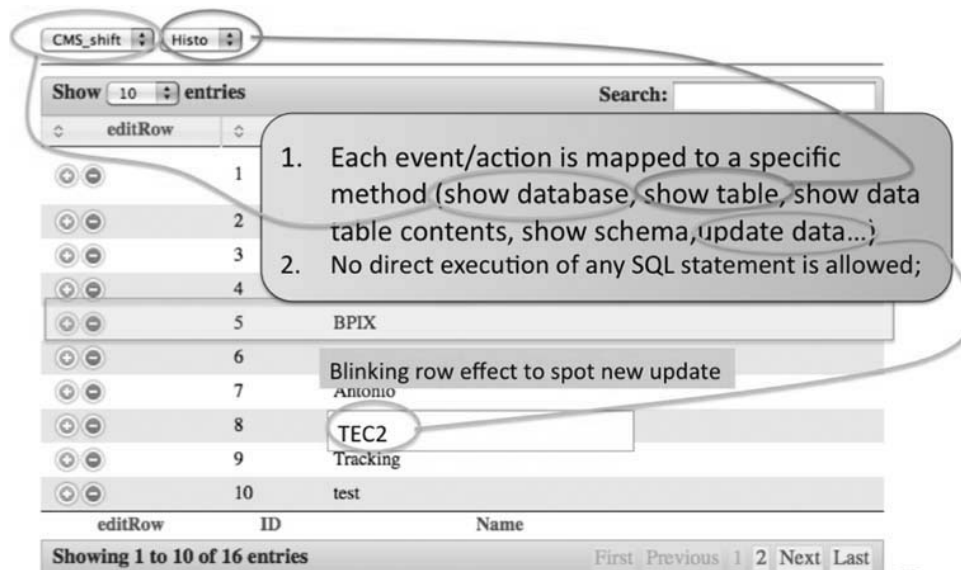


Figure 5.5: Front-end of *CMS condition DB web application*. The *jSpyDB* GUI supports a wide range of operations on data-tier without writing any SQL statements.

Thanks to SQLAlchemy we can make database-independent queries accessing relational databases such as Oracle, MySQL, PostgreSQL, and SQLite.

In detail the *Logical Tier* is responsible of the following activities.

- Filter all request from outside in order to prevent unauthorized access and inappropriate use of data.
- Mapping user requests to the handler that connects to the database and performing database calls.
- Making cache checks to allow using cache files if it is less than sixty seconds old in order to get a faster response, and to avoid overloading the back-end database systems.
- Converting data retrieved from database into JSON strings that are sent back to the *presentation Tier*.

Figure 5.6 explains the communication between the different tiers. First component is the client side, represented by a standard browser, communicating with the *Apache*

HTTP server (Apache) using the *Web Socket*. Second component is the Apache server that by means of the *Web Server Gateway Interface* (WSGI) module can communicate with *CherryPy* [71]. The last component, *CherryPy*, allows us to easily connect to Oracle database also through the CMS Software framework, thanks to its *Pythonic nature*.

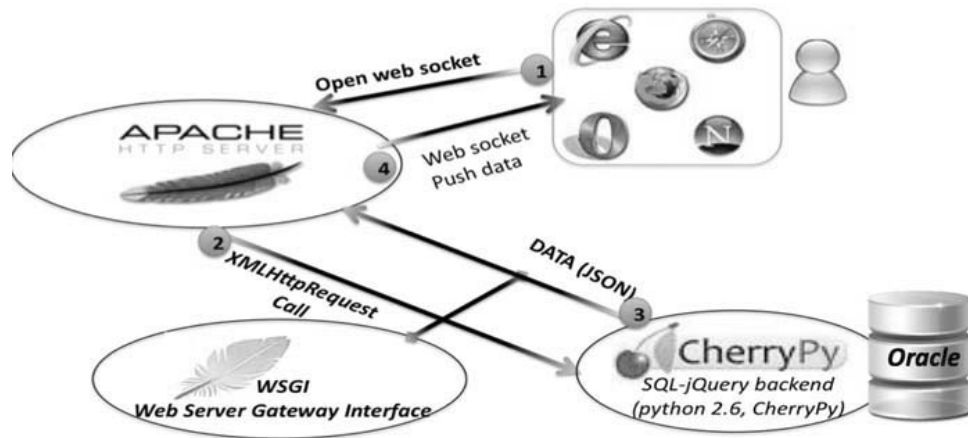


Figure 5.6: Web-request work-flow.

For the *Logic Tier*, we provide *RPM package*⁴, so that the installation could be easily automated. In case of installation on other Operating System, we provide the source files of whole *Logic Tier*; in this case the build of the executable program from source code files will be required.

The *jSpyDB logic-tier* is currently used by the following web application regarding the CMS conditions data: *popcon monitoring* [72], *Global TAG browser* [73] and finally *Payload Inspector* [74], a web application for displaying the condition data. These three web applications are useful for having a centralized control of all the work-flows for the population of the production DBMSs, and to reduce any web-related security issues,

⁴The name RPM refers to two things: software packaged in the .rpm file format, and the package manager. The package manager is a collection of software tools to automate the process of installing, upgrading, configuring, and removing software packages for a computer's operating system in a consistent manner.

CMS condition DB web application - Data Tier

The instances of the *jSpyDB* support most of the commercial and open source DBMS to handle their administration over the world wide web.

In the *jSpyDB* context the data-tier is an entity that contains all the database object and instance object which connections are used by the logical tier application.

The *CMS conditional DB web server* data-tier can settle different databases such as Oracle, MySQL, PostgreSQL and also FroNTier system to access the caching servers for the condition data of the CMS experiment.

5.5 HTML version 5 vs HTML version 4

At the time of this thesis, HTML4 had been in service for more then a decade. In this time, the World Wide Web has evolved. New and innovative websites are being created every day pushing the boundaries of HTML in every direction and really pointing out the need to upgrade the core markup language of the web to the next major revision, namely HTML5.

The HTML5 [75] reflects an effort, started in 2004, to study contemporary HTML implementations and deployed content. The draft:

- defines detailed processing models to foster interoperable implementations.
- introduces markup and APIs for emerging idioms, such as *web applications*.

In this section we focus on three APIs from HTML5, *Web Workers*, *Web Storage*, and *Web Sockets* discussing the benefits that can be gained from using them, in the context of accessing and monitoring the CMS condition database.

5.5.1 Web Workers

The next round of innovative applications will require more computing power. The hardware vendors are responding, not with faster CPUs, but with multi-cores and many-cores, often combined with *Graphics processing unit* (GPUs) and other types of accelerators [76].

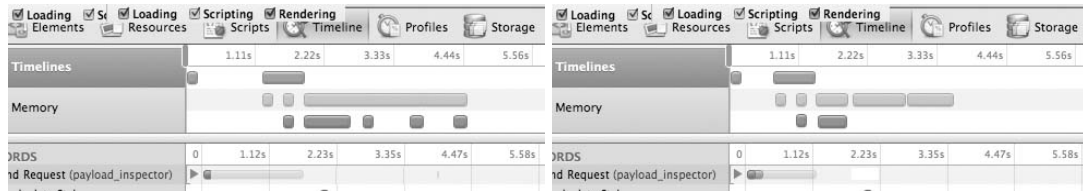
In order to exploit the capabilities of these new architectures, the programming language community has responded with new tools for parallel programming. All the traditional application-programming languages (C, C++, Java, etc.) are increasing support for parallel computations in multiple different ways through extensions, libraries, virtual execution environments, automated parallelization tools, parallel programming patterns, and so forth. Surprisingly, while these approaches have been at least partially successful in their own application domains, their adoption into the web-oriented programming environments and scripting programming languages, used by a larger and larger number of developers, had never been addressed.

Only recently, thanks to the most recent developments such as HTML5, the API *Web Workers* allow parallel processing of web client scripts. In detail, *HTML5 Web Workers* provide background-processing capabilities to web applications and typically run on separate threads so that JavaScript applications using such APIs can take advantage of multi-core CPUs.

One way *Workers* can be useful, in the context of accessing the CMS detector condition data, is to allow the code to perform processor-intensive calculations without blocking the user interface thread. In our context, each account may have over two hundred TAGs, each having a thousand IOVs for which you can get payload in different formats: XML, plot, trend plot. Executing a single-threaded JavaScript code will be very expensive in term of processing effort, and, at the same time, the user will not be able to use the GUI while the page is rendering all the contents of a specific database account. In order to avoid this bottleneck, we allow the rendering work of the HTML page to be divided among each processor through different JavaScript threads. For example, in the case of ECAL account, the page rendering of HTML elements is divided in five JavaScript threads, each run separately in series or/and in parallel according to the number of cores used.

The benchmark results comparing *Web Workers* with *single-thread JavaScript* for the *Payload Inspector* web application are presented in Figure 5.7. The web page rendering timeline depends on several factors including page size, web browser and, most of all, parallel processes running on workstation and connection speed. By using *Web Workers* it has been possible to reduce the JavaScript execution time by over 20%

and, as the processing of JavaScript thread responsible of web page rendering has been set before, the rendering process time by over 52%. The test has been performed with a 1.86 GHz Intel Core 2 Duo processor.



(a) Timeline for loading the Payload Inspector web page using a single JavaScript thread. (b) Timeline for loading the Payload Inspector web page using Web Workers.

Figure 5.7: Snapshots from *Safari developer tool* [77].

Nevertheless, such a test of the *Payload Inspector* does not provide clues to better understand CPU utilization efficiency, since, in this case, the JavaScript thread execution depends on other processes and services, amongst which network resources and connections are not completely controlled. To obtain a more reliable result about the contribution of *Web Workers*, a computation has been made by removing any dependency on external agents such as loading operations from network external resources. Such a benchmark computation has been performed on Safari 5.0 run over an Intel Core i7-860 (2.8 GHz, quad core, Hyper Threading) processor. Table 5.1 shows the results obtained by running a JavaScript algorithm [78]. The test computes $2^{2048M} \bmod(97777)$ by using the *Chinese Remainder Theorem* [79] and the *map/reduce problem* to merge all threads before computing the final result [78].

Number of workers	Average time	Improvement rate
1 Worker	20836 ms	100%
2 Workers	10920 ms	52%
4 Workers	6054 ms	29%
8 Workers	3338 ms	16%
16 Workers	3325 ms	16%

Table 5.1: Results of JavaScript benchmark processes using different numbers of Web Workers: 1,2,4,8 and 16.

The *Web Storage* specification defines an API for persistent data storage in the web applications without the need for server side storage. It means that we can store data that, assuming end-users do not clear their browser data, will be available to us the next time they come back on the client side.

One example of *Web Storage*, useful in the context of access to the CMS detector condition data, is to cache the TAG containing the full sequence of IOVs. Since in the condition database, for each account, most of the data do not change, we can store in the client disk these data and make them available to the user without getting them every time from the server. This method has several advantages:

- save network bandwidth and load on the server: every time you load a page that has some cached elements, the browser will poll the server to check if the cached file has been updated; if not, then it will not download it. By doing so, the load on the server is considerably reduced, and the bandwidth usage will be also minimized.
- reduce time response by over 90%: files that are cached locally will load much faster.

Figure 5.8 shows four snapshots from *Safari developer tool* [77]. Each pane graphs the order and speed at which website components load over the network. This tool lets you sort data based on loading parameters such as latency, response time, and duration. The Figures 5.8a, 5.8b show page resources by load time for *PopCon monitoring* [80] using HTML4 and the load time for *PopCon monitoring* using *Web Storage*, respectively. The test was performed, for both web applications, from the CERN-GPN. On the right, the Figures 5.8c, 5.8d show the page resources by size for *Payload Inspector* using HTML4 and the size for *Payload Inspector* using *Web Storage*, respectively. By using *Web Storage* we reduce network traffic by over 98% (772 KB using Web Storage vs 17.18MB using HTML4).

5.5.2 The Web Sockets API

Using HTML4, *polling operation* is the default way of getting data from the server. A client sends request to the server, and the latter responds providing the required

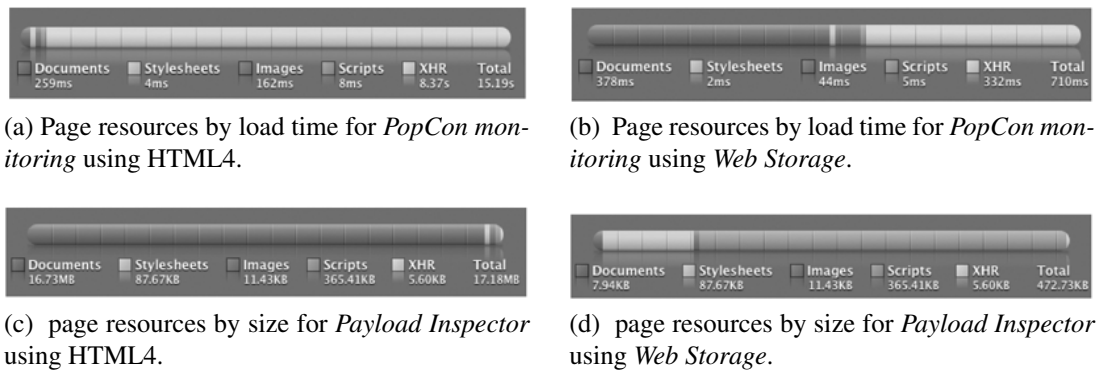


Figure 5.8: Four snapshots from *Safari developer tool*.

data. The server cannot send any data to the client by itself, the client must ask for it. This is called transitory connection, when only one side can request for data. With new web emerging technologies, more functionalities from websites are demanded, such as full communication between server and clients, or allowing server requests to the client. Such functionality was impossible to achieve in HTTP protocol without additional workarounds, because it was not designed to work in such a way. Over the years there has been various techniques developed to simulate server pushing the data to the client, like continuously polling in constant time intervals, which brings a lot of unnecessary internet traffic, and long polling - when the client polls the server but the server responds only when new data arrives or after a suitable time amount, so reducing internet traffic. Using *Web Sockets* no such workarounds are needed anymore. A continuous connection between the server and the client is created and it is not automatically closed after the server sends the data back to the client, thus allowing bidirectional communication between client and server. No headers are exchanged once the single connection has been established, server polling for checking if new data are found can be avoided. These futures reduce both internet traffic and server load.

In the context of CMS condition database we use the Web Sockets API in order to keep condition metadata updated. In detail, it works in this way: a daemon script, running on the server, looks for data changing by checking just the IOV size. If the IOV size has been changed since the previous report, the daemon script performs the

query to condition database. The daemon script sends to the client just the new IOV entries for each TAG: a few bytes are enough to keep the client updated, so that users can see the change without waiting for page refreshing.

5.5.3 Web Security

Web security experts are warning that new security issues may arise due to the spreading of new Web standard may favor security issues, this enabling a new generation of powerful Web based attacks.

Web security experts agree that the new specification will greatly increase the “attack surface” of HTML - providing more avenues by which malicious code can be delivered through the web. Since the HTML5 draft is still being continuously modified, we have decided not to put in production some of these APIs like *Web Socket*. However, since this technology applied to our development cycle has proven to increase the performance and reliability of our web application, we are ready to put this feature from HTML5 in production as soon as it becomes a standard feature in all browsers.

5.6 Open Issues

Here we list some of the problems encountered whilst still we were developing this tool.

5.6.1 Display Data Updates in Real-Time

At the moment, we are using AJAX to make the data display real-time. With the current web technology, polling the server with `Ajax` is the easiest way to retrieve the latest data and events from the server for a Web application. The Web application essentially polls the server on a regular basis, based on a timer. The freshness of the data is directly proportional to the polling frequency. The cost of polling frequently is very high, both in terms of network bandwidth as well as server infrastructure needed to support a huge number of polling requests. The server must have the capability to

buffer the data that it needs to send to the client until the next request comes from the client (see Figure 5.9). One major limitation of AJAX is that the browser must initiate the request for data from the server. The Communication section in the HTML 5 specification solves this problem by allowing the server to push data to the browser at any time.

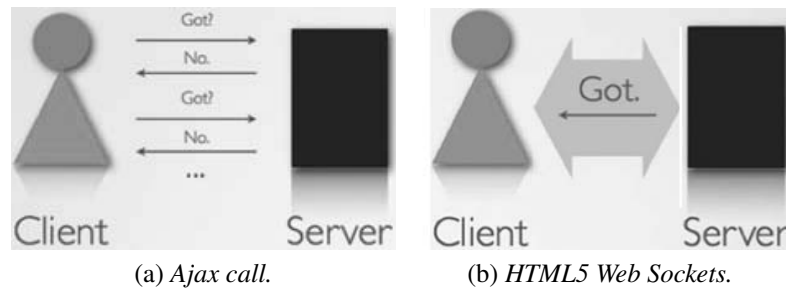


Figure 5.9: Ajax call vs Web Socket.

The Communication section in the HTML5 introduces *Server Sent Events* and *WebSockets*: these new features enable a new paradigm of Web application-programming that will revolutionize the way to develop and deploy Web applications. Thanks to WebSockets we can implement an *event-driven* communications: when an event occurs in database back-end (insert, edit, delete record in database), it implies that something changes in database back-end. In this case the *Logical Tier* triggers the client to upload the GUI with new data. Since these specifications are not yet supported in all browsers, only *Google Chrome* and *Safari* implement these APIs, we have not yet put into production. Whilst waiting for the implementation of these standards in the *Firefox 3.7* and *Internet Explorer 9*, we started, successfully, the test cycle with these APIs.

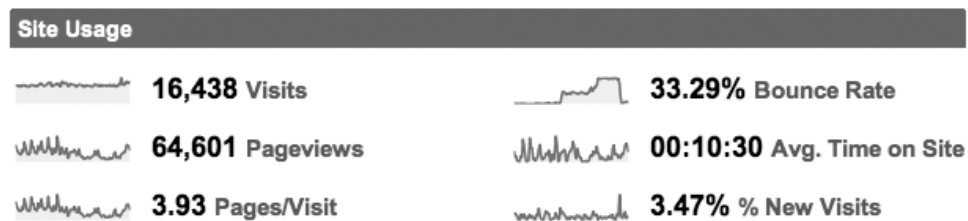
5.7 Results and conclusions

The results of using the tool in practice are displayed in the form of two simple graphs reporting the site usage statistics for the last four months. During this period the *jSpyDB logic-tier* has had an average number of contacts of ~ 180 different Internet Protocol Addresses (IPs) per day without any service interruption. Hence, the system

has proven to be reliable and very robust with high performance and availability both for GUI and database back-end. Figure 5.10a shows the number of visits per day per *CMS condition database web applications*. Figure 5.10b shows different statistics usage. Among others you can see the total number of visitors and the total page viewed.



(a) Graph showing number of visits per day.



(b) CMS condition database web usage statistics.

Figure 5.10: *Google Analytics* [81] snapshot covering the period from 3 August to 13 November 2010.

We have demonstrated with some examples that the combined use of *Web Storage* and *Web Sockets* increasing the performance and reducing the costs in term of computing power, memory usage and network bandwidth for client and server. Above all, the *Web Workers* allow creating different scripts that can be executed using multi-thread mode, exploiting modern multi-core microprocessors. *Web workers* have been employed in order to substantially decrease the web page rendering time to display the condition data stored in the CMS database accounts. Despite the significant benefits coming from these three HTML5 APIs, the web security expert warn about possible security issues related to this new technology, especially *Web Sockets*, currently under

development. For this reason in this work, we have evaluated the use of this technology without putting into production the *Web Wockets*, waiting that it passes the compatibility tests about security issues across all browsers. The other two technologies, *Web Workers* and *Web Storage*, being standardized in almost all current browsers, are already exploited successfully in the current web application accessing the off-line condition database accounts with data coming from the 7.5 TeV proton-proton collisions and the ongoing 2.76 TeV Pb-Pb collisions.

Chapter 6

Conclusion

This doctoral thesis work has been carried out in the frame of the Compact Muon Solenoid (CMS) experiment at LHC. Its focus has been the design and development of tools and services related to the control and the management of the condition data produced by the CMS experiment.

The author has been a member of the *CMS Database core project* for the last three years. He has been responsible for providing web services, taking part at the compilation of the requirements, as well as the design and development of different tools and services, for which no commercial solution can be found. Here we list the services for which he was the main contributor:

- The *PopCon monitoring*: a web-based application that collects many operational metrics from different agents; the agents are responsible for the population of condition objects and their activities are not covered by standard Oracle monitoring applications. *PopCon monitoring* can produce various graphs and specific logs to quickly identify problems.
- The *Payload Inspector*: a web service providing condition data in different formats. Examples of such formats includes:
 - two-dimensional colour map for a better understanding of the current status of a specific CMS sub-detector

- trend plot for the condition data historical representation, useful to spot the parameters changing during different data taking runs.
- The *off-line DropBox service*: a system able to load data into CMS condition master database from the off-line network. This tool is able to spot problematic dataset that could block the express data processing rejecting the calibration data which metadata formats and metadata values are not satisfactory. This tool is the only component that ensure a consistent data flow between the off-line and the on-line databases.

All these services are currently used by the CMS Collaboration.

The design of these tools has required a deep knowledge of the detector and of the various sub-detectors, as well as their behavior under different working conditions. The development and maintenance of such services required to establish a strong collaboration among the members of the various CMS sub-detectors and the central online and off-line teams, the *CERN IT Department*, and the *CMS Computer Services* (Tier-0 center and CMS CERN Analysis Facility), with clearly defined responsibilities.

The work mentioned in this doctoral thesis is acknowledged and described in several scientific papers published on physics journals and conference proceedings. Hereby the list follows of all the papers issued during the doctoral activity where the author appears as the main reference author.

- Fast access to the CMS detector condition data employing HTML5 technologies. Presented at the 18th International Conference on Computing in High Energy and Nuclear Physics (CHEP-2010), waiting for publishing on the Journal of Physics.
- jSPyDB, an open source database-independent tool for data management. Presented at CHEP-2010, waiting for publishing on the Journal of Physics.
- *SQL-jQuery Client* a Tool Managing the DBMS Back-end of the CMS Software Framework through the Web Browser. Presented at Real-Time 2010 Conference, waiting for publishing on IEEE Conference Proceedings.

- Development of tools for the automatic upload of the calibration data into the CMS condition database. Presented at the 11th ICATPP Conference, published in *Astroparticle, Particle And Space Physics, Detectors And Medical Physics Applications* - Vol.5, 2009.
- CMS conditions database web application service. Presented at the 17th International Conference on Computing in High Energy and Nuclear Physics (CHEP-2009), published in the Journal of Physics: Conf. Ser. 219 (2010).
- Web application for detailed real-time database transaction monitoring for CMS condition data. Presented at the 15th International Conference on Distributed Multimedia Systems (DMS 2009), published in the DMS 2009 Proceedings in one volume.
- Probing methods for solving version-dependent errors in CMS software. Presented at the 7th International Conference of Computational Methods in Sciences and Engineering (ICCMSE 2009), published in the Proceedings of ICCMSE 2009.

Besides of these, the author contributed in writing other 5 scientific papers following the current PhD project [85], [86], [87], [88], [89].

The tools and services developed proven to comply with all CMS specifications and have been successfully integrated in the CMS framework in 2009. Regular upgrades and improvements have been performed during the 2010, following the tight CMS software update schedule. A total of ~ 2 TBs condition object were successfully managed and certified in 2009 and ~ 3.5 TBs in 2010. The growth rate of condition data is $\sim 150GBs$ per month.

This first year of extensive data taking has also been used as a system benchmark and the condition database services lived up to the challenge. Heavily used by physicists, detector-experts and database administrators, these tools have shown their critical importance for a correct detector operation and for a reliable certification and reconstruction of data for physics analyses. The early physics results of the CMS experiment, with proton-proton collisions at 7 TeV, presented after only about 3 months in

major international conferences, such as ICHEP, are proofs of the vital contribution given by the condition database services.

Appendix A

Riassunto

A.1 Introduzione

Nel triennio 2008-2010 ho svolto la mia attività di ricerca presso il Dipartimento di Fisica di Bari e presso il CERN di Ginevra nell'ambito dell'esperimento CMS (Compact Muon Solenoid) al Large Hadron Collider (LHC).

La presenza al CERN, per brevi periodi di tempo, si è resa necessaria, in quanto membro del gruppo “DB project”. Questo gruppo, costituito per decisione del *Management Board* di CMS, ha avuto il compito di curare uno degli aspetti più critici per un esperimento di alta energia, ossia la gestione e il monitoraggio del Condition Database. Questo database contiene dati fondamentali ed indispensabili per una accurata ricostruzione ed analisi dei dati acquisiti dall'esperimento CMS e, quindi, in ultima analisi per ottenere significativi risultati di fisica.

Nel corso di questi tre anni ho sviluppato vari strumenti informatici volti alla gestione e alla validazione delle costanti di calibrazione, delle variabili di allineamento e di altre informazioni legate allo stato dei sotto-rivelatori: tali dati, detti di condizione sono indispensabili per la corretta ricostruzione delle osservabili fisiche derivanti da eventi di collisione.

Dal 2009 sono stato nominato responsabile dei tool di monitoraggio dei dati di condizione di CMS avendo messo in produzione e mantenuto diversi server in cui erano installati gli strumenti per il controllo di qualità delle costanti di calibrazione e allinea-

mento dell'esperimento CMS. Mi sono occupato sia di comprendere le potenzialità dei metodi per rilevare possibili problemi nella validazione delle costanti di calibrazione, mediante trend plot o visualizzazione interattiva di mappe 2D, sia degli aspetti tecnici legati alla necessità di elaborare una grande mole di dati.

Questi stessi lavori hanno contribuito alla stesura di diversi articoli per conferenze internazionali di computing della fisica delle alte energie [57], [60], [61], [69], [80], [83], [82], [84].

A.2 Strumenti per l'archiviazione e il monitoraggio delle costanti di calibrazione e allineamento

A.2.1 Ambiente di archiviazione dei dati di condizione

Tutti i database online di CMS sono basati sulla tecnologia ORACLE e sono dei puri database relazionali. Questi comprendono:

- il database di costruzione: dove sono immagazzinati tutti i dati raccolti in fase di costruzione e test degli apparati sperimentali,
- il database di configurazione che contiene tutte le informazioni necessarie a configurare l'elettronica di front-end, il trigger ed il sistema di acquisizione dati (DAQ),
- il database delle condizioni (condition) dove sono conservati tutti i dati relativi al funzionamento degli apparati e che sono comunemente chiamati "condition data".

Il database di calibrazione di CMS è sviluppato sulla tecnologia POOL-ORA (sviluppato al CERN) ed è un database, di solo lettura, ad oggetti basato sempre su ORACLE. Questo database è popolato da una parte dei dati online tramite *PopCon* (vedere Sezione A.2.2) più altri dati che sono prodotti da processi di calibrazioni ed analisi offline tramite un tool chiamato *DropBox offline* (vedere Sezione A.2.3). Questi dati sono poi letti ed analizzati dal Data Quality Monitoring (DQM) che produce tutta una

serie di risultati atti a verificare il corretto funzionamento dell'apparato sperimentale, con un ritardo di qualche ora rispetto alla presa dati, e che serve quindi come una ulteriore verifica, più dettagliata, da aggiungere a quelle ottenute con il DQM online.

A.2.2 PopCon

Il condition database gioca un ruolo fondamentale in quanto contiene tutti quei dati che consentono di analizzare le performance dei rivelatori e del trigger. Molti di questi vengono usati nella ricostruzione degli eventi e nell'analisi dei dati fisici, in quanto occorre conoscere lo stato di ogni sottosistema mentre vengono registrati i dati provenienti da eventi di collisione. Gran parte dei dati del condition database devono quindi essere trasferiti dal database online a quello offline in modo da essere accessibili in fase di ricostruzione ed analisi. Questo trasferimento era sempre stato fatto in modo manuale ed usando tecniche diverse dai vari sottosistemi ed è stato quindi il primo tool che abbiamo deciso di sviluppare centralmente e di fornire ai sottosistemi. Il tool, chiamato *PopCon*, sviluppato dal gruppo centrale del DBMS (Data Base Management System), è perfettamente integrato all'interno del framework offline di CMS (CMSSW). Questo strumento consente all'utente:

- di eseguire una o più queries sul database online,
- di creare uno o più oggetti contenenti i risultati dell'interrogazione,
- di assegnare un intervallo di validità (IOV) ad ogni oggetto,
- di scrivere questi oggetti nel database offline.

PopCon è attualmente usato dal 100% dei sottosistemi ed è in fase di produzione nei run di collisione protone-protone e Pb-Pb che CMS sta effettuando.

Data la complessità dell'ambiente informatico in cui questo programma doveva essere utilizzato, il framework di CMSSW, fondamentale è stato il contributo che ho dato nello sviluppo di uno strumento per il troubleshooting di questa applicazione. Questa ricerca è stata presentata e pubblicata in occasione del Symposium on Computing in Experimental High Energy Physics del 2009 [84].

A.2.3 DropBox offline

Per migliorare tutta la fase di sviluppo, test ed integrazione delle release del software di CMS senza disturbare la fase di produzione abbiamo deciso di modificare la struttura del database offline creando due ulteriori database server. Il primo può essere usato in lettura e scrittura da ogni sottosistema per verificare il funzionamento del proprio software, mentre il secondo è usato centralmente per i test di integrazione della nuova release. Solo quando anche questa seconda fase è terminata il database d'integrazione è replicato in quello di produzione dove solo i processi centrali di CMS possono accedere attraverso la *DropBox offline*. Questa nuova struttura e questo nuovo sistema, la *DropBox offline*, in uso da un anno e mezzo, hanno notevolmente semplificato diversi aspetti critici della gestione del condition database:

- rendendo sicuro il lavoro di produzione di una nuova release,
- rendendo possibile la procedura di aggiornamento dei dati di condizione prodotti da processi di calibrazioni ed analisi offline,
- riducendo al minimo la possibilità di compiere errori catastrofici dovuti all'interferenza tra il lavoro di test e quello d'integrazione di un software così complesso.

Il mio contributo a questo lavoro di ricerca è stata la progettazione di un sistema sicuro e stabile, in grado di eseguire la comunicazione, e quindi il trasferimento e aggiornamento dei dati di condizione calcolati sulla rete offline, nella rete online in cui avviene la presa dati dal rivelatore. Questo lavoro che ho presentato alla conferenza internazionale ICATPP è stato anche pubblicato sul Nuclear Instruments and Methods in Physics Research [57].

A.2.4 PopCon monitoring

Il gruppo centrale del DBMS di CMS ha sviluppato il tool *PopCon monitoring*, che consente a tutti gli utenti di monitorare il trasferimento dei dati di condizione in modo da essere accessibili in fase di ricostruzione ed analisi. Questo sistema centrale di monitoraggio ha facilitato la ricerca logica e sistematica delle cause di fallimento dei

trasferimenti di tali dati. Infatti, grazie ai vari log e alla relativa reportistica forniti da questo strumento, la maggior parte degli utenti è stata in grado di risolvere il problema facendo tornare i processi, responsabili dei trasferimenti, nuovamente operativi, contattando gli esperti di database solo in caso di problemi più complicati, quali la non corretta implementazioni di *data model* e *work-flow*. Fondamentale è stato il mio contributo nella progettazione del tool su una struttura multi-tier, sperimentando le ultime tecnologie web e utilizzando intensamente i servizi più innovativi. Anche a questo lavoro, hanno fatto seguito pubblicazioni scientifiche, che ho presentato a diverse conferenze internazionali [82], [80].

A.2.5 Payload Inspector

Il gruppo centrale del DB ha sviluppato il tool *Payload Inspector* (sempre in CMSSW), che consente a tutti gli utenti di accedere in lettura ai dati del condition database e di visualizzare i risultati (istogrammi, plot, mappe 2D, e tabelle) con un'interfaccia WEB. Questo sistema centrale di monitoraggio dei dati ha risolto sia alcuni problemi di sicurezza dei database online e offline che alcuni problemi di performance che si venivano a creare quando tutti i sottosistemi tentavano di accedere in lettura al database in modo asincrono ed usando tecniche diverse. Questo tool consente inoltre di operare un set di queries e di immagazzinare i risultati in un file XML in modo da consentire analisi offline dettagliate delle prestazioni dei vari sottosistemi ed eventualmente di esportare i dati in tutto il mondo. Anche questo tool è attualmente usato dalla gran parte dei sotto-progetti ed è in fase di produzione nei run di collisione protone-protone e Pb-Pb del 2010.

Oltre alla fase di progetto e sviluppo che ho personalmente condotto nella realizzazione e messa in produzione di questo strumento, cruciale è stata la fase di performance tuning. Dovendo il tool fornire una reportistica completa in vari formati dei dati di condizione di tutti i sotto-rivelatori, fondamentali sono state alcune personali scelte tecniche per garantire la scalabilità dello strumento al crescere dei dati raccolti dal rivelatore [83].

A.3 Conclusione

Il lavoro di questa tesi è stato svolto come membro del gruppo dei database, il cui compito è quello di coordinamento di tutti i database di CMS, online ed offline.

Il goal principale del lavoro è stato quello di uniformare in tempi rapidi i vari sotto-progetti (strutture dei database, linguaggi usati e tools) in modo da convergere verso un solo ed unico *database project*, che possa quindi essere supportato centralmente nei prossimi anni.

Il risultato finale del mio lavoro è stato la realizzazione di un sistema dalle caratteristiche uniformi, che ha dato un contributo fondamentale al processo di produzione dei primi risultati di fisica con collisioni a 7 TeV di CMS, presentati dopo soli circa tre mesi a conferenze di rilievo internazionale quali, ad esempio, ICHEP.

Appendix B

Acronyms and Abbreviations

AJAJ	Asynchronous Javascript and JSON
AJAX	Asynchronous JavaScript and XML
ALCA	<i>(CMS experiment)</i> Alignment and Calibration
ALICE	A Large Ion Collider Experiment
APD	Avalanche Photo Diode
API	Application Programming Interface
APV	Analog Pipeline Voltage
ASM	Automatic Storage Manager
ATLAS	A Toroidal LHC ApparatuS experiment
AlCaRECO	<i>(CMS experiment)</i> Alignment and Calibration Reconstructed files. The name consists of two parts: “AlCa” indicates that the events are used for alignment and calibration, and “RECO” indicates that the events are selected directly at the Reconstructed events level.
AlCaRaw	<i>(CMS experiment)</i> The name consists of two parts: “AlCa” indicates that the events are used for alignment and calibration, and “Raw” indicates that the events are selected directly at the raw data level.

AlCaSkim	(<i>CMS experiment</i>) Skim indicates that the datasets selected contains only the information relevant for calibration and alignment purposes.
Apache	Apache is an open-source web server ensuring HTTP service delivery in accordance with the latest HTTP standards, boasting a wide range of functionalities such as support for CGI (Common Gateway Interface)
BLOB	Binary Large Objects
BX	Bunch Crossing
C++	C++ is an object-oriented programming language based on the C programming language. It has support for multiple inheritance, operator overloading, templates and exception handling.
CDR	Central Data Recording
CERN-GPN	CERN General Purpose Network
CERN-IT	CERN Information Technology Services
CERN	Conseil Européen pour la Recherche Nucléaire
CLOB	Character Large Object
CMS-CAF	CMS CERN Analysis Facility
CMS-TN	CMS Technical Network
CMSSW	CMS Software Framework
CMS	Compact Muon Solenoid experiment
CORAL	COMmon Relational Access Layer CORAL defines a vendor independent API for relational database access, data and schema manipulation.

CPU	Central Processing Unit
CRUZET	Cosmic RUn at Zero Tesla
CSA06	Computing Software and Analysis Challenge 2006
CSC	<i>(CMS experiment)</i> Cathode Strip Chambers
CherryPy	CherryPy is an object-oriented web application framework using the Python programming language.
CmsRun	The CMSSW executable. It is configured at run time by the user's job-specific configuration file.
DBA	Database Administrator
DCS	<i>(CMS experiment)</i> Detector Control System
DQM	<i>(CMS experiment)</i> Data Quality Monitoring
DT	<i>(CMS experiment)</i> Drift Tubes
ECAL	<i>(CMS experiment)</i> Electromagnetic Calorimeter
EDM	<i>(CMS experiment)</i> Event Data Model
FTP	File Transfer Protocol
FroNTier	The FroNTier system distributes data from the central databases located at Tier-0 to all the Tier sites around the world and uses Squid to cache the HTTP objects at every site.
GRUMM	Global Run in Middle-March
HCAL	<i>(CMS experiment)</i> Hadronic Calorimeter
HEP	High-Energy Physics

HLT	(<i>CMS experiment</i>) High-Level Trigger
HTML	HyperText Markup Language. HTML is the predominant markup language for web pages.
HTTPS	Hypertext Transfer Protocol Secure
HTTP	HyperText Transfer Protocol
ICHEP	International Conference on High-Energy Physics
IOV	(<i>CMS condition database</i>) Interval Of Validity Since the condition data may vary over time, for instance due to changes to the detector, each condition data item has a validity range, <i>Interval of Validity</i> (IOV), with a start time (since) and an end time (till).
IP5	LHC Interaction Point 5
ISO	International Organization for Standardization
JQuery	Cross-browser JavaScript library designed to simplify the client-side scripting of HTML.
JSON	JavaScript Object Notation JSON is a lightweight data-interchange format.
JavaScript	JavaScript is scripting language supported by most web tools and browsers for use in web development. It gives HTML pages interactive functions.
KSI2K	Kilo Specmarks Integer year 2000 KSI2K is the benchmark used to measure the computing power of all the HEP experiments.
LCG	LHC Computing Grid Project
LHC-b	Large Hadron Collider beauty

LHC	Large Hadron Collider
LKCT	Lepton Kinematic Template Cones
LSF	Load Sharing Facility. LSF is a commercial computer software job scheduler.
MIT License	Free software license originating at the Massachusetts Institute of Technology (MIT)
MTCC	Magnet Test and Cosmic Challenge
MySQL	Open source relational database management system.
O2O	<i>(CMS condition database)</i> Online to Offline
OMDS	<i>(CMS condition database)</i> Online Master Database System
ORA	Object Relational Access
ORCOFF	<i>(CMS condition database)</i> Off-line Reconstruction Condition database for OFF-line use.
ORCON	<i>(CMS condition database)</i> Off-line Reconstruction Condition database for ON-line use.
PHP	Hypertext Preprocessor General-purpose scripting language that was originally designed for web development to produce dynamic web pages.
PL/SQL	Procedural Language/Structured Query Language PL/SQL is Oracle Corporation's procedural extension language for SQL and the Oracle relational database.
POOL	P ool O f persistent O bjects for L H C
PhpPgAdmin	Web application, written in PHP, for managing PostgreSQL databases

PopCon	<i>(CMS condition database)</i> Populator of Condition Objects. PopCon operates the online to off-line condition data transfer.
PostgreSQL	Open source object-relational database management system
PromptReco	<i>(CMS experiment)</i> The prompt reconstruction is one of the workows for which a low latency of code corrections is important. Fast deployment of bug xes may generally become necessary when new major releases have been deployed for the rst time, when running conditions change drastically, or in the early stage of a data taking period.
Python	Interpreted, general-purpose high-level programming language whose design philosophy emphasizes code readability. Apart from CERN, large organizations that make use of Python include Google, Yahoo! and NASA.
QCD	Quantum ChromoDynamics
RAC	Real Application Clusters
RAW	<i>(CMS experiment)</i> Raw data file
RDBMS	Relational Database Management System
RECO	<i>(CMS experiment)</i> RECO ⁿ structed Data It contains selected objects from reconstruction modules; it is still quite large, at the level of 500 kB/event.
RMS	Root mean square
ROOT	Object oriented framework for large scale data analysis.
RPC	<i>(CMS experiment)</i> Resistive Plate Chambers
S/N	Signal To Noise Ratio
SAN	Storage Area Network

SQLAlchemy	Open source SQL toolkit and object-relational mapper for the Python programming language released under the MIT License.
SQLite	SQLite is a software library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine.
SQL	Structured Query Language
SSO	Single Sign-On SSO is a property of access control of multiple, related, but independent software systems. With this property a user logs in once and gains access to all systems without being prompted to log in again at each of them.
SST	<i>(CMS experiment)</i> Silicon Strip Tracker
Squid	Squid is a caching proxy for the Web supporting HTTP, HTTPS and FTP.
TAG	<i>(CMS condition database)</i> Human-readable name which includes the CMS sub-detector element (e.g. ECAL, HCAL, RPC), the type or nature of the data (e.g. pedestal, geometry, temperature), the version of the CMSSW (e.g. 31X, 34X), a progressive version number.
TEC	<i>(CMS experiment)</i> Tracker End Cup
TIB	<i>(CMS experiment)</i> Tracker Inner Barrel
TID	<i>(CMS experiment)</i> Tracker Inner Disk
TOB	<i>(CMS experiment)</i> Tracker Outer Barrel
Timestamp	Time at which an event is recorded by a computer.
WLCG	Worldwide LHC Computing Grid

WSGI	Web Server Gateway Interface WSGI defines a simple and universal interface between web servers and web applications or frameworks for the Python programming language.
XML	Extensible Markup Language

Bibliography

- [1] The LHC Project. LHC Design Report, Volume I: the LHC Main Ring. Technical Report CERN-2004-003-V-1, CERN, Geneva, 2004.
- [2] CMS TriDAS Project, Volume 1: Trigger Technical Design Report. CERN/LHCC 2000 - 38, CMS TDR 6.1. December 15, 2000.
- [3] The ATLAS Collaboration, G Aad et al., “The ATLAS Experiment at the CERN Large Hadron Collider”, Journal of Instrumentation 3 (S08003): S08003. doi:10.1088/1748-0221/3/08/S08003, 2008.
- [4] K. Aamodt et al. (ALICE collaboration), “The ALICE experiment at the CERN LHC”, Journal of Instrumentation 3 (8): S08002. doi:10.1088/1748-0221/3/08/S08002, 2008.
- [5] A. Augusto Alves Jr. et al. (LHCb Collaboration), “The LHCb Detector at the LHC”, Journal of Instrumentation 3: S08005. doi:10.1088/1748-0221/3/08/S08005, 2008.
- [6] W. Venturini-Delsolaro, “Status of the LHC Machine”, paper presented at this Conference.
- [7] M. Jeitler for the CMS Collaboration, “The CMS experiment: status, performance and selected results”, CMS CR -2010/271, 2010.
- [8] L. Scodellaro, “Top quark physics at CMS”, paper presented at IHEP - Protvino - Russian Federation, 2010.

-
- [9] D. Giordano for CMS collaboration. “Tracking Performance at CMS”, paper presented at Moriond/QCD: XLVth Rencontres de Moriond on QCD and High Energy Interactions, CMS CR-2010/057, 2010.
- [10] M. Hoch., “CMS Silicon Strip Tracker Operations and Performance”, Contributed given at IEEE2010: 2010 Nuclear Science Symposium, Medical Imaging conference and RTDS workshop, Knoxville, TN (United States), 30 Oct. Nov 2010.
- [11] CMS Collaboration, “The Hadron Calorimeter Project Technical Design Report”, CERN/LHCC 97-31 (1997).
- [12] CMS Collaboration, “The Electromagnetic Calorimeter Technical Design Report”, CERN/LHCC 97-33 (1997).
- [13] P. Gras on behalf of the CMS Collaboration, “Commissioning and performance of the CMS calorimeter system with proton-proton collisions at the LHC”, Presented at ICHEP2010, CMS CR-2010/272.
- [14] The CMS Electromagnetic Calorimeter Group, “Intercalibration of the barrel electromagnetic calorimeter of the CMS experiment at start-up”, 2008 JINST 3 P10007.
- [15] The CMS Collaboration, “Electromagnetic calorimeter calibration with 7 TeV data”, PAS EGM-2010-003 (2010).
- [16] The CMS Collaboration, “Performance of the CMS hadron calorimeter with cosmic ray muons and LHC beam data”, CMS Collaboration 2010 JINST 5 T03012.
- [17] The CMS Collaboration, “Jet Performance in pp Collisions at $\sqrt{s} = 7$ TeV”, CMS PAS JME-10-003 (2010).
- [18] The CMS Collaboration, “Measurement of the Jet Energy Resolutions and Jet Reconstruction Efficiency at CMS”, CMS PAS JME-09-007.

- [19] The CMS collaboration, “Performance of the CMS drift-tube chamber local trigger with cosmic rays”, CMS Collaboration 2010 JINST 5 T03003.
- [20] D. Acosta et al., “Large CMS cathode strip chambers: design and performance”, Nucl. Instrum. Meth. A 453 182, 2000.
- [21] Z. Jaworskia, I. M. Kudlab, W. Kuzmicza and M. Niewczasa, “Resistive Plate Chamber (RPC) based Muon Trigger System for the CMS experiment - Pattern Comparator ASIC”, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment Volume 419, Issues 2-3, Pages 707-710, 21 December 1998.
- [22] The CMS collaboration, “Performance of muon identification in pp collisions at $\sqrt{s} = 7\text{TeV}$ ”, CMS PAS MUO-10-002.
- [23] The CMS Collaboration, GRUMM analysis page,
<https://twiki.cern.ch/twiki/bin/viewauth/CMS/GRUMMAnalysis>
- [24] The CMS Collaboration, CRUZET analysis page,
<https://twiki.cern.ch/twiki/bin/viewauth/CMS/CRUZETAnalysis>
- [25] International CMS Workshop on cosmic ray data analysis - CRAFT 2009, Torino, Italy, 11-13 Mar 2009.
- [26] Alcaraz, J (CIEMAT Madrid) et al., “CSA06 Computing, Software and Analysis challenge at the Spanish Tier-1 and Tier-2 sites”, CMS-NOTE-2007-022 ; CERN-CMS-NOTE-2007-022, 2007.
- [27] The CMS Collaboration, The Computing Software and Analysis Challenge 2007 page, <https://twiki.cern.ch/twiki/bin/view/CMS/CSA07>.
- [28] The CMS Collaboration, “CMS Physics TDR, Volume I: Detector Performance and Software”, Technical Report CERN-LHCC-2006-001; CMSTDR-008-1, CERN, Geneva, 2006.

-
- [29] Simone Frosali, on behalf of CMS Silicon Strip Tracker, “The CMS tracker calibration workflow: experience with cosmic ray data”, J. Phys.: Conf. Ser. 219 032040, 2010.
- [30] The CMS Collaboration, “The CMS experiment at the CERN LHC”, 2008 JINST 3 S08004.
- [31] Dave Dykstra and Lee Lueking, “Greatly improved cache update times for conditions data with FronTier/Squid”, CHEP2009 Proceedings.
- [32] Rene Brun and Fons Rademakers, “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/>.
- [33] Papadopoulos, I. et al., “POOL development status and plans”, CHEP2004 Proceedings.
- [34] Oracle Database - <http://www.oracle.com/>.
- [35] MySQL Home Page, www.mysql.com/
- [36] SQLite Home Page, <http://www.sqlite.org/>
- [37] M. De Gruttola, S. Di Guida, D. Futyan, F. Glege, G. Govi, V. Innocente, P. Paolucci, P. Picca, G. A. Pierro, D. Schlatter, Z. Xie, “Persistent storage of non-event data in the CMS databases”, Published in JINST 5:P04003, 2010.
- [38] G. Govi, V. Innocente, Z. XIE. CMS Offline Conditions Framework and Services. 2010 J. Phys.: Conf. Ser. 219 042027.
- [39] C.D. Jones 2006, “Access to non-event data for CMS”, Proc. Conf. for Computing in High Energy and Nuclear Physics(Mumbai).
- [40] M. Janulis et al. “CMS Online Database Experience with First Data”, CMS CR-2010/218.

- [41] Timciuc, Vladlen (Caltech), “Database usage for the CMS ECAL Laser Monitoring System”, CMS CR-2009/159.
- [42] V. Timciuc et al. “Database usage in the CMS ECAL laser monitoring system” 2010 J. Phys.: Conf. Ser. 219 022045.
- [43] C.Grandi, D.Stickland, L.Taylor et al. “The CMS Computing Model”, CERN-LHCC-2004-035/G-083, 2004.
- [44] CERN batch service Site <http://batch.web.cern.ch/batch/>.
- [45] Lo Presti, G. Barring, O, Earl, A, Garcia Rioja, R.M, Ponce, S, Taurelli, G, Waldron, D, Coelho Dos Santos, M. “CASTOR: A Distributed Storage Resource Facility for High Performance Data Processing at CERN”, 24th IEEE Conference on Mass Storage Systems and Technologies, 2007. MSST 2007.
- [46] Worldwide LHC Computing Grid Technical Site. <http://lcg.web.cern.ch/lcg/>.
- [47] G. Cerminara for the CMS Collaboration, “Operation of the CMS detector with first collisions at 7 TeV at the LHC”, CMS CR-2010/264.
- [48] The CMS collaboration (G.L. Bayatian et al.), CMS technical design report, volume II: Physics performance, J.Phys.G34:995-1579,2007.
- [49] D. Futyan for the CMS Collaboration, “Commissioning the CMS alignment and calibration framework”, Journal of Physics: Conference Series 219 (2010) 032041.
- [50] The CMS Collaboration, “Electromagnetic calorimeter calibration with 7 TeV data”, CMS-PAS-EGM-10-003.
- [51] S. Argiro, “Triggers and streams for calibration in CMS”, Proceedings of the International Conference on Computing in High Energy and Nuclear Physics (CHEP2010), 18-22 Oct 2010, Taipei, Taiwan.

-
- [52] T. Lampen, “CMS Silicon Strip Tracker calibration work-flow and tools”, Proceedings of the International Conference on Computing in High Energy and Nuclear Physics (CHEP2010), 18-22 Oct 2010, Taipei, Taiwan.
- [53] R. Mankel for the CMS collaboration, “Alignment and calibration experience under LHC data-taking conditions in the CMS experiment”, 2010 J. Phys.: Conf. Ser. 219 032041.
- [54] The OpenMP API specification for parallel programming Home page, <http://openmp.org>.
- [55] D. Bonacorsi, From Commissioning to Collisions: Preparations and Execution of CMS Computing. IPRD 2010, Siena, Italy - 08 June 2010.
- [56] Govi, G. et al., “POOL integration into three experiment software frameworks”, Computing in High Energy Physics and Nuclear Physics 2004, Interlaken, Switzerland, 27 Sep - 1 Oct 2004, pp.479.
- [57] G. A. Pierro, I. Kuzborskij, S. Di Guida, V. Innocente, “Development of Web Tools for the automatic Upload of Calibration Data into the CMS Condition Database”, Published in Astroparticle, Particle And Space Physics, Detectors And Medical Physics Applications (pp 724-728), 2009.
- [58] CMS Computing TDR, CERN-LHCC-2005-023, <http://cdsweb.cern.ch/record/838359> 20 June 2005.
- [59] The CMS Collaboration, CMS Physics Technical Design Report, Volume II: Physics Performance. J. Phys. G, 34(6):995–1579, 2007.
- [60] G. A. Pierro, K. M. Dziedziniwicz, D. Giordano, V. Innocente, A.-C. Le Bihan, “CMS conditions database web application service”, Published in J. Phys.: Conf. Ser. 219 072048, 2009.
- [61] A. Pierro, S. Di Guida, V. Innocente, “A machine learning approach to error detection in a database transaction system”, Presented at ICCMSE 2009: Symposium on Computing in Experimental High Energy Physics, CMS CR-2010/024.

- [62] S. Di Guida, F. Cavallari, V. Innocente and G. A. Pierro, “Real time monitoring system for applications performing the population of Condition Database for CMS non-event data”, CMS CR-2010/072, 2010.
- [63] Maria S.Mennea, A. Regano, G. Zito, “CMS tracker visualization tools”, Published in Nuclear Inst. and Methods in Physics Research A, Vol 548/3 pp 391-400, 2005.
- [64] phpPgAdmin: a web-based administration tool for PostgreSQL,
<http://phppgadmin.sourceforge.net/>
- [65] PhpMyAdmin: a free software tool written in PHP intended to handle the administration of MySQL over the World Wide Web. www.phpmyadmin.net
- [66] Oracle SQL Developer: a free and fully supported graphical tool for database development, <http://www.oracle.com/>
- [67] A. Tuininga - cx-Oracle, July 13, 2010. <http://cx-oracle.sourceforge.net/>.
- [68] A. Jardine - DataTables, <http://www.datatables.net/>.
- [69] G. A. Pierro, F. Cavallari, S. Di Guida, V. Innocente, “CMS DB Client Browser - An Open Source Tool Managing the DB Backend of the CMS Software Framework Through Web Browser”, Published in Conference Proceeding of the 17th IEEE NPSS Real Time Conference, 2010.
- [70] Sqlalchemy: The Python SQL Toolkit and Object Relational Mapper.
<http://www.sqlalchemy.org/>.
- [71] CherryPy: a pythonic, object-oriented HTTP framework.
<http://www.cherrypy.org/>.
- [72] M. De Gruttola, S. Di Guida, D. Futyan, F. Glege, G. Govi, V. Innocente, P. Paolucci, G. A. Pierro, D. Schlatter, “First experience in operating the population of the condition database for the CMS experiment”, Published in J. Phys.: Conf. Ser. 219 042046, 2010.

-
- [73] Database group of CMS Offline project, Global tag Browser for CMS Condition Database, Home page. <http://cms-conddb.cern.ch/gtlist/>
- [74] Database group of CMS Offline project. “CMS Payload Inspector”: a web application for displaying the condition data.
http://cms-conddb.cern.ch/payload_inspector/
- [75] HTML5 - W3C Working Draft
<http://dev.w3.org/html5/spec/Overview.html>
- [76] Data Parallel Programming for the Emerging Web - Richard L. Hudson, Tatiana Shpeisman, Adam Welc, Ali-Reza Adl-Tabatabai - Intel Labs
- [77] Safari developer tools.
<http://developer.apple.com/>
- [78] Javascript Web Workers Test v1.4.0 - Pedro Verruma
<http://pmav.eu/stuff/javascript-webworkers/>
- [79] Kak, Subhash (1986), “Computational aspects of the Aryabhata algorithm”, Indian Journal of History of Science 21 (1): 6271.
- [80] G. A. Pierro, I. Butenas, S. Di Guida, V. Innocente, “PopCon Monitoring: Web Application for Detailed Real-time Database Transaction Monitoring”, Published in Conference Proceeding of the 15th International Conference on Distributed Multimedia Systems, vol.1, 2009.
- [81] Google Analytics Home page, <http://www.google.com/analytics/>
- [82] G. A. Pierro, F. Cavallari, S. Di Guida, V. Innocente, “Fast access to the CMS detector condition data employing HTML5 technologies”, Published in Conference Proceeding of the 18th International Conference on Computing in High Energy and Nuclear Physics, CMS CR-2010/222, 2010.

- [83] G. A. Pierro, F. Cavallari, S. Di Guida, V. Innocente, “jSPyDB an open source database-independent tool for data management”, Published in Conference Proceeding of the 18th International Conference on Computing in High Energy and Nuclear Physics, CMS CR-2010/279, 2010.
- [84] G. A. Pierro, S. Di Guida, V. Innocente, “Probing methods for solving version-dependent errors in CMS software”, Published in Conference Proceedings of IC-CMSE 2009, Symposium on Computing in Experimental High Energy Physics, CMS CR-2010/024, 2010.
- [85] S. Di Guida, V. Innocente, F. Cavallari, G. A. Pierro, “Time-critical database condition data handling in the CMS during the first data taking period”, Proceedings of the International Conference on Computing in High Energy and Nuclear Physics (CHEP2010), 18-22 Oct 2010, Taipei, Taiwan.
- [86] M. De Gruttola, S. Di Guida, V. Innocente and G. A. Pierro, “Time-Critical Database Conditions Data-Handling for the CMS Experiment”, CMS CR-2010/144, 2010.
- [87] S. Di Guida, M. De Gruttola, V. Innocente, F. Cavallari, G. A. Pierro, “Real Time Monitoring System for Applications Performing the Population of Condition Database for CMS Non-Event Data”, Published in Conference Proceeding of the 17th IEEE NPSS Real Time Conference, 2010.
- [88] M. De Gruttola, S. Di Guida, V. Innocente and G. A. Pierro, “Web application for detailed real-time database transaction monitoring for CMS condition data”, CMS CR-2010/021.
- [89] M. De Mattia, D. Giordano, A.-C. Le Bihan, G. A. Pierro, “Historic Plotting Tool for Data Quality Monitoring”, Published in Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment Volume 617, Issues 1-3, Pages 263-265, 2010.

Acknowledgments

The completion of this thesis would not have been possible without the guidance and the help of several people who in one way or another contributed with their valuable assistance to the preparation and completion of this work.

First and foremost, my utmost gratitude to my supervisors, Prof. Giorgio Pietro Maggi, Dr. Vincenzo Innocente and Dr. Domenico Giordano, whose good teaching, sound advices, sincerity and encouragement I will never forget.

Deep gratitude goes as well to the CMS Offline Coordinator, Dr. Lucia Silvestris, who gave me the possibility to work at CERN on the project that led to my current thesis.

I am indebted to many of my colleagues for their support. I would like to express my gratitude to the CMS condition database group, and in particular to the group convener, Dr. Francesca Cavallari, and to the responsible of the database operation, Dr. Salvatore Di Guida. I have always been happy to see their interest in my projects. The regular meetings had brought valuable discussions which allowed me to see different aspects of my research in a broader perspective.

Then I would like to thank students in Computer Science and Software Engineering from Vilnius University, Ignas Butenas, Ilja Kuzborskij, Nikolaj Kondrat, Algirdas Beinaravicius, Martynas Pumputis, Gediminas Mazrimas, Yan Rymarchik, Mindaugas Polujanskas and Saulius Bucka for their collaboration in this project and for having shared together our ideas.

I am deeply indebted to Dr. Nicola De Filippis for reading the first chapter of this thesis and for his observations and suggestions.

I wish to thank my dear friends Guido, Raffaello, Salvatore and Giovanni for help-

ing me getting through some difficult times and for all the emotional support, comradeship and entertainment they provided.

Last but not least, I wish to thank my girlfriend Gaia, my parents, my sister, my nephew Gaetano and my nice Chiara. They bore me, raised me, supported me, taught me, and loved me. This thesis is dedicated to them.