Alma Mater Studiorum · Università di Bologna

Scuola di Scienze
Dipartimento di Fisica e Astronomia
Corso di Laurea Magistrale in Fisica

Performance Studies of CMS workflows using Big Data technologies

Relatore:
Prof. Daniele Bonacorsi

Correlatore:
Dott. Claudio Grandi

Presentata da:
Luca Ambroz

Sessione I

Anno Accademico 2015/2016

Abstract

At the Large Hadron Collider (LHC), more than 30 petabytes of data are produced from particle collisions every year of data taking. The data processing requires large volumes of simulated events through Monte Carlo techniques. Furthermore, physics analysis implies daily access to derived data formats by hundreds of users. The Worldwide LHC Computing Grid (WLCG) - an international collaboration involving personnel and computing centers worldwide - is successfully coping with these challenges, enabling the LHC physics program. With the continuation of LHC data taking and the approval of ambitious projects such as the High-Luminosity LHC, such challenges will reach the edge of current computing capacity and performance. One of the keys to success in the next decades - also under severe financial resource constraints - is to optimize the efficiency in exploiting the computing resources.

This thesis focuses on performance studies of CMS workflows, namely centrally-scheduled production activities and unpredictable distributed analysis. The work aims at developing and evaluating tools to improve the understanding of the monitoring data in both production and analysis. For this reason, the work comprises two parts. Firstly, on the distributed analysis side, the development of tools to quickly analyze the logs of previous Grid job submissions can enable a user to tune the next round of submissions and better exploit the computing resources. Secondly, concerning the monitoring of both analysis and production jobs, commercial Big Data technologies can be used to obtain more efficient and flexible monitoring systems. One aspect of such improvement is the possibility to avoid major aggregations at an early stage and to collect much finer granularity monitoring data which can be further processed at a later stage, just upon request.

In this thesis, a work towards both directions is presented. Firstly, a lightweight tool to perform rapid studies on distributed analysis performances is presented as a way to enable physics users to smoothly match the job submission tasks to changing conditions of the overall environment. Secondly, a set of performance studies on the CMS Workflow Management and Data Management sectors are performed exploiting a CMS Metrics Service prototype, based on ElasticSearch/Jupyter Notebook/Kibana technologies, that contains high granularity information on CMS production and analysis jobs, exploiting the HTCondor ClassAdds.

Chapter 1 provides an overview of the Standard Model. Chapter 2 discusses the LHC accelerator complex and experiments with main focus on CMS. Chapter 3 introduces Computing in High Energy Physics and describes the CMS Computing Model. Chapter 4 presents the development of an original tool for evaluating the performances of local analysis jobs. Chapter 5 describes how data from the CMS Metrics Service can be analyzed to provide insights on the CMS global activities.

Sommario

Al Large Hadron Collider (LHC) ogni anno di acquisizione dati vengono raccolti piú di 30 petabyte di dati dalle collisioni. Per processare questi dati è necessario produrre un grande volume di eventi simulati attraverso tecniche Monte Carlo. Inoltre l'analisi fisica richiede accesso giornaliero a formati di dati derivati per centinaia di utenti. La Worldwide LHC Computing GRID (WLCG) è una collaborazione interazionale di scienziati e centri di calcolo che ha affrontato le sfide tecnologiche di LHC, rendendone possibile il programma scientifico. Con il prosieguo dell'acquisizione dati e la recente approvazione di progetti ambiziosi come l'High-Luminosity LHC, si raggiungerá presto il limite delle attuali capacitá di calcolo. Una delle chiavi per superare queste sfide nel prossimo decennio, anche alla luce delle ristrettezze economiche dalle varie funding agency nazionali, consiste nell'ottimizzare efficientemente l'uso delle risorse di calcolo a disposizione.

Il lavoro mira a sviluppare e valutare strumenti per migliorare la comprensione di come vengono monitorati i dati sia di produzione che di analisi in CMS. Per questa ragione il lavoro è comprensivo di due parti. La prima, per quanto riguarda l'analisi distribuita, consiste nello sviluppo di uno strumento che consenta di analizzare velocemente i log file derivanti dalle sottomissioni di job terminati per consentire all'utente, alla sottomissione successiva, di sfruttare meglio le risorse di calcolo. La seconda parte, che riguarda il monitoring di jobs sia di produzione che di analisi, sfrutta tecnologie nel campo dei Big Data per un servizio di monitoring piú efficiente e flessibile. Un aspetto degno di nota di tali miglioramenti è la possibilitá di evitare un'elevato livello di aggregazione dei dati giá in uno stadio iniziale, nonché di raccogliere dati di monitoring con una granularitá elevata che tuttavia consenta riprocessamento successivo e aggregazione "on-demand".

In questa tesi viene presentato un lavoro che riflette queste due visioni. In primo luogo, viene sviluppato un strumento leggero per effettuare veloci studi di performance per analisi distribuite al fine di coniugare, nel miglior modo possibile, le sottomissioni di job da effettuare con le strutture di calcolo a disposizione. In secondo luogo, sono stati effettuati una serie di studi di performance sui settori di CMS Workload Management e Data Management, sfruttando un prototipo chiamato "CMS Metrics Service" basato su tecnologie ElasticSearch/Jupyter Notebook/Kibana, il quale contiene informazioni ad alta granularitá sui job di analisi e produzione di CMS (mediante HTCondor ClassAdd).

Il Capitolo 1 fornisce una visione generale del Modello Standard delle particelle elementari. Il Capitolo 2 presenta l'acceleratore LHC e i principali esperimenti all'acceleratore, focalizzandosi in particolare su CMS. Il Capitolo 3 discute le problematiche legate al Computing nella settore della fisica delle alte energie, in particolare in CMS. Il Capitolo 4 presenta lo sviluppo di uno strumento originale per valutare, localmente, le performance di job di analisi. Il Capitolo 5 descrive in dettaglio come i dati presenti nel CMS Metrics Service possano essere analizzati per fornire informazioni preziose sulle performance delle attivitá globali di CMS, e guidare scelte future.

Contents

1 The Standard Model **8**
1.1 Particles of the Standard Model . . . 8
1.2 The fundamental interactions . . . 10
1.2.1 The electromagnetic interaction . . . 11
1.2.2 The strong interaction . . . 12
1.2.3 The electroweak interaction . . . 15

2 High Energy Physics at the LHC **17**
2.1 The LHC accelerator at CERN . . . 17
2.2 The experiments at the LHC . . . 21
2.2.1 ALICE . . . 21
2.2.2 ATLAS . . . 23
2.2.3 CMS . . . 24
2.2.4 LHCb . . . 24
2.2.5 Other experiments at LHC . . . 25
2.3 The CMS detector . . . 26
2.3.1 Tracker . . . 28
2.3.2 Electromagnetic Calorimeter . . . 29
2.3.3 Hadron Calorimeter . . . 30
2.3.4 Magnet . . . 31
2.3.5 Muon detector . . . 32
2.3.6 Trigger . . . 33

3 Computing in High Energy Physics **35**
3.1 Introduction . . . 35
3.2 Grid technologies and WLCG . . . 35
3.3 The CMS Computing model . . . 38
3.4 CMS computing services . . . 40
3.5 CMS Distributed Analysis with CRAB . . . 41

4 Analysis performances: a CRAB-based tool 44
4.1 Design . . . 44
4.2 Single task analysis . . . 46
4.3 Multiple Task Analysis . . . 51
4.4 Job submissions for benchmarking . . . 53

5 Performance studies of CMS workflows using ElasticSearch, Jupyter Notebook and Kibana 54
5.1 Big Data Analytics in CMS . . . 54
5.2 Hadoop . . . 56
5.3 ElasticSearch and Kibana . . . 57
5.4 The CMS Metrics Service from University of Nebraska . . . 57
5.5 Workload Management view . . . 59
5.5.1 Use-case: jobs count across production and analysis . . . 61
5.5.2 Use-case: total core hours per production campaign . . . 65
5.5.3 Use-case: jobs count on many computing centers . . . 66
5.5.4 Use-case: campaign-based job failure modes . . . 72
5.5.5 Use-case: campaign-based CPU efficiency . . . 76
5.5.6 Performance comparison of two Tier-2: CERN HLT and IT Legnaro 82
5.6 Data Management view . . . 84
5.7 Quasi real-time monitoring . . . 89

A Important ElasticSearch attributes for the CMS implementation 93

B Example of Elasticsearch query 97

List of Figures

1.1 Fundamental vertex of the QED. 11
1.2 Screening mechanism in Quantum ElectroDynamics for an electron (top) and α_{EM} as a function of probe energy (bottom). 12
1.3 Fundamental vertexes of the QCD. 13
1.4 Coupling constants $\alpha_{QCD} \equiv \alpha_s$ and α_{QED} as a function of the quadratic transferred four-momentum Q^2 [5]. 14
1.5 Examples of fundamental vertexes of the weak interation. 15
1.6 Production of the Higgs boson at a hadron accelerator. 16

2.1 The LHC collider inside the underground tunnel (copyright CERN). . . . 17
2.2 Scheme of the acceleration complex (copyright CERN). 18
2.3 Cross section of LHC dipole (copyright CERN). 19
2.4 Experiments at the LHC (copyright CERN). 21
2.5 Picture of the ALICE detector under maintenance (copyright CERN). . . 22
2.6 Picture of the ATLAS detector under construction (copyright CERN). . . 23
2.7 Picture of the CMS detector while open (copyright CERN). 24
2.8 Picture of the CMS detector while open (copyright CERN). 25
2.9 Section of the CMS detector (copyright CERN). 27
2.10 Left: CMS Tracker stand−alone p_t resolution for muons, without beam constraint. Right: Combined CMS Tracker and Muon Chamber p resolution, in the central and forward rapidity regions of the detector [27]. 29
2.11 Artistic representation of a 20° wedge. The ECAL is on the right and HCAL on the left (copyright CERN). 31
2.12 A schematic view of a muon trajectory inside the detector (copyright CERN). 32
2.13 A schematic representation of the dataflow in the CMS Trigger. The two consecutive steps are highlighted: the L1 Trigger and the HLT (copyright CERN). 33

3.1 The Tier structure of the WLCG sites [36] 37
3.2 Data flow in the CMS computing model. 39

3.3 Overview of the CMS Computing Services [46]. 40
3.4 Overview of the CRAB architecture [58]. 43

4.1 Tool design. The tool performs two types of analysis on the CMS working directory: a single task analysis for each task individually, and a multiple task analysis which looks at each task as a whole. More details will be provided in Sections 4.2 and 4.3. 45
4.2 Number of jobs as a function of the CPU Time for a workflow that is described in detail in the text. 47
4.3 Number of jobs as a function of the Execution Time for a workflow that is described in detail in the text. 47
4.4 Number of jobs as a function of the CPU Efficiency for a workflow that is described in detail in the text. 48
4.5 Occurrences of the number of events for a workflow that is described in detail in the text. 49
4.6 Execution Time as a function of the CPU Time for a workflow that is described in detail in the text. 50
4.7 Execution Time as a function of the number of events for a workflow that is described in detail in the text. 50
4.8 Averages of CPU Time for four related workflows (the error is the standard deviation). 51
4.9 Averages of Execution Time for four related workflows (the error is the standard deviation). 52
4.10 Averages of CPU Efficiency for four related workflows (the error is the standard deviation). 52
4.11 Averages of number of events for four related workflows (the error is the standard deviation). 53

5.1 Schematic representation of the structured and unstructured metadata in CMS [64]. 55
5.2 Diagram of the component architecture of Nebraska / CMS Metrics Service. 58
5.3 Daily sum of Core Hours for different Site WMS. 60
5.4 Job count for the 10 most relevant “processing activities”. See text for further explanations. Blue bars refer to user analysis activities and yellow bars refer to different Monte Carlo production campaigns. 61
5.5 Total Core Hours for jobs of production and analysis in May 2016. 63
5.6 RunIISpring16DR80 campaign progress over time. 64
5.7 RunIISummer15GS campaign progress over time. 64
5.8 Total Core Hours for the major production campaigns ongoing in May 2016. 65

5.9 Fraction of jobs in the WLCG Sites that contributed the most to the CMS experiment activities (both production and analysis) in the time frame under study. 67
5.10 Total number of completed jobs, with breakdown on the computing sites. 69
5.11 Total Core Hours for the major computing centers, showing also the fraction of the job succeeded and failed. 69
5.12 Total number of completed jobs, with breakdown on the computing sites, showing also the fraction of production versus analysis jobs. 70
5.13 Relative fraction of production and analysis jobs at different Tier levels over the time frame under study. 71
5.14 The five most frequent exit codes of CMSSW jobs belonging to the RunIISpring16DR80 campaign, for jobs that ran on-site (note the log scale). . 73
5.15 The five most frequent exit codes of CMSSW jobs belonging to the RunIISpring16DR80 campaign, for jobs that ran off-site (note the log scale). . 74
5.16 The five most frequent exit codes of CMSSW jobs belonging to the RunIISummer15wmLHEGS campaign, for jobs that ran on-site (note the log scale). 75
5.17 Average CPU efficiency of various CMS workflows over the last 12 months (source: CMS Dashboard). 77
5.18 Fraction of Wall Clock Hours for the different Task Type for the RunIISpring16DR80. 78
5.19 CPU efficiency in intervals of 10% shown in terms of wall clock hours for the RunIISpring16DR80 campaign. 79
5.20 CPU efficiency in intervals of 10% shown in terms of wall clock hours for the RunIISummer15wmLHEGS campaign. 80
5.21 CPU efficiency in intervals of 10% shown in terms of wall clock hours. It is also displayed whether the jobs where access on the desired DataLocations (on-site: green) or not (off-site: yellow). 81
5.22 Total Core Hours as function of time for the Tier-2 CERN HLT and Tier-2 It Legnaro in the time frame under study. 82
5.23 Average daily CPU efficiency for the Tier-2 CERN HLT and Tier-2 It Legnaro in the time frame under study. 83
5.24 Total data volume in input to CMSSW jobs for the major production campaigns over last 3 months. 84
5.25 Total data volume in input to CMSSW jobs, with breakdown on the countries (note the log scale). 86
5.26 Total data volume in input to CMSSW jobs, with breakdown on the Tier levels (note the log scale). 86
5.27 Total data volume in output to storage from CMSSW jobs, with breakdown on the countries (note the log scale). 87

5.28 Total data volume in output to storage from CMSSW jobs, with breakdown on the Tiers level (note the log scale). 87
5.29 Total OutputGB as a function of the Total InputGB for different computing sites. 88
5.30 Average CPU efficiency of different Tier resources over just few hours on a random day. 89
5.31 Total core hours in different Tier resources over just few hours on a random day. 90

List of Tables

1.1 Charge and mass of elementary particles. 9
1.2 Quantum flavour numbers. 9
1.3 Force-carrying bosons. 11

2.1 Main technical characteristics of the LHC accelerator. 20
2.2 Comparison of various scintillating crystals [28]. 30

5.1 Approximate time required for executing queries to obtain a few selected plots presented in this thesis using the CMS Metrics Service with Jupyter Notebook. 72

Chapter 1

The Standard Model

1.1 Particles of the Standard Model

The Standard Model is a comprehensive theory, which describes the interactions among elementary particles through the concept of quantum fields.

The particles of the SM can be split into two main classes according to their intrinsic angular momentum: fermions, that have half-integer spin, and bosons, which have integer spin. Furthermore, for every particle, its own antiparticle exists: a particle that has same spin and mass, but opposite internal quantum numbers. There are twelve fermions that can be divided in six leptons and six quarks. Moreover, quarks can have three different "colors". The color charge regulates the strong interaction among quarks. There are three colors: blue, red and green.

Leptons have either unitary or null electric charge, where the charge is measured in units of the module of the electron charge. They can be organized in three generations:

$$\begin{pmatrix} \nu_e \\ e \end{pmatrix} \begin{pmatrix} \nu_\mu \\ \mu \end{pmatrix} \begin{pmatrix} \nu_\tau \\ \tau \end{pmatrix}$$

In each generation the particles on the bottom are called, from left to right, electron, muon and tau while the particles on the top are the respective neutrinos. Electron, muon and tau interact via the electromagnetic and weak forces, whereas neutrinos only through the weak force. The strong interaction does not affect leptons. The electron and the neutrinos are stable particles, whereas the muon and the tau decay after a short period of time (mean lifetime: $\tau_\mu = 2.19 \cdot 10^{-6}$ and $\tau_\tau = 2.90 \cdot 10^{-13} s$). The name "lepton", from the Greek "small", originates from the fact that the first leptons discovered had a mass significantly smaller than the other particles. Today, the masses of the electron, the muon and the tau are known with great accuracy; however, for the masses of the neutrinos, only the upper limits are known from the decay of the tritium (see Table 1.1).

Conversely, the six quarks also interact through the strong force. They can be organized in three generations:

$$\begin{pmatrix} u \\ d \end{pmatrix} \begin{pmatrix} c \\ s \end{pmatrix} \begin{pmatrix} t \\ b \end{pmatrix}$$

Quarks are called according to their first letter: up, down, charm, strange, top and bottom. Each quark is characterized by a flavour quantum number. Tables 1.1 and 1.2 summarize the main properties of leptons and quarks [1].

	ν_e	ν_μ	ν_τ	e	μ	τ
Charge (e)	0	0	0	-1	-1	-1
Mass (MeV)	$< 2 \times 10^{-6}$	$< 0.19 \times 10^{-6}$	$< 18.2 \times 10^{-6}$	0.51	106	1770

	u	d	c	s	t	b
Charge (e)	$\frac{2}{3}$	$-\frac{1}{3}$	$\frac{2}{3}$	$-\frac{1}{3}$	$\frac{2}{3}$	$-\frac{1}{3}$
Mass (MeV)	350	350	1500	500	180000	4500

Table 1.1: Charge and mass of elementary particles.

		u	d	c	s	t	b
I, I_3	Isospin	$1, \frac{1}{2}$	0	0	0	0	0
I, I_3	Isospin	0	$1, \frac{1}{2}$	0	0	0	0
C	Charm	0	0	1	0	0	0
S	Strangeness	0	0	0	-1	0	0
T	Topness	0	0	0	0	1	0
B	Bottomness	0	0	0	0	0	-1

Table 1.2: Quantum flavour numbers.

Until now, quarks have been observed only in composite states called hadrons. Two types of hadrons are known precisely: mesons, constituted of a pair, quark and an anti-quark; and baryons, constituted of three quarks or three anti-quarks. Moreover, candidates of pentaquarks (hadrons made of five quarks) have been observed by several experiments [2].

1.2 The fundamental interactions

In nature there are four fundamental forces that have been observed: gravity, strong interaction, weak interaction, and electromagnetic interaction. The SM provides a quantitative description of only the last three through the same theory: the Gauge Theory [3]. This theory is based on the concept of symmetry. A symmetry of a physical system is a mathematical or physical feature of the system, which remains unchanged under some transformations. The mathematical description of symmetries uses group theory. Symmetries may be categorized as global or local. A global symmetry holds at all points of space-time, whereas a local symmetry is one that has different symmetry transformations at different points of spacetime. From a mathematical stand point, a local symmetry transformation is parametrised by the space-time coordinates. Local symmetries play an important role in physics as they form the basis for gauge theories.

A Gauge theory is a particular field theory in which a certain global, continuous symmetry of the theory is transformed into a local symmetry. In this way, a new field (the gauge field) is introduced that is characterized by its own dynamics and couples to the particles and fields which are associated to the symmetry. In the SM, the Gauge symmetric group is:

$$SU(3)_c \times [SU(2)_L \times U(1)_Y]$$

where $SU(3)_c$ is the symmetry group of the strong interaction and $SU(2)_L \times U(1)_Y$ is the symmetry group of the electroweak interaction, a group that unifies the weak and the electromagnetic interaction. The groups allow to introduce respectively 8, 3, and 1 spin-1 gauge fields. These gauge fields are bosons that act as mediators for the interactions. The photon (γ) is the responsible for the electromagnetic interaction, the $W^{\pm}$ and Z carry the weak interaction and 8 gluons (g) mediate the strong interaction. Some of their properties are summarized in Table 1.2.

The $SU(3)_c$ is an exact symmetry, whereas $SU(2)_L \times U(1)_Y$ is said to be spontaneously broken. This means that the Lagrangian, which describes this symmetry, is not invariant under the transformations of the group in its ground state. Indeed, the Higgs mechanism consists of the spontaneous breaking of the gauge symmetry $SU(2) \times U(1)$, which give mass to the bosons $W^{\pm}$ and Z. Lastly, the final fundamental constituent of the SM is the Higgs boson. This boson is a neutral particle with zero spin which is the base of the Higgs mechanism.

In the next sections, the electromagnetic interaction, the weak interaction and strong interaction will be briefly discussed with the aid of Feynman diagrams. These are powerful tools that help visualize the subatomic processes and calculate their probabilities.

Force	Boson	Electric charge	Spin	Mass (GeV)	Force range (fm)
Strong	$g_1, .., g_8$	0	1	0	1
Weak	$W^{\pm}, Z$	$\pm 1, 0$	1	80.4 , 91.2	10^{-3}
Electromagnetic	γ	0	1	0	∞

Table 1.3: Force-carrying bosons.

1.2.1 The electromagnetic interaction

The Quantum ElectroDynamics, also known as QED, is a relativistic quantum field theory that describes the electromagnetic interaction. Since this theory is based on the abelian group $U(1)$, it implies that the interaction, among charged particles, is carried through a massless bosons known as photons. Indeed, the QED Lagrangian density for a charged particle:

$$\mathcal{L}_{QED} = \bar{\psi}(i\gamma^{\mu} D_{\mu} - m)\psi - \frac{1}{4} F^{\mu\nu} F_{\mu\nu}$$

where $F_{\mu\nu} = \partial_{\mu} A_{\nu} - \partial_{\nu} A_{\mu}$ is the electromagnetic field tensor and A_{μ} is the gauge field of the photon; ψ and $\bar{\psi}$ are the bispinor field and the adjoint bispinor field for a charge particle of spin $\frac{1}{2}$; γ^{μ} are the Dirac matrices; $D_{\mu} = \partial_{\mu} + i\frac{e}{\hbar} A_{\mu}$ is the covariant derivative; e is the unitary electric charge; and m is the mass of the charged particle.

The fundamental Feynmann vertex in QED is:

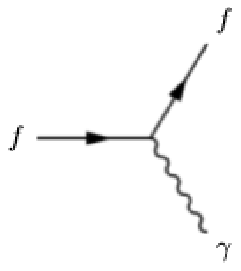


Figure 1.1: Fundamental vertex of the QED.

where the probability of the interaction is proportional to the fine-structure constant:

$$\alpha_{EM} = \frac{e^2}{4\pi\epsilon_0\hbar c}$$

The coupling in QED varies according to the energy scale of the process (running coupling). At low energies the coupling constant is $\alpha_{EM} \approx \frac{1}{137}$, whereas at the energy scale of mass of the Z boson ($E \approx 90GeV$) the coupling constant has a value of $\alpha_{EM} \approx \frac{1}{127}$. This is due to the fact that the vacuum polarizes creating virtual pairs electron−positron for a very short period of time (from the time−energy *Heisenberg's uncertainty principle* $\Delta t \approx \frac{\hbar}{\Delta E}$ with E representing the energy scale). Since opposite charges attract each others, the virtual positrons tend to be closer to a negatively charged particle (e.g electron see Figure 1.2) and screen its electric charge from a probe.

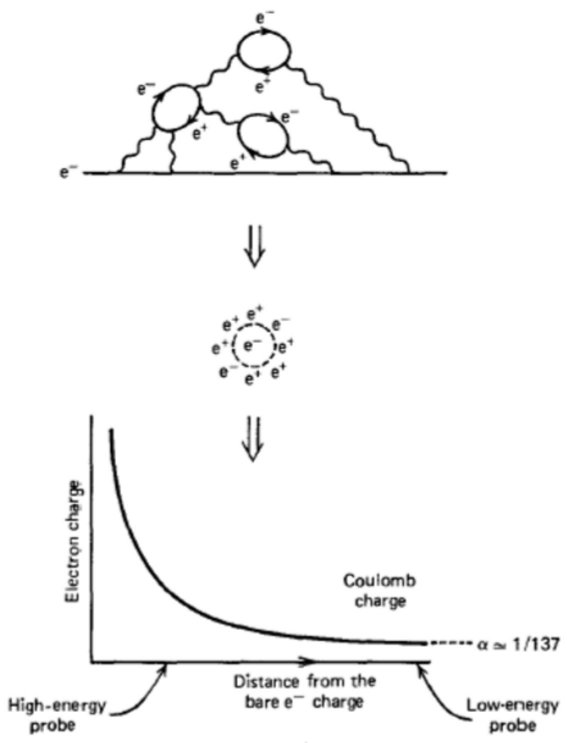


Figure 1.2: Screening mechanism in Quantum ElectroDynamics for an electron (top) and α_{EM} as a function of probe energy (bottom).

1.2.2 The strong interaction

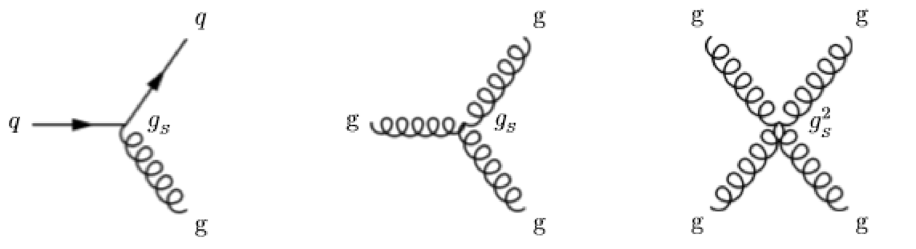
The theory that describes the strong interaction is Quantum ChromoDynamics, which is also known as QCD. QCD has been developed in analogy to the QED. More specifically, the abelian group $U(1)$ of the QED has been substituted with the non-abelian group $SU(3)_c$ that corresponds to the 3 color charges. The color states generate a basis in a 3−dimensional complex vector space; a generic color state of a quark is a vector belonging to this vector space. The color state can be rotated by 3×3 unitary matrices.

The QCD has been developed enlarging the *Yang−Mills theory* of isotropic $SU(2)$ symmetry to $SU(3)_c$ [4]. Yang and Mills extended the concept of global internal isotopic symmetry to a local isotopic gauge theory, in which the gauge scale factor depends explicitly on the space-time coordinates.

The color space forms a fundamental representation of $SU(3)_c$. From group theory it is known that the fundamental representation of $SU(3)_c$ has eight generators so a octet of gauge bosons (gluons) will emerge. These particles are the force carriers of the strong interaction and they carry color and anticolor in eight possible combinations:

$$r\bar{b}, b\bar{r}, r\bar{g}, g\bar{r}, b\bar{g}, g\bar{b}, (r\bar{r} - b\bar{b}), (r\bar{r} + b\bar{b} - 2g\bar{g})$$

The "white" color combination is not admitted. Since the QCD is not abelian, the interactions among glouns are allowed. Thus, the fundamental vertexes of the QCD are:



(a) Interaction quark-quark-gluon. (b) Interaction among three glouns. (c) Interaction among four glouns.

Figure 1.3: Fundamental vertexes of the QCD.

where the coupling constant for the strong force is g_s and α_s is defined as

$$\alpha_s(Q^2) = \frac{g_s^2}{4\pi} = \frac{1}{\frac{33-2n_f}{12\pi}\ln(Q^2/\Lambda_{QCD})}$$

where n_f is the number of different considered flavors, Q^2 is the quadratic transferred four-momentum, and $\Lambda_{QCD} \sim 200\ MeV$. It is important to notice that α_s depends on the energy (Q^2) at which the interaction takes place. Λ_{QCD} specifies the energy scale at which $\alpha_s \sim 1$. At lower energies, α_s increases (*Landau pole*), thus prohibiting perturbative calculations. *Lattice QCD* is a non$-$perturbative approach that allows to make QCD prediction of strongly interacting systems through numerical simulations.

The potential energy between two quarks can be written as:

$$U_s = -\frac{4\alpha_S \hbar c}{3r} + kr$$

This indicates that the interaction for small distances is repulsive while for big distances is attractive. When two quarks are too far apart, the bond between them breaks, and the energy stored in the bond is used to create a new couple quark-antiquark which combine with the previous quarks to form two new mesons.

The QCD Lagrangian density is

$$\mathcal{L} = \sum_{j=1}^{n_f} \bar{q}_j (iD_\mu \gamma^\mu - m_j) q_j - \frac{1}{4} \sum_{A=1}^{8} F^{A\mu\nu} F^{A}_{\mu\nu}$$

where $F^{A\mu\nu}$ are the kinetic terms associated to the gluon fields. D_μ is the covariant derivative: $D_\mu = \partial_\mu - igG_\mu$ with $G_\mu = \sum_{A=1}^{8} t^A G^A_\mu$ in which G^A_μ are the eight gluon fields and t^A are the generators of the group $SU(3)_c$ that are related to the Gell-Mann matrices λ_A: $t_A = \frac{\lambda_A}{2}$ [3].

QCD shows two different behaviors at high and low energies compared to Quantum Electrodynamics (Figure 1.4):

- *Confinement* constrains colored particles in bound states of white color at low energies.
- *Asymptotic freedom* is the property that causes bonds between particles to become asymptotically weaker as energy increases and distance decreases.

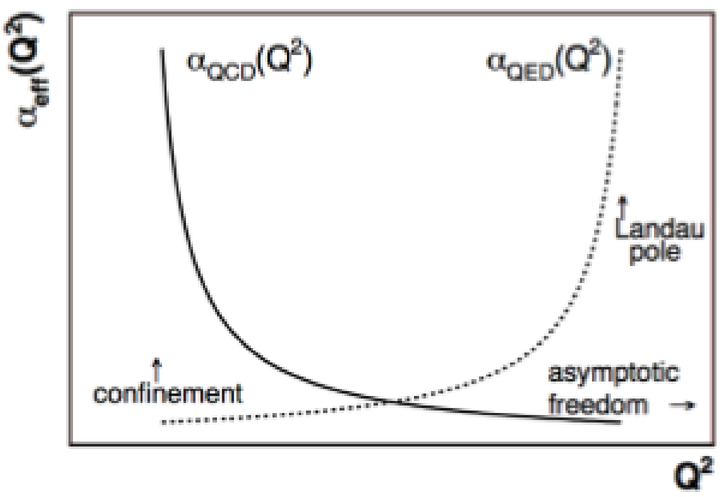
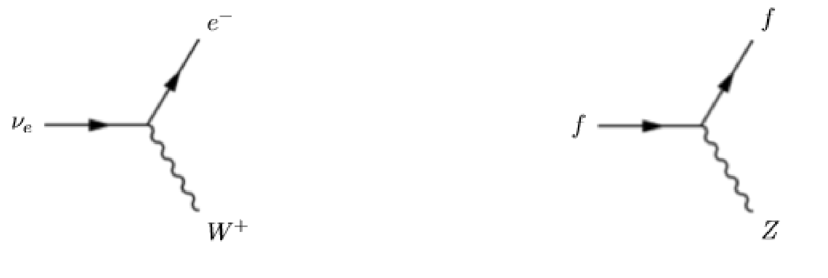


Figure 1.4: Coupling constants $\alpha_{QCD} \equiv \alpha_s$ and α_{QED} as a function of the quadratic transferred four-momentum Q^2 [5].

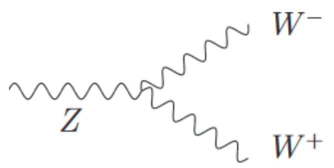
1.2.3 The electroweak interaction

The first successful attempt of describing the weak interaction, in particular the β decay, was carried on by Fermi in 1933 [6]. His theory was a phenomenological interpretation inspired by the electromagnetism. However, this theory had several limits among which the fact that the interaction had to be point-like and that it did not include the violation of parity. After few decades and several experimental evidences, this theory was overcome by Glashow−Weinberg−Salam electroweak theory which included also the electromagnetism [7]. In fact, above the unification energy ($100GeV$), the electromagnetic force and weak force merge into a unique electroweak force. Furthermore, this gauge theory also predicted the existence of the massive vector bosons $W^{\pm}$ and Z. $SU(2)_L \times U(1)_Y$ is the group of the electroweak theory, where Y is the hypercharge defined by the Gell−Mann−Nishijima relation $Q = I_3 + \frac{Y}{2}$ with Q the electric charge expressed in multiples of the electron charge, and I_3 is the third component of the weak isospin. This theory is non−abelian and allows the existences of vertexes among only the vector bosons γ, $W^{\pm}$ and Z (see Figure 1.5).



(a) Interaction among a lepton, a neutrino and a W^+.

(b) Interaction among two fermions and a Z.



(c) Auto-interaction vertex among vector bosons

Figure 1.5: Examples of fundamental vertexes of the weak interation.

The electroweak theory proposed by Glashow presented a substantial problem: it was not able to take into account the fact that the photon was massless whereas the Z and $W^{\pm}$ were not. This conundrum was solved independently by Englert, Brout [8], Higgs [9], Guralnik, Hagen and Kibble [10]. They came up with a mechanism (often called *Higgs mechanism*) that is responsible for generating the masses of the $W^{\pm}$ and Z bosons in the SM through the spontaneous symmetry breaking of the gauge symmetry $SU(2) \times U(1)$. Particles interact with the Higgs boson proportionally to their masses. Thus the processes which include the quark top, which is the heaviest of all quarks, are very valuable for investigating the nature of the Higgs boson. The Higgs boson can be produced in a hadron accelerator (see Figure 1.6).

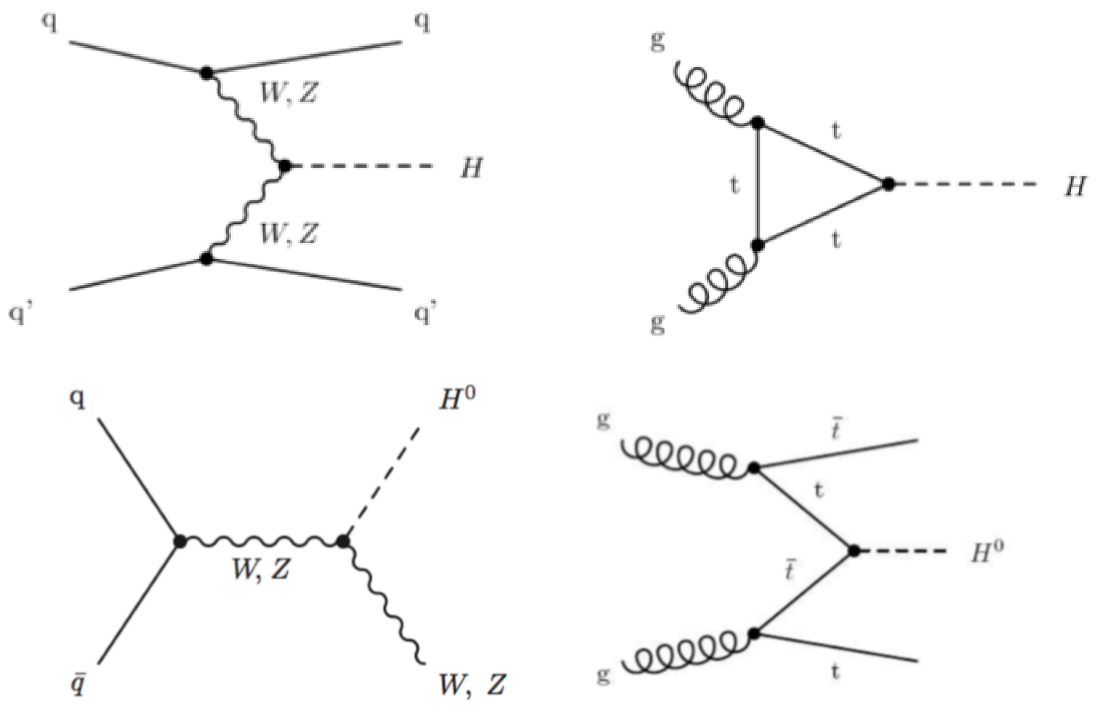


Figure 1.6: Production of the Higgs boson at a hadron accelerator.

Chapter 2

High Energy Physics at the LHC

2.1 The LHC accelerator at CERN

The Large Hadron Collider (LHC) [11, 12] is a particle collider (see Figure 2.1) built at the European Organization for Nuclear Research (CERN), located beneath the Franco-Swiss border near Geneva in Switzerland, where the Large Electron-Positron collider (LEP) previously existed [13].

The purpose of the LHC is to provide scientists an experimental apparatus that would allow them to test theories in high energy physics, such as the existence of the Higgs boson, of supersymmetric particles, and beyond. As an example of one of its first results, the discovery of a new particle was publicly announced on July 4th, 2012 compatible with the Higgs boson predicted by the Standard Model [14, 15, 16].

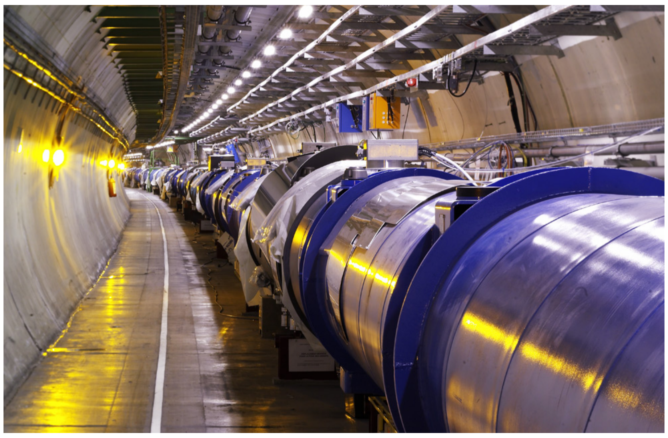


Figure 2.1: The LHC collider inside the underground tunnel (copyright CERN).

Protons and heavy ions are accelerated at the LHC. The acceleration process for protons is conducted in five stages (see Figure 2.2). At the beginning, hydrogen atoms are ionized in order to produce protons and are injected in the LINAC 2, a linear accelerator. When protons reach the end of LINAC 2, they have gained an energy of $50MeV$ and then enter the Booster, where their energy increases up to $1.4GeV$. Subsequently, they enter the Proton Synchrotron (PS) where 277 conventional electromagnets push the protons to 99.9% the speed of light; each proton has now an energy of $25GeV$. Afterwards, proton bunches are accelerated in the Super Proton Synchrotron (SPS), a circular particle accelerator with a circumference of $7km$. When protons have reached an energy of $450GeV$, they are injected into the LHC in two separate pipes in which they move in opposite directions. Here, through magnets, particles can be accelerated to their maximum designed energy of $7TeV$.

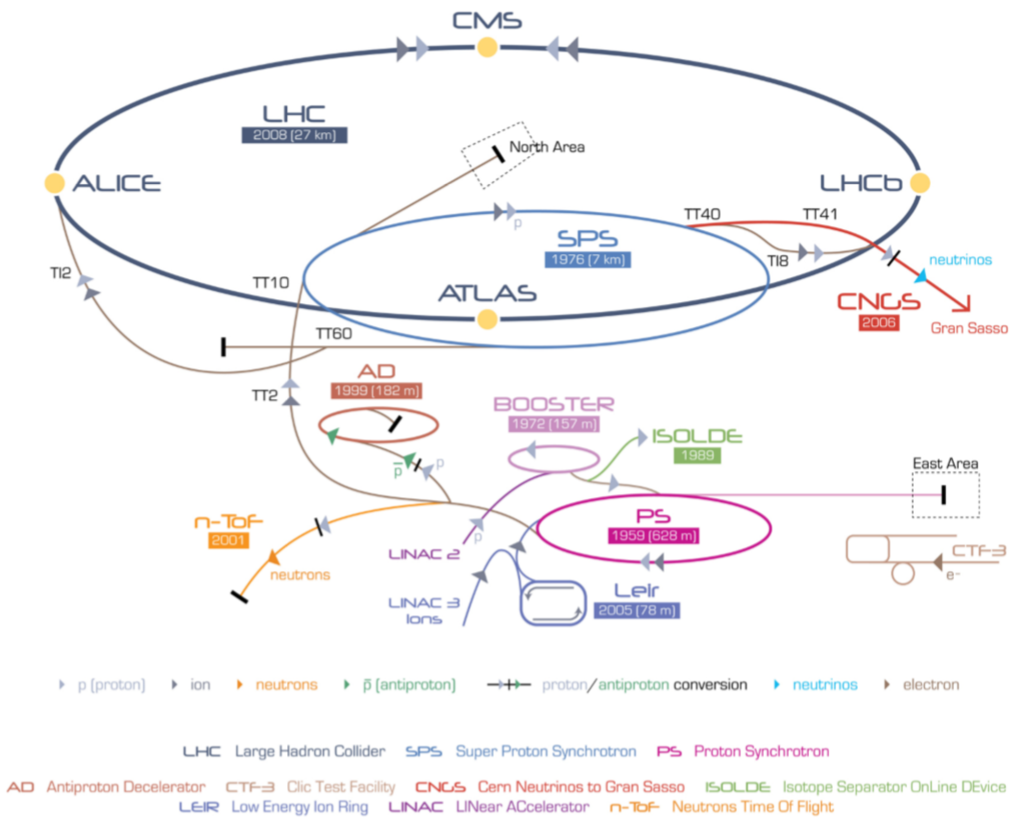


Figure 2.2: Scheme of the acceleration complex (copyright CERN).

The two pipes of the LHC intersect in four caverns, where the four detectors are placed. Here protons collide and the products of the collisions can be detected.

A vacuum system is required so the particles do not lose energy in the acceleration process due to impacts with air molecules. The LHC vacuum system is constituted of three individual vacuum systems: the insulation vacuum for cryomagnets, the insulation vacuum for helium distribution, and the beam vacuum.

To keep the path of the subatomic particles stable, the LHC exploits more than 1600 superconducting magnets constituted of an alloy based of $NbTi$. There are 1232 magnetic dipoles (see Figure 2.3) whose duty is to curve the beam along the circumference, 392 magnetic quadrupoles whose purpose is to focus the beam when it approaches the four detectors, and various smaller correcting magnets. The magnets operates at a temperature of $1.9K$: this enables the magnets to generate a magnetic field up to $8.4T$. The temperature is kept constant thanks to a powerful cryogenic system which exploits the properties of the superfluid helium.

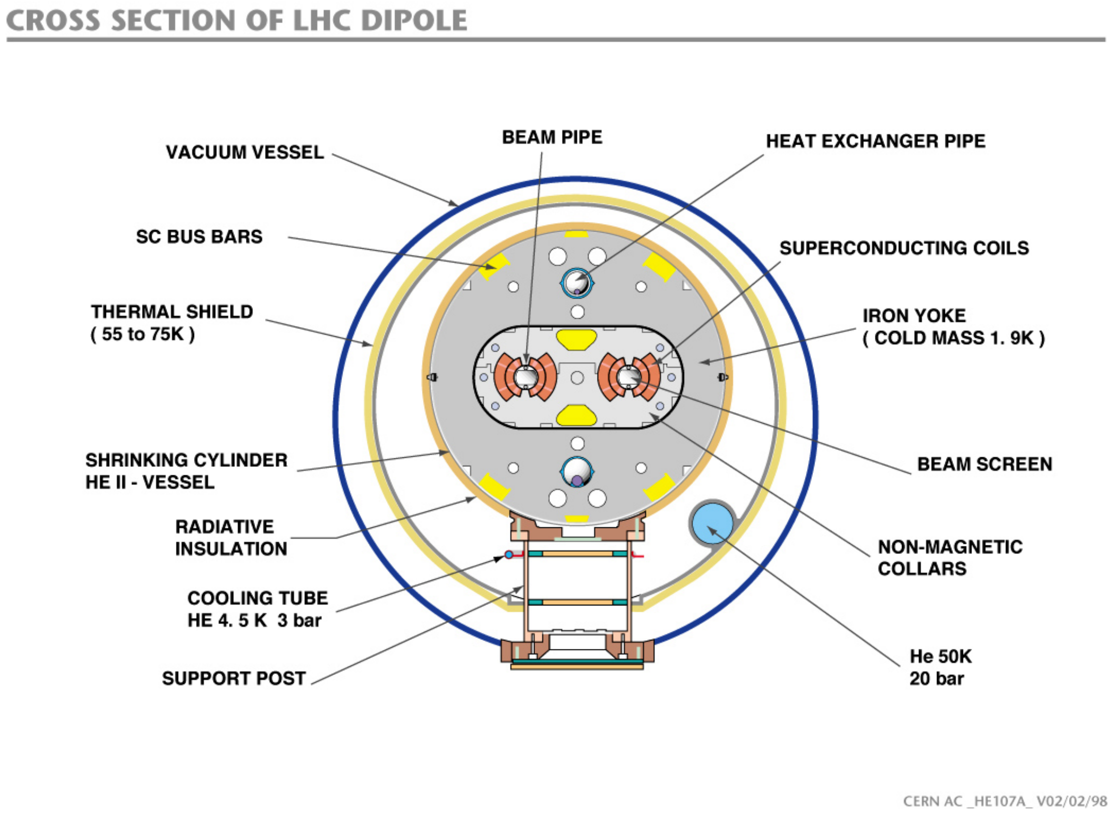

CROSS SECTION OF LHC DIPOLE
VACUUM VESSEL
BEAM PIPE
HEAT EXCHANGER PIPE
SC BUS BARS
SUPERCONDUCTING COILS
THERMAL SHIELD (55 to 75K)
IRON YOKE (COLD MASS 1. 9K)
SHRINKING CYLINDER HE II - VESSEL
BEAM SCREEN
RADIATIVE INSULATION
NON-MAGNETIC COLLARS
COOLING TUBE HE 4. 5 K 3 bar
SUPPORT POST
He 50K 20 bar
CERN AC _HE107A_ V02/02/98

Figure 2.3: Cross section of LHC dipole (copyright CERN).

Charged particles inside the LHC are accelerated through radio-frequency (RF) cavities. They are metallic chambers in which an oscillating electromagnetic field is able to to accelerate the particles to their maximum speed. The bunches of particles have to arrive in synchronism with the electromagnetic field in order to receive a push forward. When particles reach the designed velocity they will no longer feel accelerating voltage. Conversely, the RF will try to keep all the particles as close as possible to the design velocity. At these energies the main source of energy loss is synchrotron radiation. There are 16 RF cavities (8 per beam) around the LHC and they all operate in a superconducting state. They contain high-power klystrons which are specialized linear-beam vacuum tubes that work as amplifiers. Each cavity can deliver up to $2MV$, resulting in $16MV$ for each beam.

An important parameter that characterizes a particle accelerator is the machine luminosity ($\mathcal{L}$) defined as:

$$\mathcal{L} = \frac{f_{rev} n_b N_b^2 \gamma_r}{4\pi \epsilon_n \beta^*} F$$

in which f_{rev} is the revolution frequency, n_b is the number of bunches per beam, N_b is the number of particles in each colliding beam, ϵ_n is the normalized transverse beam emittance, β^* is the beta function at the collision point, γ_r is a relativistic factor and F the geometric luminosity reduction factor. Each second, the number of events that occur is:

$$N_{event} = \mathcal{L}\sigma_{event}$$

The ATLAS and CMS experiments (see Section 2.2) run at high machine luminosity $\mathcal{L} = 10^{34} cm^{-2} s^{-1}$, whereas ALICE and LHCb operates, respectively, at a lower luminosity $\mathcal{L} = 10^{27} cm^{-2} s^{-1}$ and $\mathcal{L} = 10^{32} cm^{-2} s^{-1}$.

Some of the main technical characteristics of the LHC are summarized in Table 2.1.

Particles accelerated	Protons and heavy ions (Lead 82+)
Accelerator circumference	$26659m$
Injected beam energy	$450GeV$ (protons)
Nominal beam energy for physics	$7TeV$ (protons)
Magnetic field at $7TeV$	$8.4T$
Operating temperature	$1.9K$
Number of magnets	1232
Number of quadrupoles	858
Number of correcting magnets	6208
Number of RF cavities	16
Frequency of RF cavities	$400MHz$
Maximum Voltage of a single RF	$2MV$
Maximum Luminosity	$\mathcal{L} = 10^{34} cm^{-2} s^{-1}$
Power consumption	$\sim 180MW$

Table 2.1: Main technical characteristics of the LHC accelerator.

2.2 The experiments at the LHC

At the LHC there are four main experiments, each one located in its own cavern where beams collide (see Figure 2.4).

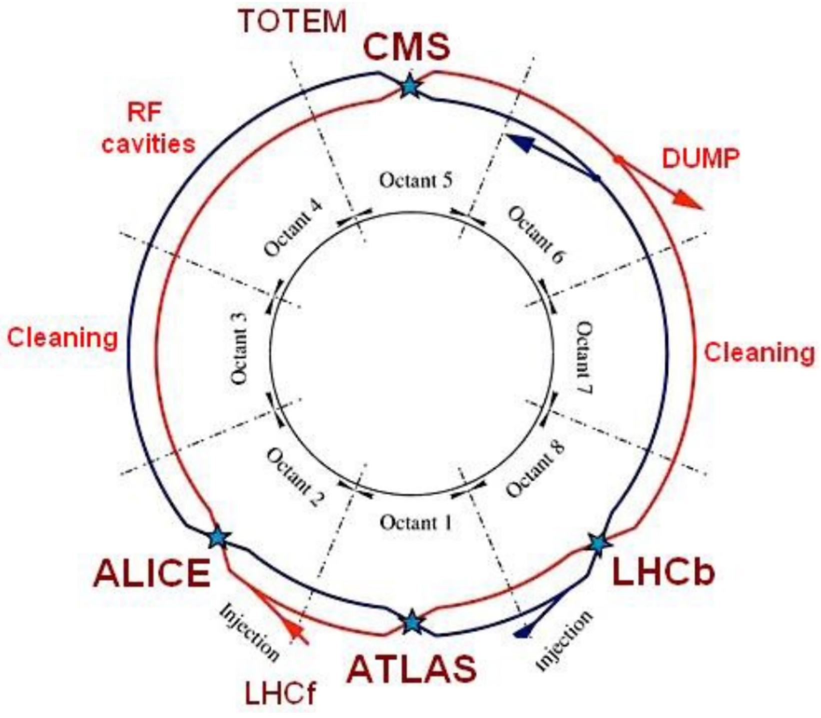


Figure 2.4: Experiments at the LHC (copyright CERN).

In the next paragraphs, few introductory details about each experiment are provided. A full description of each detector, its purpose, and its design features are out of the scope of this work; references and further readings can be found in [17, 18, 19, 20, 21].

2.2.1 ALICE

A Large Ion Collider Experiment (Figure 2.5) [17] is a general-purpose, heavy-ion detector which studies the strong interaction at extreme values of energy density and temperature during the collision of heavy nuclei (Pb). The detector has been designed to identify a great number of single events which happens during each collision of heavy nuclei. The main purpose of this experiment is to study Quark Gluon Plasma and the restoration of chiral symmetry [22, 23], in particular the whole detector has been designed to cope with the extremely high multiplicity that characterises heavy ion collisions (from three to four orders of magnitude higher than a proton-proton collision). Even though Quark Gluon

Plasma is predicted to be formed in only heavy-ion collision, proton-proton collisions and proton-ion collision provide extremely valuable informations for calibrations.

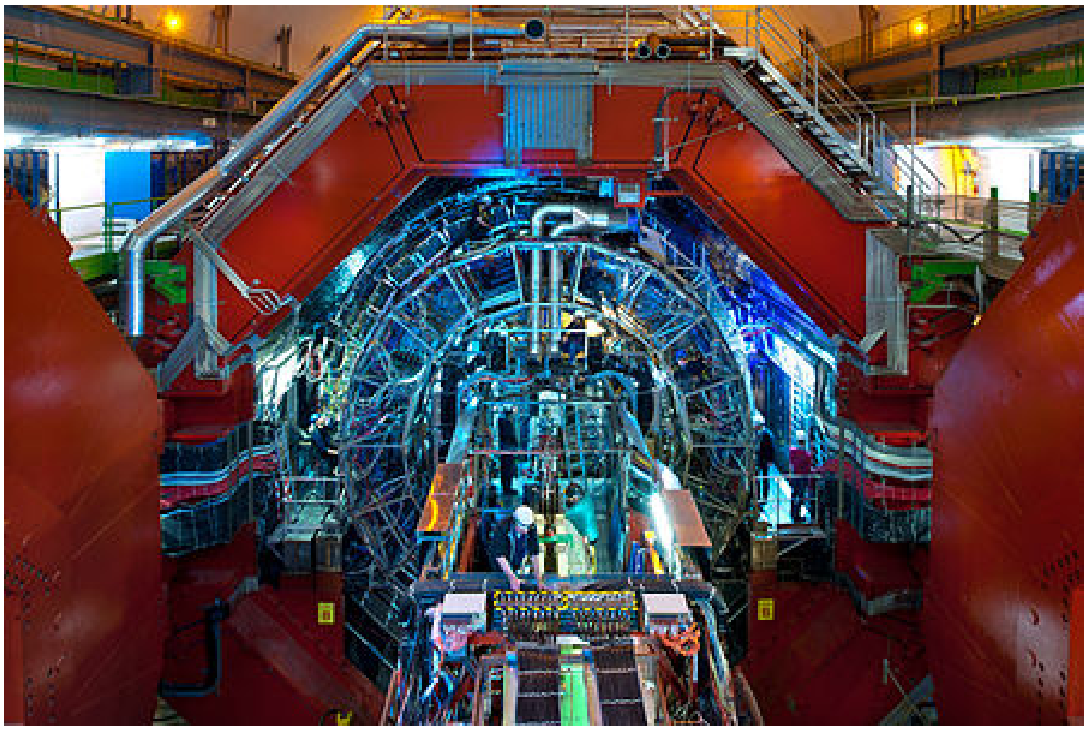


Figure 2.5: Picture of the ALICE detector under maintenance (copyright CERN).

The ALICE detector is constituted of several subdetectors. Moving from the beam pipe outwards, there are the following subdetectors: the Inner Tracking System, the Time Projection Chamber, the Time Of Flight, the Ring Imaging Cherenkov HMPID, the Transition Radiation Detector, the Electromagnetic Calorimeters and the Muon Spectrometer. The process of Particle Identification plays a crucial role in understanding Quark Gluon Plasma. For this reason, ALICE exploits several different subdetectors to perform Particle Identification: the Time Of Flight, which is able to separate π/K and K/p for $p_t < Gev/c$ with a 3σ precision; the High Momentum Particle Identification, which is exclusively dedicated to particle identification of hadrons with $p_t > 1GeV/c$; and the Transition Radiation Detctor, whose major purpose is to identify electrons of $p_t > 1GeV/c$. Furthermore, close to the beam pipe, the ALICE detector has Inner Tracking System based on silicon pixels and microstrips. This subdetector is very useful to reconstruct heavy flavours decays. The most complex subdetector in ALICE is the Time Projection Chamber that provides informations both on the particles trajectories and the energy deposited. An additional information of the energy deposited is provided by two electromagnetic calorimeters, one is made of the scintillating crystal $PbWO_4$ and the other is Pb scintillator sampling calorimeter; the former provides a better energy resolution and granularity than the latter, however the latter cover a volume significantly bigger than the former. The detector weights $10'000$ tonnes and consists of a barrel part, which measures hadrons, electrons, and photons, and a muon spectrometer in the forward region.

2.2.2 ATLAS

A Toroidal LHC ApparatuS (Figure 2.6) [18] is an experiment whose main purpose is to investigate new physics beyond the Standard Model exploiting the extremely high energy at the LHC. It also searches the existence of dark matter and extra dimensions.

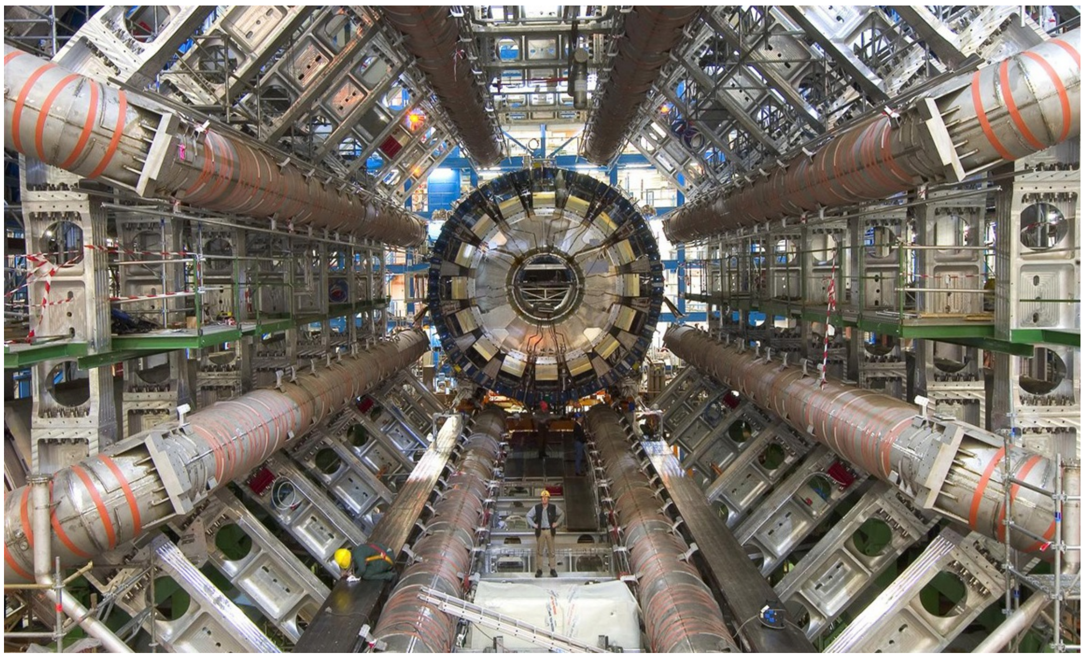


Figure 2.6: Picture of the ATLAS detector under construction (copyright CERN).

The detector is made of four main subdetectors (Inner Tracker, Electromagnetic Calorimeter, Hadron Calorimeter, and the Muon Spectrometer) and a complex system of solenoidal and toroidal magnets. The Inner Detector is responsible for tracking the particles, its main components are the pixels and silicon microstrip trackers, which offer high granularity, and the Transition Radiation Tracker, which also helps with particle identification. The electromagnetic calorimeter is based on Liquid Argon and is placed inside the solenoid of the detector which provides a $2T$ axial field. Outside the solenoid, there is the hadronic calorimeter that is a sampling calorimeter, in which steel is used as absorber and the active material is scintillating tiles. Moving outwards, the muon spectrometer constituted of Resistive Plate Chamber and Drift Tubes allows to track the muons escaping the detector. Finally, air-core toroids generate a magnetic field for the muon chamber to reconstruct the muon momentum. The apparatus is $46m$ long with a diameter of roughly $25m$ and weights approximately $7'000$ tonnes.

2.2.3 CMS

The Compact Muon Solenoid (Figure 2.7) [19, 20] is a multi-purpose detector designed to observe a wide variety of phenomena in proton-proton and heavy ion collisions. Its main goal is to investigate the nature of electroweak symmetry breaking which is explained in the Standard Model through to the Higgs mechanism. This experiment will be covered in greater detail in section 2.3.

Figure 2.7: Picture of the CMS detector while open (copyright CERN).

2.2.4 LHCb

The Large Hadron Collider beauty [21] is a detector specialized in the study of the certain rare decays of the B meson. In particular, the values of the Cabibbo Kobayashi Maskawa

Matrix are measured in order to better understand the CP violation, which is strictly linked with the concentrations of matter and antimatter in the universe. At $E = 14TeV$ the cross section for the production of a pair $b\bar{b}$ is $\sigma_{b\bar{b}} \approx 500\mu b$. The LHCb operate at a reduced luminosity of $\mathcal{L} = 10^{32}cm^{-2}s^{-1}$ by tuning the focusing of the beams before the interaction point. This choice is due to limit the pile-up (two or more particle-particle interaction for a single bunch crossing) in order to simplify the later analysis; incidentally, this choice also reduce the radiation damages.

LHCb is a single-arm spectrometer with a forward angular covarage (Figure 2.8); this design is due to the fact that b and $\bar{b}$ hadrons are produced in the same forward or backward cone. The main requirements for the detector are: optimum vertex and momentum resolution to discriminate among the various final states.

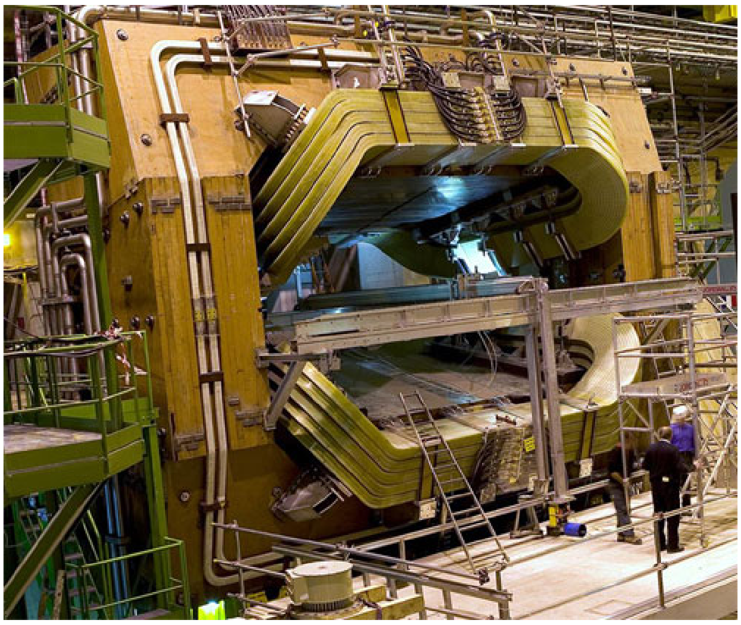


Figure 2.8: Picture of the CMS detector while open (copyright CERN).

Moving from the interaction point outwards, the LHCb detector is made of the following subdetectors: the vertex locator VELO, a first aerogel Ring Imaging Cherenkov, a silicon Trigger Tracker, the magnet followed by three tracking stations, a second aerogel Ring Imaging Cherenkov, a Scintillator Pad Detector and Preshower, an Electromagnetic Calorimeter, a Hadronic Calorimeter and lastly a series of muon chambers.

2.2.5 Other experiments at LHC

There are three other smaller experiments along the LHC tunnel:

- **TOTEM**: the goal of the TOTEM experiment is to measure total cross section, elastic scattering, and diffractive processes in the very forward region [24]. TOTEM detectors are placed on both sides of the interaction point at CMS over a distance of hundred of meters.
- **LHCf**: the purpose of the LHCf experiment is to study the cross section of neutral particles produced in the very forward region [25]. This studies are crucial for understanding how the cosmic rays cascade develops through the atmosphere. The LHCf consists of two detectors which are placed along the beamline, at $140m$ either side from collision point of the ATLAS experiment.
- **MOEDAL**: the Monopole and Exotics Detector at the LHC is designed to search for new highly ionizing particles, such as the predicted magnetic monopoles [26]. The MoEDAL detector is placed in the same cavern as the LHCb experiment.

2.3 The CMS detector

The CMS experiment [19, 20] is a multi−purpose detector designed to explore the Standard Model and investigate new physics beyond the SM. The discovery, together with ATLAS, of a particle that fits the signature of the Higgs boson represents one of the greatest successes of the CMS collaboration. When protons collide at their maximum designed energy ($\sqrt{s} = 14TeV$), 10^9 events/s will occur. The on−line selection process has to trigger only 100 events/s to be saved. This high flux of particles needs custom electronics capable of enduring a high flux of radiation while simultaneously being able to make extremely challenging selections.

When the detector was designed, it had to meet the following general requirements:

- identify and track muons with high precision;
- have a high precision electromagnetic calorimeter for the measurements of electrons, positrons and photons;
- have an effective tracker for the measurement of particles' momenta;
- cover almost the whole solid angle, in order to be able to detect all the particles produced during each collision.

To fulfill these criteria, the detector uses a powerful solenoid that bends the trajectory of charged particles. Different particles release energy differently in each part of the detector. Combining the knowledge of the particles' path and momenta, it is possible to trace back the particle involved in a certain event and determine other information such as their masses. The detector (see Figure 2.9) is constituted of five concentric layers: the tracker, the electromagnetic calorimeter (ECAL), the magnet, the hadronic calorimeter (HCAL), and the muon system. Moreover, a Trigger and a Data Acquisition System is required to collect and process the data exiting the detector. A few details about each system is provided in the following paragraphs (Sections 2.3.1-2.3.6).

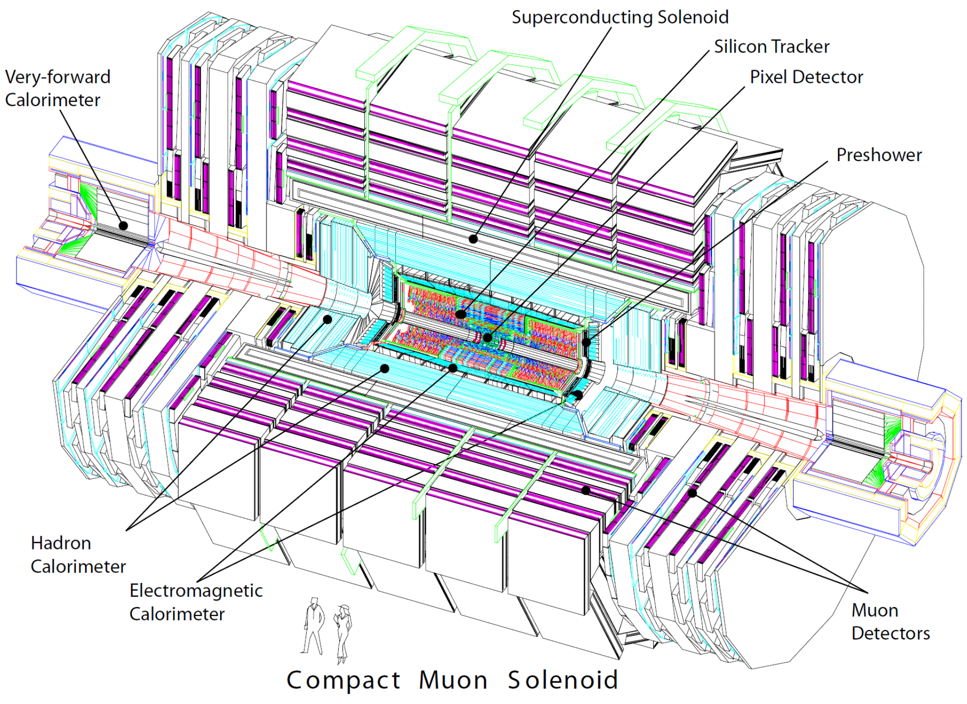


Figure 2.9: Section of the CMS detector (copyright CERN).

The detector can be divided in three main regions: the barrel, which consists of the central region, made of five wheels and orthogonal to the beam axis; two endcaps that seal the barrel, composed of three disks each; and the very forward regions, which consists of a series of subdetectors placed near the beam axis.

To proper describe the detector itself and the physics quantities, a right-handed cartesian reference frame must be defined as followed:

x axis is defined as the horizontal axis, which connects the interaction point inside CMS to the center of the LHC ring.

y axis is defined vertical exiting from the interaction point.

z axis is defined tangent to the beam line.

Furthermore, since CMS is characterized by cylindrical symmetry, from the Cartesian coordinates it is possible to define polar coordinates (r, ϕ, θ). The previous definitions allow to introduce a series of quantities:

- Particle transverse momentum: $p_t = \sqrt{p_x^2 + p_y^2}$
- Transverse energy: $E_t = E\sin(\theta)$
- Transverse mass: $m_t = \sqrt{m^2 + p_t^2}$
- Missing Transverse Energy (MET): $E_t^{missing} = -\sum_i \vec{p_t^i}$
- Rapidity (Lorenz invariant): $y = \frac{1}{2}\ln\frac{E+p_z}{E-p_z}$
- Pseudo-rapidity: $\eta = -\ln(\frac{\theta}{2})$

2.3.1 Tracker

The tracker is able to detect muons, electrons, hadrons and tracks coming from the decay of short-lived particles (e.g. b quarks). These events are of primary importance for the physics at the LHC. Since it is the innermost element of the detector it has to interfere the least with the particles produced. Most measurements are accurate to the $10\mu m$ level. The tracker has been built exclusively using silicon-based technologies; this choice allowed it to meet requirements such as radiation hardness and speed of acquisition. The tracker covers a cylindrical region of $115cm$ of radius, and $540cm$ of length.

Isolated tracks of high p_t particles are reconstructed with a resolution of $\frac{\delta p_t}{p_t} \approx (15 \cdot p_t \oplus 0.5)\%$ with p_t measured in TeV for $|\eta| < 1.6$ [27]. The resolution progressively degrades as η increases (see Figure 2.10 Left). Moreover, the outer muon chamber system, provides a second measurement for the transverse momentum of the muon which can be combined with the measurement obtained by the tracker in order to improve the overall resolution (see Figure 2.10 Right). This feature allows to reconstruct muons with an efficiency above 98%. Charged hadrons with a $p_t > 10GeV$ are reconstructed with an efficiency close to 95% whereas hadrons with $1GeV < p_t < 10GeV$ are reconstructed with an efficency higher than 85%. The efficiency for high energy electrons is better than 90%.

The tracker relies on two type of technologies for the detection of particles: pixels and microstrips. The signals are then transmitted through optic fibers cables outside the detectors.

Pixels are placed both on the barrel and the endcaps. On the barrel there are three layers of pixels 4, 7, and $11cm$ away from the beam, whereas on the endcaps there are two layers divided in 24 sections which are tilted to take into account the Lorenz drift angle (Lorentz drift angle = $32°$ for electron in silicon with a magnetic field of $4T$). Each layer is constituted of modular detector units. The modules (around 66 millions) contain

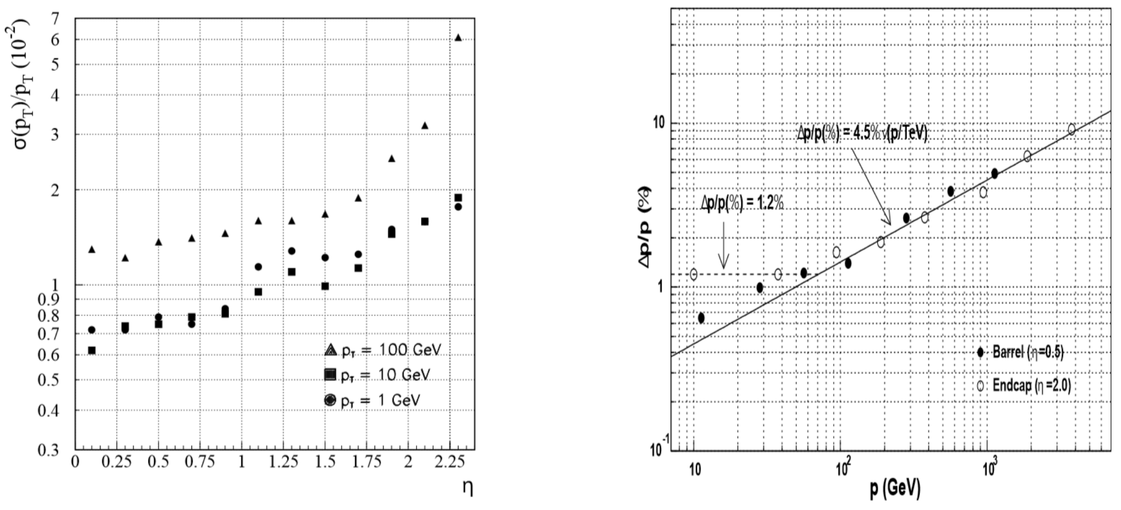


Figure 2.10: Left: CMS Tracker stand−alone p_t resolution for muons, without beam constraint. Right: Combined CMS Tracker and Muon Chamber p resolution, in the central and forward rapidity regions of the detector [27].

a sensor plate attached to highly integrated readout chips. The active area of each pixels is $150 \times 100 \mu m^2$. Each pixels generates around $50 \mu W$, thus an integrated cooling system is needed.

The microstrip system contains 15200 sensitive modules clustered in 10 million detector strips. The modules are attached to 80,000 red-out microelectronic chips. The silicon microstrips, compared to the pixels, necessitate of a much lower quantity of read-out electronics per unit area.

2.3.2 Electromagnetic Calorimeter

The electromagnetic calorimeter is designed to detect particles that interact according to the QED such as photons and electrons [28]. The main components of the ECAL are lead tungstate ($PbWO_4$) crystals that cover the entire solid angle: when the crystals are hit by a particle, scintillation occurs. The lead tungstate has several desirable properties compared to other crystal scintillators used in other experiments: short radiation length, relatively high critical energy, fast light emission (see Table 2.2). Furthermore, the production techniques for this material have been mastered in the last decades, especially in Russia and China. The main drawback of this material is the low light output emission compared to other inorganic scintillators; for this reason, the electromagnetic calorimeter necessitates very large photodiodes.

With the purpose of recording the light emitted by the crystals, Avalanche Photo-Diodes (APDs) are placed around the calorimeter and Vacuum PhotoTriodes (VPTs)

	NaI	BGO	CSI	BaF_2	CeF_3	$PbWO_4$
Density $[g/cm^3]$	3.67	7.13	4.51	4.88	6.16	8.28
Radiation length $[cm]$	2.59	1.12	1.85	2.06	1.68	0.89
Interaction length $[cm]$	41.4	21.8	37.0	29.9	26.2	22.4
Molire radius $[cm]$	4.80	2.33	3.50	3.39	2.63	2.19
Light decay time $[ns]$	230	60 300	16	0.9 630	8 25	5 (39%) 15 (60%) 100 (1%)
Refractive index	1.85	2.15	1.80	1.49	1.62	2.30
Maximum of emission $[nm]$	410	480	315	210 310	300 340	440
Relative light output	110	18	20	20/4	8	1.3

Table 2.2: Comparison of various scintillating crystals [28].

are placed in the endcaps. A preshower detector is placed at the end of the ECAL to distinguish single high-energy photons from pairs of low-energy photons. The preshower detector consists of two planes of lead and several silicon sensors. A photon that hits the lead generates an electromagnetic shower, made of electron-positron pairs, that is detected accurately by the sensors. It is then possible to trace back the initial energy of the photon.

The energy resolution of the electromagnetic calorimeter is described by the following formula:

$$\left(\frac{\sigma}{E}\right)^2 = \left(\frac{a}{\sqrt{E}}\right)^2 \oplus \left(\frac{\sigma_n}{E}\right)^2 \oplus c^2$$

where a accounts for the stochastic contribution, σ_n the noise and c^2 is a constant. The stochastic term describes the fluctuations in the shower development. The noise term includes contributions from electronic noise (dominant at low energy) and energy pile-up (dominant at high luminosity). For E expressed in GeV, the constants for the energy resolution for electrons in beam tests are: $a = 2.8\%$, $\sigma_n = 12\%$, $c^2 = 0.3\%$ [29].

2.3.3 Hadron Calorimeter

The hadron calorimeter measures mainly hadron jets, neutrinos (in the form of Missing Energy Transverse) and exotic particles. It is able to detect particles up to $|\eta| = 5$ [30].

The HCAL is a sampling calorimeter which uses "absorbers" to allow the development of the hadronic particle shower, and fluorescent scintillator materials to collect the light when a particle passes through them. All the light measured by the sensors is then added up to estimate the energy of the particles. HCAL consists of 20° wedges inside the magnet (see Figure 2.11). The choice for the materials had to be quite unconventional since the hadronic calorimeter is completely embedded in a $4T$ magnetic field generated by the surrounding solenoid. Previous experiments showed that the light production from scintillator increases when inserted in a magnetic field [31]. However this effect saturates when above $2T$ for Kuraray SCSN81, which is the plastic scintillator used in the barrel. This material can also sustain the high flux of radiation required and it characterized of its long term stability. The scintillator are then connected to photodiodes through wavelength shifting−fibers.

The absorbers are made of plates of brass and steal from 5 to $8cm$ thick and the interaction length associated to them varies as a function of η with values ranging from $5.25\lambda_0$ to $10.5\lambda_0$. Also the energy resolutions vary in the different regions of the hadronic calorimeter. Nonetheless, they are described by the same formula:

$$\left(\frac{\sigma}{E}\right) = \frac{a}{\sqrt{E}} + c$$

where E is measured in GeV; c is equal to 5%; and a to 65% in the barrel, 85% in the endcap, and 100% in the forward hadronic calorimeter. The calibrations parameters and the energy response of the hadronic calorimeter have been determined with cosmic rays and beam tests [32].

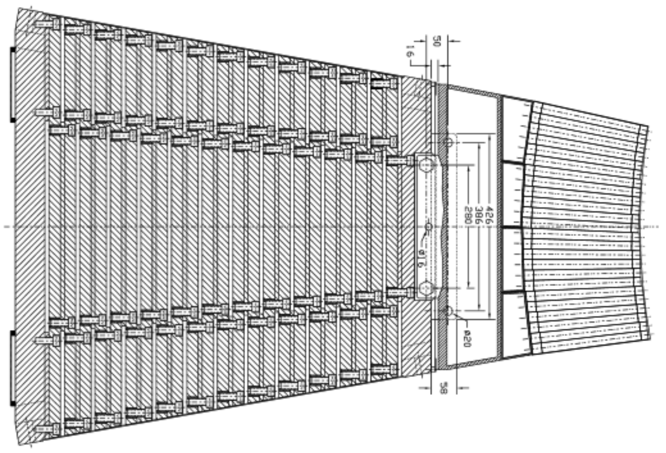


Figure 2.11: Artistic representation of a 20° wedge. The ECAL is on the right and HCAL on the left (copyright CERN).

2.3.4 Magnet

The magnet, which contains the tracker and the calorimeters (ECAL and HCAL), is a superconducting solenoid in which a current of $19500A$ flows that generates a uniform magnetic field up to $4T$. In order to maintain this magnetic field, the solenoid is constantly kept at a temperature of $4K$.

The solenoid cables are made of an alloy of Niobium and Titanium, which wind around the hadronic calorimeter in four layers. The whole magnet is $12.5m$ long with an inner diameter of $6m$. The magnets also provides mechanical stability to the detector.

At more than $5m$ from the beam pipe there is an external iron yoke. This component has two purposes: stopping all the particles except muons and neutrinos from reaching the muon detector, and providing the return of the magnetic field.

2.3.5 Muon detector

The muon detector is placed on the outer layer of the detector, as muons are relatively non-interacting particles; in fact, they are able to pass through several meters of iron without loosing much energy. Needless to say, since they give their name to the detector, they are extremely important in several process, such as the decay of the Higgs boson in four muons. The paths of the muons are obtained by interpolating a curve through the points of the detector hit by the particles (see Figure 2.12).

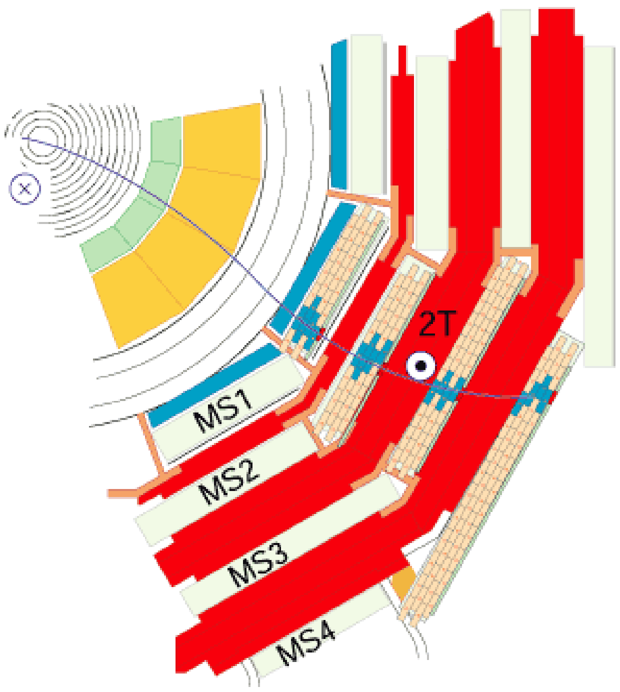


Figure 2.12: A schematic view of a muon trajectory inside the detector (copyright CERN).

There are three types of subdetectors for muons' identification and all of them measure the ionization of produced by the muon in a gas. There are 1400 muon chambers;

250 drift tubes (DT) and 540 cathode strip chambers (CSC) to identify particles' position and give a trigger. 610 resistive plate chambers (RPC) form a second trigger. DTs are organized in cells of area $6.24cm^2$ and are filled with Ar (85%) and CO_2 (15%). Consecutive DT layers are misaligned by half cell in order to improve coverage and reduce blind spots. CSCs are multi−wire proportional chambers filled with Ar (30%), CO_2 (50%) and CF_4 (20%). They provide a two−dimensional information of the position of the muon. Lastly, RPCs are filled with $C_2H_2F_4$ (96.5%) and C_4H_{10} (3.5%). Since they are very fast (response time $3ns$), they are used for triggering. DTs and RPCs are displaced around the beam line whereas CSCs and RPCs complete the endcaps disks at both ends of the barrel.

2.3.6 Trigger

The LHC is designed to operate at $\mathcal{L} = 10^{34}cm^{-2}s^{-1}$ and there are on average over twenty inelastic pp collisions per bunch crossing. Since the bunches are only $25ns$ apart, the total data output is too big to be stored. However, the events of physical interest are only a fraction of the total. The Trigger and Data Acquisition System operates a first selection so that data can be stored at the archival storage capability of $O(10^2)Hz$ at data rates of $O(10^2)MB/s$. In order to be able to reject roughly 10^5 events for every accepted event, the event selection in CMS is divided in two stages (see Figure 2.13): firstly, the Level-1 Trigger (L1 Trigger) reduces the rates of accepted events to $\sim 100kHz$ [33]; secondly, the High−Level Trigger (HLT) reduces the rates of accepted events to $\sim 100Hz$ [34].

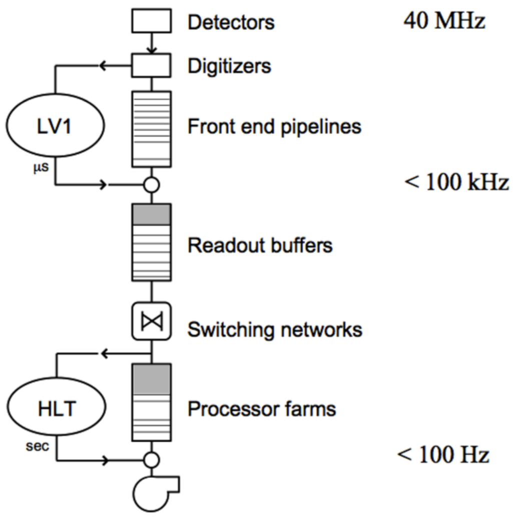


Figure 2.13: A schematic representation of the dataflow in the CMS Trigger. The two consecutive steps are highlighted: the L1 Trigger and the HLT (copyright CERN).

- The L1 Trigger is made of custom electronics (ASICs and FPGAs) and performs a first online rough particle identification based mainly on the information obtained by the calorimeters and the muon system. It is constituted of three primary components: the L1 calorimeter trigger, the L1 muon trigger, and the L1 global trigger. They all have to decide whether to save the event or not within $3\mu s$ after each collision. In fact, after this period of time, data temporally saved in the buffers are overwritten.

- The HLT is implemented via software and takes a step farther the selection of the L1 trigger. The hardware of the HLT consists of a farm of commercially available processors and each event is assigned to only one of them. When the HLT evaluates a L1 candidate, firstly it continues the L1 reconstruction, then, if the candidate is not discarded, it reconstructs its tracks with also the information from the tracker. In fact, the latter process is one of the most CPU−expensive process among all [35]. In order to minimize the time used by the CPUs for computations, the software that runs on the HLT satisfies the following guidelines: it reconstructs certain parts of the events only if it is strictly required and the reconstruction is interrupted if more calculation would not modify the final outcome.

Chapter 3

Computing in High Energy Physics

3.1 Introduction

The LHC produces a huge amount of data that has to be stored and later analyzed by scientists. During each collision, swarms of particles are produced and signals leaving the detector are recorded. About 30 Petabytes (PB) of data are produced at the LHC every year. Thus, efficient storage and data management systems are required. In order to cope with all this information, a complex computing infrastructure has been designed and deployed, characterized by computing centers distributed worldwide. This infrastructure is known as the Grid.

3.2 Grid technologies and WLCG

The Worldwide LHC Computing Grid project (WLCG) [36, 37] is a global collaboration responsible for building and maintaining a data-oriented infrastructure required by the experiments at the LHC. The purpose of this infrastructure is to provide computing resources to store, distribute and analyze the data produced by the LHC to the users of the collaborations regardless of where they might be located. The idea of a shared computing infrastructure constitutes the basis of the concept of the Grid. The WLCG cooperates with several Grid projects such as the European Grid Infrastructure (EGI) [38] and Open Science Grid (OSG) [39]. In the HEP context, the Grid has been created with the intent to give a common middleware to all the experiments; then each experiment can then run its specific applications on top of the middleware. This choice has allowed to keep the costs of maintaining and upgrading the necessary computing resources under control. The middleware projects provide the software layers on top of which the experiments add their own different application layers. At the middleware level, the logical elements, which constitute every Grid site, are:

- **Computing Element** (CE) manages the user's requests for computational power at a Grid site. This power is provided by using clusters of computers organized in farms and managed by software tools. The CE manages the jobs submitted by the user and the interactions with the services of the Grid.
- **Worker Node** (WN) is where the computation actually happens on a site farm. Here, scripts can be used to configure the environment.
- **Storage Element** (SE) gives access to storage and data at a site. Data are stored on tapes and disks. Tapes are used as long-term secure storage media whereas disks are used for quick data access for analysis. The protocol SRM (Storage Resource Manager) offers a common interface to access data remotely. The main types of data stored are raw date generated by the detector, data produced by the analysis of the users and simulation through Monte Carlo techniques.
- **User Interface** (UI) is the machine on which a user interacts with the Grid; through the UI, a user can access remote computing resources.
- **Central services** to help users access computing resources. Some examples are data catalogues, information systems, workload management systems, and data transfer solutions.

The Storage Federation, which is a fairly new infrastructure developed in parallel to the site SEs, provides read access to the same data, but it does not rely on a catalogue to locate the files. Instead, it uses a set of "redirectors": when the user searches a file, the redirector first looks in the local storage. Then, if the redirector does not find the data locally, it asks the SE in its federation whether it has the file. In case the file is not found, the redirector can further ask a higher level redirector if it can find the file. This process is iterated until either the file is found or the highest redirector does not find anything.

The security on the Grid is based on X.509 certificates which provide authentication for both the user and the services. The user is endowed with a Grid certificate which is used to access the services he needs. A private key is assigned to each holder of a certificate and a public key is used to make requests to a certain service. The authorization is based on the Virtual Organization Management System (VOMS) [40]. VOMS contains all the users of the Grid and the tasks that they can execute on the Grid itself.

The computing centres around the world are categorized into four types of "Tiers" [41] according on the kind of services they provide (see Figure 3.1):

Tier-0 There are two physical locations for a logical unique Tier-0 function: one is the CERN Data Centre in Geneva (Switzerland) and the other is located at the Wigner

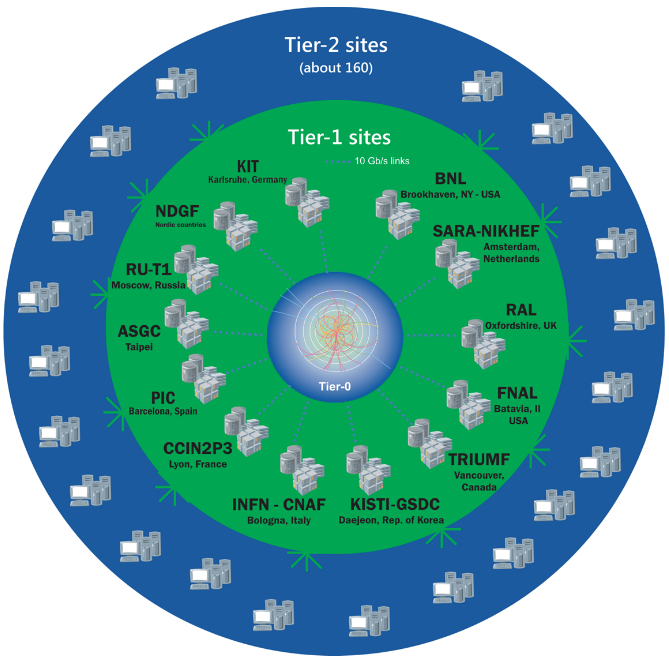


Figure 3.1: The Tier structure of the WLCG sites [36]

Research Centre for Physics in Budapest (Hungary). The Tier-0 is responsible for keeping RAW data, for a first data reconstruction, for the distribution of the RAW data and the reconstruction output to the Tier-1s, and lastly for data reprocessing when the LHC is not acquiring data.

Tier-1 There are 13 LHC Tier-1 sites, of which 7 were available to CMS during Run-1. They are exploited for large-scale, centrally-organized activities and can exchange data among them and to/from all Tier-2 sites. They are responsible for storing RAW and RECO data, for large-scale data reprocessing, and for safe-keeping of the corresponding output. Moreover, a share of the simulated data are produced at the Tier-2s. Lately, they have been commissioned as sites where users can perform their analysis.

Tier-2 There are about 160 Tier-2s in LHC, displaced around the world (about 50 were available to CMS during Run-1). They are usually universities or scientific institutes, and they often provide significant CPU resources for user analysis. Tier-2s do not have tape archiving and do not provide long-term custodial storage. However, they take care of data generation and simulation.

Tier-3 A Tier-3 can be, for example, a computer cluster of relatively small size, which is connected to the Grid. There is no formal agreement between WLCG and Tier-3s on their respective roles. This makes Tier-3s the most flexible Tier level.

3.3 The CMS Computing model

To carry out CMS physics analysis, scientists have to be able to access the huge amount of data collected by the detector and stored in the Grid sites. Moreover, they need to be granted a lot of computational power in order to run their analysis and generate Monte Carlo simulations. To these requests, one has to add the difficulties originating from the fact that CMS is an experiment with collaborators from several nations. To handle these challenges CMS exploits the Grid. More specifically, the Tiers in the CMS Computing model [42, 43] have the roles described in the following paragraphs.

Tier-0 The tasks of a CMS Tier-0 are:

1. accepting data from the Data Acquisition System [44];
2. storing RAW data on tapes;
3. performing a first reconstruction;
4. distributing the RAW data to the Tier-1s so there are two copies of all RAW data.

Tier-1 The main functions of a Tier-1 for CMS are:

1. providing a great amount of CPU for data reprocessing and data analysis;
2. data storage of RAW and RECO data;
3. data distributions to/from Tier-2s;

Tier-2 The Tiers-2s of CMS provide:

1. data transfer from/to Tier-1s;
2. services for data analysis;
3. productions of Monte Carlo events;

Tier-3 CMS Tier-3s are not formally bound to WLCG even though they can offer flexible computing capabilities.

Data in CMS are catalogued according to various characteristics. The *Event* is the fundamental unit of the data, which corresponds to a single bunch crossing. *LumiSections* are a collection of subsequent events in which the instantaneous luminosity is assumed to be constant. The integrated luminosity of a data sample can be calculated from the single LumiSections that constitutes it. Moreover, several LumiSections form the *Run* which lasts approximately the time that the beam is stable in the accelerator. Furthermore, Events originating from one or more Runs are grouped in *Datasets*. These Events share

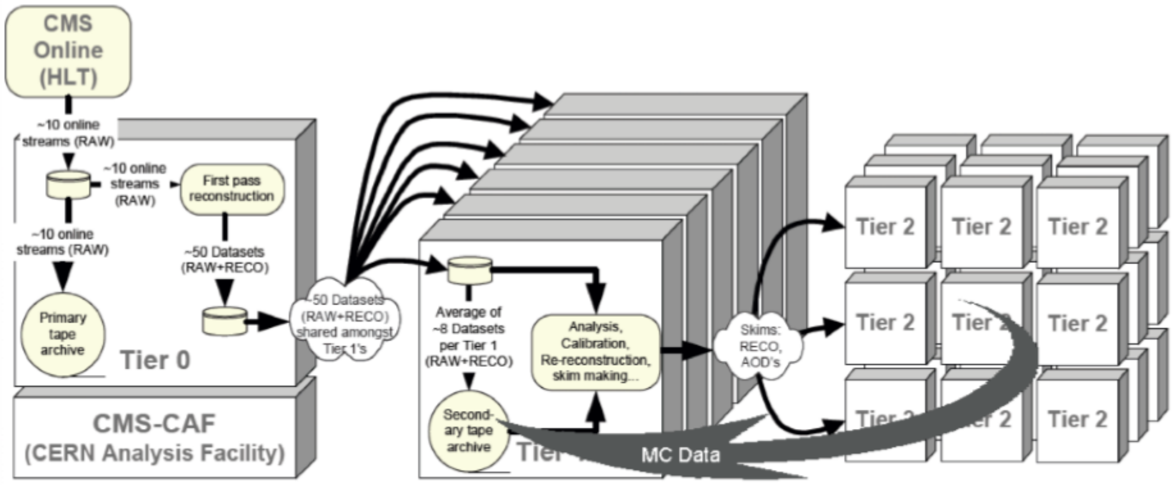

CMS Online (HLT)
~10 online streams (RAW)
~10 online streams (RAW)
First pass reconstruction
~10 online streams (RAW)
~50 Datasets (RAW+RECO)
Primary tape archive
Tier 0
CMS-CAF (CERN Analysis Facility)
~50 Datasets (RAW+RECO) shared amongst Tier 1's
Average of ~8 Datasets per Tier 1 (RAW+RECO)
Analysis, Calibration, Re-reconstruction, skim making...
Second-ary tape archive
Skims: RECO, AOD's
Tier 2
MC Data

Figure 3.2: Data flow in the CMS computing model.

some common specific traits. Lastly, there are *file blocks* which are an aggregation of datasets.

There are various types of data that flow through the Grid for the CMS experiment. Some of the most important (see Figure 3.2) are:

RAW The data as they are recorded by the detector. They include all the information from the detector, a record of the trigger decisions and other low level data. RAW data in general are not managed directly by the user and thus further processing is needed before data analysis can occur.

RECO An elaborated (reconstructed) form of data that later could be used for analysis. Specific detector reconstruction and compression algorithms are used to produce high level physics objects such as tracks, vertices, jets and etc. RECO data are not permanent: in fact they can be recalculated when newer calibration constants of the detector are available.

AOD Analysis Object Data. This data type is a subset of RECO and contains information suitable for most analyses. AODs contain high-level physics objects such as tracks, particles, four-momentums, etc..

AODSIM AODSIM are events simulated through Monte Carlo methods. They contain high-level information and are used for physics analyses.

MiniAOD MiniAOD is a new type of CMS data tier introduced in 2016, which is approximately ten times lighter than a conventional Run-1 AOD. They will be able to provide sufficient information for the majority ($\sim$ 80%) of the analyses during Run-2 [45].

Furthermore, the CMS collaboration necessitates other types of data that are not based on the Event as a unit (*Non-Event Data*). There are four categories of non-event data: construction data, which include information produced during the assembling of the detector; equipment management data contain calibraton information of the CMS subdetectors; configuration data consists of calibration and alignment parameters; and conditions data which describe the current operative condition of the detector [46]. Non-events data are stored in several central databased. These data are generated both during on-line data acquisition and off-line analysis.

3.4 CMS computing services

CMS offers a series of computing services to help the user to interact with the Grid. In the next paragraphs, some of the most important services will be presented in the next paragraphs (see Figure 3.3) [46]. Note that this is only a general high-level view, and some satellite services which may too rapidly evolve over the years are not mentioned.

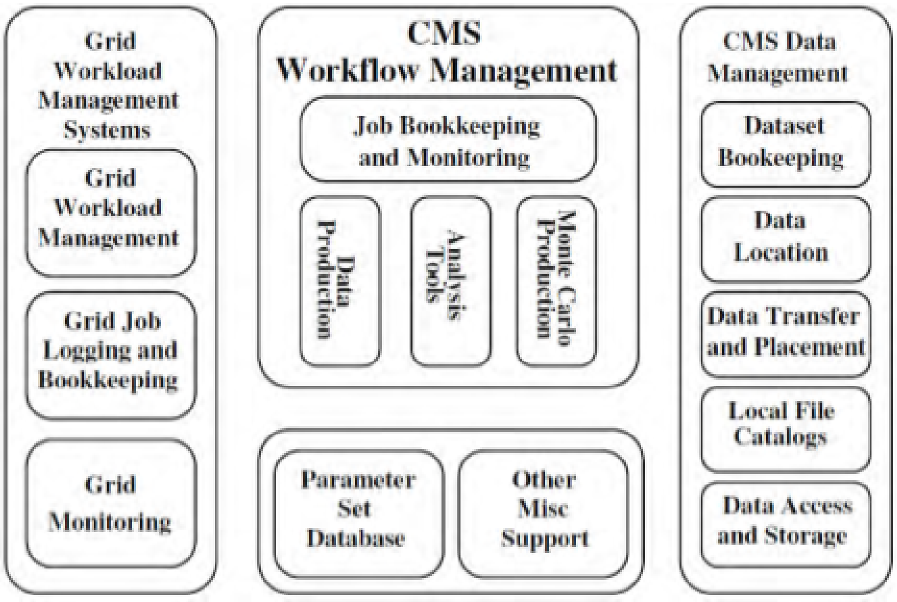


Figure 3.3: Overview of the CMS Computing Services [46].

Data management. The CMS collaboration needs services to catalogue, locate, access CMS data. The *Data Bookkeeping Service* (DBS) is a data catalogue which tells you which data exists. The *Data Aggregation Service* (DAS) is a service whose purpose is to provide a data search through a query language. *Physics Experiment Data Export* (PhEDEx) is a reliable and scalable dataset replication system in production for CMS since 2004, and stands as one of the first developed Grid based system for HEP that is still in production today [47, 48, 49].

Workload management. The analyses which run on the Grid are intrinsically independent from each other. This fact allows the workload management to distribute the workload according to the computing center capabilities and the location of the data. Then, the actual execution of the tasks is performed by *WMCore/WMAgent*. The *Berkeley Database Information Index* (BDII) [50] informs the workload management about how the various Grid sites are being exploited at the moment. Furthermore, when the analysis has been assigned to a specific computing center, a job wrapper performs the set up of the environment, then it runs the user analysis on the local dataset and delivers the requested data to user. This procedure is further supported by other various CMS tools, one in particular *CRAB* will be discuss in details in the Section 3.5. Monte Carlo simulations are managed by a web-based service: the *Monte Carlo Management* (McM). Up to now, CMS has simulated over twelve billion events, organized in approximately sixty different campaigns each one characterized by specific detector conditions and LHC running conditions [51]. More details of the campaigns production will be provided in Section 5.5.1.

3.5 CMS Distributed Analysis with CRAB

The analysis in CMS, both of data and of simulated samples, is carried out on the Grid using a toolkit called CRAB (CMS Remote Analysis Builder) [52, 53, 54]. CRAB provides an interface for users to interact with the distributed computing infrastructure (see Figure 3.4). In general, an analysis usually includes two main steps: first, the user develops his analysis code and tests it locally on a small scale; second, the user prepares and submits a large set of jobs to run on an actual large dataset using CRAB. Usually, an analysis is constituted of hundreds of jobs that are created and managed by CRAB. CRAB is a tool which is under continuous development, in this thesis the versions two and three will be used, i.e. CRAB2 and CRAB3.

In CRAB2, to perform an analysis, the user has to write a configuration file in a specific meta-language. This file has the default name `crab.cfg`. In CRAB3, the configuration file is written in Python.

The `crab.cfg` in CRAB2 is divided in various sections: CRAB, USER, CMSSW and GRID. The following parameters for the `crab.cfg` are mandatory:

- `jobtype`: the type of job that has to be executed;

- `scheduler`: the name of the scheduler that has to be used;
- `datasetpath`: the complete path and name of the dataset which has to be analyzed;
- `pset`: the name of the CMSSW configuration file;
- `out_file`: the output file name generated by the `pset` file;
- `return_data`: enables to retrieve the output in the local working area. The options are 0 or 1;
- `copy_data`: enables to copy the output to a remote SE. The options are 0 or 1.

Furthermore, it is necessary to specify whether the jobs are going to be split according to the luminosity, which is mandatory for real data, or by the number of event, which is possible only for Monte Carlo events. In both cases, two parameters have to be specified out of a list of three. For job splitting by luminosity:

- `total_number_of_lumis`: defines the number of luminosity blocks to be analyzed;
- `lumis_per_job`: specifies the number of luminosity blocks that a job can access;
- `number_of_jobs`: establishes the total number of jobs that are going to run.

For job splitting by event:

- `total_number_of_events`: the total number of events to be analyzed;
- `events_per_job`: assigns to each job the number of events it can access;
- `number_of_jobs`: specifies the total number of jobs that are going to run.

Once the proper working environment has been set-up and both the `crab.cfg` and the CMSSW configuration have been written, it is then possible to create the jobs via the command [55] [56]:

```
crab -create
```

This command creates the jobs according to the CRAB configuration file. The following step is to submit the jobs to the Grid. This is done via the command:

```
crab -submit
```

Now, it is possible to check the status of the jobs via the command:

```
crab -status
```

The CMS Dashboard is a standard tool used for monitoring the status of the jobs. The CMS Dashboard provides the user with a large set of monitoring metrics and useful information. In particular, a task monitoring service is in place that offers monitoring specifically targeted to help a user track the status of his jobs over time, including successes versus failures, etc [57]. Moreover, further information is provided in the CRAB log files that can be retrieved with the command:

```
crab -getoutput
```

The above command retrieves only log files of the jobs which are marked "Done". For the other jobs, either they have not completed yet (so the user must wait to recover the output), or they have failed (and the user can debug their failure reasons, and possibly re-submit them again via CRAB).

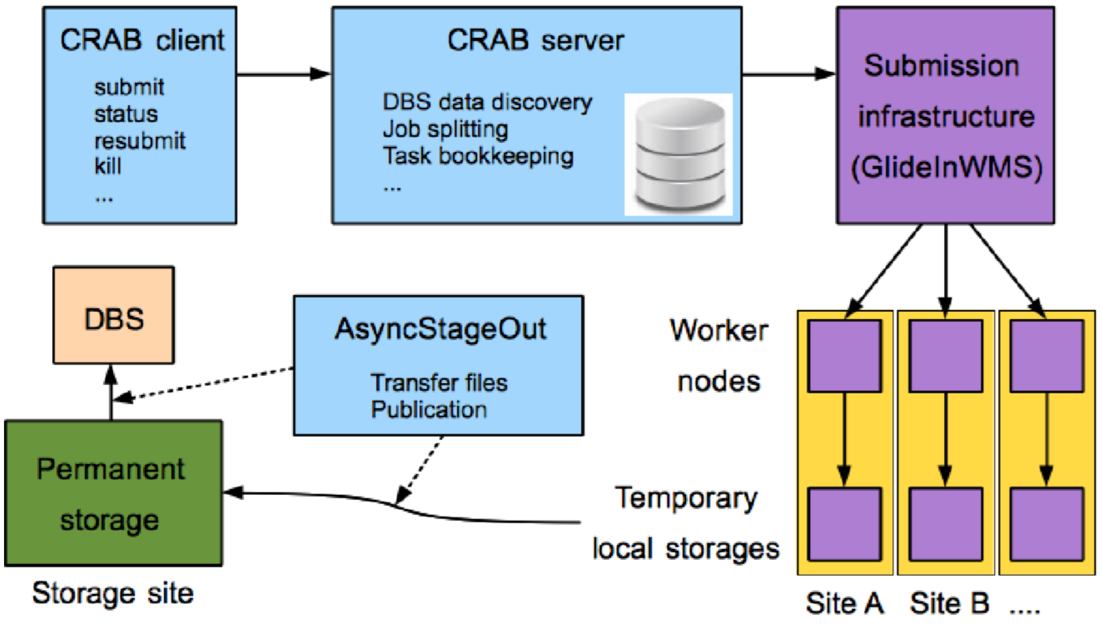


Figure 3.4: Overview of the CRAB architecture [58].

Chapter 4

Analysis performances: a CRAB-based tool

Computing resources in CMS are used mainly for two types of activities: Monte Carlo productions and data analysis. The former activity is centrally coordinated, while the latter fulfills the local requests of the research groups in CMS. These two workflows require that the computing resources are managed and monitored in ways tailored to the use-cases. On one hand, Monte Carlo productions are monitored globally at high level of details, and this approach will be discussed in Chapter 5. On the other hand, data analysis -performed with CRAB- is monitored both centrally and by the scientists who have submitted such jobs, regardless of the actual location of the resources.

This chapter presents the design and development of a tool capable of providing a quick and intuitive analysis of the measured performances in CRAB job submissions, on any Grid or Cloud resources.

The tool has been tested with job submissions on Grid and Cloud infrastructures specifically designed for benchmarking.

4.1 Design

First of all, a quick evaluation of possible programming approaches to address the monitoring needs was performed. We considered commonly used scripting languages like Perl, Python, go-lang, bash, etc. Given its wide adoption in CMS, we opted for Python. To be as consistent as possible with the CMS code base, Python 2.7 was used. A quick overview of the available monitoring systems in CMS for distributed analysis was performed, to identify the actual needs and to avoid overlap with existing tools. The focus of this effort has been to design a lightweight tool that any user can exploit with basically no learning curve to optimize the sequence of creation, submission and monitoring of analysts jobs, thus achieving a higher efficiency in physics analysis. A detailed description of the tool can be found in the following paragraphs.

In order to operate it, a user that has carried his analysis in a specific working directory (specified in the `crabcfg.py` file) for several tasks should retrieved the logs through the CRAB command:

```
crab getlog <working-directory>/<task-directory>
```

Then he simply needs to run the tool giving as argument the working directory:

```
python tasks_analyzer.py <working-directory>
```

and, at the end, the program creates a directory called `Tasks_Analysis_<working-directory>` containing both the information about the single tasks and on overall analysis of the tasks (see Figure 4.1).

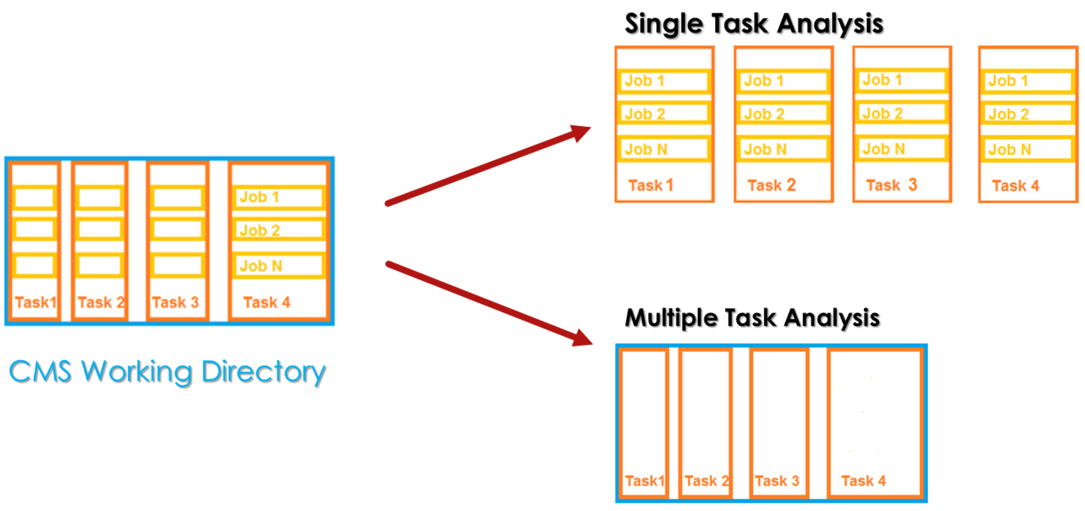


Figure 4.1: Tool design. The tool performs two types of analysis on the CMS working directory: a single task analysis for each task individually, and a multiple task analysis which looks at each task as a whole. More details will be provided in Sections 4.2 and 4.3.

The metrics that have been studied are very simple. The idea is indeed to start with some simple observables and let the user express additional needs, or implement more observables himself. Such metrics are:

- `CPU Time`: time the application spent on WN CPU.

- `Execution Time`: overall job execution time.

- `CPU Efficiency`: parameter which evaluates the CPU's efficiency as follows:

$$\texttt{CPU Efficiency} = \frac{\texttt{CPU Time}}{\texttt{Execution Time}}$$

- `Number of events`: total number of events per each task.

4.2 Single Task Analysis

For each task contained in the working directory the tool performs several actions:

1. It unpacks the log files and stores them as .xml and .log files.
2. It retrieves information from these files and produces a dump of them in a .csv file which can be then use for a more sophisticated analysis using packages such as R[59].
3. It produces plots in .png format, using the metrics described above. Moreover, the graphs are stored in root files named `<task-name>.root` for further analysis exploration with ROOT [60].

As a demonstration of the features of this tool, in Figure 4.2-4.11, the plots produced by analyzing four tasks submitted to a Cloud infrastructure in Bologna are shown. These tasks were used to compare the performance of a Cloud infrastructure to the Worldwide LHC Computing Grid and provide a proof of concept of the elastic extension of the CMS-Bologna Tier-3 Site on an external Cloud infrastructure, implemented on OpenStack [61]. A newly designed LSF [62] configuration was used in what it can be considered a “Cloud Bursting” of a traditional CMS Grid Site [63]. The project aimed to implement an extension of the traditional” Grid resources to a Cloud infrastructure in the most transparent manner for the user. However, there are small differences between the two architectures and these can be detected thanks to this type of analysis tool.

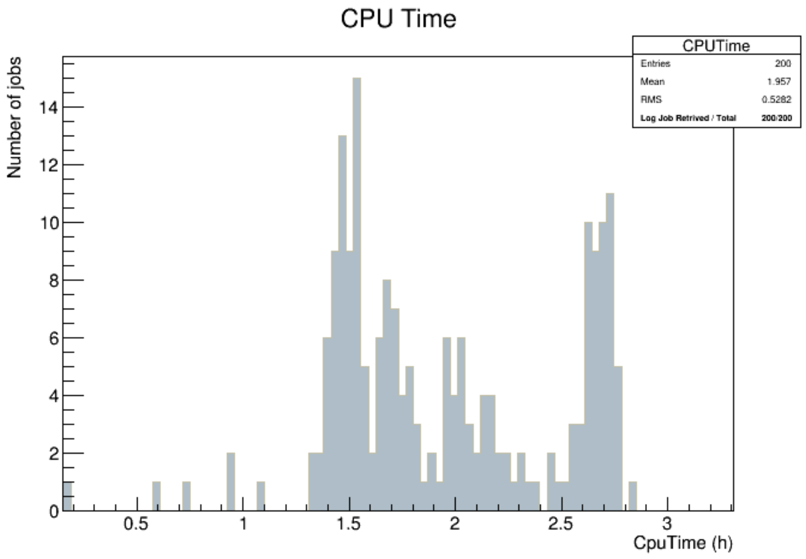


Figure 4.2: Number of jobs as a function of the CPU Time for a workflow that is described in detail in the text.

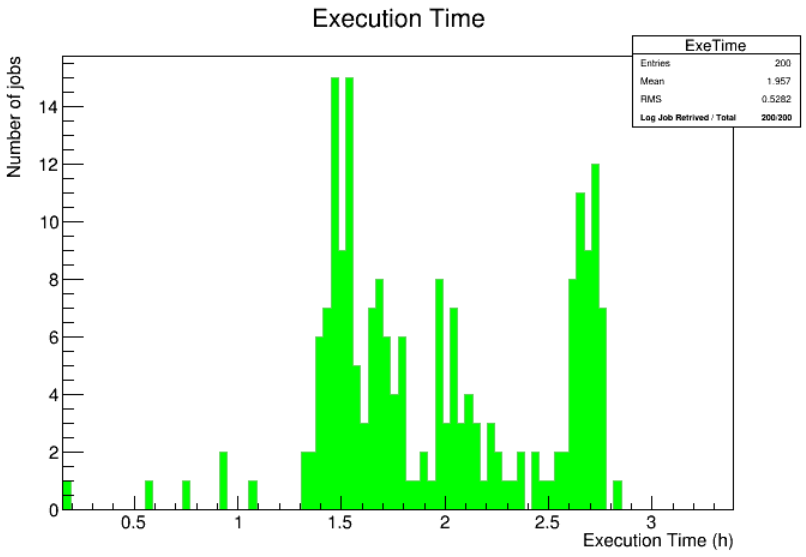


Figure 4.3: Number of jobs as a function of the Execution Time for a workflow that is described in detail in the text.

The plots in Figure 4.2 and 4.3 provide a general overview of the time of execution of the single jobs. With this type of information, an analyst can optimize his job submission exploiting his computing resources without having moments in which there are no jobs running. For example, take Figure 4.2 and consider this case. An analyst typically will have to submit many tasks consisting of many jobs each, and would aim to see them finished in the minimum amount of time. It is realistic that he submits some jobs only, or a small task, first to figure out how much time it may take to run all tasks in the work queue. Running a small task and/or only few jobs may not be statistically significant, i.e. they may end up in underperforming Grid sites (thus overestimating the overall work completion time), or only few of them may succeed (thus yielding a not significant statistics of successful jobs to draw any conclusion). It is probably wiser to submit a realistic, medium-size task and use the tool documented in this project to analyze the actual CPU time and CPU efficiency patterns of all the jobs in such task. Consider for example that this is done and the streamlined analysis of its results allows to conclude that the vast majority of jobs in the test task complete in less than 8 hours running on 5 different (in performance, location, etc) computing centers. It will hence be straightforward to presume that also all the other tasks may do the same: as a consequence, having this insight at hand, a proper scheduling of the user's submissions in the next few days is possible, and ultimately the whole work completion time will be largely reduced with respect to a "let's submit randomly and see what happens" approach.

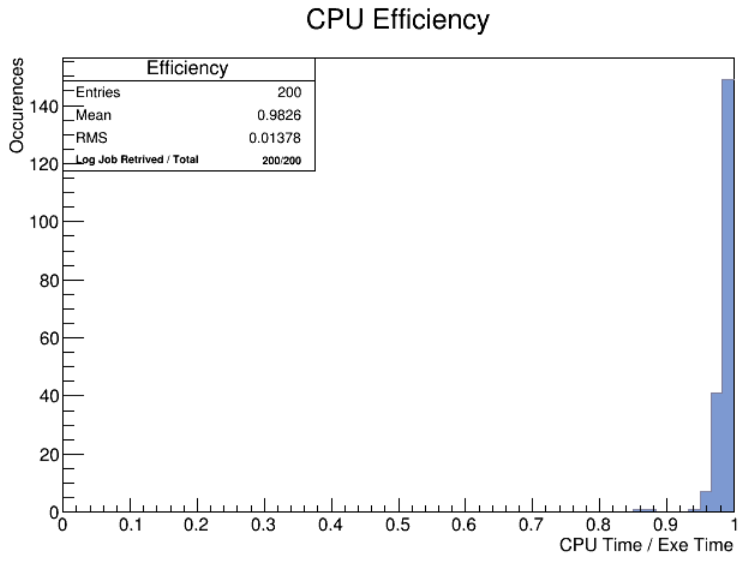


Figure 4.4: Number of jobs as a function of the CPU Efficiency for a workflow that is described in detail in the text.

The plot in Figure 4.4 shows if the computational resources are used efficiently or whether there are problems for certain tasks. Furthermore, this metric is fundamental to compare the performances of the Grid to any Cloud infrastructure. Indeed, in the context of the comparison previously described, one expects to obtain different peaks of CPU efficiency for the Tier-3 Cloud infrastructure in Bologna and the Grid. In the end, it is a fast way to evaluate two completely different and unrelated architectures.

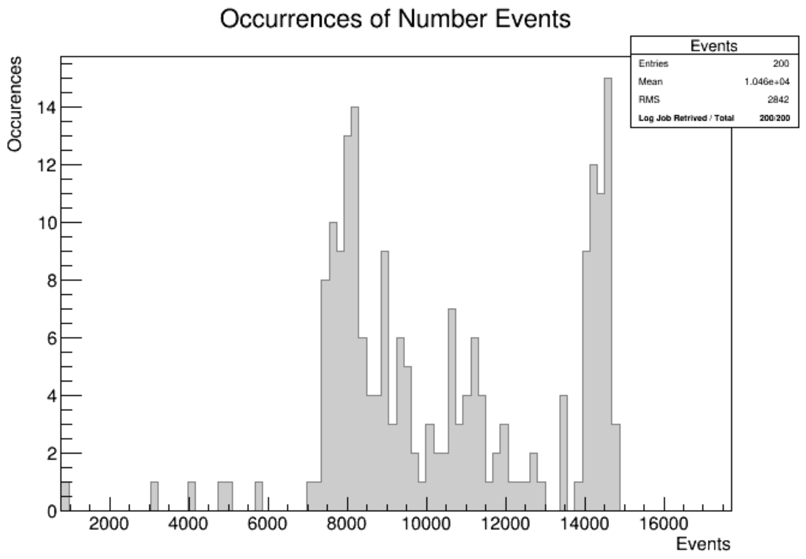


Figure 4.5: Occurrences of the number of events for a workflow that is described in detail in the text.

It is also interesting to have a quick graphical way to monitor how many events have been processed - as from the info in the logs - without actually looking at the logs themselves but by just parsing them and filling an histograms, as shown in Figure 4.5.

A feature which is highly desirable in this kind of tools is to been able to plot correlations between different observables. Thus, this tool can produce bi-dimensional plots of all the metrics discussed above according to the user's needs. The plot in Figure 4.6 shows how the execution time is correlated to the CPU time. The graph in Figure 4.7 looks for the degree of correlation between the Execution Time of each task and the Number of Events contained in each task.

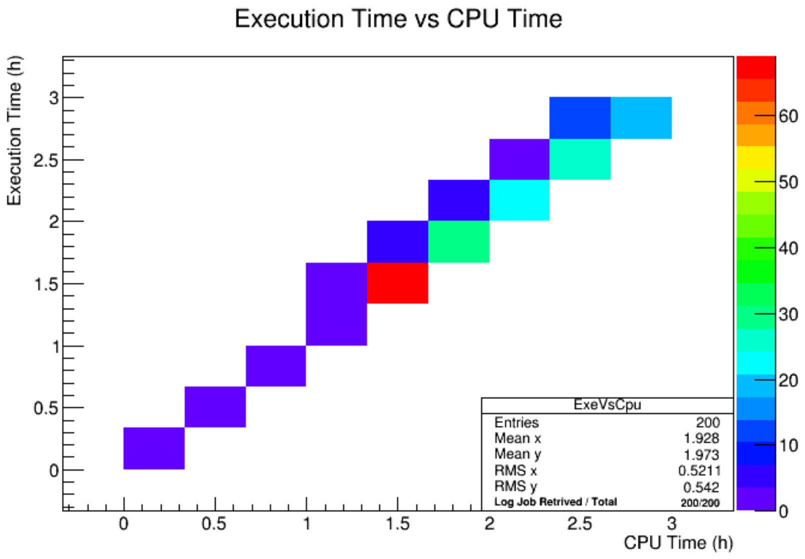


Figure 4.6: Execution Time as a function of the CPU Time for a workflow that is described in detail in the text.

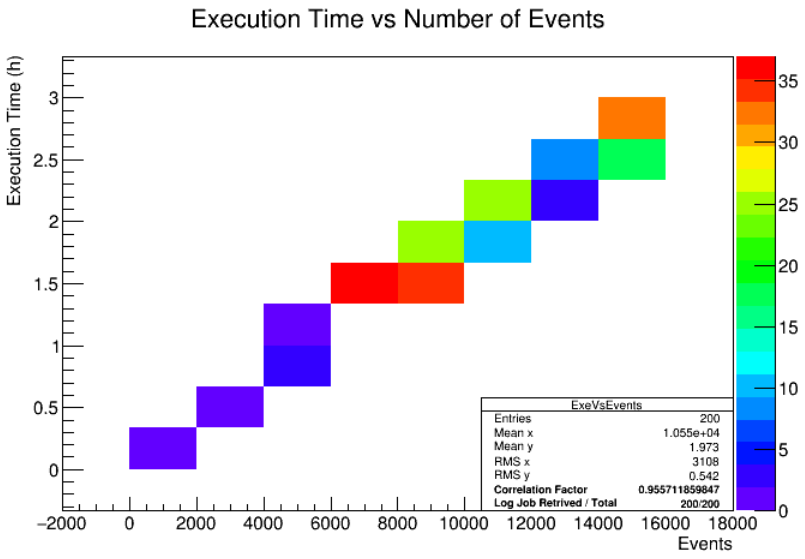


Figure 4.7: Execution Time as a function of the number of events for a workflow that is described in detail in the text.

4.3 Multiple Task Analysis

So far, we discussed a per-task monitoring, in which all entries in a plot refer and belong to just one task. Another useful aggregation in CRAB is per Working Directory. In fact, typically, users need to submit plenty of tasks, some of which may become useless as the analysis proceeds, but some other may be a reference for all the subsequent tasks creation and submission. It is hence interesting, inside a unique working directory, to graphically compare - for example - the average CPU time (or other observables) *per task*, in a single *per working directory* plot.

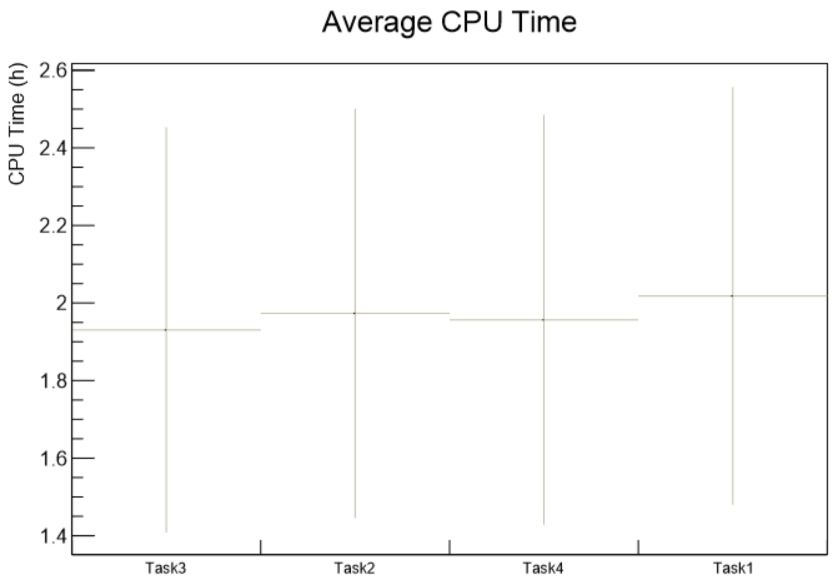


Figure 4.8: Averages of CPU Time for four related workflows (the error is the standard deviation).

In Figure 4.8, the averages of CPU time of different tasks are plotted. This may help to spot the tasks that use more CPU time in their jobs' execution. Similarly, in Figure 4.9, the averages of Execution Time for different tasks are plotted. This could help identify which tasks take overall more time to finish.

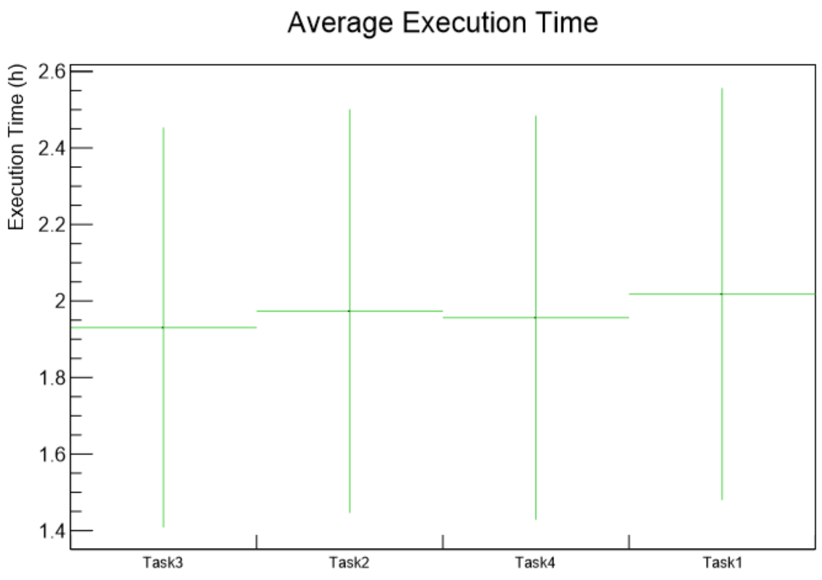


Figure 4.9: Averages of Execution Time for four related workflows (the error is the standard deviation).

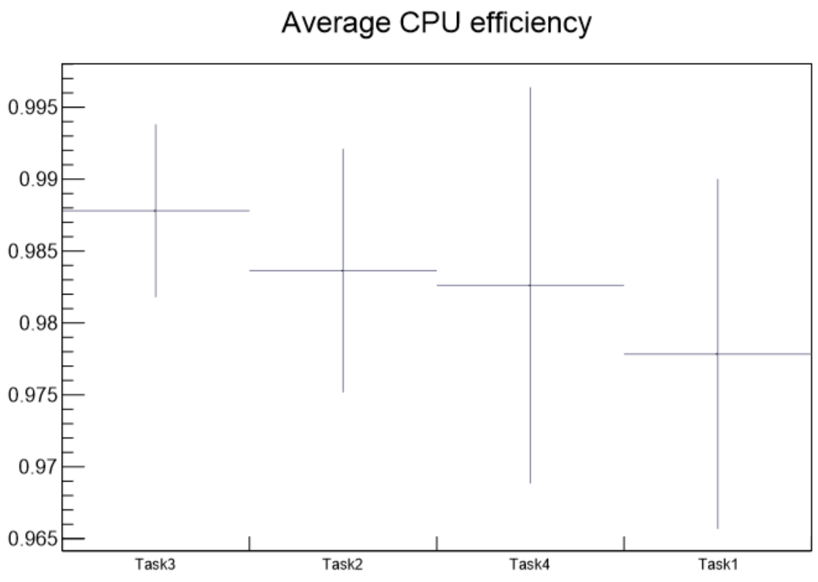


Figure 4.10: Averages of CPU Efficiency for four related workflows (the error is the standard deviation).

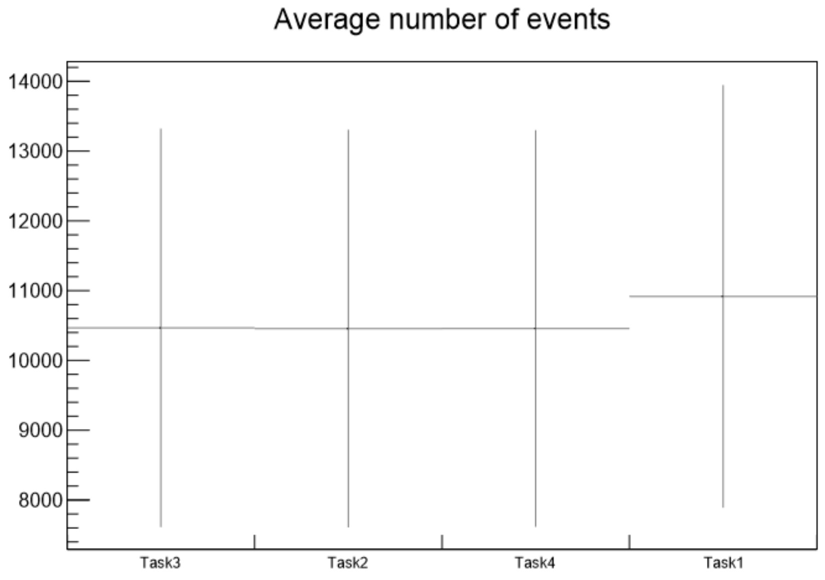


Figure 4.11: Averages of number of events for four related workflows (the error is the standard deviation).

From the combined analysis of these plots it is possible to extract even more valuable informations. For example, let's consider the plots in Figure 4.10 and 4.11. In Figure 4.10, Task1 has a lower CPU efficiency compared to other similar tasks. Now observing the average number of events of the tasks in Figure 4.11, one can conclude that having to access a larger number of events might be the cause of the lower CPU efficiency. This has only been possible by observing the four tasks in parallel.

4.4 Job submissions for benchmarking

The tool has been tested with job submissions on Grid and Cloud specifically designed for benchmarking. The first tests of the tool were performed on six small tasks in which it was fixed the `NJOBS` to 10 whereas the `unitsPerJob` were progressively increased (1, 2, 5, 10, 20, 50). These tests were used to initially develop the tool. Once the reliability of the tool was reassured, the tool was tested with log files coming from the analysis of the Tier-3 Cloud infrastructure in Bologna containing up to 200 jobs. All the tests were based on the same workflow: an analysis task in the context of the fully hadronic top physics [63].

Chapter 5

Performance studies of CMS workflows using ElasticSearch, Jupyter Notebook and Kibana

5.1 Big Data Analytics in CMS

Big data analytics is a collection of newly developed techniques for evaluating large data sets containing a multitude of data types to discover correlations, users preferences and unknown patterns. Big data analytics has been exploited successfully in various business areas to improve sales, revenue opportunities, operational efficiency, and customer service.

Lately, these techniques have also been exploited in the CMS experiment. In fact, beyond the data produced by the physics collisions or the Monte Carlo simulations, there are plenty of metadata concerning the performances of the computing facilities, which are seldom analyzed and big data analytics will help shed some light on it. Metadata in CMS comprehend different types of non−physics heterogeneous data, which can be divided in:

- **Structured data** is any data that resides in a fixed field within a record or file and can be stored, for example, in a SQL database.
- **Semi structured data** is a type of information which is not located in a relational database; however, since it has some organizational properties, it can be analyzed without too much difficulties and could be farther process in order to be stored in a database.
- **Unstructured data** is data which cannot be easily stored in a database.

Figure 5.1 provides a schematic representation of the structured and unstructured metadata in CMS. These data includes machine data, monitoring logs, accounting information, etc.

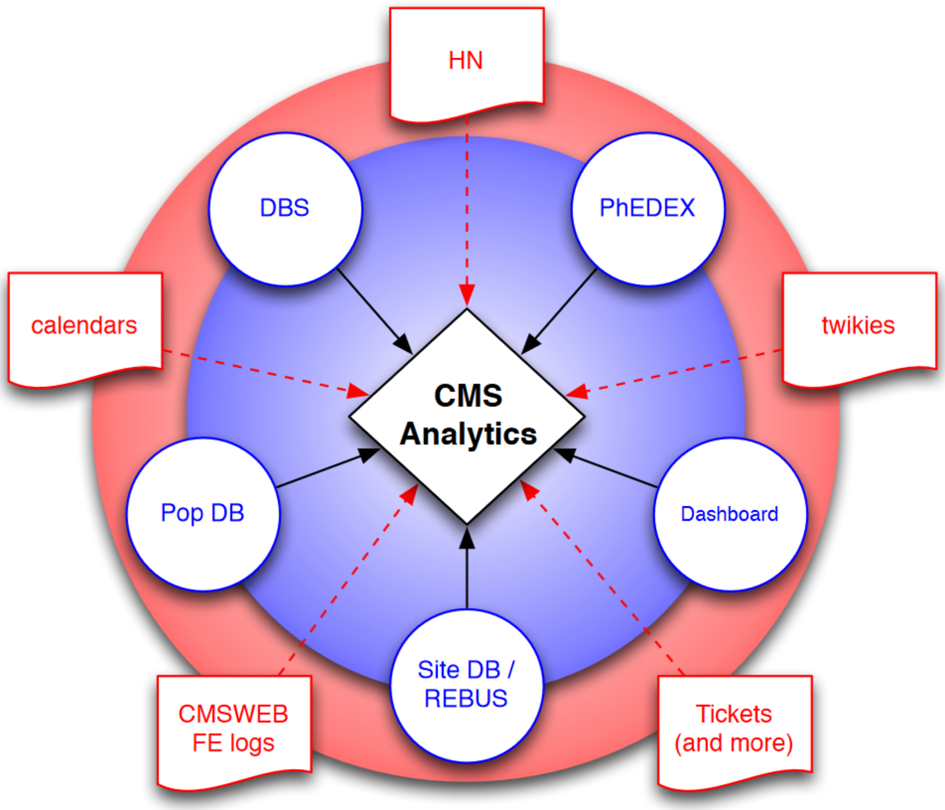


Figure 5.1: Schematic representation of the structured and unstructured metadata in CMS [64].

Currently, CMS is carrying on the Data Analytics project to analyze metadata [64]. This project is divided in several sub–projects that have different scopes as well as timelines. The Data Analytics project primary goal is to create adaptive data–driven models of CMS Data Management (DM) and Workload Management (WM) activities. This system aims at being capable of forecasting, for example, future behaviors of the jobs' submissions from the measurements of their performances in the recent past. However, as medium–term goal, the intent of the Data Analytics project is to improve the efficiency with which the CMS computing resources are exploited, and to better understand the metadata produced so far ("Run–1" and "Run–2").

In this context, also adaptive computing model have to be developed in order to cope with this metadata which may change and evolve in the future.

5.2 Hadoop

Hadoop is a software platform and framework for distributed storage and distributed processing of big data on computer clusters: it is a platform because it is a long-running system which executes tasks, and it is a framework because it gives a high layer of abstraction to application developers [65]. Hadoop is written mainly in Java and it is being developed under the Apache license. Cutting and Cafarella, the founders of Hadoop, developed their idea based on two papers released by Google in 2003 and 2004 in which Google discusses how it organizes and manages its data [66]. Nowadays, Hadoop represents a standard for Big Data analysis. This is the reason why, Yahoo! and several other companies use Hadoop. The core Apache Hadoop project is constituted of three major resources:

- **HDFS** (Hadoop Distribuited File System) is the place in Hadoop in which data are stored. HDFS is designed to run on clusters of inexpensive commodity servers. It is optimized for read intensive operations. The system makes several copies of each data, thus it is improbable to lose data. However, failures are not completely avoidable. Furthermore, HDFS is a WORM-ish (Write Once Read Many) file system: files cannot be overwritten but only extended. The main task of HDFS is the sequential read of large files, which are generally divided in large blocks in size ($64MB$ or more) distributed among the clusters.

- **YARN** is the management framework for exploiting Hadoop resources such as CPU, RAM, and disk space. YARN operates in a space between the data and where MapReduce runs.

- **MapReduce** is the framework for analyzing the distributed data. MapReduce is made of Java applications called mappers and reducers. Mappers convert a first set of data into a second set of data in which the single elements are tuples (key / value pairs). Then, reducers analyze the output produced by the mappers.

For the advantages described in the previous paragraphs, HDFS has been implemented in several Tier-2 CMS. It has shown to be a viable alternative to the standard Grid Storage Element especially for what concerns scalability, reliability, and manageability [67].

The work performed in this thesis is based on a prototype , non-production installation set up for test and commissioning purposes by collaborators in the CMS Software/Computing team. At the current deployment stage and utilization level, this set-up (described in more details in Section 5.4) is considered to be largely sufficient for this study. In view of an upgrade to a CMS production service and an extension to more complex usage patterns by a large users base, an inclusion of technologies from the Hadoop ecosystem is being considered. In particular, at the moment of writing this thesis, a complete duplication of the system onto CERN resources (Hadoop based installation) is being planned.

5.3 ElasticSearch and Kibana

ElasticSearch is a open source search server developed in Java with an HTTP web interface for searching text [68]; it is powered by Lucene, which is an open source full−text search library.

The primary data−type that ElasticSearch uses is JSON. However, internally, the JSON is converted to fields for Lucene's key or value API. Complex JSON types are also supported, with the aid of both arrays and object notation. Furthermore, ElasticSearch documents are able to manage more complex relations, for example, parent−child relationships and nested documents. A Google Chrome plugin, Sense, provides a JSON aware interface for ElasticSearch.

In ElasticSearch data organized in Indexes, which represent the single largest unit of data. Although ElasticSearch allows cross−index searches, it is preferable to organize related data inside the same index. Indexes can be thought as analogous to databases from a relational perspective. The CRUD commands (create, read, update and delete) are the four basic functions in databases. In ElasticSearch, they are performed through: PUT, GET, POST, DELETE.

Moreover, ElasticSearch is endowed of a Python library to support queries also through Python code. One way to display ElasticSearch queries and aggregations is through an open source tool called Kibana [69].

Through this thesis, the queries that have been submitted to ElasticSearch have been designed in Python and managed using Jupyter Notebook [70]. This choice has allowed a higher flexibility in the code development. A Kibana interface (see Section 5.4) has been used at the beginning to familiarize with the context of the ElasticSearch instance studied (see Section 5.4) and to validate some of the first results obtained.

All the Python code base is open to the CMS community and it has been commented thoroughly.

5.4 The CMS Metrics Service from University of Nebraska

The CMS Metrics Service is a comprehensive database of all production and analysis jobs run in the CMS Global Pool (based on Condor [71]). For this section, input from colleagues in the CMS management and in the CMS team of University of Nebraska was crucial [72]. Special acknowledgments for the technical work should go to Brian Bockelman. The main purpose of this infrastructure is to reuse modern, scalable database technology (ElasticSearch) and integrate it directly with HTCondor via the Python binding API.

The goal of this database is to enable job-level analytics involving many different job attributes; some attributes are specific to the CMS ecosystem (such as number of events processed, bytes read by CMSSW, dataset processed, task name) while others are basic attributes recorded by HTCondor (such as CPU, memory, and disk usage). A list of the attributes that have been used in this work can be found in Appendix A.

This service exploits the expressivity of HTCondor ClassAds - which allows CMS to keep CMS-specific, schema-free data as part of the job description - and the NoSQL nature of the ElasticSearch database. This service has been active since mid-February 2016. Between one and two million new job records are collected each day. The component architecture of this service is diagrammed in Fig. 5.2.

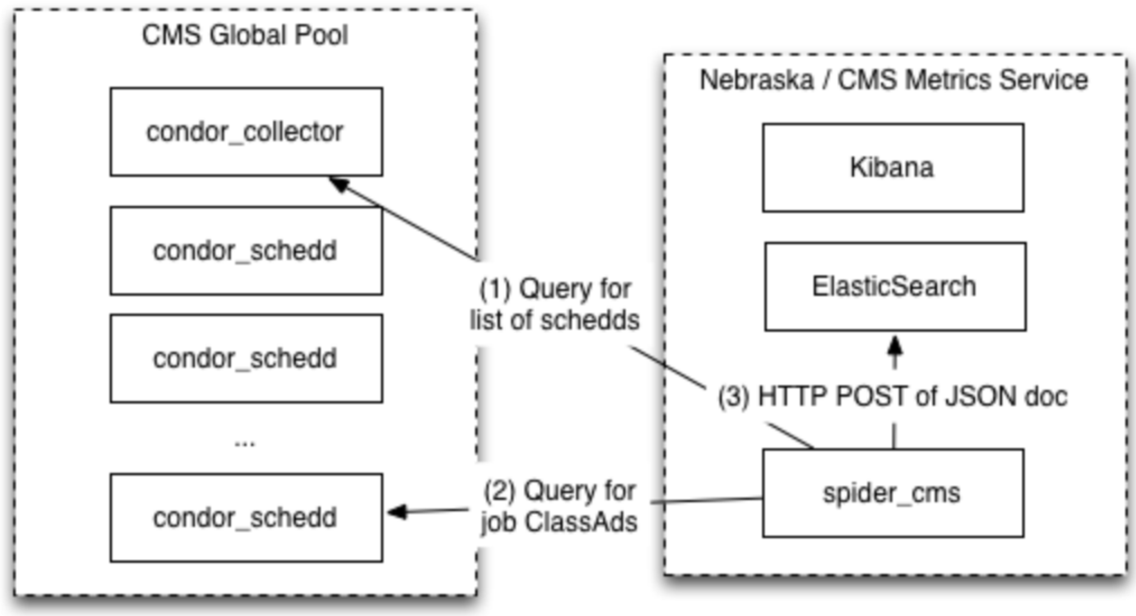

CMS Global Pool
condor_collector
condor_schedd
condor_schedd
...
condor_schedd
(1) Query for list of schedds
(2) Query for job ClassAds
(3) HTTP POST of JSON doc
Nebraska / CMS Metrics Service
Kibana
ElasticSearch
spider_cms

Figure 5.2: Diagram of the component architecture of Nebraska / CMS Metrics Service.

As stated above, the CMS Metrics Service is based on an ElasticSearch database. A CMS-custom script, `spider_cms`, runs approximately every 15 minutes to do the following:

1. Query the `condor_collector` on the CMS Global Pool for a list of active `condor_schedd` services.

2. For each active schedd, query all the active jobs in the queue and recently completed jobs.

3. For each job, we convert the HTCondor ClassAd to a JSON document. This often involves unit conversion, attribute name normalization (some CMS applications use slightly different names), and aggregating multiple attributes as appropriate.

4. Each jobs JSON-based description is uploaded to the ElasticSearch database.

To perform analytics on the database contents, it is also provided an Kibana-based interface.

This service was initially deployed as a prototype and is awaiting upgrade to a more traditional production deployment of ElasticSearch.

Right now, it is powered by a single virtual machine on the "Anvil" OpenStack service at the Holland Computing Center at the University of Nebraska-Lincoln that has 8 cores and 64GB of memory. The database is located on a 1TB Ceph-backed block device. All services run on the same VM. CMS planned to roughly quadruple the available hardware in July 2016.

The ElasticSearch version is currently 2.2.0 and Kibana is version 4.4.1. The only custom software for this service is `spider_cms` [73].

5.5 Workload Management view

In the CMS WM sector, plenty of useful information resides in the log of the jobs submitted to Grids and Clouds. Production jobs and analysis jobs represent different families of CMS workflows, and the tools used to deal with production and analysis needs are different, thus also the log files are different. Despite such diversity, plenty of knowledge on CMS computing operations lies in them, and in principle they represent the most important source of high granularity details any performance study may hope to access.

On the other hand, in some cases there are quicker and more effective ways to store for future access at least a set of such information, which is perceived as sufficient to perform most of the a-posteriori analysis, without needing to access the full set of log files.

CMS is exploring both approaches. In the former case, there is a development project called WMArchive which aims at collecting and archiving at CERN all logs from all jobs belonging to Monte Carlo production activities. This is in progress and is not focus of the current work. The latter approach is the one described in 5.4, and we will focus on this in the rest of the thesis.

Since almost 2 years, CMS jobs are submitted via a condor Global Pool (see Section 5.4). HTCondor is nowadays one of the most adopted site Workload Management systems at Grid sites, see e.g. Figure 5.3 for its usage in CMS. As explained in 5.4, HTCondor ClassAds provide us an impressive (despite not complete, i.e. no full logs) amount of information on which we can base all the explorations presented in the following sections. A selected list of attributes available on such resource can be found in Appendix A.

In this study, we focus on a few selected monitoring use-cases and we demonstrate that it is possible to fulfill the monitoring need by exploiting the CMS Metrics Service, while it would be impossible to do so through official CMS or WLCG monitoring mechanisms.

Note that the CMS Metrics Service set-up is relatively recent, so the plots in the following sections have been produced only over the time window covered by the database so far, i.e. from mid February 2015 to mid June 2016. The date of mid June is indeed the date in which the plots in this thesis have been updated for the last time before including them in the thesis. In the following we will refer to the mid February - mid June period as the "time frame under study".

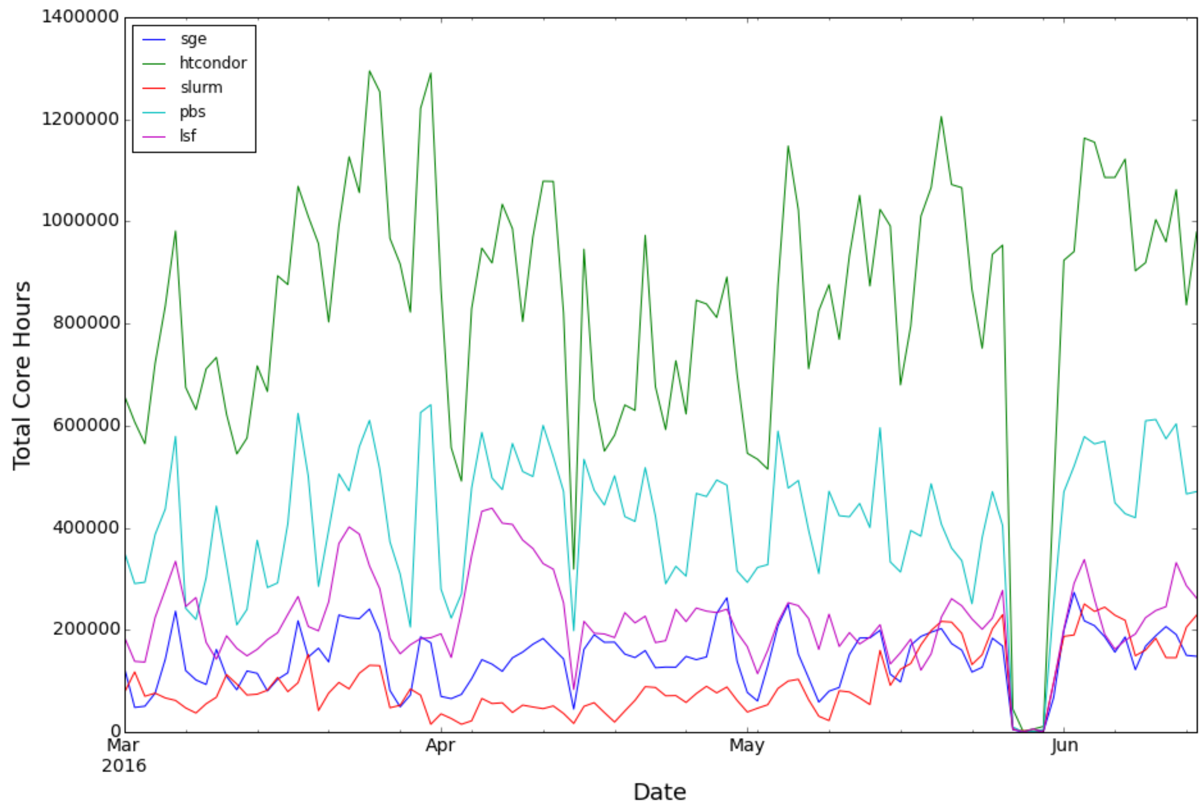


Figure 5.3: Daily sum of Core Hours for different Site WMS.

5.5.1 Use-case: jobs count across production and analysis

The first metric of interest is the number of completed CMS jobs belonging to different CMS activities. In particular, we can focus only on two categories: centrally-submitted simulation jobs and distributed analysis jobs. The number of completed jobs is a very simple metric to extract, and the CMS Dashboard can be used for it. On the other hand, the CMS Dashboard does not have the concept of campaign, so it is particularly interested to explore this metric in the CMS Metrics Service.

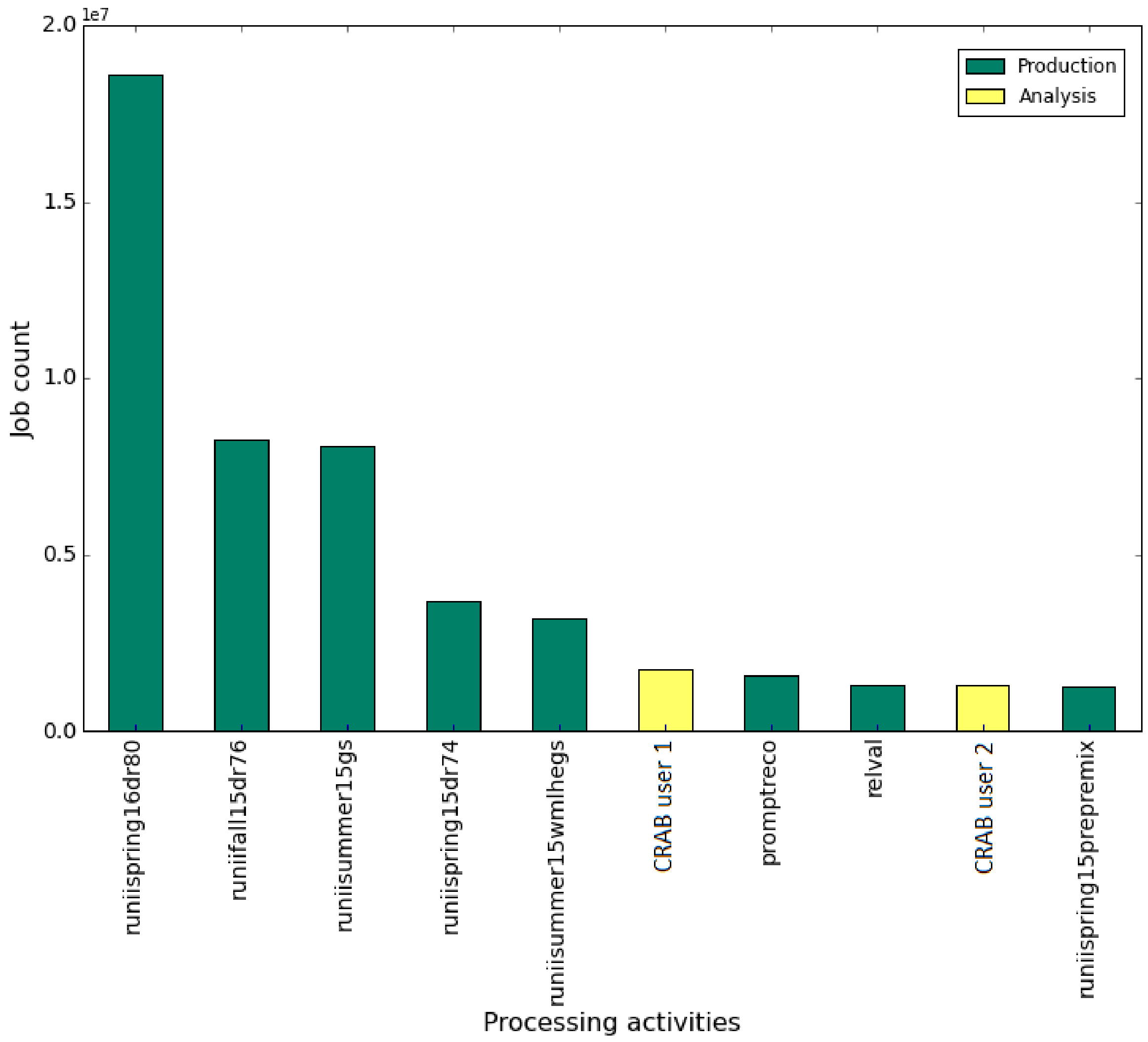


Figure 5.4: Job count for the 10 most relevant "processing activities". See text for further explanations. Blue bars refer to user analysis activities and yellow bars refer to different Monte Carlo production campaigns.

In Figure 5.4 the number of jobs (job count) for the 10 most relevant CMS "activities" in the time frame under study is shown. The visualization includes both production and analysis jobs. By "analysis activities" in this context we mean jobs submitted by single analysis users via the CRAB3 tool. By "production activities" in this context we mean major Monte Carlo production efforts that focus on a specific processing step and produce in output a specific data tier. The latter entity is called production "campaign" in the CMS jargon: e.g., "GS" indicates a Monte Carlo campaign that produces GEN-SIM outputs, "DR" indicates a Monte Carlo campaign that reads GEN-SIM inputs and PU events and produces Digi-Reco outputs, etc).

In general, it can be observed that most CPU resources in CMS are allocated to production campaigns (green bars in Figure 5.4) and only a couple of power users manage to get into the top 10 list of jobs count (yellow bars in Figure 5.4). The relative fraction of production versus analysis jobs is roughly constant over time, as can be seen for example in Figure 5.5 where this is displayed for May 2016: apart from the end May period (when a monitoring glitch occurred in Nebraska and no data filled the plot), the fraction of analysis never was beyond $25 - 30\%$ of the total of core hours used by CMS.

Concerning the analysis jobs in Figure 5.4 also a couple of analysis users activities can be seen (i.e. yellow bars). These are due to individuals submitting massive amounts of CRAB3 jobs to the Grid in the time frame under study: this is not unknown, i.e. the largest of these bins, the sixth bin in Figure 5.4, correspond to the Grid certificate of a CMS expert that is used in an automatic submission mechanism of lightweight test jobs on a regular basis to verify the status of the overall WLCG Tiered infrastructure. It is interesting to note that these jobs count is more than actual Prompt Reconstruction jobs and Release Validation jobs, both important categories of CMS Grid jobs - and the relative contribution of each category to the overall CPU utilization is matter of discussion now in CMS. These pieces of information at this level of granularity could not have been extracted without the set-up described in this thesis.

The breakdown into campaigns deserves further discussion. Figure 5.4 - with respect to plots doable from the CMS Dashboard - is able to disentangle where most of the CMS users spent CPU hours, i.e. in which specific CMS activity. In the time frame under study, CPU time was mostly spent in 3 major campaigns: RunIISpring16DR80, RunIIFall15DR76, RunIISummer15GS. They alone represent the 71% of the total jobs submitted by CMS in this time window, the first two heaviest campaign being both DR ones. Specifically on each:

- RunIISpring16DR80 is a massive Digi-Reco campaign requested by the CMS Physics coordination team for the analyses in view of the ICHEP 2016 conference in Chicago in Summer 2016 [74]; this campaign started in the Spring 2016, and is now in the tails, producing already more than 5.6 billion of events.

- RunIIFall15DR76 is a Digi-Reco campaign similar to the previous one, with a less recent CMSSW version (CMSSW 7.6 instead of the current CMSSW 8.0) requested

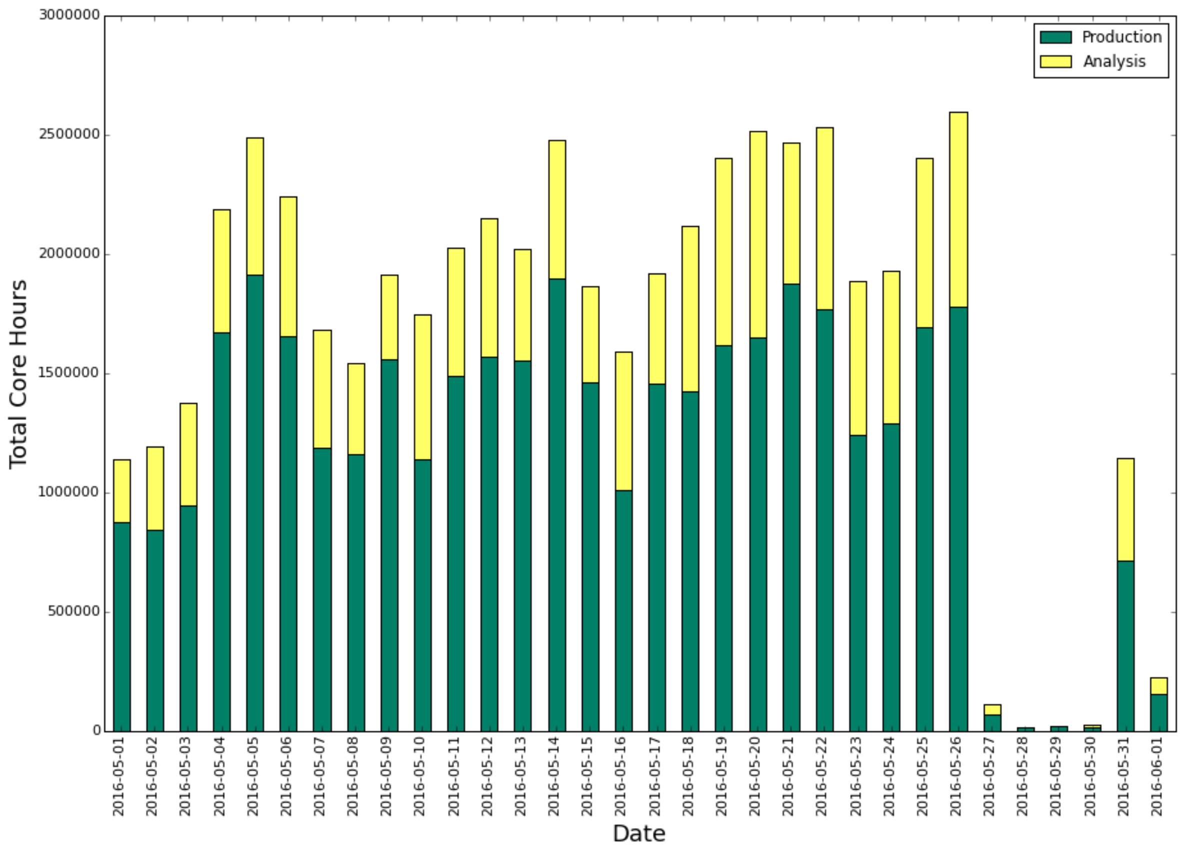


Figure 5.5: Total Core Hours for jobs of production and analysis in May 2016.

by CMS Physics coordination in view of the analyses to be presented at the Moriond conference in Spring 2016 [75]; this campaign started in late 2015 and continued through early 2016, producing close to 9 billion of events.

- RunIISummer15GS is instead a campaign aimed at populating the library of GEN-SIM events for DR campaigns afterwards, among which the 2 quoted above; this campaign started in Summer 2015 and is still open, now at more than 10 billions of events produced.

Taking as DR and GS examples the RunIISpring16DR80 campaign and the RunIISummer15GS campaign respectively, their event throughput over time is shown in Figure 5.6 and Figure 5.7, based on plots from the CMS Physics Data and MC Validation (PdmV) team [76] under the CMS Physics Performances and Dataset (PPD) project [77]. Note that these 2 campaigns are ranked first and third in 5.4, and the DR campaign corresponds to roughly twice as many jobs as the GS campaign. This correctly indicates that the load associated to DR is higher than the one associated to GS, but fails in quantitatively assess it. A best metric to do so is the total number of core hours per campaigns, as discussed in the next section.

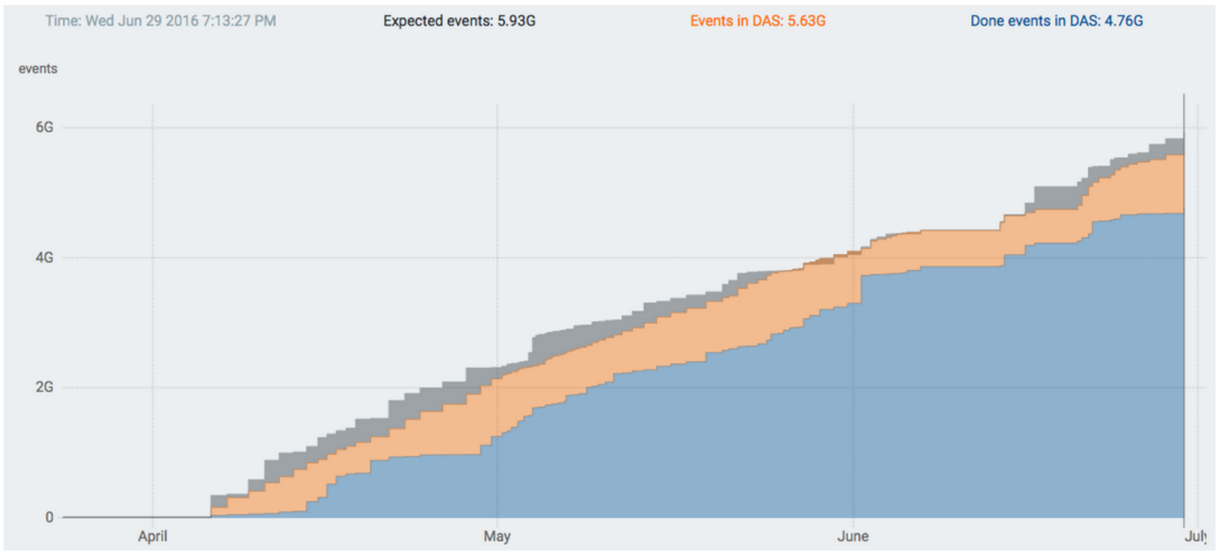


Figure 5.6: RunIISpring16DR80 campaign progress over time.

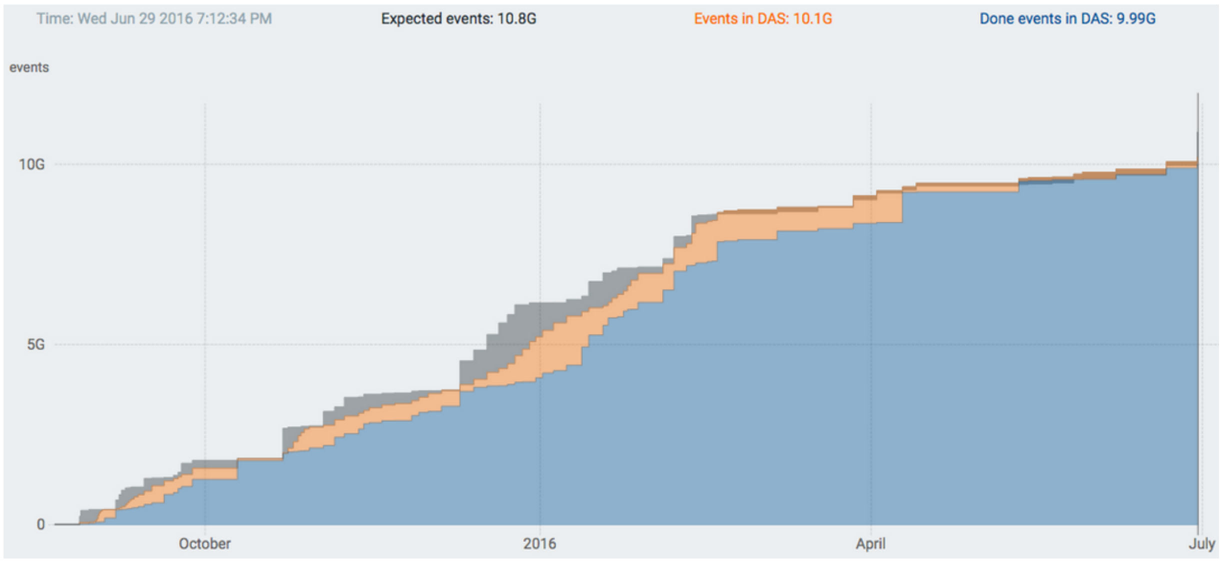


Figure 5.7: RunIISummer15GS campaign progress over time.

5.5.2 Use-case: total core hours per production campaign

The breakdown into the total number of core hours per Monte Carlo campaigns in May 2016 is shown in Figure 5.8. The choice to display May 2016 is motivated by the fact that this is a full month in which this production went at full steam, as: *i)* it started only on April 6th, so April is not a complete month of production; *ii)* it began to slow down and even plateau in June.

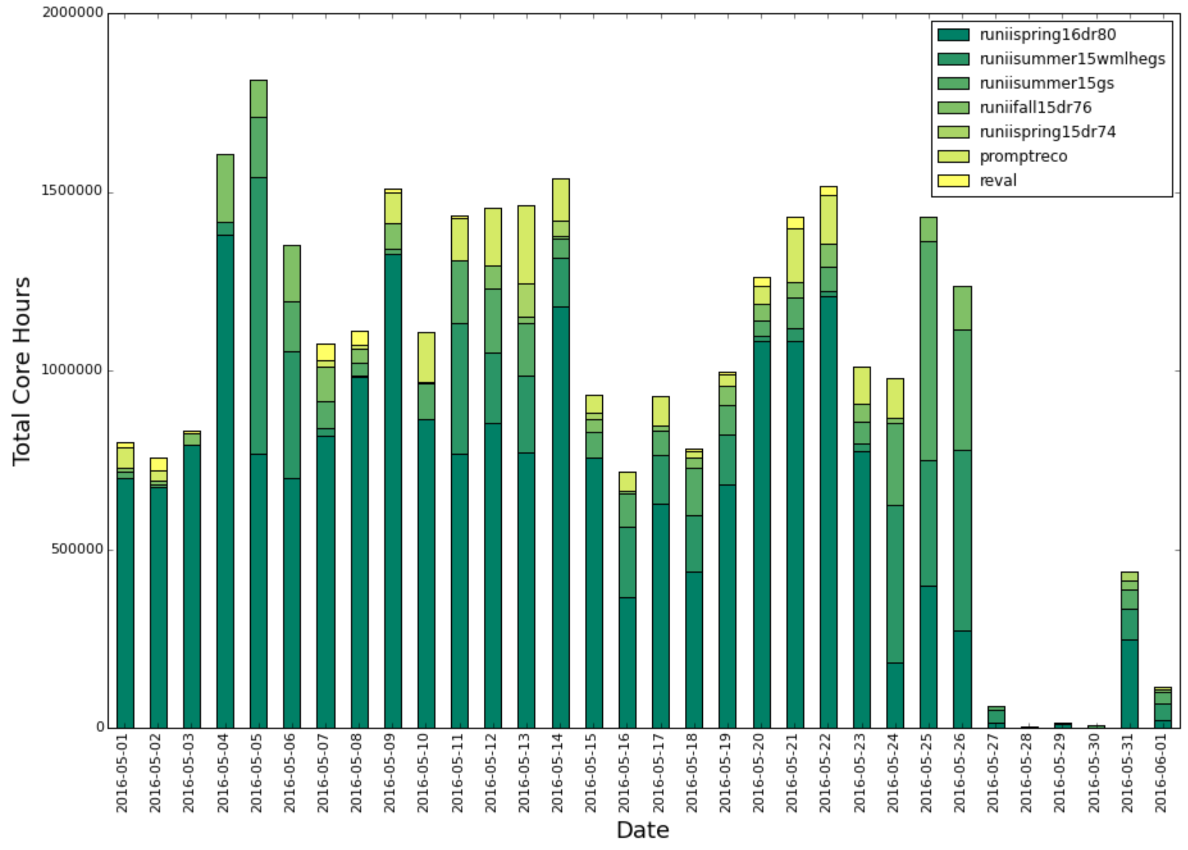


Figure 5.8: Total Core Hours for the major production campaigns ongoing in May 2016.

During the selected month, details on a daily basis can be observed, which is particularly valuable as a piece of information. In fact, CMS production proceeds on a constant basis: operations experts check on a hourly basis progresses and issues, while managers overview the situation with a daily or even weekly resolution only. A useful way to monitor how a production campaign is going on is to use information from the Global Pool, clean them from analysis jobs information, and monitor the progress - per campaign - on a daily basis for a relatively long period of time - exactly what is done in plots like Figure 5.8. Doing so, it was clearly visible that RunIISpring16DR80 have been using most of the CMS CPU power almost every day in May 2016, while other campaigns have been using their share at the level of up to $10 - 20\%$ each.

Another interesting observation is that - despite the Run-II data taking period - the fraction of CPU power used by PromptReco is relatively low compared to the massive DR campaign going on. One can also observe that the presence of RelVal (CMSSW release validation) jobs is intermittent - basically these jobs are submitted in connection to needs in the CMSSW release deployment plans. One last observation is that later in the month the CPU usage decreased - due to the fact that in early June 2016 the RunIISpring16DR80 campaign moved towards the completion of its bulk and work continued only in dealing with the tails of the campaign itself (i.e. resubmission of failed jobs, troubleshooting, etc). This decrease is partially hidden by a monitoring glitch that blinded the plot for few days, but it could be confirmed by external checks. As for the Figure 5.4, also the information in Figure 5.8 would not be available for plotting via the standard existing monitoring tools in CMS or WLCG, and it has been possible only via the CMS Metrics Service set-up discussed in this thesis.

As a partial conclusion of this part, it is worth commenting that metrics like "core hrs" tend to be more practically useful than "number of jobs" plots: the latter is frequently used in monitoring and accounting system, but as a matter of fact the complexity and load of an experiment activity lies in the complexity of the workflow, on which better observables are e.g. core hrs indeed, or memory usage, or even fraction of failed jobs due to infrastructure load, etc. On the other hand, a metric as simple as the job count could still be useful for some general overview needs, as discussed in the following section.

5.5.3 Use-case: jobs count on many computing centers

If ones wants to directly compare how much work a computing site is doing to fulfill his share of the global work load in both large, distributed Monte Carlo production for CMS and also distributed analysis over few months, monitoring the number of jobs on WLCG Tiers could be a simple but valid starting point.

First of all, at any given time each computing Tier may have a variable mix of completed, running, removed or failed jobs (both production and analysis). Integrating over wide time windows, the number of running jobs may become less relevant indeed, but the fraction of e.g. removed or failed jobs may sum up to relatively high value, comparable to completed ones. As a consequence, the first metric to consider is to just sum them all up. Figure 5.9 shows the fraction of jobs in each of the WLCG Tiers that contributed to the CMS experiment activities (both production and analysis) in the whole time frame under study, cutting at 15 sites. Starting from the top right part and moving anti-clockwise, one can see that the US Tier-1 in Fermilab was the major contributor, with about 14% of the total jobs, followed by the CERN Tier-2 center, a bunch of US Tier-2 sites, the CERN Tier-0 (that often contributes to the Monte Carlo production activities when data taking is off), the largest German Tier-2 in DESY, and so on.

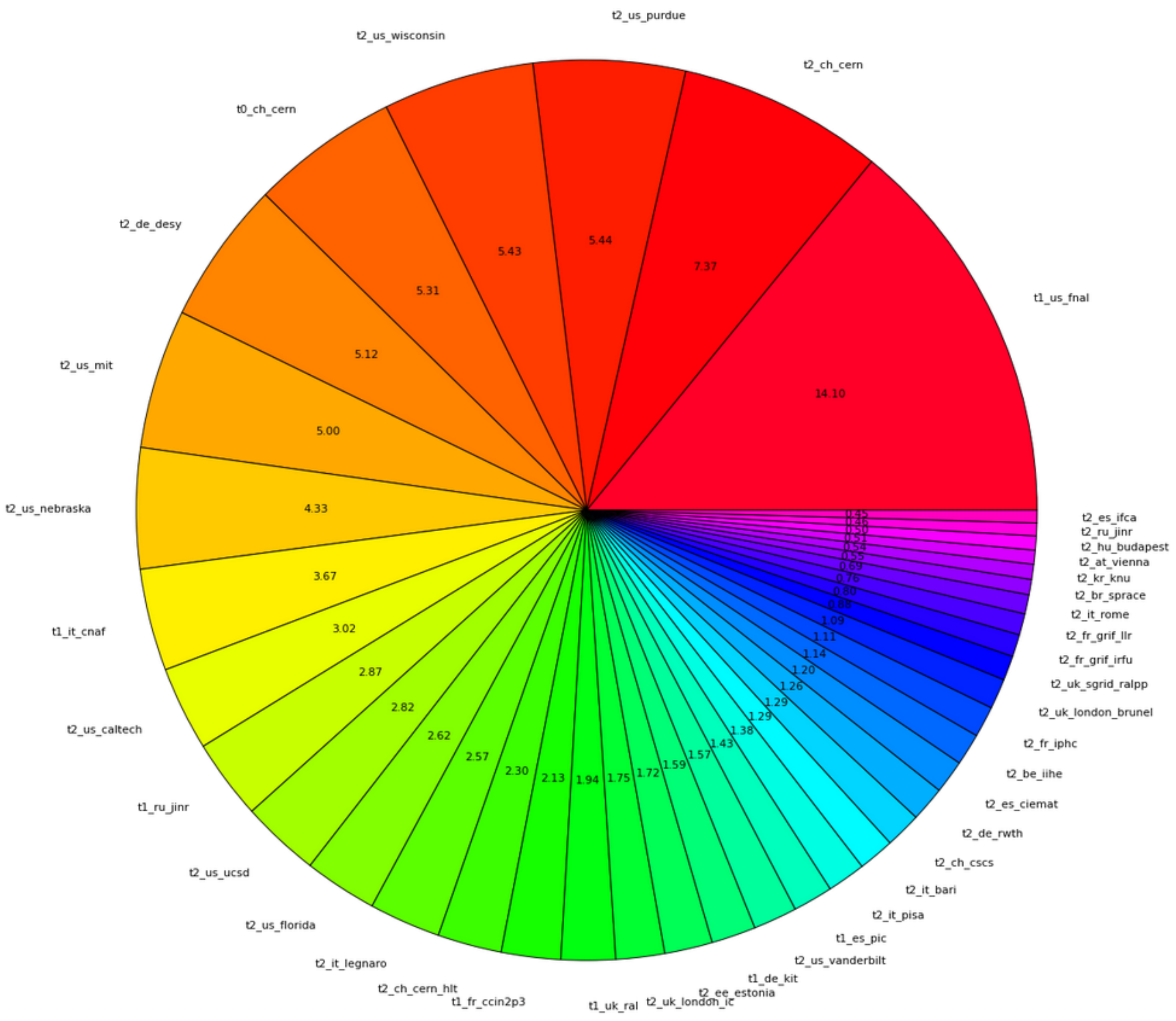


Figure 5.9: Fraction of jobs in the WLCG Sites that contributed the most to the CMS experiment activities (both production and analysis) in the time frame under study.

As different job statuses are actually mixed up in Figure 5.9, it may be misleading to use such plot to imply which are the sites contributing the most to CMS production and analysis. In order to overcome such limitation, one could consider to plot only the total number of completed jobs by site - thus excluding all other statuses - as in Figure 5.10. This plot yields a proper site ordering, in the sense discussed in the following.

The Fermilab (FNAL) contribution - in terms of completed jobs - to CMS production and analysis is about twice (or more) as much as the contribution from any other individual computing centre in CMS. This is known and explained by FNAL volume of resources and quantity and maturity of computing manpower and expertise on-site. This plot also shows that, FNAL apart, the rest of the sites do not differ of large factors: they contribute similarly, at least in terms of number of completed jobs in the same time window, but it can be observed that such contributions are not necessarily ordered according the Tier level as from the MONARC hierarchy. It is indeed visible that some Tier-2 sites (e.g. the large US Tiers) contribute more than each of the European Tier-1 sites. It is not uncommon to observe this in CMS operations: this is an evolution of the CMS computing model through LHC Run-I and Long Shutdown I (happening also for other LHC experiments), and Tier-2 sites are becoming large computing centers capable of running services and sustaining job loads comparable to those that were originally foreseen to be sustainable only by Tier-1 centers. The boundaries among Tiers is becoming more blurry over recent years, and the trend is continuing.

One question still remains open: in this mix of Tier-1 and Tier-2 that contribute similarly to CMS production and analysis, a proper ranking in terms of actual events throughput cannot be extracted from Figure 5.9: as discussed in the previous section, the total core hours would be a more appropriate metric for this purpose. The number of both successful and failed jobs, for production and analysis, for the 25 sites contributing the most, displaying only the completed jobs is shown in Figure 5.11. The resulting plot is - not unexpectedly - very similar to the previous one, but the site ranking in terms of contribution to CMS activities is now more precise and useful.

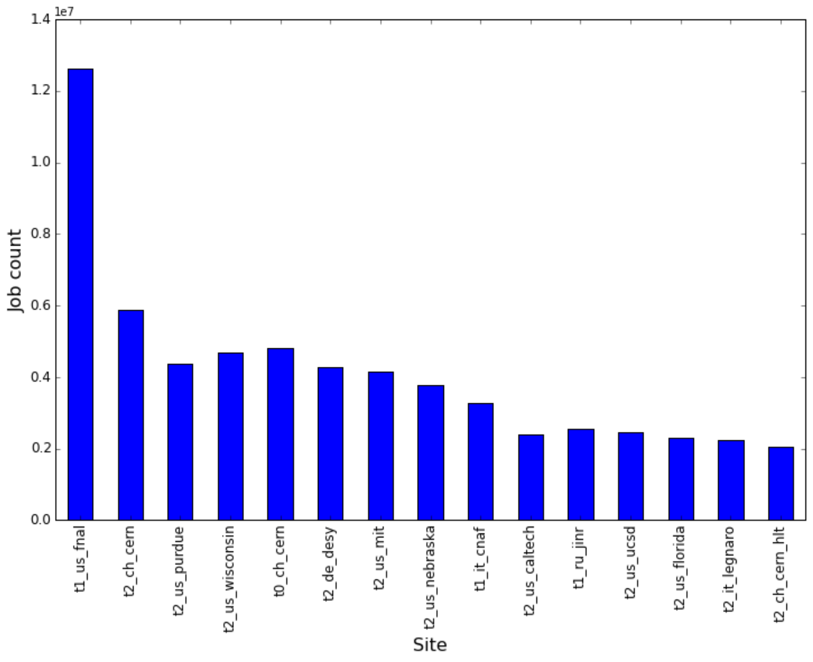


Figure 5.10: Total number of completed jobs, with breakdown on the computing sites.

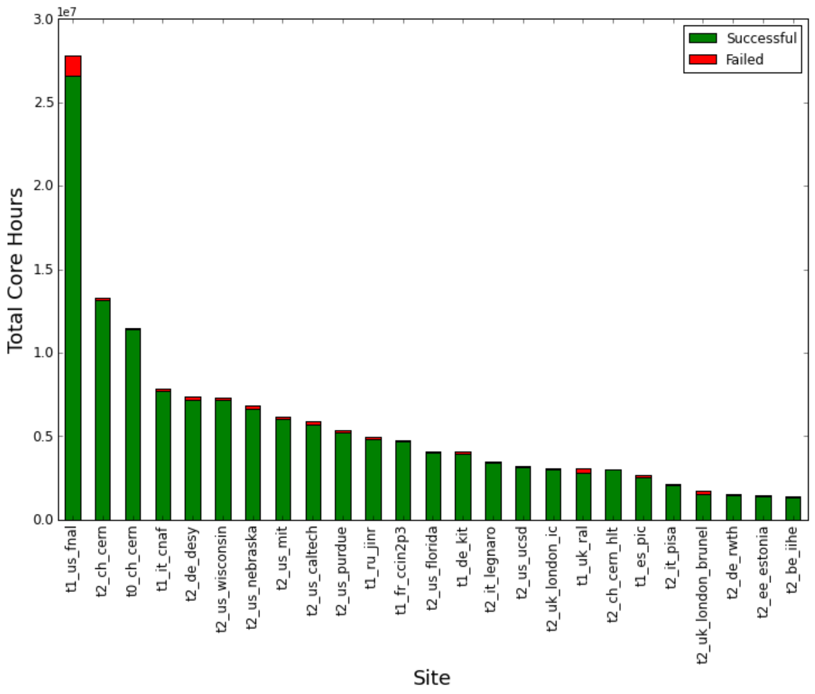


Figure 5.11: Total Core Hours for the major computing centers, showing also the fraction of the job succeeded and failed.

Additionally, the fraction of analysis versus production could be shown. Figure 5.12 shows the total number of completed jobs, with breakdown on the computing sites, together with the fraction of production versus analysis jobs on each site. All sites (Tier-1 and Tier-2) also run a fraction of analysis jobs: this fraction seems around 50% for Tier-2 centers, and much smaller for Tier-1 centers: both aspects are as foreseen by the CMS Computing model. I is interesting to note that the Tier-0 and the offline usage of the High Level Trigger farm at CERN are both contributing to production only: indeed, they are closed for CRAB3 analysis jobs submissions. This kind of plots is of interest to the CMS Computing operations team, in babysitting both the distributed analysis efforts and the production campaigns, and needing to regularly check how much each site contributed to which activity, in order to plot misbehaviors, unbalanced or operational mistakes: this information at this level of granularity is available only in the CMS Metrics Service set-up discussed in this thesis.

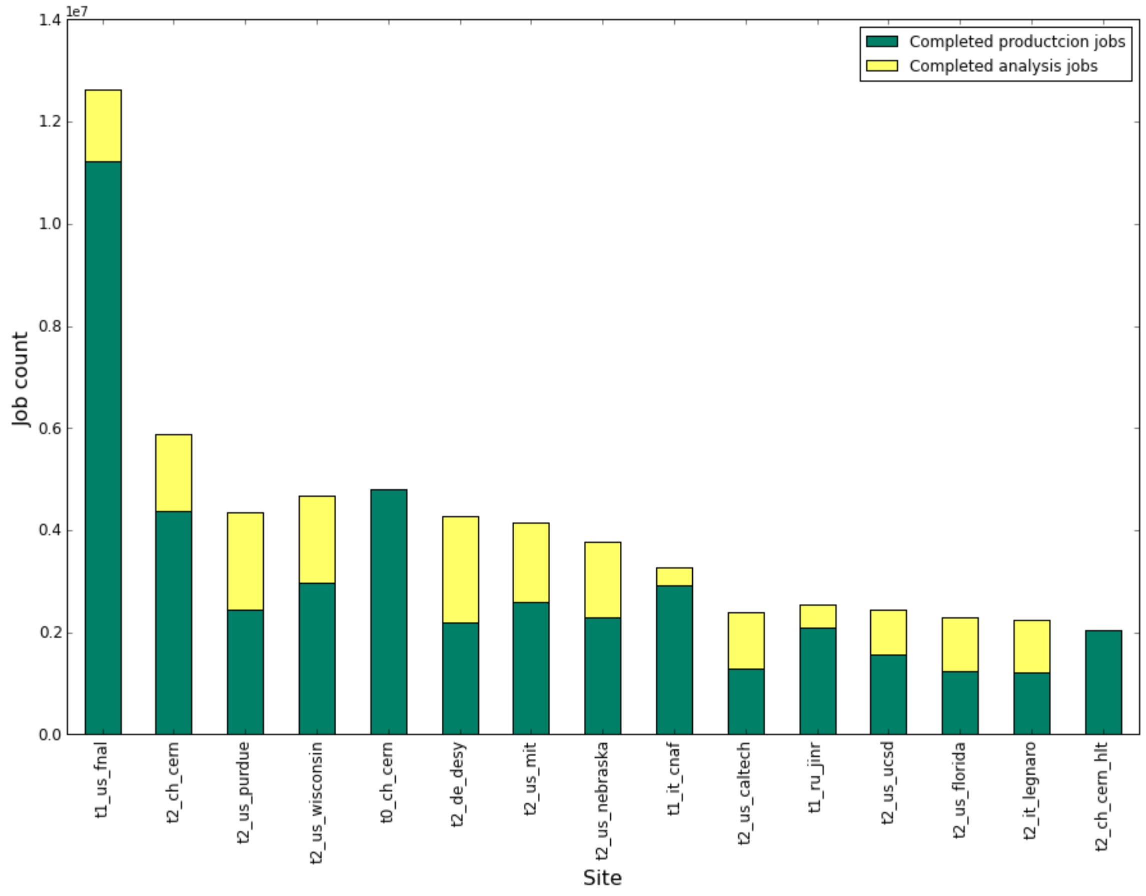


Figure 5.12: Total number of completed jobs, with breakdown on the computing sites, showing also the fraction of production versus analysis jobs.

Furthermore, a coarser granularity of the previous plot would be more interesting for a different monitoring customer, i.e. the CMS management. While the CMS Computing operations team in interested in the site breakdown for their daily work on the technical side, the CMS management may be interested in less details on the sites breakdown, e.g. for their work on the planning side. A metric as simple as the relative fraction of production versus analysis jobs at different Tier levels over the time frame under study is shown in Figure 5.13. This plot visually confirms and strengthens what was argued above, i.e.:

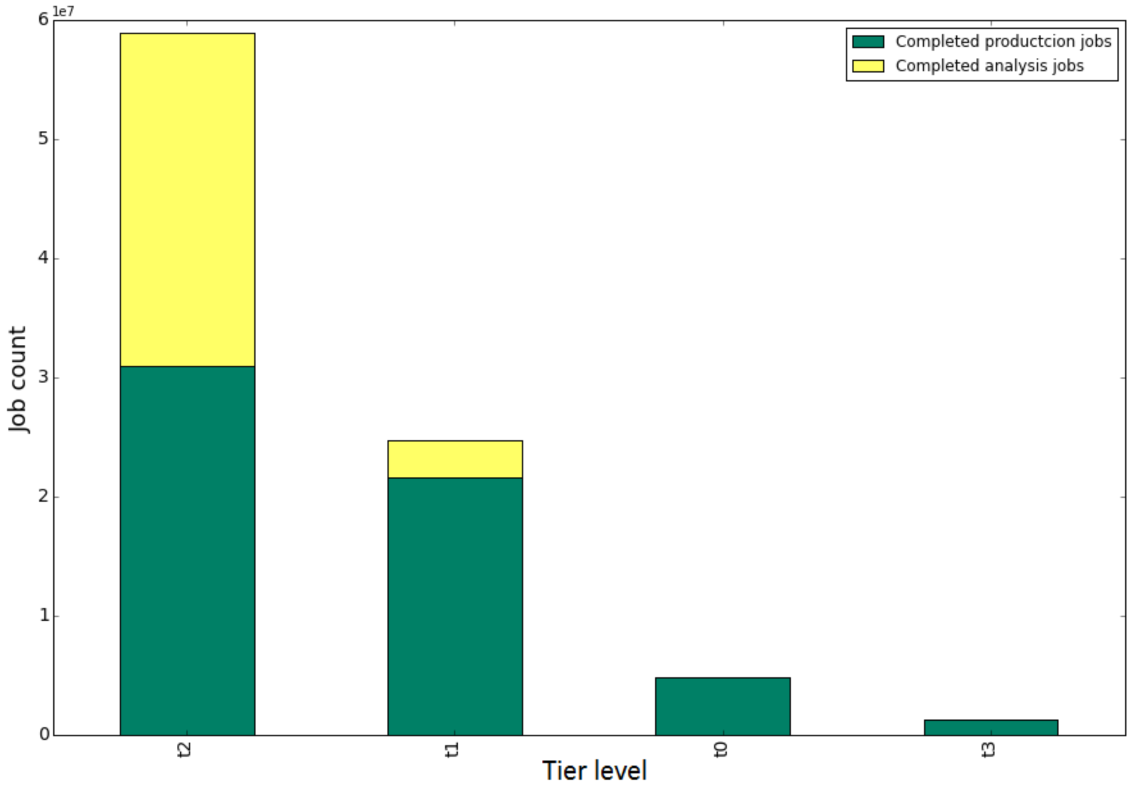


Figure 5.13: Relative fraction of production and analysis jobs at different Tier levels over the time frame under study.

- Tier-2s are the most important source of CPU power for CMS, and they process production and analysis jobs at a relative share of approximately 50% – 50%;
- Tier-1 sites host large resources than most Tier-2s but are fewer in number (and the Tier-2 sites are growing in size over the years), so the Tier-1 sites contribute less in production and definitely - as per design - much less in analysis (despite it is possible to run analysis jobs at Tier-1s);

- Tier-0 is sometimes used to expand the global production pool but at a controlled level, as its primary activity is connected to data taking and must be protected;
- The aggregate contribution from Tier-3 centers to production is smaller but still visible.

It is worth observing that Figures 5.9 to 5.13 are not uniquely obtainable via the CMS Metrics Service set-up used in this thesis, but can also be extracted via the CMS Dashboard. But it is worth reminding that in case a breakdown by campaign is needed, it can be done for all these plots in the CMS Metrics Service while it cannot in the CMS Dashboard.

The time requested to perform the queries in Figures 5.9 to 5.13 are displayed in Table 5.5.3, as provided by the Kibana internal statistics service. For a preliminary evaluation of the viability of the CMS Metrics Service for what concerns time performances. In case this prototype becomes a production service for CMS, such performances will be subjected to ad-hoc optimization work. Nevertheless as of now, those times are comparable to the ones that can be obtained from producing similar plots using the CMS Dashboard.

Task	CMS Metrics Service
Query to obtain Figure 5.9	$7 \pm 3s$
Query to obtain Figure 5.10	$3 \pm 2s$
Query to obtain Figure 5.12	$4 \pm 2s$
Query to obtain Figure 5.13	$4 \pm 2s$

Table 5.1: Approximate time required for executing queries to obtain a few selected plots presented in this thesis using the CMS Metrics Service with Jupyter Notebook.

5.5.4 Use-case: campaign-based job failure modes

The ratio of successful versus submitted jobs is an obvious success metrics for any processing task. This information is as easy to collect as potentially incomplete to plan and execute concrete actions that may mitigate the impact of job failure to the overall activity. The root causes of the job failures vary a lot, and largely depend on two classes of factors, namely the stability of the underlying infrastructure and the solidity (or lack of) of the workflows being submitted.

As stated in the previous sections, the standard monitoring tools like the CMS Dashboard do not allow investigations of any metric at the campaign level: the failure modes per campaign may actually be quite instructive, and are being quickly explored in this section using the CMS Metrics Service. Focus is on the same Monte Carlo production campaigns discussed previously in some details, i.e. RunIISpring16DR80 and RunIISummer15wmLHEGS [1].

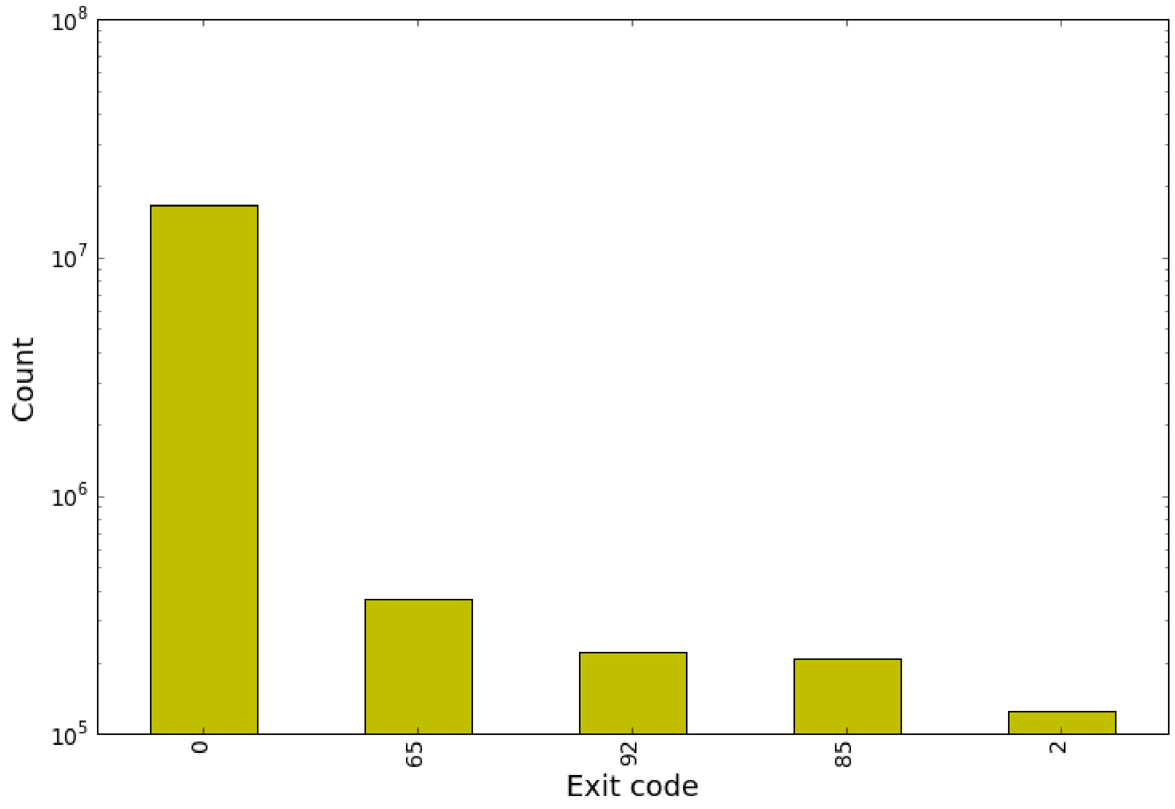


Figure 5.14: The five most frequent exit codes of CMSSW jobs belonging to the RunIISpring16DR80 campaign, for jobs that ran on-site (note the log scale).

For the RunIISpring16DR80 campaign, so far 86.3% of the CMS jobs ran on-site, while 13.7% ran off-site: this means that the majority of jobs were submitted on sites where the input (GS) data where resident and (potentially) accessible, while the rest had to access the file remotely via WAN access. Figure 5.14 shows the 5 most frequent exit codes of CMSSW jobs belonging to the RunIISpring16DR80 campaign, for jobs that ran on-site, i.e. the majority of them. In general, the vast majority of jobs are successful, i.e. exit code 0 (note also the logarithmic scale), only 7% of the jobs fail due to various reasons. Among the main failure modes, the dominant ones are exit code 65 (end of job from user application - CMSSW), exit code 85 and exit code 92 (both related to a job that failed to open local and fallback files) [78].

[1]RunIISummer15wmLHEGS is similar to RunIISummer15GS - described in Section 5.5.1 - with the respect of the metrics analyzed in this Section.

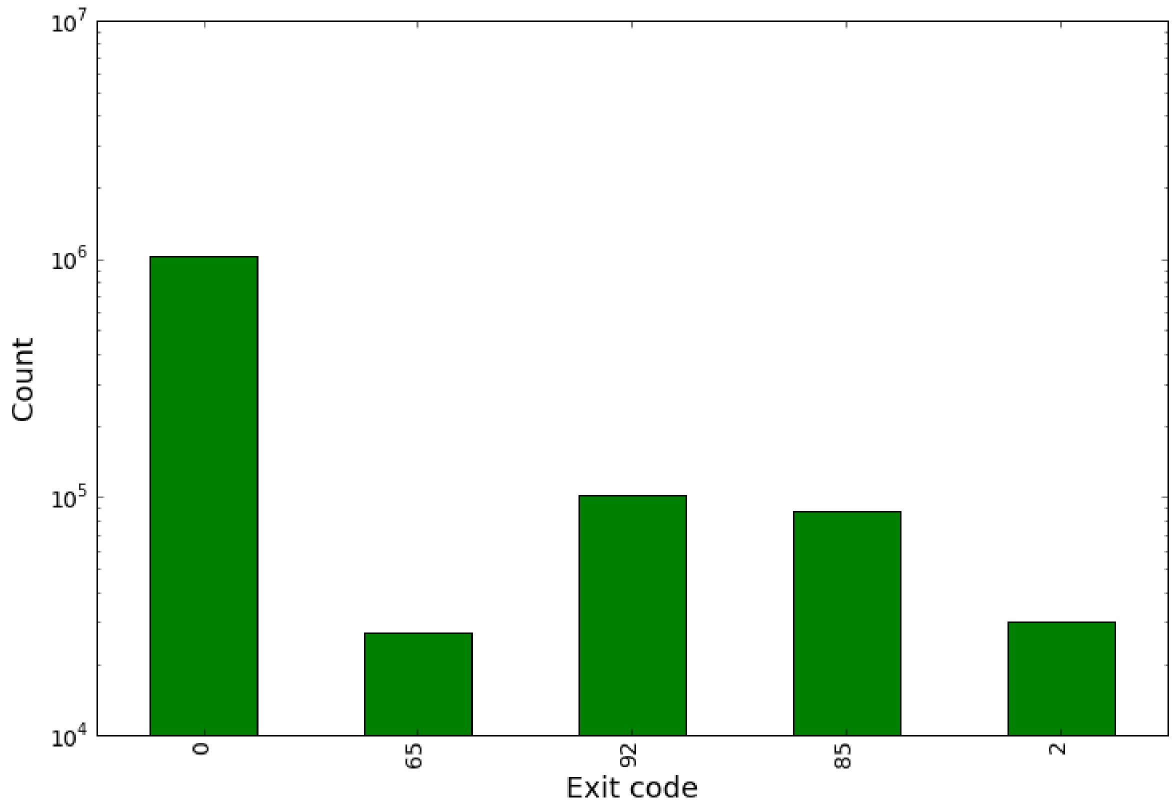


Figure 5.15: The five most frequent exit codes of CMSSW jobs belonging to the RunIISpring16DR80 campaign, for jobs that ran off-site (note the log scale).

Figure 5.15 shows instead the 5 most frequent exit codes of CMSSW jobs belonging to the RunIISpring16DR80 campaign - i.e. same as Figure 5.14 - but for jobs that instead ran off-site. As before, most jobs are successful, and the fraction of failed jobs increases to 19%, thus higher than jobs that ran locally. The main failure modes for off-site jobs are reported to be exit code 85, exit code 92 and exit code 65 i.e. same as above, but with different relevance: a bigger fraction of exit codes 92 and 85 is observed, indicating that jobs that are forced to access input data from remote location are more prone to data access errors.

It is instructive to compare what is observed for a DR campaign in terms of failure modes frequencies with what is observed with a GS campaign instead.

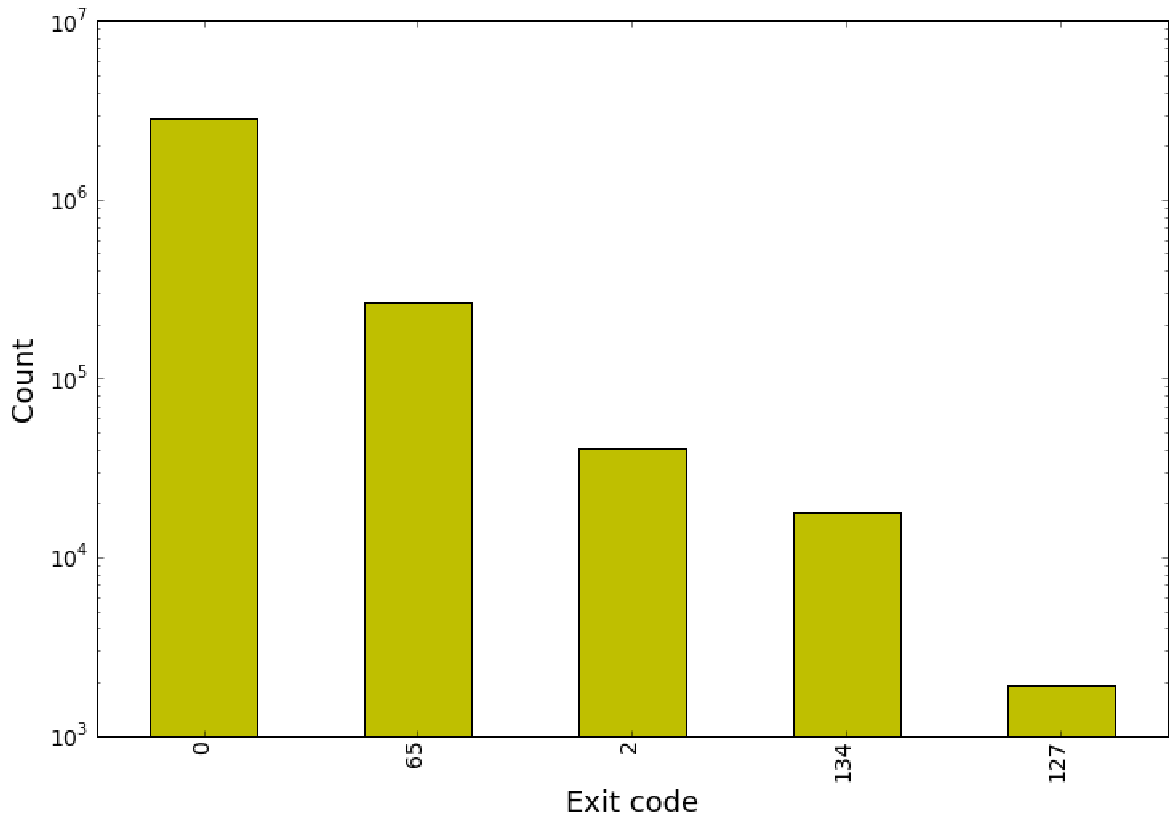


Figure 5.16: The five most frequent exit codes of CMSSW jobs belonging to the RunIISummer15wmLHEGS campaign, for jobs that ran on-site (note the log scale).

In Figure 5.16 the 5 most frequent exit codes of CMSSW jobs belonging to the RunIISummer15wmLHEGS campaign, for jobs that ran on-site, are shown. We focus here only on on-site jobs as the GS workload is much lighter - i.e. reading small files in input (if any) but no PU files, and writing out GEN-SIM files - thus showing no major complexity in terms of PU serving and consequently no need to off-load such input loading to WAN access techniques. Apart from the vast majority of successful jobs, 11% are failed jobs, with exit code 65 (end of job from user application (CMSSW)), exit code 2 (interrupt (ANSI)) and exit code 134 (IOT trap (4.2 BSD)). Regardless of the details of the failures, it is evident that the data access (local or remote) is less dominant as a root cause of errors wth respect to jobs belonging to a DR campaign: this is explained by the fact that a DR campaign has indeed a higher IO load profile with respect to a GS campaign.

It has been showed through a couple of examples that the CMS Metrics Service is able to offer failure modes information per campaign. The characteristics of each campaign are well known since the preparation phase, before the first job submission occurs. The load of a campaign (e.g. in terms of I/O) is known, and it represents its fingerprint: one can expect specific failure modes and - on a relatively stable infrastructure - one is also able to predict their frequency of occurrence: by being able to stably monitor the rates of such failure modes per campaign (and not only in general for all jobs) one can actually closely monitor if such rates are increasing with respect to expected values during the evolution of a campaign, and thus promptly spot problems in the computing system as a whole (e.g. network problems, storage problems, site problems, etc) and take mitigation actions or use this information to potentially build a more adaptive submission system. Of course, all this would be impossible without such monitoring information which are available only in the CMS Metrics Service prototype.

5.5.5 Use-case: campaign-based CPU efficiency

Another very relevant information to look at in order to properly monitor the way computing resources are utilized in massive processing activities is the CPU efficiency (defined in section 4.2). Ideally, one would like to have a very high CPU efficiency regardless of the kind of job that is being submitted: this is obviously unrealistic. Several factor may contribute to decrease the actual measured CPU efficiency of an experiment workflow, e.g. high I/O production workflows, peculiarities in the job scheduling process by the pilot resource provisioning system, remote data reads on the WAN (Wide Area Network) through storage federations, bottlenecks in a site data serving system, etc.

A first interesting check is to use a general monitoring system like the CMS Dashboard to check the CPU efficiency of various CMS workflows over e.g. last 12 months, in order to get a feeling of major offenders. This is shown in Figure 5.17. Excluding jobs of categories like "test" or "hctest" - as well as the "unknown" ones which are known issues in associating jobs to precise job types (ans assuming ignoring them would not impact much the general considerations), it is interesting to observe that *i)* production jobs have among the highest CPU efficiency; *ii)* analysis jobs has a lower CPU efficiency with respect to production jobs (well known due to the variability of non optimizations that may occur in user-driven processing with respect to centralized expert processing), and *iii)* the most inefficient (despite massive in terms of core hours needed) workflow is the digitization - labeled as "reprocessing" (digi-reco) in the plot. This is related to the PU reading, i.e. plenty of WCT spent in reading input data instead of spending it as CPT, thus contributing to a low CPU efficiency: this needs to be carefully monitored, and in the short term CMS plans to mitigate it via pre-mixing solutions (not yet in production).

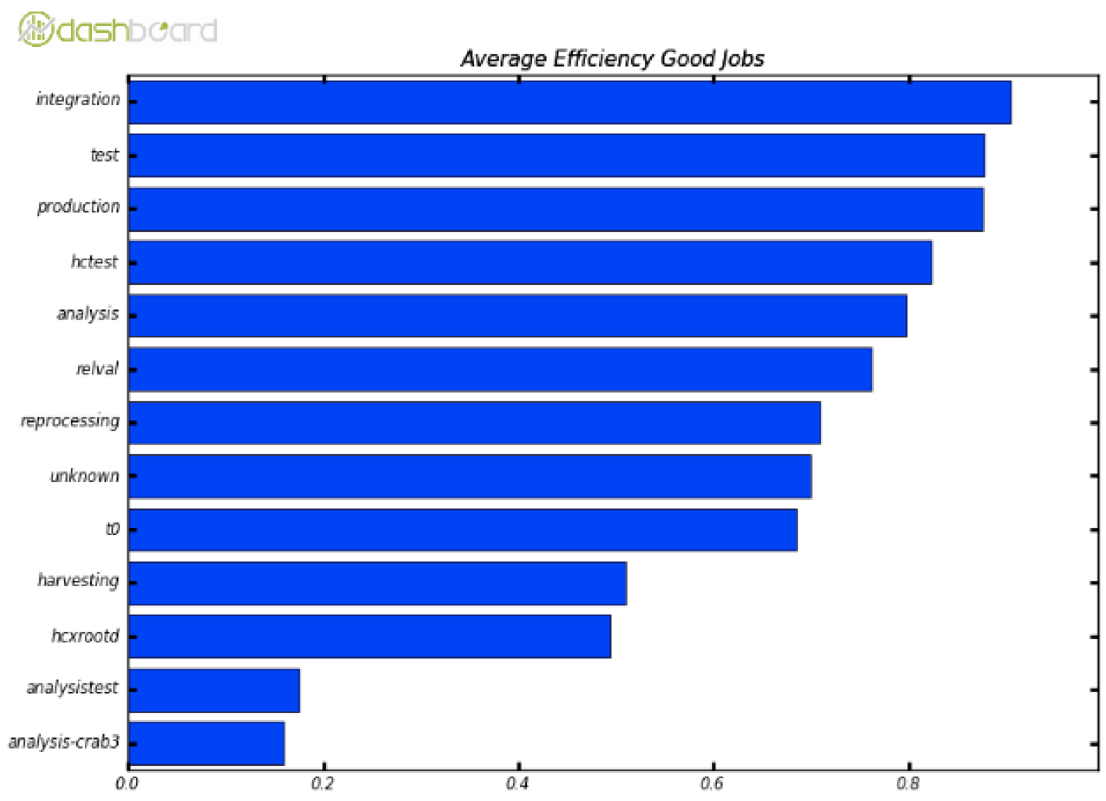


Figure 5.17: Average CPU efficiency of various CMS workflows over the last 12 months (source: CMS Dashboard).

Also, concerning the possible other causes of low CPU efficiency, an interesting one is the remote data reads on the WAN. The average CPU efficiency of jobs reading input data locally/remotely over 12 months - regardless of the job type - has been measured to be about 82%/76% respectively: so, it seems that the negative impact on CPU efficiency of remote data access is not large, only of the order of 5% compared to jobs reading data locally. Despite the variability across sites, one of the smoking gun on this topic seems indeed to be the high-I/O profile of DR workflows due to PU-reading.

It is interesting to evaluate over a relatively long time period (one month may suffice), the difference in CPU efficiency among production jobs and analysis jobs: over last months (approximately June 2016), production jobs used about 53M core hours and yielded a CPU efficiency of 75%, whereas analysis jobs used about 29M core hours and yielded a CPU efficiency of 69%. The figures are relatively close, but analysis is indeed more prone to low CPU efficiencies.

Restricting to production jobs only, in view of the discussions done in previous sections, it is of interest to explore how CPU efficiency looks like per Monte Carlo campaign.

Starting from RunIISpring16DR80, the total clock hours spent is more than 20 million and it is shared among different task types as shown in Figure 5.18, with the digi workflow clearly dominant. The CPU efficiency for the RunIISpring16DR80 campaign, in intervals of 10% and shown in terms of wall clock hours (and not job count) is shown in Figure 5.19. It can be observed that a good fraction of the jobs have high CPU efficiency, but the distribution definitely has a long tail at all values of lower CPU efficiencies. This is indeed related to the heavy workload under study: every job in this DR campaign requires reading PU information from local or remote storage, thus causing very wide variations of CPU time (CPT) and thus of CPU efficiency.

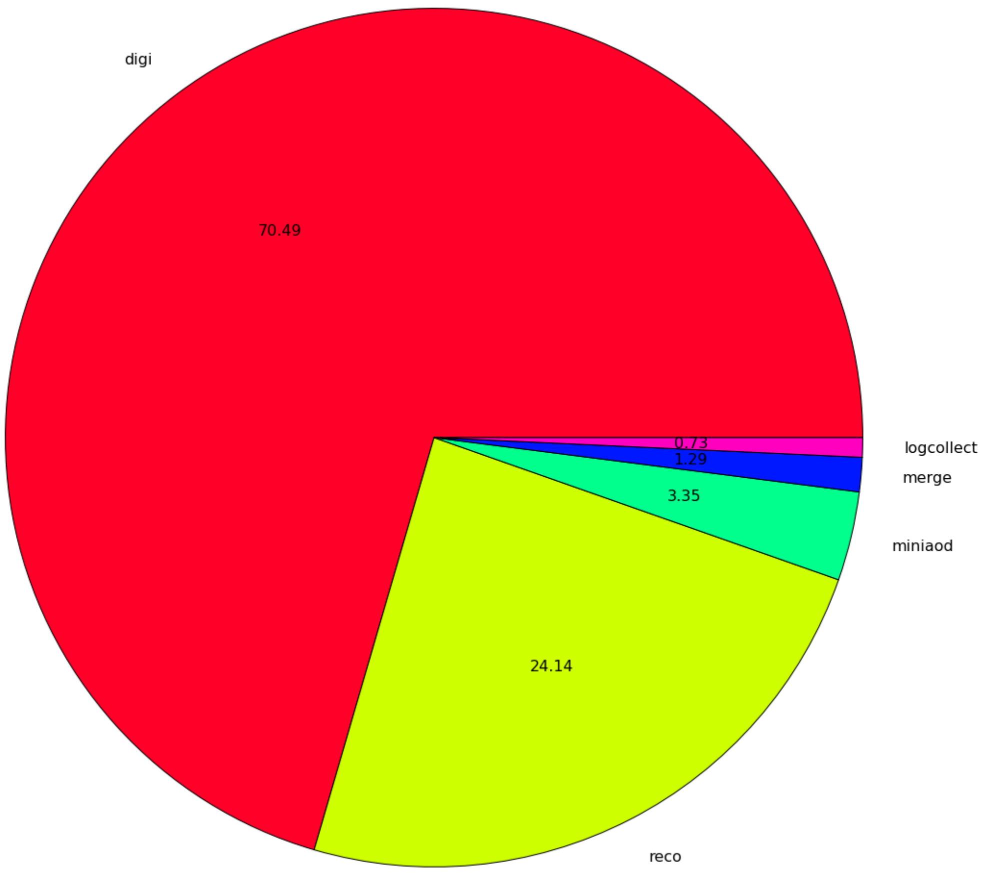


Figure 5.18: Fraction of Wall Clock Hours for the different Task Type for the RunIISpring16DR80.

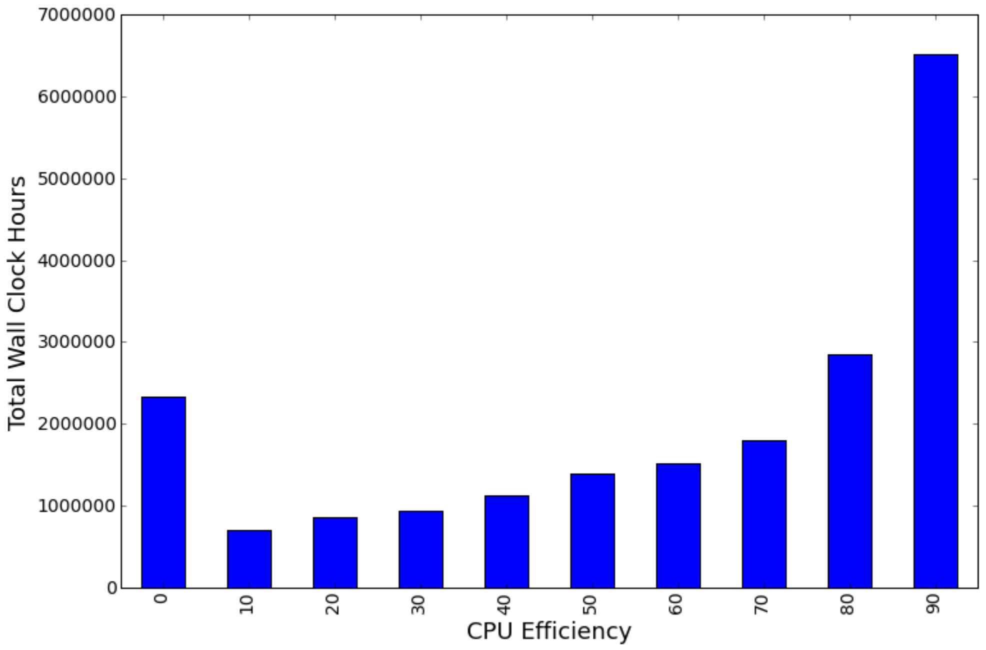


Figure 5.19: CPU efficiency in intervals of 10% shown in terms of wall clock hours for the RunIISpring16DR80 campaign.

The CPU efficiency, in intervals of 10%, in terms of wall clock hours for the RunIISummer15wmLHEGS campaign insetad is shown in Figure 5.20. With respect to the RunIISpring16DR80 campaign, the CPU efficiency in this case is much higher, with the majority of jobs showing values as high as $> 80 - 90\%$ and definitely less low CPU efficiency jobs (note that the large quantity of jobs at null CPU efficiency are jobs failing - or being killed for some reason - soon after being submitted). This behavior is as expected: it originates from the very different nature of the input data needed for processing jobs of the two campaigns, and can now be measured properly measured by campaign.

This information at the campaign level is extremely relevant for CMS planning purposes, when new campaign start to be submitted and new CPU efficiency profiles start to be seen. The load of CMS processing in a specific campaign - if monitored - allows to choose the most appropriate computing resources to target for such campaign: this information is being used to e.g. include more IO-performing sites into the resources pool for a given campaign and to exclude less IO-performing sites. As there is a wide plethora of Monte Carlo campaigns on which a site can contribute, such tuning allows CMS to achieve the highest production throughput while permitting all sites to contribute at their best, based on their resources and IO serving capacity.

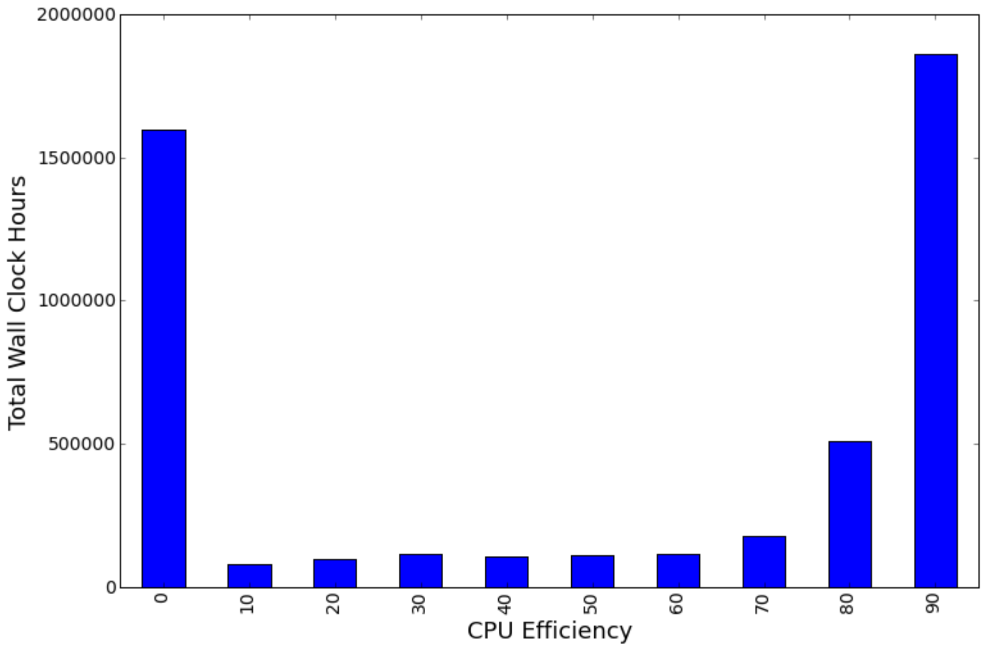
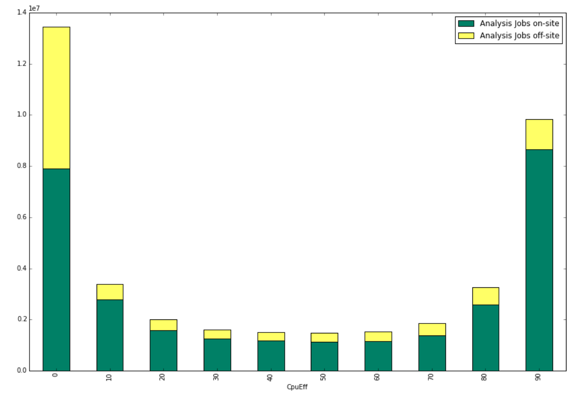
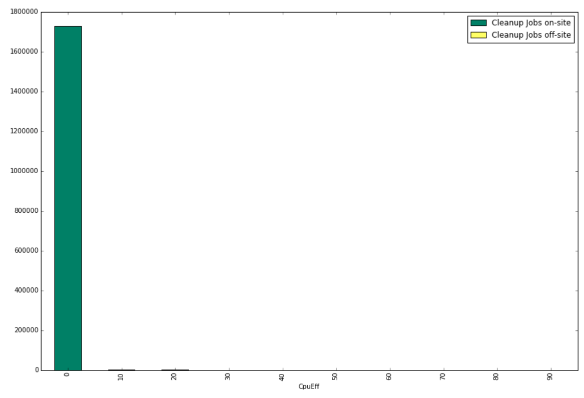


Figure 5.20: CPU efficiency in intervals of 10% shown in terms of wall clock hours for the RunIISummer15wmLHEGS campaign.

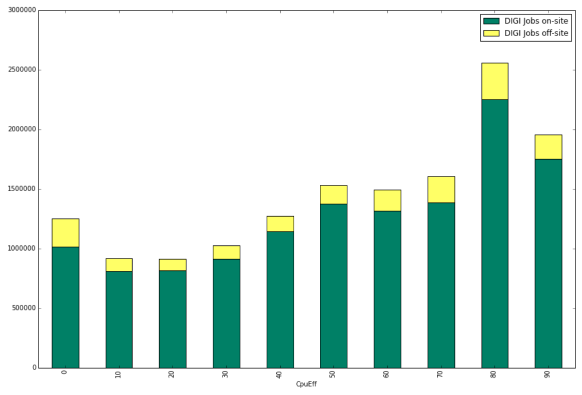
As a final comment on CPU efficiency, its value for different job types can be explored for all the jobs collected in the time frame under study (see Figure 5.21). RECO jobs are among the most efficient, while Analysis and Cleanup jobs are quite inefficient. Furthermore, it is quite remarkable to notice that the CPU efficiency of a particular task is independent from whether the jobs access the data on a desired site or they had to look for the data elsewhere.



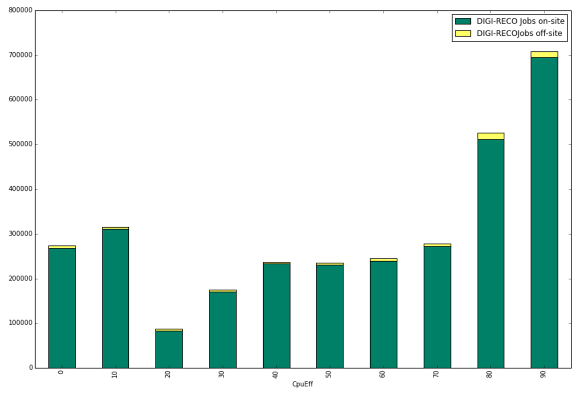
(a) Analysis



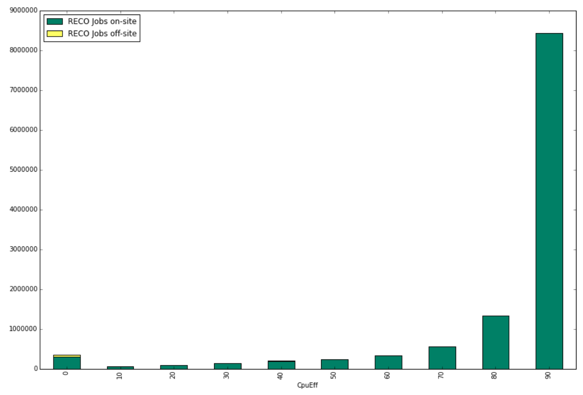
(b) Cleanup



(c) DIGI



(d) DIGI-RECO



(e) RECO

Figure 5.21: CPU efficiency in intervals of 10% shown in terms of wall clock hours. It is also displayed whether the jobs where access on the desired DataLocations (on-site: green) or not (off-site: yellow).

5.5.6 Performance comparison of two Tier-2: CERN HLT and IT Legnaro

The CERN HLT is a Cloud infrastructure which is very different from the other Tier-2. When the LHC is running, the HLT computational power is used as a first offline trigger, whereas, when the LHC is not running, the HLT is used for Monte Carlo production (see Section 2.3.6). From Figure 5.9, it seems reasonable to compare the CERN HLT to the Italian Tier-2 at Legnaro because the two infrastructures have handled roughly the same amount of jobs in the time frame under study. However, if we analyze how the Total Core Hours varies each day, we see some remarkable differences (Figure 5.22). In fact, the HLT infrastructure had several periods of time in which it did not work, even though it had more CPUs than Legnaro. The overall result is that it handled about the same amount of jobs. Similarly, the measurement of the CPU efficiency in the same period of time confirms this hypothesis (Figure 5.23). The complete Elasticsearch query and analysis is reported in Appendix B.

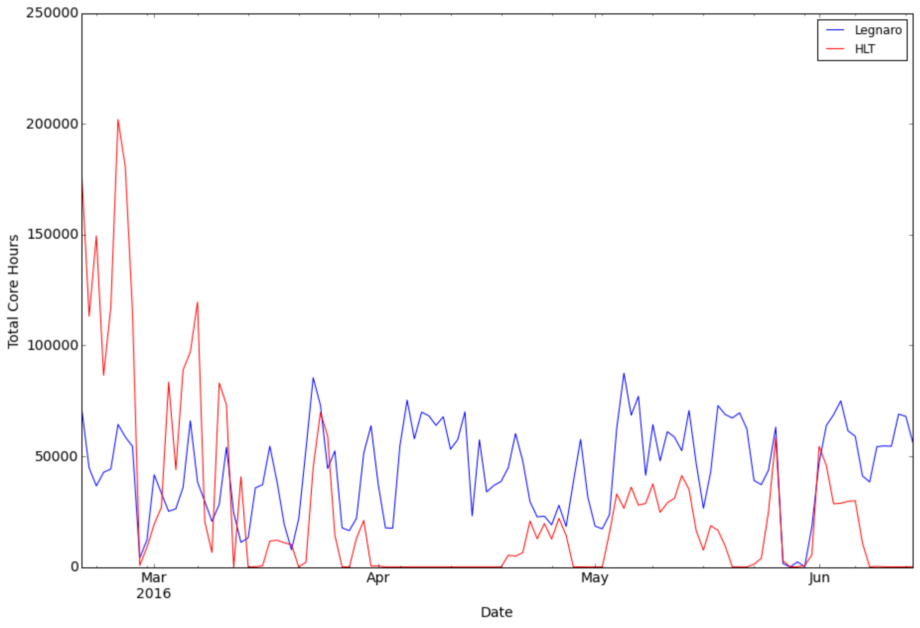


Figure 5.22: Total Core Hours as function of time for the Tier-2 CERN HLT and Tier-2 It Legnaro in the time frame under study.

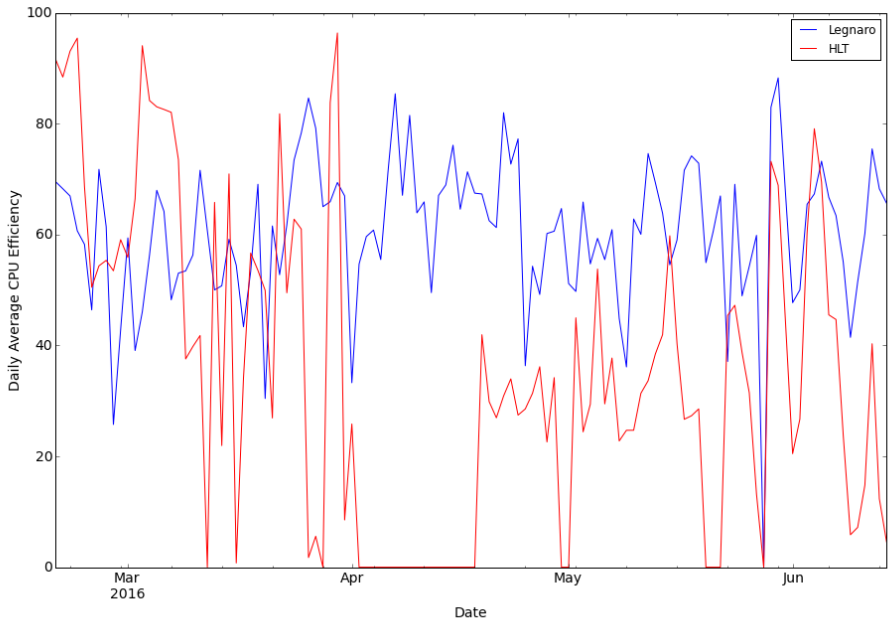


Figure 5.23: Average daily CPU efficiency for the Tier-2 CERN HLT and Tier-2 It Legnaro in the time frame under study.

5.6 Data Management view

In the previous section we focused on dealing with information on Workload Management in CMS. The performance analysis presented in this thesis is in fact complete only when data from the WM sector are indeed completed with data from the DM sector: despite several aspects lie across the two areas, the investigations are presented separately and in this section we will focus on Data Management aspects specifically. Before starting, it must be underlined that a complete coverage of all DM aspects (e.g. all details of data transfers, etc) would be much beyond the scope of this work. Still, some insights on information on volume and diversity of data accessed in input to the jobs and data written in output from the jobs is worth to be considered. We will focus on this in the following.

As stated earlier, the CMS Monte Carlo campaigns have (among others) a distinctive signature in the amount of I/O operations necessary to the workflow completion. So, first metrics to look at are just the quantity of input data and the quantity of output data (i.e. the data read from and written to Grid sites storage during the job execution). The amount of data read by the CMSSW process (expressed in GB) will be referred to as InputGB in the following, while the amount of data written by the CMSSW process to some Grid storage (expressed in GB) will be referred to as OutputGB (see Appendix A for more details).

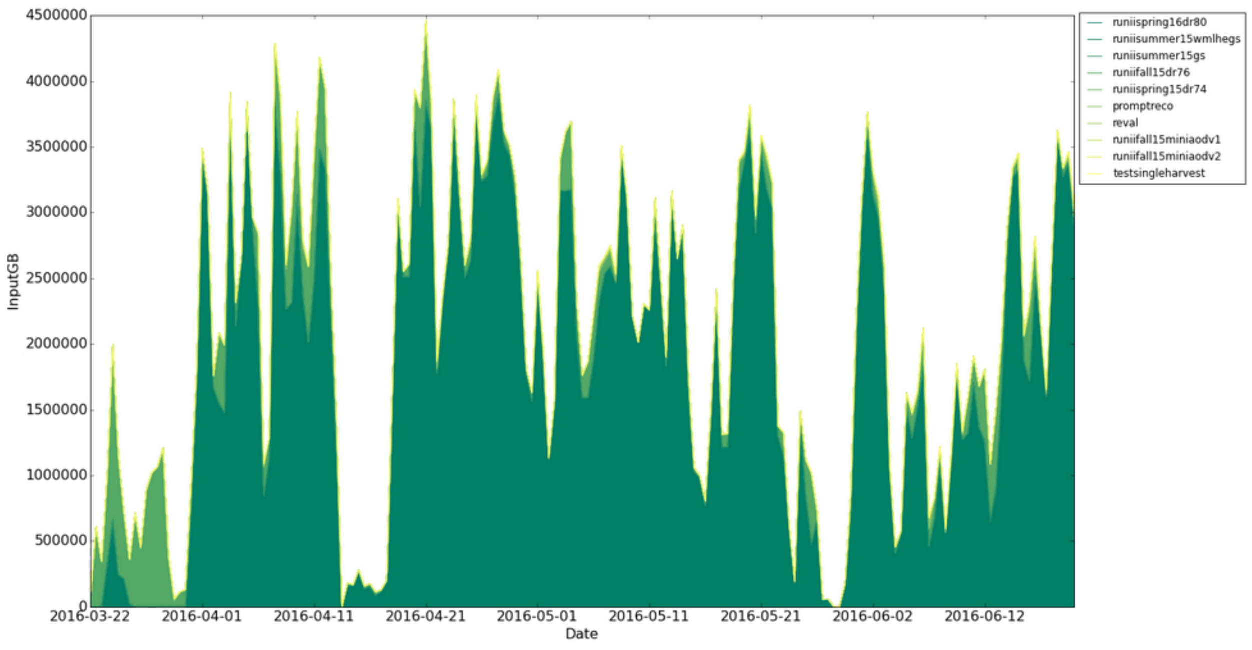


Figure 5.24: Total data volume in input to CMSSW jobs for the major production campaigns over last 3 months.

In Figure 5.24, the data in input to CMSSW application (i.e. InputGB) is shown, with breakdown per campaign, in its evolution over time over the last 3 months. It can be observed that the RunIISpring16DR80 campaign is so dominant over other secondary campaigns that the latter ones are hardly visible [2]. The input volumes to the DR campaign is as high as hundreds of GB per day for extended times. As a cumulative value, this ends up being quite impressively high, and it can interestingly be explored grouped by country, as done in Figure 5.25 (note the log scale). It can be inferred that all the most relevant regions contributing to Computing in CMS (US, IT, DE, UK, CH, RU, SP, FR) hosted over last 90 days CMSSW applications that ingested in total more than 10^7 GB, i.e. 10 PB of data, each - peaking in the US region where the data ingestion volume is even greater than 10^8 GB, i.e. 100 PB of data. As we know that the dominating activity is the DR campaign, this impressive number reflects an extraordinary load on the overall infrastructure caused by RunIISpring16DR80 itself (mostly), and a huge data serving effort shared by all CMS Tier levels. This can be seen also in Fig 5.26: aggregating over Tier-1 and Tier-2 levels - i.e. regardless of the regional domain - both levels contribute by ingesting more than 10^8 GB: this offers a metric to re-state that Tier-1 and Tier-2 resources approximately contribute in similar manner to the overall CMS processing effort: indeed, Tier-2 sites have more CPU power than Tier-1 but they also perform more analysis and only about 50% of them contribute to DR efforts, while the Tier-1 sites have less CPUs but they all contribute to DR, so the two may compensate, with potentially Tier-2 digesting more jobs (i.e. more analysis as well as more other production which have less IO load). This confirms the direction the computing model is evolving towards, i.e. decrease the differences among Tiers of different levels, and this is an important information to be monitored regularly, in order to make sure that the resources exploitation is appropriate to efficiently serve the experiment needs.

The total data volume in output to storage from CMSSW jobs, instead, with breakdown on the countries, is shown in Figure 5.27. It can be seen that the shape of the distribution is roughly the same as InputGB, but it is scaled down by an order of magnitude, as expected from the workflows characteristics (i.e. we know that data volume in input is larger than data volume in output by that approximate amount, due to pile-up serving). The breakdown on Tier levels for OutputGB is shown in Figure 5.28. Again, the ratio among the two is similar, as discussed above.

The results indicate the need to review the CMS mechanism for pile-up serving to jobs, in view of alternative solutions which are less heavy on computing resources.

[2]The same plot in logarithmic scale could be showed, but the focus of the observation in the next is not to disentangle the different minor contributions in their relative shares, but instead to focus on the dominance of the DR campaign in data input

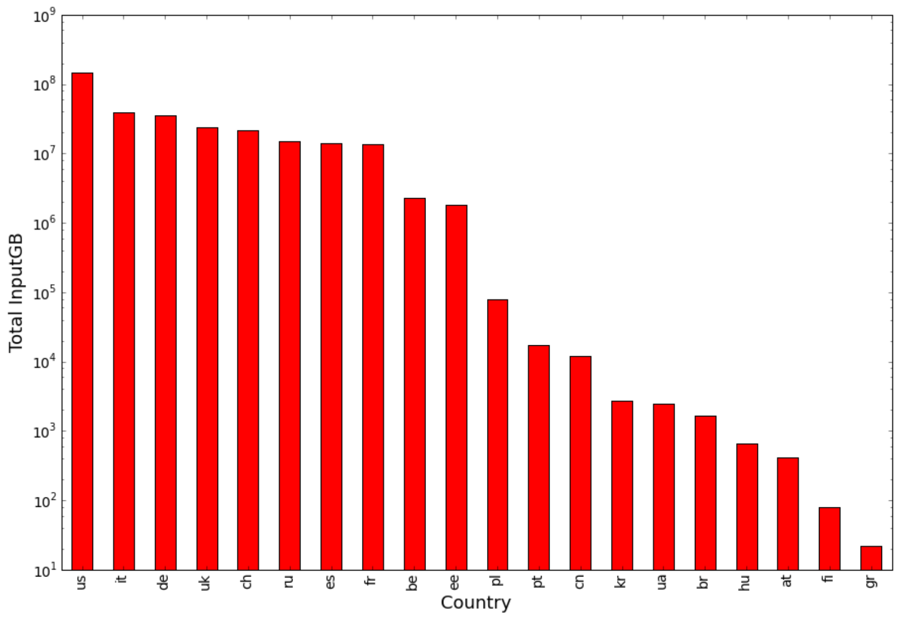


Figure 5.25: Total data volume in input to CMSSW jobs, with breakdown on the countries (note the log scale).

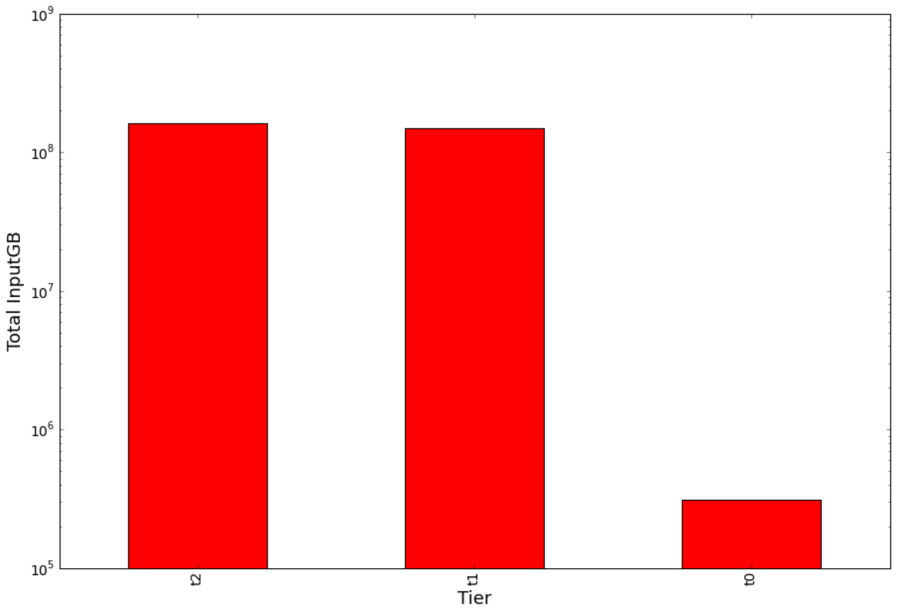


Figure 5.26: Total data volume in input to CMSSW jobs, with breakdown on the Tier levels (note the log scale).

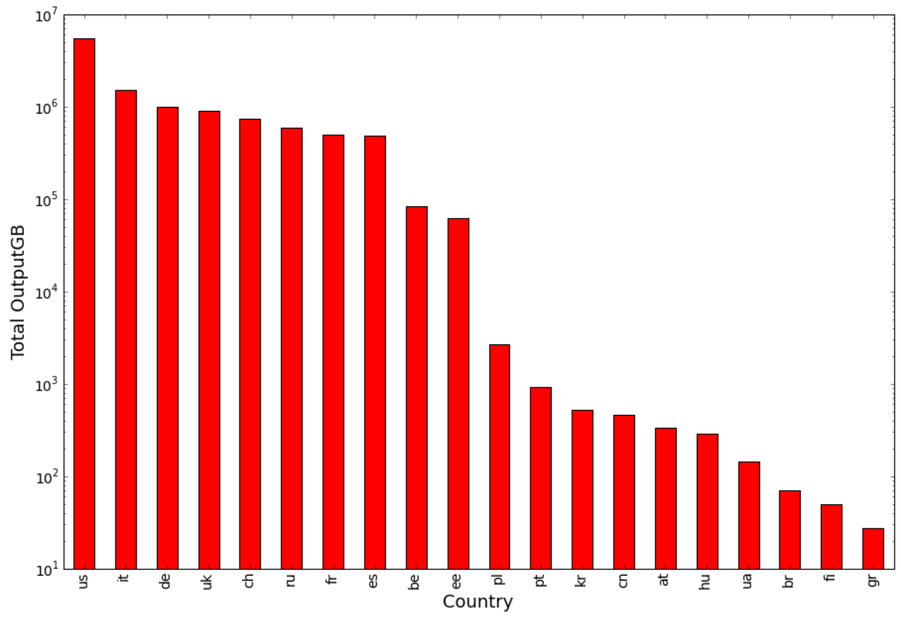


Figure 5.27: Total data volume in output to storage from CMSSW jobs, with breakdown on the countries (note the log scale).

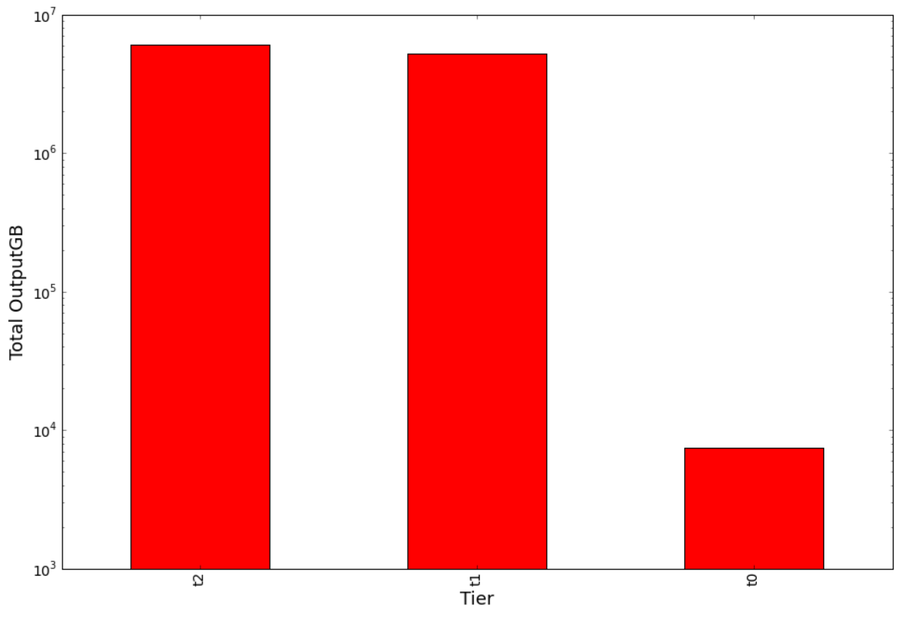


Figure 5.28: Total data volume in output to storage from CMSSW jobs, with breakdown on the Tiers level (note the log scale).

As discussed, OutputGB and InputGB should be correlated, and this is (only visually) checked in 5.29. It can be seen that all sites behave as expected, apart from T1_DE_KIT. The German Tier-1 center seems to have a larger fraction of input data read in over output data written out: as the workflows are the same as other Tier-1 centers, this plot may indicate to the CMS Computing operations team that the set-up at KIT needs to be verified, to e.g. check for possible "lazy download" remnant implementations from the past (when all input data files - and not only the fraction needed in input - were loaded locally on the worker nodes). This is being investigated, and represent an example of how useful this kind of data gathering and immediate visualization may be to avoid unnecessarily wasting computing resources (either CPU or storage).

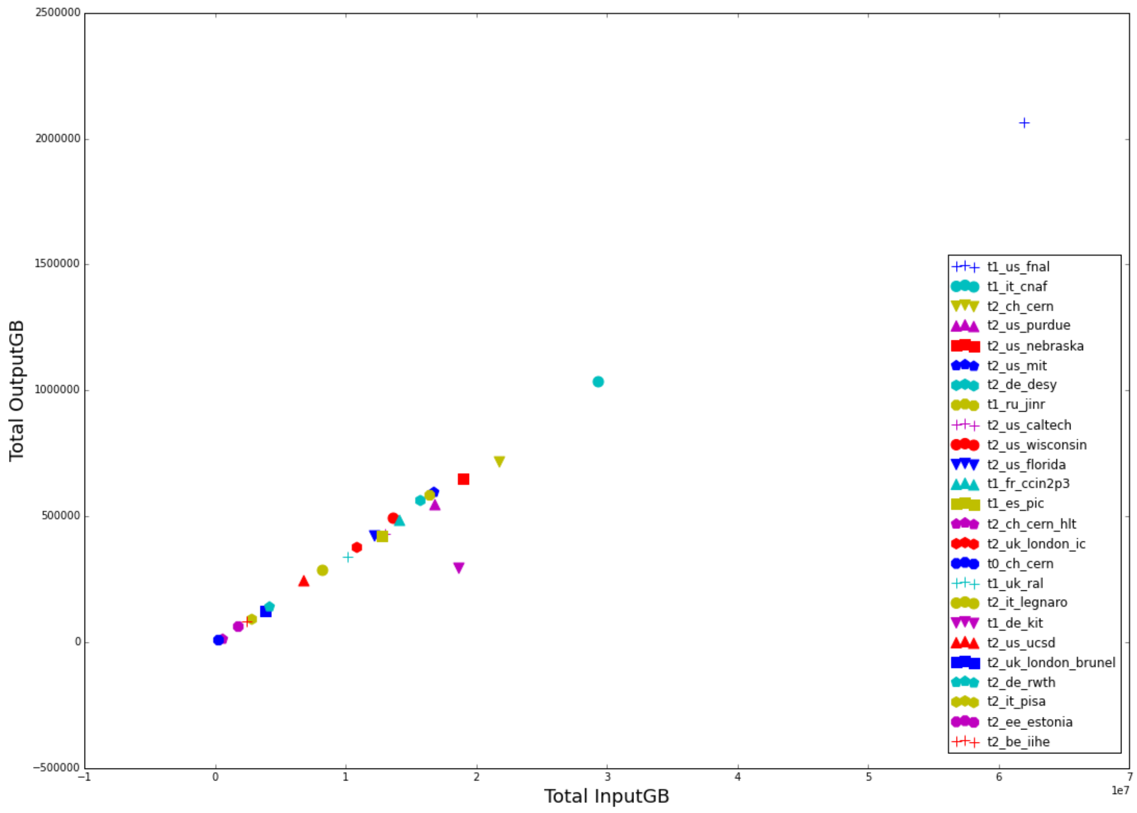


Figure 5.29: Total OutputGB as a function of the Total InputGB for different computing sites.

5.7 Quasi real-time monitoring

The presence of CMS-specific details into the monitoring data is one of the key points of CMS Metrics Service, thanks to the expressiveness of HTCondor ClassAdds. This was exploited in the previous section to demonstrate how the CMS Metrics Service can be useful to monitor, investigate and understand in details various observables over large time windows. An additional asset may come from the fact that this data are also recorded at a very fine time granularity, so also daily or hourly plots can be produced.

Figures 5.30 and 5.31 are a couple of examples of this: critical observables like CPU efficiency or total core hours can be plotted with hourly granularity, and can help to spot misbehaviors in a quasi real-time manner.

Potentially, CMS Metrics Service may hence be considered also as a precious tool for operational choices based on quasi real-time monitoring, i.e. could be considered to be used by CMS Computing shifters to monitor sites performances and immediately (in principle, even automatically) notify site admins with tickets in case misbehaviors are observed.

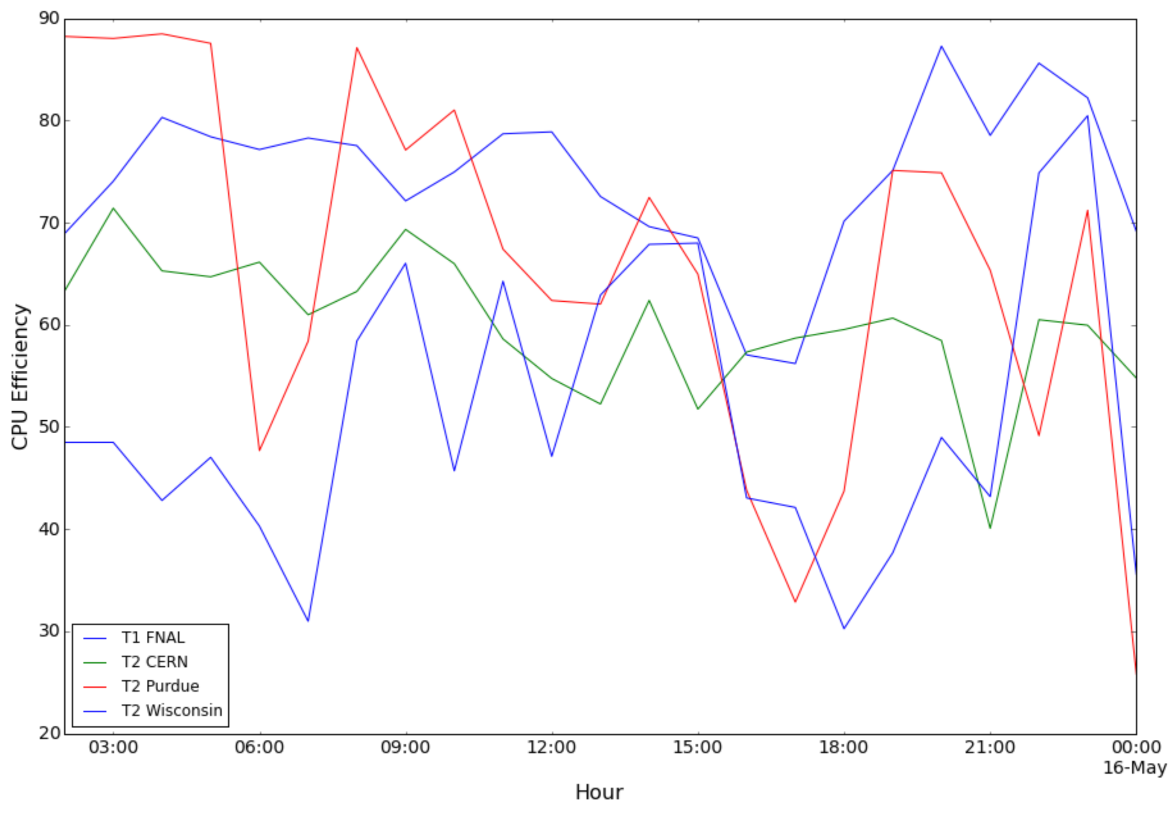


Figure 5.30: Average CPU efficiency of different Tier resources over just few hours on a random day.

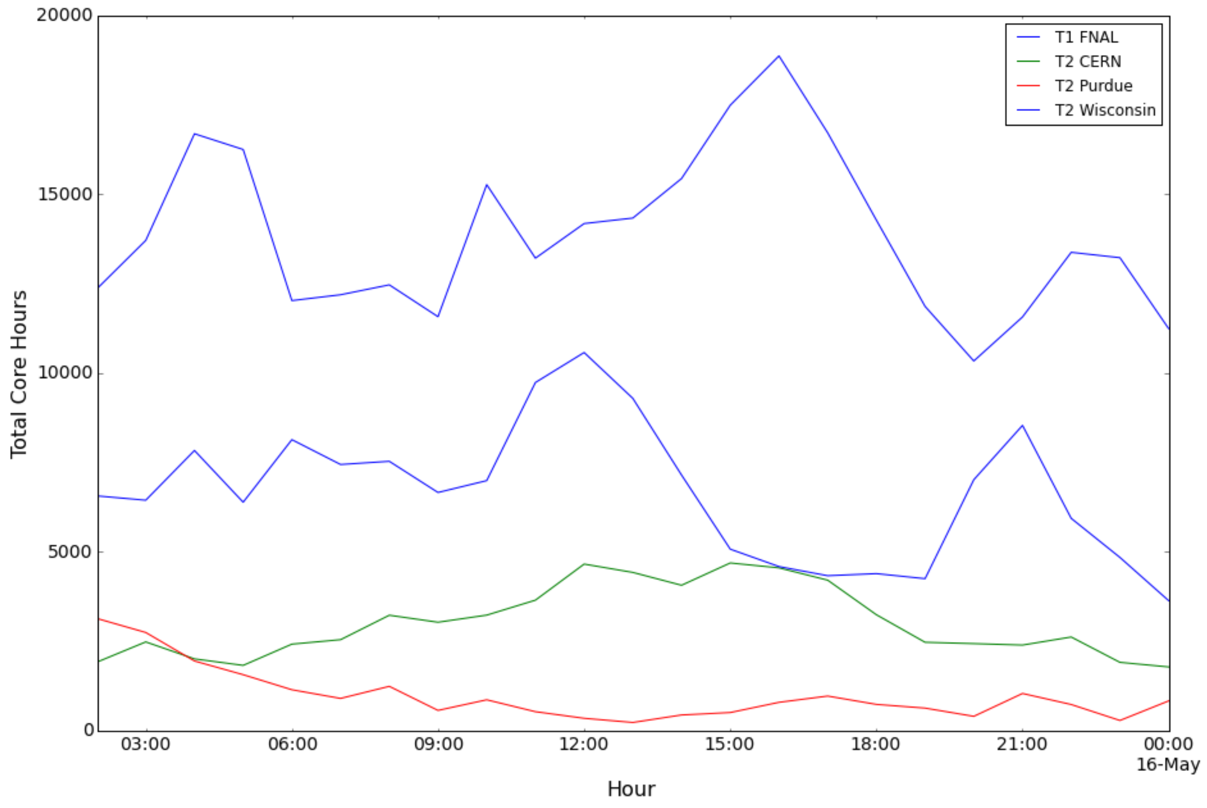


Figure 5.31: Total core hours in different Tier resources over just few hours on a random day.

Conclusions

The purpose of this thesis was to make performance studies of CMS workflows, namely centrally-scheduled production activities and unpredictable distributed analysis.

In order to monitor analysis jobs, a lightweight tool has been developed to provide easy access to the metrics contained in the logs to end users. The tool has been tested with job submissions of increasing complexity on Grid and Cloud infrastructures specifically designed for benchmarking: from logs of a toy-analysis personally conducted on Grid, to logs of a real physics analysis in the fully hadronic top research on a newly deployed Cloud infrastructure. In the top analysis sector, no major inefficiencies and misbehaviors were found: the developed tool has demonstrated to work on a real physics analysis, and is ready to be applied to other sectors (Higgs analyses, SUSY analyses, etc). Moreover, the tool emerged as a versatile and light mechanism for any physics analysis team to tune the Grid/Cloud job submissions and target the best performing sites. In conclusion, the tool has shown to have a potentially wide usage in CMS; the code is public and ready to be exploited by all collaboration members.

In order to improve the monitoring of both analysis and production jobs, commercial Big Data technologies have been exploited. The analysis of the data of the newly deployed ElasticSearch installation, known in CMS as the CMS Metrics Service, highlighted a few major directions of improvement in the CMS Computing Model:

- Workload Management. An extensive comparison of total core hours and total job count showed that the former, currently less used, provides a more accurate assessment of the actual CMS work loads, especially with regards to different production campaigns. An exhaustive investigation of workload splitting across Tiers in different regions (e.g. US, Italy,..) and Tier levels (e.g. Tier-1, Tier-2,..) showed the evolution paths of the CMS computing model. One of the results is the quantitative evidence of Tier-2 centers evolving to be the most important source of CPU power for CMS. Another result is the observation of high Tier-2 reliability, and the capability to run massive production campaigns with low failure rates comparable to Tier-1 centers. Furthermore, two very different campaigns have been chosen and closely studied in terms of failure modes, average CPU efficiencies, total wall clock time. Their fingerprints (in terms of e.g. CPU efficiency) allows to build better matching of CMS workloads to resources: it is important to note that this type of study at the campaign level can be performed only now with the new CMS Metrics Service. Lastly, the performance of two Tier-2 centers - CERN HLT and Legnaro Tier-2 - with different CPU capacity but equal production throughput delivered to CMS, has been studied. CPU efficiency and core hours showed again to be the most relevant metric to use. Thanks to the high granularity of the monitoring data in the CMS Metrics Service, a CPU efficiency profile lower than expected in certain days was measured for the HLT offline processing resource.

- Data Management. The amount of I/O operations necessary for the workflows completion was the core of the investigations. Data volume in input to (and output from) the processing CPUs at Tiers in different regions and for different Tier levels has been measured and studied over time. The main result is the measurement of the total aggregate I/O needed to be supported by Tier-2s. The results indicate the need to review the CMS mechanism for pile-up serving to jobs, which is a work in progress now in CMS.

- Infrastructure management. The possibility of performing quasi real-time monitoring has been briefly explored. In fact, all the queries performed through this work could be rerun over much shorter periods of time (e.g. $\sim 1h$). This direction of work is promising: for example, CMS Computing shifters may be equipped with a new powerful tool to make quick choices, and in general future work on more automated and proactive monitoring tools is envisioned.

A set of plots obtained in the studies performed in this thesis has been presented in the plenary session of the CMS Week in June 2016 by the coordinators of the CMS Software/Computing and PPD projects.

Appendix A

Important ElasticSearch attributes for the CMS implementation

This section documents the most commonly used and most useful attributes for doing queries over the CMS implementation of Hadoop through ElasticSearch [73].

Generic attributes:

- `RecordTime`: It is either the time when the job exited the queue or when the JSON document was last updated, depending on which event occurred first.
- `CoreHr`: It is the number of core-hours utilized by the job. The core-hours is the product of the number of CPU cores available for the job times the number of hours it takes.
- `CpuTimeHr`: It is the total amount of CPU time in hours.
- `QueueHrs`: It is the number of hours the job spent in queue before running.
- `WallClockHr`: It is the number of hours the job spent running. This is invariant of the number of cores.
- `CpuEff`: It is the total scheduled CPU time (user and system CPU) divided by core hours, in percentage. An example is the following: the job lasted for 24 hours, utilized 4 cores, and used 72 hours of CPU time. In conclusion, the `CpuEff` is 75%.
- `CpuBadput`: It is the badput associated with a job expressed in hours. It is the sum of all unsuccessful job attempts.
- `MemoryMB`: It is the total amount of RAM used by the job.

- `RequestCpus`: It is the number of cores used by the job.
- `RequestMemory`: It is the amount of memory requested by the job expressed in MB.
- `ScheddName`: It is the name of HTCondor scheduler where the job ran.
- `Status`: It is the status of the job: `Completed`, `Running`, `Idle`, or `Held`.
- `x509userproxysubject`: It is the DN of the grid certificate associated with the job; however, for CMS jobs, this is not the best attribute to use to identify a user. In fact, it is preferable: `CRAB_UserHN`.

CMS-specific attributes:

- `Campaign`: It is the campaign associated to the job; derived from the WMAgent workflow name. This attribute is only present for production jobs.
- `Workflow`: It is the human-readable workflow name. Example: `HIG-RunIISpring16DR80-01026_0`
- `WMAgent_RequestName`: It is the WMAgent request name. This attribute is only present for production jobs; example: `pdmvserv_task_HIG-RunIISpring16DR80-01026__v1_T_160530_083522_4482`.
- `WMAgent_SubTaskName`: It is the WMAgent subtask name. This attribute is only present for production jobs. An example: `/pdmvserv_task_HIG-RunIISpring16DR80-01026__v1_T_160530_083522_4482/HIG-RunIISpring16DR80-01026_0`
- `CMSGroups`: It is the name of the CMS group associated with the request. This attribute is only present for production jobs. Example: `HIG`.
- `Type`: It is the type of the job: `analysis` or `production`.
- `TaskType`: It is a more detailed task type classification, based on the CMSSW configuration. Values are `analysis`, `DIGI`, `RECO`, `DIGI-RECO`, `GEN-SIM`, and `Cleanup`.
- `MegaEvents`: It is the number of events processed by the job expressed in millions.
- `KEvents`: It is the number of events processed by the job expressed in thousands.
- `CMSSWKLumis`: It is the number of lumi sections processed by the job expressed in thousands.

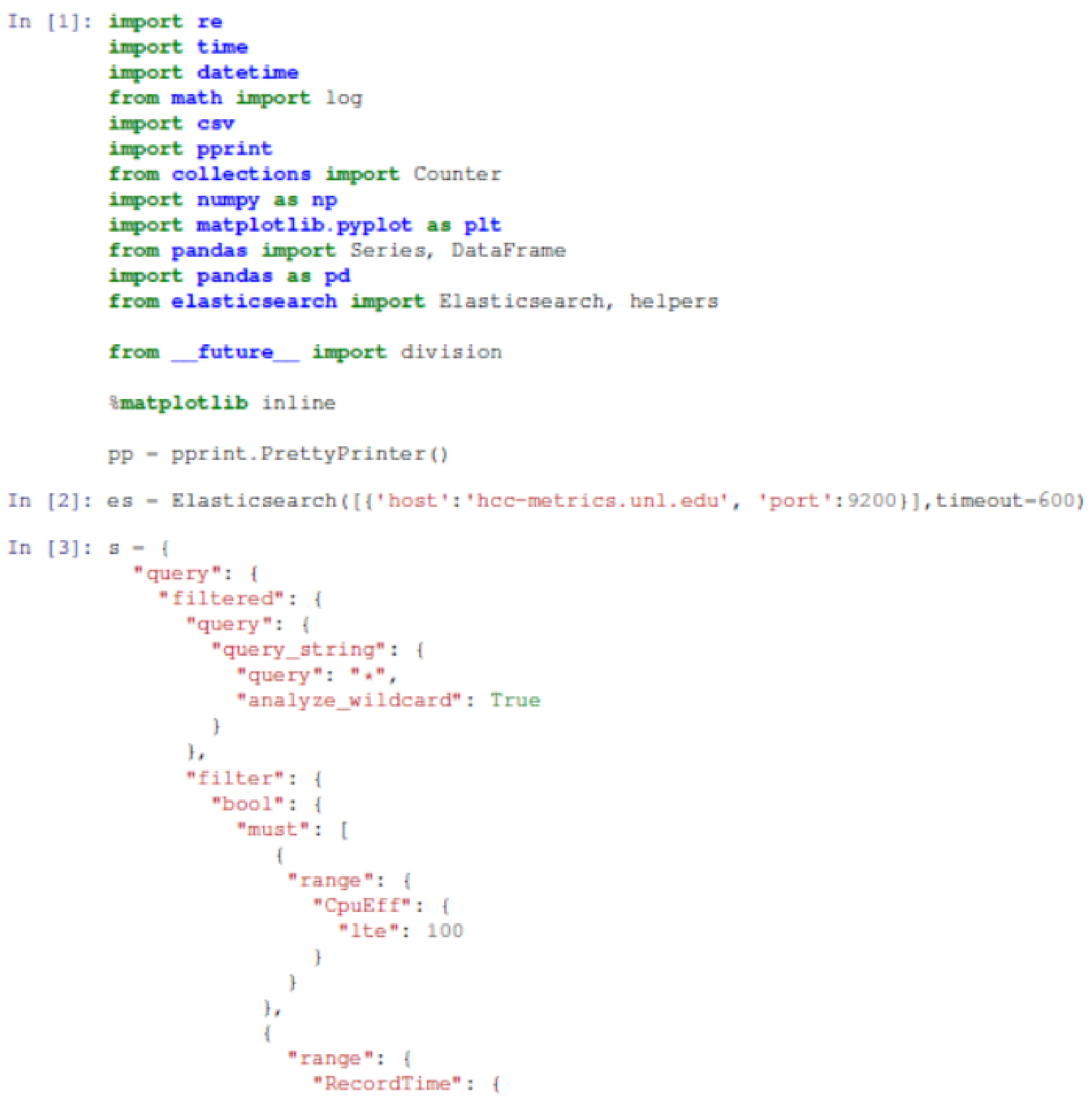
- ChirpCMSSWMaxEvents, ChirpCMSSWMaxFiles, ChirpCMSSWMaxLumis: They are respectively the maximum number of events, files, and lumis a job will process before exiting. These values may not be known for all jobs; in that case, -1 is reported.
- ExitCode: It is the exit code of the job. If available, this is the exit code of the last CMSSW step.
- CRAB_AsyncDest: It is the output destination CMS site name for a CRAB3 job.
- DESIRED_CMSDataset: It is the primary input dataset name.
- CRAB_DataBlock: It is the primary input block name. It is available for CRAB3 jobs only.
- DESIRED_Sites: It is the list of sites where the job could run on.
- DataLocations: It is the list of known primary input dataset locations.
- CMSSWWallHrs: It is the number of wall hours reported by cmsRun.
- StageOutHrs: It is an estimate of the stageout time, in hours. Calculated from WallClockHr-CMSSWWallHrs.
- InputData: Values are Onsite if Site is in the DataLocations list; Offsite otherwise.
- OutputGB: It is the total amount of output written by the CMSSW process expressed in gigabytes.
- InputGB: It is the total amount of data read by the CMSSW process expressed in gigabytes.
- ReadTimeHrs and ReadTimeMins: They are the total amount of time CMSSW spent in its IO subsystem for reads expressed respectively in hours and minutes.
- ReadOpPercent: It is the percentage of reads done via a single read operation. This reading method is opposed to a vectored readv operation.
- ReadOpSegmentPercent: percentage of read segments done via a single read operation.
- ChirpCMSSWReadOps: It is the number of read operations performed (this excludes readv activity).
- Site: It is the CMS site where the job ran. For example: T2_CH_CSCS.

- `Country`: It is the country where the job ran. For example: `CH`.
- `Tier`: It is the Tier type where the job ran. For example: `T2`.
- `CRAB_UserHN`: It is CMS username of user that submitted the task. This attribute is only present for analysis jobs.

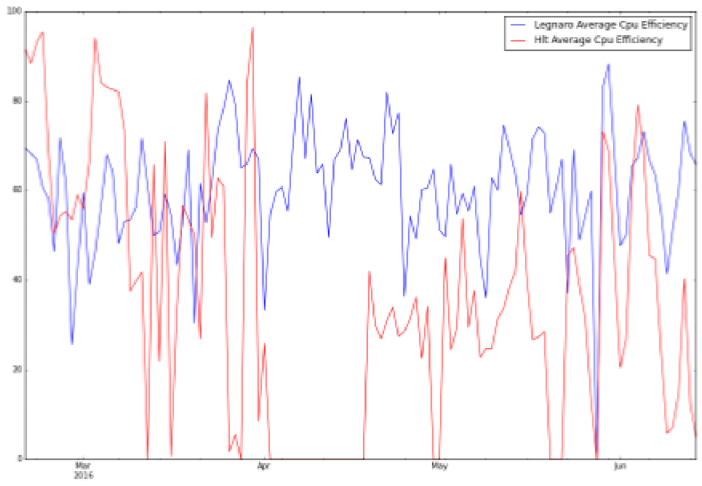
Appendix B

Example of Elasticsearch query

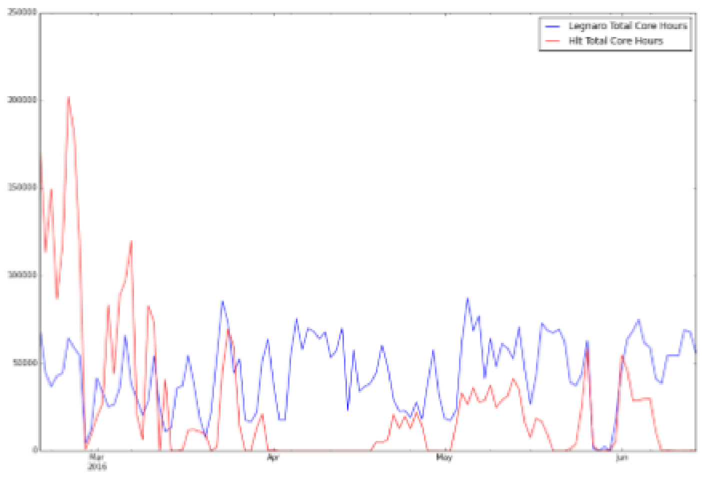
Below it is reported an example of Elasticsearch query, written using Jupyter Notebook, to compare to computing sites: `t2_ch_cern_hlt` and `t2_it_legnaro`.



```
In [1]: import re
        import time
        import datetime
        from math import log
        import csv
        import pprint
        from collections import Counter
        import numpy as np
        import matplotlib.pyplot as plt
        from pandas import Series, DataFrame
        import pandas as pd
        from elasticsearch import Elasticsearch, helpers

        from __future__ import division

        %matplotlib inline

        pp = pprint.PrettyPrinter()

In [2]: es = Elasticsearch([{'host':'hcc-metrics.unl.edu', 'port':9200}],timeout=600)

In [3]: s = {
          "query": {
            "filtered": {
              "query": {
                "query_string": {
                  "query": "*",
                  "analyze_wildcard": True
                }
              },
              "filter": {
                "bool": {
                  "must": [
                     {
                      "range": {
                        "CpuEff": {
                          "lte": 100
                        }
                      }
                    },
                    {
                      "range": {
                        "RecordTime": {
                          "gte": 1455926400000,
                          "lte": 1465941600000,
                          "format": "epoch_millis"
                        }
                      }
                    }
                  ],
                  "should": [
                     {"match":{"Site":"t2_ch_cern_hlt"}},
                     {"match":{"Site":"t2_it_legnaro"}}
                  ]
                }
              }
            }
          },
          "size": 0,
          "aggs": {
            "Date": {
              "date_histogram": {
                "field": "CompletionDate",
                "order":{"_key" : "asc"},
                "interval": "1d",
                "time_zone": "Europe/Berlin",
                "min_doc_count": 1,
                "extended_bounds": {
                  "min": 1455926400000,
                  "max": 1465941600000
                }
              },
              "aggs": {
                "Site": {
                  "terms": {
                    "field": "Site",
                    "order":{"_term" : "desc"}
                  },
                  "aggs": {
                    "AvgCpuEff": {
                      "avg": {
                        "field": "CpuEff"}
                      },
                     "TotCoreHr": {
                      "sum": {
                        "field": "CoreHr"}
                      },
                    }
                }
              }
            }
          }
        }

In [4]: agg = es.search(index='cms-*', body=s, request_timeout=12000)

In [5]: bucketsDate = agg['aggregations']['Date']['buckets']
In [6]: with open('Hlt_Legnaro.csv', 'w') as g:
            writer = csv.writer(g, lineterminator='\n')
            for b1 in bucketsDate:
                unix_date = b1['key']
                try:
                    Legnaro_AvgCpuEff = b1['Site']['buckets'][0]['AvgCpuEff']['value']
                    Legnaro_TotCoreHr = b1['Site']['buckets'][0]['TotCoreHr']['value']
                except:
                    Legnaro_AvgCpuEff = 0
                    Legnaro_TotCoreHr = 0
                try:
                    Hlt_AvgCpuEff = b1['Site']['buckets'][1]['AvgCpuEff']['value']
                    Hlt_TotCoreHr = b1['Site']['buckets'][1]['TotCoreHr']['value']
                except:
                    Hlt_AvgCpuEff = 0
                    Hlt_TotCoreHr = 0
                writer.writerow([
                unix_date,
                Legnaro_AvgCpuEff,
                Legnaro_TotCoreHr,
                Hlt_AvgCpuEff,
                Hlt_TotCoreHr,
                ])

In [7]: df = pd.read_csv('Hlt_Legnaro.csv', names=['Unix_Date','Legnaro_AvgCpuEff',
        'Legnaro_TotCoreHr', 'Hlt_AvgCpuEff', 'Hlt_TotCoreHr'])

In [8]: time_unix = df['Unix_Date'].tolist()
        readable_datetime = []
        for i in time_unix:
            readable_datetime.append(datetime.datetime.fromtimestamp(i/ 1e3))
        df.insert(0, 'Date', readable_datetime)
        del df['Unix_Date']

In [9]: df[0:4]

Out[9]:         Date  Legnaro_AvgCpuEff  Legnaro_TotCoreHr  Hlt_AvgCpuEff  \
        0 2016-02-20          69.530985       70766.839722      91.606359
        1 2016-02-21          68.260280       44681.791111      88.406579
        2 2016-02-22          66.958104       36566.495278      93.071425
        3 2016-02-23          60.682856       42733.059722      95.427299

           Hlt_TotCoreHr
        0  176435.488889
        1  113174.547222
        2  149387.157222
        3   86527.957222

In [10]: L_Eff = pd.Series(df['Legnaro_AvgCpuEff'].tolist(), index=df['Date'].tolist())
         H_Eff = pd.Series(df['Hlt_AvgCpuEff'].tolist(), index=df['Date'].tolist())

In [11]: ax1=L_Eff.plot(figsize=(15,10), label='Legnaro Average Cpu Efficiency',
         legend=True)
         H_Eff.plot(ax=ax1, label='Hlt Average Cpu Efficiency', legend=True, color='r')
         fig1 = ax1.get_figure()
         fig1.savefig('Legnaro_Hlt_Avg_Cpu_Eff.png')
```



```
In [12]: L_Eff = pd.Series(df['Legnaro_TotCoreHr'].tolist(), index=df['Date'].tolist())
         H_Eff = pd.Series(df['Hlt_TotCoreHr'].tolist(), index=df['Date'].tolist())

In [13]: ax2=L_Eff.plot(figsize=(15,10), label='Legnaro Total Core Hours', legend=True)
         H_Eff.plot(ax=ax2, label='Hlt Total Core Hours', legend=True, color='r')
         fig2 = ax2.get_figure()
         fig2.savefig('Legnaro_Hlt_Total_Core_Hours.png')
```



Bibliography

[1] K.A. Olive et al. "*Particle Data Group*", Chin. Phys. C, 38, 090001 (2014).

[2] LHCb Collaboration "*Observation of $J/\psi p$ resonances consistent with pentaquark states in $\Lambda_b^0 \to J/\psi K p$ decays*", Phys. Rev. Lett. 115, 072001 (2015).

[3] C. Quigg "*Gauge Theories of the Strong, Weak, and Electromagnetic Interactions: Second Edition*", Princeton University Press (2013).

[4] C. N. Yang and R. L. Mills "*Conservation of Isotopic Spin and Isotopic Gauge Invariance*", Phys. Rev. Volume 96, 1 (1954).

[5] R. Craig Group "*Measurement of the Inclusive Jet Cross Section Using the Midpoint Algorithm in Run II at CDF*", Dissertation presented to the graduate School of the University of Florida in partial fulfillment of the requirements for the degree of doctor of philosophy (2006).

[6] E. Fermi "*Tentativo di una teoria dei raggi β* La Ricerca Scientifica 2 (1933).

[7] S. L. Glashow, "*Partial-symmetries of weak interactions*", Nucl.Phys. 22 579 (1961).

[8] F. Englert and R. Brout "*Broken Symmetry and the Mass of Gauge Vector Mesons*", Phys. Rev. Lett. 13 321 (1964).

[9] P. Higgs "*Spontaneous Symmetry Breakdown without Massless Bosons*", Phys.Rev. 145 1156 (1966).

[10] G. S. Guralnik, C. R. Hagen, T. W. B. Kibble, "*Global Conservation Laws and Massless Particles*", Phys. Rev. Lett. 13 585 (1964).

[11] The Large Hadron Collider, `http://home.web.cern.ch/topics/large-hadron-collider`

[12] "*The CERN Large Hadron Collider*", `http://jinst.sissa.it/LHC/`, IOP/Sissa

[13] R. Aßmann, M. Lamont, S. Myers, "*A Brief History of the LEP Collider*", Nucl. Phys. B, Proc. Suppl. 109 (2002) 17-31.

[14] The CMS Collaboration, "*Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC*", Phys. Lett. B **716** (2012) 30.

[15] S. Chatrchyan *et al.* [CMS Collaboration], "*A New Boson with a Mass of 125 GeV Observed with the CMS Experiment at the Large Hadron Collider*", Science **338** (2012) 1569.

[16] The ATLAS Collaboration, "*Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC*", Phys. Lett. B, **716** (2012) 1.

[17] ALICE experiment web page, `http://aliceinfo.cern.ch/Public/Welcome.html`

[18] ATLAS experiment web page, `http://www.atlas.ch/`

[19] CMS Collaboration, "*The CMS experiment at the CERN LHC*", JINST **3** S08004 (2008).

[20] CMS experiment web page, `http://cms.web.cern.ch/`

[21] LHCb experiment web page, `http://lhcb-public.web.cern.ch/lhcb-public/`

[22] M. Procura, et al. "*Nucleon mass: from lattice QCD to the chiral limit*", Phys. Rev. D, **73**, 114510 (2006).

[23] U. Heinz "*Concepts of heavy-ion physics*", 2002 European School of high-energy physics, Pylos, Greece, 25 Aug-7 Sep 2002: Proceedings (2004).

[24] The TOTEM Collaboration, `http://totem.web.cern.ch/Totem/`

[25] The LHCf Experiment, `http://hep.fi.infn.it/LHCf/`

[26] The MOEDAL Collaboration, `http://home.web.cern.ch/about/experiments/moedal`

[27] CMS Collaboration "*The CMS tracker system project : Technical Design Report*", CERN-LHCC-98-006, CMS-TDR-5 - Geneva CERN, (1997).

[28] CMS Collaboration "*The CMS electromagnetic calorimeter project : Technical Design Report*", CERN-LHCC-97-033, CMS-TDR-4 - Geneva CERN, (1997).

[29] P. Adzic et al. "*Energy resolution of the barrel of the CMS electromagnetic calorimeter*", JINST 2 (2007) P04004, doi:10.1088/1748-0221/2/04/P04004.

[30] CMS Collaboration "*The CMS hadron calorimeter project: Technical Design Report*", CERN-LHCC-97-031 ; CMS-TDR-2 - Geneva CERN, (1997).

[31] S. Bertolucci, M. Cordelli, M. Curatolo, et al. "*Influence of Magnetic Fields on the Response of Acrylic Scintillators*", Nucl. Instr. and Meth. A254 (1987) 561.

[32] CMS Collaboration "*Performance of the CMS Hadron Calorimeter with Cosmic Ray Muons and LHC Beam Data*", physics.ins−det, (2010).

[33] CMS Collaboration, "*The TriDAS Project Technical Design Report, Volume 1: The Trigger Systems*", CERN/LHCC 2000/38, CMS Technical Report 6.1, (2000).

[34] CMS Collaboration, "*The TriDAS Project Technical Design Report, Volume 2: Data Acquisition and High-Level Trigger*", CERN/LHCC 2002/26, CMS Technical Report 6.2, (2002).

[35] V. Gori "*The CMS High Level Trigger)*", International Journal of Modern Physics: Conference Serie (2014).

[36] J. D. Shiers, "*The Worldwide LHC Computing Grid (worldwide LCG)*", Computer Physics Communications 177 (2007) 219–223.

[37] WLCG, `http://lcg.web.cern.ch/lcg/`

[38] European Grid Infrastructure, `http://www.egi.eu/`

[39] Open Science Grid, `http://www.opensciencegrid.org`

[40] Virtual Organization Membership Service, `http://toolkit.globus.org/grid_software/security/voms.php`

[41] D. Bonacorsi, "WLCG Service Challenges and Tiered architecture in the LHC era", IFAE, Pavia, April (2006).

[42] CMS Collaboration, "The CMS Computing Model", CERN LHCC 2004-035.

[43] CMS Collaboration, "The CMS Computing Project Technical Design Report", CERN-LHCC-2005-023.

[44] G. Bauer et al., "*The data-acquisition system of the CMS experiment at the LHC*", Journal of Physics, Conference Series 331 (2011) 02202.

[45] G. Petrucciani, A. Rizzi and C. Vuosalo, on behalf of the CMS Collaboration, "*Mini-AOD: A New Analysis Data Format for CMS*", Journal of Physics: Conference Series 664 (2015) 072052.

[46] A. Breskin, R. Voss "*The CERN Large Hadron Collider: Accelerator and Experiments Volume 2: CMS, LHCb, LHCf, and TOTEM*" (2009).

[47] T. Barrass et al, "*Software agents in data and workflow management*", Proc. CHEP04, Interlaken, (2004). See also `http://www.fipa.org`

[48] D. Bonacorsi, T. Barrass, J. Hernandez, J. Rehn, L. Tuura, J. Wu, I. Semeniouk, "*PhEDEx high-throughput data transfer management system*", CHEP06, Computing in High Energy and Nuclear Physics, T.I.F.R. Bombay, India, (2006).

[49] L. Tuura et al., "*Scaling CMS data transfer system for LHC start-up*", J. Phys.: Conf. Ser. 119 072030 (2008).

[50] Berkeley Database Information Index, `https://twiki.cern.ch/twiki/bin/view/EGEE/BDII`

[51] G. Boudoul et al., "*Monte Carlo Production Management at CMS*", J. Phys.: Conf. Ser. 664 072018 (2015).

[52] CRAB online manual `http://cmsdoc.cern.ch/cms/ccs/wm/www/Crab/Docs/crab-online-manual.html`

[53] D. Spiga et al., "*The CMS Remote Analysis Builder (CRAB)*", Lect. Notes Comput. Sci. 4873 580-586 (2007).

[54] G. Codispoti et al., "*CRAB: A CMS Application for Distributed Analysis*", Nuclear Science Symposium Conference Record N02.79 (2008).

[55] Software Guide on CRAB, `https://twiki.cern.ch/twiki/bin/view/CMSPublic/SWGuideCrab`

[56] CRAB3 online tutorial, `https://twiki.cern.ch/twiki/bin/view/CMSPublic/WorkBookCRAB3Tutorial`

[57] The CMS Dashboard, `http://dashboard.cern.ch/cms/`

[58] A. Tanasijczuk, "*CRAB3 architecture and task workow*" (2014) `https://twiki.cern.ch/twiki/bin/view/CMSPublic/CRAB3TaskFlow`

[59] The R Project for Statistical Computing, `https://www.r-project.org/`

[60] ROOT Data Analysis Framework, `https://root.cern.ch/`

[61] OpenStack, http://www.openstack.org/

[62] S. Zhou, J. Wang, X. Zheng, P. Delisle, "*Utopia: A load sharing facility for large, heterogeneous distributed computing systems*", TechnicalReportCSRI-257, Computer Systems Research Institute, University of Toronto (1992).

[63] R. Di Maria, "*Elastic Computing on Cloud Resources for the CMS Experiment*", Master thesis (2015).

[64] D. Bonacorsi, V. Kuznetsov, et al., "*Exploring patterns and correlations in CMS Computing operations data with Big Data analytics techniques*" PoS ISGC2015 008 (2015).

[65] D. Miner, "*Hadoop: what you need to know*" O'Reilly Media (2016).

[66] K. Sitto and M. Presser, "*A Field Guide to Hadoop: An Introduction to Hadoop, Its Ecosystem, and Aligned Technologies*" O'Reilly Media (2015).

[67] B. Bockelman, "*Using Hadoop as a Grid Storage Element*" CSE Conference and Workshop Papers, Paper 157, (2009), http://digitalcommons.unl.edu/cseconfwork/157

[68] ElasticSearch, https://www.elastic.co/products/elasticsearch

[69] Kibana, https://www.elastic.co/products/kibana

[70] Jupyter Notebook, http://jupyter.org/

[71] HTCondor, https://research.cs.wisc.edu/htcondor/

[72] B. Bockelman, D. Bonacorsi: personal communication.

[73] ElasticSearch upload for CMS and HTCondor data, https://github.com/bbockelm/cms-htcondor-es/

[74] 38th International Conference On High Energy Physic, https://www.ichep2016.org/

[75] Rencontres de Moriond, http://moriond.in2p3.fr/

[76] Physics Data and MC Validation (PdmV), https://twiki.cern.ch/twiki/bin/viewauth/CMS/PdmV

[77] Physics Performances and Dataset (PPD) project, https://twiki.cern.ch/twiki/bin/view/CMS/PhysicsPerfomanceDatasetHome

[78] Error codes currently sent from CMS jobs to the dashboard, https://twiki.cern.ch/twiki/bin/view/CMSPublic/JobExitCodes