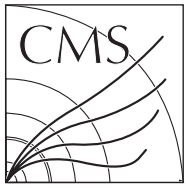
View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by CERN Document Server

Available on CMS information server

CMS CR -2009/086



The Compact Muon Solenoid Experiment

Conference Report

Mailing address: CMS CERN, CH-1211 GENEVA 23, Switzerland

12 May 2009

Use of the gLite-WMS in CMS for production and analysis

G.Codispoti, C.Grandi, A.Fanfani, D.Spiga, M.Cinquilli, F.Farina, E.Miccio, F.Fanzago, A.Sciaba', S.Lacaprara, S.Belforte, D.Bonacorsi, A.Sartirana, D.Dongiovanni, D.Cesini, S.Wakefield, J.Hernández, S Lemaitre, M.Litmaath, Y.Calas and E.Roche

Abstract

The CMS experiment at LHC started using the Resource Broker (by the EDG and LCG projects) to submit Monte Carlo production and analysis jobs to distributed computing resources of the WLCG infrastructure over 6 years ago. Since 2006 the gLite Workload Management System (WMS) and Logging & Bookkeeping (LB) are used. The interaction with the gLite-WMS/LB happens through the CMS production and analysis frameworks, respectively ProdAgent and CRAB, through a common component, BOSSLite. The important improvements recently made in the gLite-WMS/LB as well as in the CMS tools and the intrinsic independence of different WMS/LB instances allow CMS to reach the stability and scalability needed for LHC operations. In particular the use of a multi-threaded approach in BOSSLite allowed to increase the scalability of the systems significantly. In this work we present the operational set up of CMS production and analysis based on the gLite-WMS and the performances obtained in the past data challenges and in the daily Monte Carlo productions and user analysis usage in the experiment.

Presented at *CHEP09: International Conference On Computing In High Energy Physics And Nuclear Physics, 21-27 Mar 2009,Prague,Czech Republic,15/05/2009*

Use of the gLite-WMS in CMS for production and analysis

G Codispoti[1], C Grandi[1], A Fanfani[1], D Spiga[2], M Cinquilli[3], F Farina[5], E Miccio[4], F Fanzago[6], A Sciaba'[2], S Lacaprara[7], S Belforte[8], D Bonacorsi[1], A Sartirana[4], D Dongiovanni[4], D Cesini[4], S Wakefield[9], J Hernández[10], S Lemaitre[2], M Litmaath[2], Y Calas[2] and E Roche[2]

[1]Università and INFN Bologna
[2]CERN
[3]INFN Perugia
[4]INFN-CNAF
[5]CERN and INFN Milano Bicocca
[6]CERN and INFN-CNAF
[7]INFN-LNL
[8]Università di Trieste and INFN
[9]Imperial College London
[10]CIEMAT

E-mail: `Giuseppe.Codispoti@bo.infn.it`

Abstract. The CMS experiment at LHC started using the Resource Broker (by the EDG and LCG projects) to submit Monte Carlo production and analysis jobs to distributed computing resources of the WLCG infrastructure over 6 years ago. Since 2006 the gLite Workload Management System (WMS) and Logging & Bookkeeping (LB) are used. The interaction with the gLite-WMS/LB happens through the CMS production and analysis frameworks, respectively ProdAgent and CRAB, through a common component, BOSSLite. The important improvements recently made in the gLite-WMS/LB as well as in the CMS tools and the intrinsic independence of different WMS/LB instances allow CMS to reach the stability and scalability needed for LHC operations. In particular the use of a multi-threaded approach in BOSSLite allowed to increase the scalability of the systems significantly. In this work we present the operational set up of CMS production and analysis based on the gLite-WMS and the performances obtained in the past data challenges and in the daily Monte Carlo productions and user analysis usage in the experiment.

1. Introduction

The Compact Muon Solenoid (CMS) [1] experiment will collect data produced at the CERN Large Hadron Collider (LHC) [2]. LHC is designed to produce large event statistics to cope with many physics challenges: CMS has therefore to deal with a large amount of data and to produce a comparable amount of Monte Carlo simulated data. A large experimental community, more than 3000 collaborators, distributed over more than 38 countries all around the world is involved to face technological and physics challenges.

The CMS computing model has been designed to be highly distributed with respect to resource access for the analysis and data storage. The choice of a distributed system allows reducing costs, delegating responsibilities to the individual institutes and participating national organizations and ensures load balancing of the available resources while replicating the interesting data in different sites. The system is organized in a hierarchical order of regional computing centres interconnected through the Grid middleware.

The World-wide LHC Computing Grid (WLCG) Project [3] provides, through the supporting projects EGEE, OSG and Nordugrid, a Grid to support the computing models of the LHC experiments. The Grid middleware provided by the EGEE project allows single sign-on on a User Interface (UI): the user submits jobs through the gLite Workload Management System (WMS) [4] to all CMS centers via their Computing Elements (CE) that allow transparent access to the batch queues of the sites. Logging and Bookkeeping (LB) dedicated servers, possibly shared among different WMSs, allow tracking the jobs life from the submission to the output retrieval. Storage Elements (SE) allow remote access to storage resources presenting to the user a single logical file namespace where CMS data are stored.

The Monte Carlo production system and the analysis tools have been instrumented to deal with the Grid middleware, but also with local resources such us the CERN Analysis facility, an LSF based facility for high priority tasks, and local batch systems located at the institutions. CMS designed and developed BOSSLite, a common Grid/batch interface in order to provide the workload management systems with a unique interface. Moreover, BOSSLite provides logging facilities in order to collect jobs statistics for debugging purposes.

In this paper we will describe the BOSSLite architecture (section 2 and 3) and the CMS experience using gLite WMS in the past years.

2. BOSSLite: a light weight implementation of a Batch Object Submission System

BOSSLite is a python library developed to interface the CMS Workload Management tools to the Grid middlewares and local batch systems. It relies on a database to track and log information into an entity-relation schema. Information is logically remapped into python objects that can be transparently used by the CMS framework and tools.

BOSSLite has to be highly efficient to not introduce a bottleneck in the Grid operations and to provide safe operation in a multiprocessing/multi threaded environment. For that reason, the interaction with the database is performed through safe sessions and connection pools.

A database session provides access to database operations through a generic abstract interface, enabling standard operations such as open/close connection, query, insert/update rows and so on. In the current implementation, two subclasses provide support for MySQL and SQLite databases. Support for other databases can be added by creating new subclasses. The usage of database connection pools keeps a set of open connections to the database to avoid creating and closing them every time a user requires database interaction.

In the very same way, the Scheduler part of BOSSLite provides a generic abstract interface, enabling standard operations such as job submission, tracking, cancel and output retrieval. Scheduler specific interfaces are implemented in specific plug-ins, loaded at run-time. Presently, plug-ins are implemented for EGEE (gLite), OSG and ARC (NorduGrid) middleware stacks and LSF, and SGE batch systems.

A set of high level API connects the Scheduler and the Database part, implementing the default behaviour as well as standard methods of general utility. The high level scheduler API depends on the Database high level API, allowing coherent update and handling of the information to be stored in the database.

The BOSSLite architecture is showed in Figure 1.

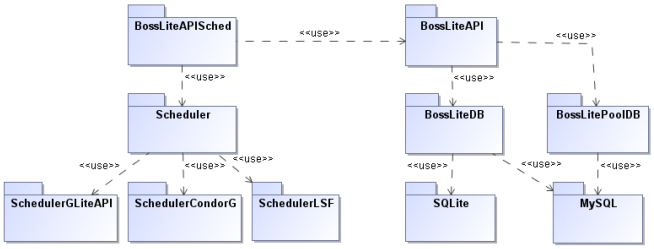


Figure 1. Schematic view of the BOSSLite Architecture.

3. Database Schema

The typical CMS analysis requires access to a huge amount of data, possibly split in many files, optimized in size. Since in general the same algorithm has to run over the whole sample, the analysis task can be split in many jobs that may run in parallel. The only difference among them is given by the parameters identifying the portion of data to be accessed. The same applies for the Monte Carlo jobs where, instead of being analyzed, data are produced by parallel jobs in small files and merged later.

The BOSSLite database structure reflects the typical CMS analysis task. A top entity called task groups similar jobs, storing common information such as common jobs requirements, dataset to be accessed/produced and so on. Jobs are then provided with a static part, characterizing the job itself by recording for instance arguments and file names, and a dynamic part. The latter stores information such as scheduling timestamps, job status and destination. Since a job may fail for many reasons, there may be as many resubmissions as needed: in order to record the execution information of every submission for logging purposes, they are stored in the so called running instances. A job may have a running instance for each resubmission.

A schematic view of the database structure is showed in Figure 2.

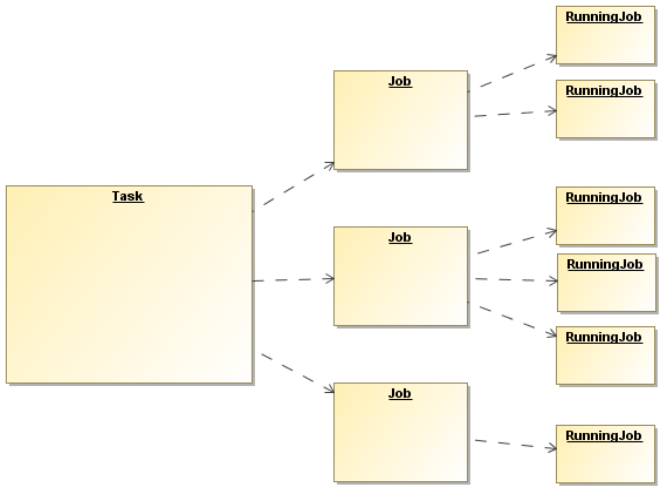


Figure 2. Task information mapping in the BOSSLite database.

4. gLite specific Interface

The BOSSLite interface to the gLite middleware is meant to make an optimal usage of the gLite features. Since different CMS workload management tools (e.g. for production or for analysis) may have different setup and workflow, the scheduler interface has to be flexible enough to ensure the coherent usage of the features optimizing the tool workflow.

Using the gLite WMS BOSSLite enables bulk job submission through gLite collections. This allows a faster job submission, but also an optimized job status tracking through bulk queries.

Since the CMS computing model uses its own data location system, the WMS match making has the main role to perform a load balancing among sites that are hosting the data to be accessed or where Monte Carlo simulated data have to be produced. The usage of the bulk match-making further reduces the actual load of the WMS.

The access to the gLite features is made through the WMProxy python API. This allows the association between BOSSLite jobs and their grid identifiers, a non-trivial task if performed through the command line. This reduces also the overhead of parsing "human readable" streams (i.e. the standard output of the command line). At the same time, the WMProxy API are not designed to be a high level tool, reflecting the actual WMS operations steps. This makes their usage quite complex so that the BOSSLite interface is not just a tiny layer. Among other things, it has to reimplement UI features such as configuration files interpretation and sandbox transfer. Furthermore they are exposed to python compatibility issues, which are often the main deployment problem. For that reason we proposed a new command line feature: a machine readable output stream (e.g. xml/json formatted) providing the same level of details of the API. This will allow to reuse all UI functionality already in place that currently is not accessible by WMProxy API. Through simple subprocesses it will then be possible to extract all information, keep a specific error logic and encapsulate environment and compatibility issues. Furthermore this will reduce the effort to maintain the BOSSLite gLite interface since all the needed features are maintained by the gLite developers in the UI software. The development of this new feature in the gLite UI CLI has been inserted in the EGEE-III program of work and is currently being implemented.

The usage of the low level LB API makes the job status tracking faster and allows an easy retrieval of other useful information at runtime: destination queue, status reason, scheduling timestamps, etc.

5. Integration in the CMS tools

As aforementioned, different setups characterize the different CMS workload management tools. BOSSLite must be able to use the optimal set of gLite features for each use case in a coherent way. The following tools currently use BOSSLite for the gLite submission:

ProdAgent [5]: an automatic, parallelized system for Monte Carlo samples simulation and for data processing; to reduce the possibility of bottlenecks, both job status tracking and output retrieval are multi-threaded; since the output data is transferred directly to the destination site, the output retrieval component has just to deal with job reports and log files;

CRAB [6][7]: a transparent interface to the Grid infrastructure, integrated with the CMS workload management system to reduce the user effort for simple analysis tasks;

CRAB Server [8]: a centralized system dealing with huge user tasks, automating the analysis workflow; its purpose is to optimize the resources usage, dealing with user tasks and implementing intensive analysis tasks such as calibration; it shares the tracking and output retrieval components with ProdAgent; it is characterized by a multi-user environment and multi-threaded process concurrently accessing the database and the user jobs.

CRAB server and ProdAgent host BossLite tables in a MySQL database while the stand-alone CRAB relies on an SQLite database. Both ProdAgent and the pure client version of CRAB access the gLite functionality through BOSSLite miming basic UI functions. CRAB Server reuses part of the ProdAgent architecture, but implements multi-threaded submission to allow many users to submit tasks concurrently. CRAB Server additionally has to manage grid credentials from the many users who can submit to it, therefore needs a robust handling of delegated proxies to avoid clashes and security flaws. With respect to the other tools, CRAB Server uses a different mechanism for sandbox handling. Being coupled with a GridFTP server, using gLite features, the sandboxes are directly transferred to/from the worker node. This allows implementing specific CMS policies on sandbox sizes, bypassing the WMS limits.

6. gLite WMS usage experience in the CMS tools

Since the gLite architecture is such that the system scales linearly with the number of gLite WMSs used, CMS Monte Carlo production and analysis jobs are balanced over many WMS. Currently 7 WMSs are deployed for the analysis, 4 for the Monte Carlo production. The typical instantaneous load of a single WMS, when no particular operations are scheduled, may reach 15k jobs per day (Figure 3), with peaks of 5k active jobs, running or idle, simultaneously handled.

Dedicated stress tests [10] showed that a single WMS may handle 30k jobs per day.

CMS started using the gLite WMS in 2006. Last year showed an increase in the activity, starting from the CCRC08 challenge in May 2008. Here we discuss for convenience results covering the period from May 2008 to March 2009. In this period, about 23 million jobs were submitted through the gLite WMS, with an average of about 75k jobs per day, divided as follows:

- 30k analysis
- 20k Monte Carlo production
- 25k other activities, such as regular submission to all sites to probe their readiness (JobRobot [11]) and private productions

The breakdown of these activities over the whole period is shown in Figure 4.

Jobs have been uniformly distributed over more than 40 sites, as shown in Figure 5

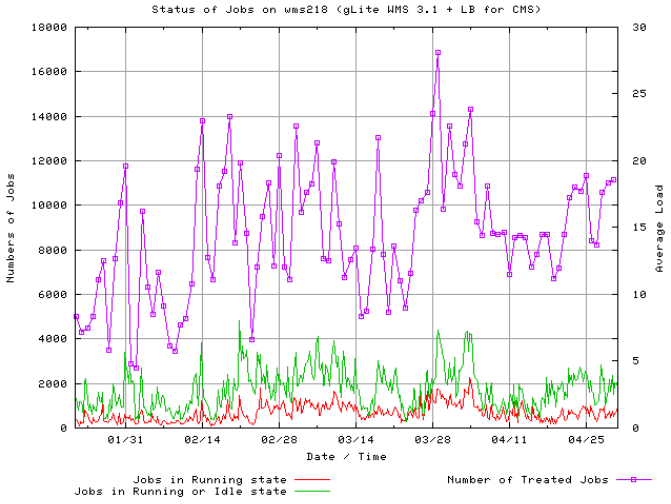


Figure 3. Typical instantaneous load of a gLite WMS used for CMS operations.

6.1. gLite WMS usage in analysis tasks

A prompt integration of the job submission tools for the analysis with the gLite middleware allowed a very early usage of the gLite WMS by CRAB. Both the middleware developers, receiving prompt feedbacks, and the CRAB developers profited by the synergy of this early usage.

In the reference period, 8.8 million analysis jobs were submitted to the WMS through CRAB: about 78% of the CMS analysis jobs total (the remaining submissions were mainly done directly via CondorG) with a daily rate of about 30kJobs/day. The actual number of distinct CMS users reached about 600 in the first 3 months of 2009.

Dedicated scale tests were set up to probe the CRAB server scalability in October 2008 (see [9]). A single CRAB Server instance in multi user mode reached 50kJobs per day, balancing jobs among two WMSs. The reached limits are mainly due to the jobs status tracking and output handling. The exercise allowed many optimizations and tweaks to further improve the scalability of the CRAB server.

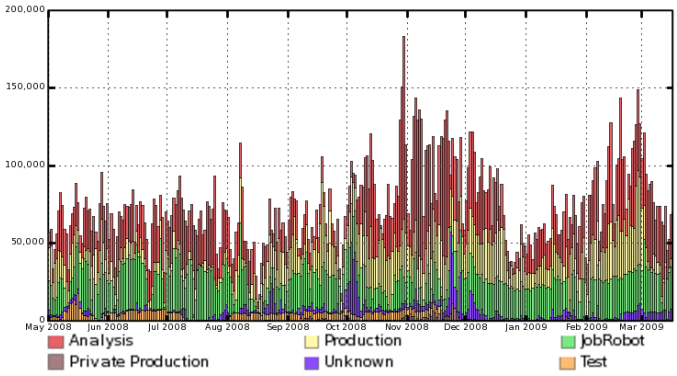


Figure 4. Distribution of CMS jobs submitted to the gLite WMSs divided by activity.

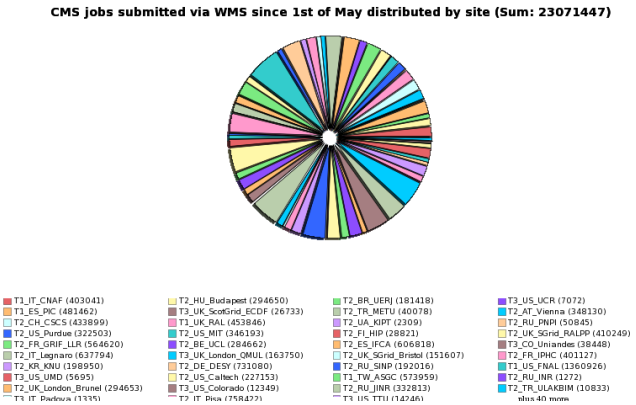


Figure 5. Geographical distribution of jobs submitted to the gLite WMSs.

As well as for the WMSs, many CRAB server instances may be deployed. In a configuration where 7 WMSs are deployed, using a few CRAB servers, it is in principle possible to cope with the expected rates for CMS in data taking regime, corresponding to 100/200 kJobs/day. The CRAB servers in production, actually used by real users, currently handle a few thousand jobs per day.

6.2. WMS usage in Monte Carlo Production

A portion of the CMS Monte Carlo production jobs are submitted through the gLite WMS. In the reference period they amounted to about 5.3 million jobs with a daily rate of about 20kJobs/day.

A single ProdAgent instance showed to be able to sustain a rate of 30kJobs/day. The maximum Monte Carlo production rate is a bit lower than the maximum analysis rate. The lower throughput is due to the fact that log files and job reports are retrieved through the WMS and most of the time is spent in the remote copy. Possible improvements would come by reducing the size and number of the files to be retrieved.

The expected Monte Carlo production rate of 50/100 kJobs/day can be reached easily by deploying few ProdAgent instances.

6.3. Success and Failure rates

Figure 6 illustrates the success rate of jobs submitted to the gLite WMS in the main CMS activities. Only 58% of the analysis jobs terminated successfully. The pie-chart shows that the main reason for the failures is application failures, not Grid problems. Application failures are expected, since analysis jobs run user code which may not have been tested thoroughly, but more detailed analysis showed that the main reason for analysis job failures is the stage out of the data output files to remote SRM servers (often those files are too large to be retrieved by

WMS inside the OutputSandBox). CMS is developing a system to asynchronously copy user data to the remote final destination using temporary buffering at the site where the jobs run, in order to mitigate the problem. The same detailed investigation also showed that a good fraction of grid failures are not due to the middleware layer, but are simply jobs that for one reason or another spend too much time on the worker node and are killed by the local batch system, appearing as aborted by the Grid.

In the Monte Carlo production the application failure rate is lower, thanks to the usage of validated code and local stage out. Grid failures are also fewer, but still jobs can be killed by local batch systems for various reasons. The most precise determination of the amount of grid failures we experience, is from the JobRobot pie-chart. The JobRobot is an automated tool for CRAB job submission, performing scheduled submissions to test the sites efficiency. The same application is used every day on every site on the same dataset, therefore the chances of jobs entering some loop and being killed are much reduced. Also all application failures are clearly site problems (NFS mount, data access etc.). For the JobRobot, the Grid failure rate is reduced to 7% of the submitted jobs. It is worth noting that many of the remaining grid failures, mostly due to CE overload or ill-configured WorkerNodes, are cured by simple resubmission of the jobs. In CMS we prefer not to use automatic resubmission by the gLite WMS since ProdAgent and CrabServer can take more educated decisions about resubmission by looking also at application failures, and those tools will anyhow hide most transient problems from the user. The rate of random middleware-induced failures we see is well manageable by our tools.

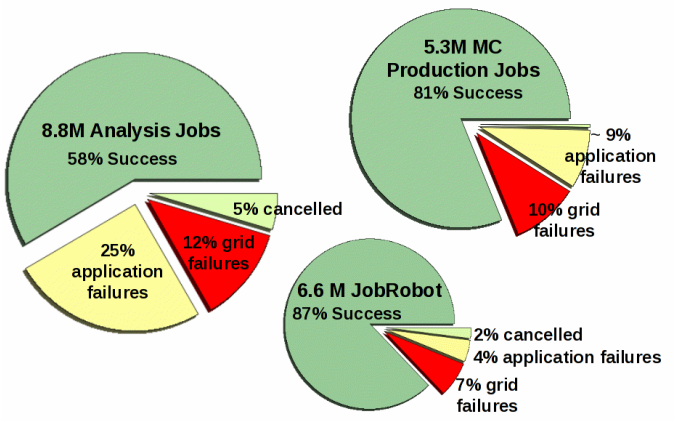


Figure 6. Success rate of CMS jobs submitted to gLite WMSs divided per activity.

7. Conclusions

CMS successfully uses the gLite WMS in MonteCarlo Production and Analysis tasks since 2006. We have shown that more than 30kJobs/day can be handled by a single gLite WMS instance. Since the CMS tools may use in parallel as many WMS as needed, they are able to submit up to 50 kJobs/day per instance. This means that we are able to cover CMS requests by delivering a few instances of CRAB and ProdAgent.

The every day experience and usage allows us to improve the system and provide feedback to the WMS/gLite developers since years. CMS can report a very fruitful collaboration with the gLite developers to fix bugs and implement new features.

The CMS experience shows that the WLCG infrastructure, when properly used, may give very good results in terms of reliability and usability.

References

[1] Chatrchyan S et al. The CMS Experiment at CERN LHC 2008 *Journal of Instrumentation*, vol 3, pp.s08004

[2] LHC Homepage http://cern.ch/lhc-new-homepage/
[3] LHC Computing Grid (LCG), Web Page, http://lcg.web.cern.ch/LCG/ and LCG Computing Grid - Technical Design Report, LCG-TDR-001 CERN/LHCC 2005-024, (2005)
[4] Andreetto P et al. The gLite Workload Management System Proceedings of Computing in High Energy and Nuclear Physics (CHEP) 2007, Victoria, British Columbia (CA), Sep 2007.
[5] Evans D et al. The CMS Monte Carlo Production System: Development and Design, 18th Hadron Collider Physics Symposium 2007 (HCP 2007) 20-26 May 2007, La Biodola, Isola d'Elba, Italy. Published in Nuclear Physics B - Proceedings Supplements,Volumes 177-178 (2008) 285-286
[6] CRAB homepage, https://twiki.cern.ch/twiki/bin/view/CMS/SWGuideCrab
[7] Spiga D et al. The CMS Remote Analysis Builder (CRAB) 14th Int. Conf. on High Performance Computing (HiPC 2007). Goa, India. Dec 18-21 2007. (vol. 4873, pp. 580-586). ISBN/ISSN: 978-3-540-77219-4.
[8] Codispoti G et al. CRAB: a CMS Application for Distributed Analysis, Conference Record N-02-79, 2008 Nuclear Science Symposium, 19 - 25 October 2008, Dresden, Germany
[9] Codispoti G et al. Distributed analysis with CRAB: the client-server architecture evolution and commissioning, ACAT 2008 - XII Advanced Computing and Analysis Techniques in Physics Research November 3-7 2008 Erice, Italy, proceedings at POS (ACAT) 029
[10] Miccio V et al. Experience in Testing the Grid based Workload Management System of a LHC Experiment FERMILAB-CONF-09-068-CD, 2008. Published in Proceedings of the 2008 International Conference on Grid Computing & Applications, GCA 2008, Las Vegas, Nevada, USA, July 14-17, 2008 Hamid R. Arabnia (Ed.), p.40-45
[11] Flix J et al The commissioning of CMS computing centres in the worldwide LHC computing Grid, Conference Record N-29-5, 2008 Nuclear Science Symposium, 19 - 25 October 2008, Dresden, Germany