Available on CMS information server **CMS NOTE 2003-014**

The Compact Muon Solenoid Experiment

CMS Note

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

May 1, 2003

CMS Test of the European DataGrid Testbed

CMS/EDG Stress-Test Task Force[1]

Abstract

Starting in the middle of November 2002, the CMS experiment undertook an evaluation of the European DataGrid Project (EDG) middleware using its event simulation programs. A joint CMS-EDG task force performed a "stress test" by submitting a large number of jobs to many distributed sites. The EDG testbed was complemented with additional CMS-dedicated resources. A total of ~ 10000 jobs consisting of two different computational types were submitted from four different locations in Europe over a period of about one month. Nine sites were active, providing integrated resources of more than 500 CPUs and about 5 TB of disk space (with the additional use of two Mass Storage Systems). Detailed descriptions of the adopted procedures, the problems encountered and the corresponding solutions are reported. Results and evaluations of the test, both from the CMS and the EDG perspectives, are described.

[1]Appendix A

Table of Content

1 Executive Summary

Introduction

The Compact Muon Solenoid experiment (CMS) [1] is one of the four particle physics experiments that will collect data at the Large Hadron Collider (LHC) [2] being built at CERN (Geneva, Switzerland) [3].

The challenge for the CMS computing infrastructure is to cope with the very large computational and data access requirements in order to perform the data analysis that will lead to physics results. The size of the resources required, the complexity of the software and the physical distribution of the CMS collaboration naturally imply a distributed computing and data access solution.

The Grid paradigm is therefore one of the most promising solutions to be investigated, and CMS is collaborating with many Grid projects around the world in order to explore the maturity and availability of middleware implementations and architectures.

The European DataGrid project (EDG) [4] is a three-year EU funded program under the V framework. The work presented in this report is an evaluation of the available middleware provided by the EDG in its second year of implementation of Grid functionality.

Though the CMS software design has yet to be finalised, as data taking is not foreseen to commence until 2007, certain fundamental architecture decisions have to be made well in advance. A part of this architecture is strongly related to Grid technology.

CMS decided to actively participate in the EDG project since its outset, with the aim of understanding how the Grid can be useful for CMS and how CMS software needs to be adapted in order to maximise the benefit of using Grid functionality and tools.

Currently the main computing activity of CMS is related to the simulation, with Monte Carlo based programs, of how the experimental apparatus will behave once it is operational. This is to better understand the physics discovery potential as well as to investigate any possible modifications required to the data acquisition and processing. This activity is known as CMS “Production and Analysis”.

The current implementation of CMS production software was used to evaluate the EDG middleware deployed on the EDG testbed. The aim was also to deliver a large number (of order one million) of simulated CMS events to be analyzed within the CMS physics programme.

Goals and achievements

The main goals of the test (referred to as a “Stress Test” because of its high demand on the availability and responsiveness of software and hardware resources) were therefore defined as:

- Verification of the portability of the CMS Production environment into a Grid environment;
- Verification of the robustness of the EDG middleware in a production environment;
- Production of data for physics studies of CMS, possibly at the level of about one million simulated events.

All three goals were met partially, though by differing amounts.

The current CMS production environment was successfully ported to the EDG implementation. This process gave some indication as to how the CMS production architecture may need to be modified to cope with Grid middleware.

Attempts to verify EDG robustness in a production environment demonstrated a lack of software maturity. Though this was to be expected, given the fact that this is only the second year of the EDG R&D program. Obtaining a robust, or at least stable, environment proved to be quite difficult during the test.

The production of CMS simulated events at the level of 1 million could not be obtained. However, more than 250 k events were produced successfully in a period of three weeks (compared to the planned four weeks).

Finally, the Stress Test demonstrated that CMS software could be run in the EDG environment, provided that some modification of the CMS Production tools are made to account for the different interfaces between the software/middleware components.

Dynamic addition of new sites and resources to the testbed was demonstrated to be possible without disruption to the system as a whole, which is promising for the future as the system scales up in complexity, size and use.

Use of an application as demanding as CMS Production has highlighted the stability of the middleware as being one of the major challenges in a large distributed environment. Grid functionality must be improved to meet application level requirements, and the analysis of the results from this test shows that improvement of the stability and quality of delivered middleware is of the highest priority.

Results

A quantitative evaluation of this Stress Test demonstrated that the efficiency of correctly completed CMS simulation jobs increased during the period of testing. At the start of the test it was of the order of 85% for simple simulation activities (the so called CMKIN jobs, which require very little computing time and have relatively simple requirements from Grid services), rising to around 95% for the final version of the EDG software that was tested. A greater improvement was measured for more complex simulation jobs (the so called CMSIM jobs, which require around 12 hours of computing time per job and have more complex requirements from Grid services), starting from about 40% efficiency and ending with around 70%.

An evaluation of the readiness of the EDG Grid implementation from the point of view of CMS Production was also performed. In this case the success or failure of the job was determined using CMS developed job-tracking tools. The resulting efficiencies were significantly higher than the ones obtained via EDG tools, as the evaluation was performed based only on the usability of the obtained data.

The total number of submitted jobs over a period of three weeks was of the order of 10 000, submitted from four different European sources and using nine different testbed sites to provide resources and services.

Better overall efficiencies were obtained as the identified problems were corrected (or patched), both for bare EDG success/failure evaluation and for CMS Production of useful events. A post-Stress-Test EDG release (version 1.4.3) was also evaluated with reduced statistical significance and details, giving a trend of increased efficiency.

A more detailed evaluation of the test led to the identification of bottlenecks as well as to some reduced functionality of the different EDG middleware components. This identification activity itself enabled a rapid response to problems, occasionally enabling corrections and workarounds to be dynamically enabled in the live environment. The coordination of the EDG testbed was also tested in depth and some inadequacies were identified, promptly recognized and were possible corrected.

Finally, many suggestions for the priority of middleware functionality to be provided in coming releases were identified during the Stress Test.

Document structure

The following chapters describe the details of the procedures and activities of the CMS/EDG Stress Test.

Chapter 2 briefly deals with the CMS software and tools for Monte Carlo productions; Chapter 3 describes the utilized components of the EDG delivered middleware; Chapters 4 and 5 describe the CMS Stress Test procedures and planned preparations.

Chapter 6 reports the phenomenology of the encountered problems and the solutions applied.

Chapter 7 gives the obtained results and Chapter 8 summarizes the Test's conclusions.

2 CMS Software for Monte Carlo Productions

The main distinguishing feature of the CMS Monte Carlo production environment is that it involves production processing on a large scale while at the same time minimizing the amount of direct human intervention. It has automated subsystems dealing with the following areas: input parameter management, robust and distributed request and production accounting, preparation of executables, management of production resources, local access to mass storage, and distributed file storage and replica management. All Monte Carlo production requests are stored in a reference database (*RefDB* [5]) at CERN. Each request contains all the input parameters needed to create the data. The request is dispatched to Regional Centers by e-mail. A set of scripts (IMPALA [6]) has been developed to automate production job creation and submission for the different steps of the production chain in a Regional Center. BOSS [7] is a system that is able to perform bookkeeping of the relevant information produced by the different types of jobs synchronously with job execution. The summary of job tracking performed by IMPALA using BOSS is sent back to the *RefDB*.

2.1 Description of CMS Monte Carlo production process

The following is a brief description of the programs used in the different steps of CMS Monte Carlo production.

1. Generation of physics channels and simulation of the CMS detector. This step has two sub-steps: the generation of physics events using CMKIN (currently based on PYTHIA, though other generators should be supported) and the simulation of tracking in the detector using CMSIM (a GEANT3 based program). Both CMKIN and CMSIM are statically linked FORTRAN programs. The output of CMKIN is a random access Zebra [8] file (*ntuple*), which is used as input to CMSIM. The output of CMSIM is a sequential access Zebra file (*FZ file*).
2. Transfer of data to an OO database and signal digitisation. This includes the translation of the information in the *FZ files* into an Objectivity/DB [9] database and the simulation of the response of the CMS data acquisition system (digitisation). Both steps are done using ORCA, a dynamically linked C++ program which uses Objectivity/DB as the underlying persistency layer.
3. Reconstruction. At the time being this implies the creation of analysis ntuples or ROOT [10] files from the digitised data using ORCA.

Since Objectivity/DB is not deployed on the EDG testbed, in this Stress Test only the two steps of the first phase were tested.

2.2 Tools for CMS official Monte Carlo productions: RefDB, IMPALA and BOSS

This section contains a brief overview of the production components used within CMS Monte Carlo official production and for the CMS Stress Test. Further details of the architecture of the production machinery can be found in the extensive Spring 2002 DAQ TDR Production note [11].

The RefDB (Reference Database) is used for parameter-tracking purposes and to guarantee a high-level overall coordination, thus greatly reducing the possibilities for operator-error. RefDB consists of an SQL database located at CERN which allows the users both to register their production requests through a web interface and - when requests are given to a production centre as an "assignment" - to get all the relevant information to be given as input to the IMPALA production component. The RefDB also keeps the summary information sent back by a Regional Centre about the status of each assignment. Communication with the RefDB is done using HTTP.

Intelligent Monte Carlo Processing And Local Activator (IMPALA) is a set of scripts that accepts a RefDB assignment as input and contacts the central database to retrieve all required parameters. It creates the appropriate scripts to submit the jobs to a scheduler and provides a system to trace the status and progress of the job submission process.

Batch Object Submission System (BOSS) has been developed to provide real-time monitoring and bookkeeping of jobs submitted to a compute farm system. The information is persistently stored in a relational database (MySQL in the current version) for further processing. BOSS extracts the specific job information to be monitored from the standard input, output and error of the job itself and stores it in the database. The user can describe the parameters to be monitored by BOSS and how to access them by defining a job type. A “job type” is defined by a “schema”, which describes the parameters to be monitored and a set of executables run with the user job containing the algorithms to determine the values of the parameters from the standard input, output and error of the job. CMS specific BOSS schemas and monitoring scripts are part of the IMPALA software. BOSS interfaces to a scheduler through a set of scripts provided by the farm administrator, using a predefined syntax. The scheduler job identification string is kept in the database together with the BOSS job identification number so that it is always possible to reach the running job via the local batch system through BOSS commands. Sample interface scripts to the EDG scheduler are included in BOSS.

2.3 Job resource requirements and use of the results

The jobs used for the Stress Test were part of an official CMS effort aimed at producing enough (of the order of 10 million in total) simulated events for particular physics studies (ECAL in-situ calibration using Level1 triggered jets) of the CMS detector’s response.

As described above, two kinds of jobs have to be run to perform the first step of the CMS Monte Carlo production process: CMKIN-like jobs and CMSIM-like ones.

The CMKIN program requirements are:

- Some control “cards” as input to define the physics process to be simulated;
- A CPU time of the order of half a second (on a ~1GHz PIII CPU) to generate a single event;
- A permanent output (on disk or MSS) of the order of 50KB/event generated.

The CMKIN step is therefore a “fast” step with an enhanced requirement on the I/O (output only).

The CMSIM program requirements are:

- The input file produced by a corresponding CMKIN job (and some additional input information “cards”);
- A CPU time of the order of 350 seconds (on a ~1GHz PIII CPU) to generate a single event;
- A permanent output (on disk or MSS) of the order of 1.8 MB/event.

The CMSIM step is therefore mostly CPU bound, even if a non-negligible I/O is needed.

Although CMS also supports SUN/Solaris, the resources used in this test only involved machines running Linux.

3 European DataGrid environment

The EDG is a 3-year EU-funded project that started in January 2001. Its aim is to set up a computational and data-intensive grid of resources for the analysis of data from scientific exploration. Its mandates are:

- develop open source middleware for fabric and Grid management;
- deploy a large-scale production-quality testbed using real-world applications with real data;
- collaborate with and complement other European and US projects as well as industries.

This section describes the system that has been deployed on the EDG testbed for the CMS test.

3.1 Architecture:

The architecture of the EDG is described in [12]. Only a brief description of the relevant concepts will be given here - it is by no means meant to be rigorous and complete, but just to provide the reader with a very basic knowledge of the Grid environment and functionality, as implemented in the EDG testbed.

3.1.1 Information Service (MDS)

The Grid Information Service collects and organizes coherently all the information about the Grid resources, including their characteristics and their status. This information is then available both for monitoring purposes and, most importantly, to the Resource Broker (RB) for the choice of the computing resources where to execute the jobs. The present implementation is based on the Globus MDS [13], and has a hierarchical structure: every Grid element publishes through its Grid resource information service (GRIS) a certain amount of information, and is listed in a Grid information index service (GIIS), which may also be listed in another GIIS, until a top level GIIS is reached.

3.1.2 Resource Broker (RB) and Information Index (II)

The RB is the heart of the workload management system (WMS). Its function is to interpret the user requirements specified using the job description language (JDL), match them against the available resources and select the most appropriate for the job, following criteria possibly specified by the user. The II is the interface to the information service, and in the present implementation is a cached copy of the content of the top GIIS.

3.1.3 Replica Manager (RM) and Replica Catalog (RC)

The RM deals with data storage, access and replication. It allows users to perform such tasks as to upload a file to the Grid, to create replicas of a file, to select the “best” replica of a file (for example, the one for which the access is fastest), and to delete files or specific replicas. A file is said to be available to the Grid when it is registered in the RC, which holds the list of all the files and their replicas contained in every storage element in the Grid. Files registered in the RC are grouped into different logical collections. The RC was implemented with a LDAP database.

3.1.4 Virtual Organization (VO) Server

In the Grid, a virtual organization is a collection of users who share the same access privileges to some set of Grid resources, and who typically belong to the same experiment, or to the same working group, etc. For example, in the EDG Testbed there was a VO for each LHC experiment. From a practical point of view, every VO has an LDAP server that contains the names of all the people in the VO and their certificate subjects. Also groups can be created inside a VO, if necessary. When, for example, a particular Grid resource is made available to a given VO (e.g. CMS), the certificate subjects of the VO members are retrieved from the VO server and automatically added to the local list of authorized users (the "grid map file").

3.1.5 Computing Element (CE)

A CE is a node, or a cluster that is known to the Grid information system and capable of accepting jobs dispatched by the WMS. The actual, physical nodes forming the cluster are called "worker nodes" (WN). The CE is operated by the Globus Resource Allocation Manager (GRAM), which interfaces the WMS to the local job scheduler (e.g. PBS, LFS, etc.). The GRIS of the CE takes care of publishing such information as the number of CPU (total and free), the number of running or idle jobs, the operating system and the software installed (even specific to a VO).

3.1.6 Storage Element (SE)

A SE is a file server interfaced to the Grid. It contains a "grid map file" with the list of the users allowed to write files into it. In the implementation used, the access control was very basic; the only distinction being based on the VO a user belongs to. Every VO authorized at a SE had a dedicated directory, called "storage directory", under which the files belonging to that VO could be written. The information regarding a SE, like the total disk space, the free space, the name of the storage directory of a given VO, etc. were published, as usual, by the GRIS. A SE may also serve as an interface to a mass storage system.

3.1.7 User Interface (UI)

The UI is just a computer (possibly a laptop) that has installed the commands and the libraries that the user needs to interact with the Grid. Typical examples are the commands used to create a proxy certificate, submit a job, query the status of a job, or delete a job, and the commands to transfer files to and from a SE, or to list all the replicas of a given file.

3.2 The "EDG-Application" Testbed

The testbeds [4] are key elements of the EDG Project as they provide the verification of usability of the Grid middleware by many users.

Many evolving and dynamically configured testbeds were foreseen and deployed. "Integration Testbed" is in charge of providing verification of correctness and functionality of the interfaces between the software objects delivered by the different middleware Work Packages; "Validation Testbed" is in charge of allowing different sites to join the "system"; "Dedicated Work Package testbeds" allow the developers to understand new functionality, compatibilities, etc. The "Application Testbed" is the one dedicated to prove the readiness for use by the current Applications' software that participates to the EDG Project. This last testbed is therefore the only one described in this context and of interest for the CMS Stress Test activity.

3.2.1 EDG Application Testbed Sites

Each main partner of the EDG project was committed to deploy a main site for tests. Sites were requested to deploy and dynamically maintain all the services described above (3.1). A considerable number of people had to be dedicated to this task, and the availability of competent personnel had to be guaranteed.

In the following we briefly list the sites and the main services provided by them, without aiming to be complete, but focusing on the "normal" services provided for the EDG Project (ad-hoc services for the CMS Stress Test or additional CMS used resources, as they were programmed or shown to be needed, are described later on).

The official main sites for the EDG Application testbed were: CERN/Geneva (CH) [3], CNAF/Bologna (IT) [14], CC-IN2P3/Lyon (FR) [15], NIKHEF/Amsterdam (NL) [16], and RAL/Oxford (UK) [17].

CERN maintained the principal RBs for the Application testbed, the "top" MDS server, at least one CE (with many WNs, ~120 CPUs) and at least one SE (with also access to the Castor MSS [18]). Some other services (not used by CMS) were also maintained for other applications (e.g. RCs and "main" SEs).

CNAF maintained the principal RC for CMS, another general-use RB, at least one CE (~18 CPUs) and at least one SE. Some other key services were also maintained for other applications and general service/backup (e.g. VO server for Atlas, RC for other Applications, backup MDS server, etc.).

CC-IN2P3 maintained the "reference" distribution of CMS software, the EDG reference software distribution, at least one CE (with many WNs, ~520 CPUs), and at least one SE (with also access to the HPSS MSS [19]).

NIKHEF maintained the unique CMS-VO server, at least a CE (~24 CPUs), and at least one SE. Other services for different applications were also maintained (e.g. Earth Observation).

RAL maintained at least one CE (~20 CPUs), and at least one SE. Other services for different applications were also maintained (e.g. Atlas servers).

The user authorization server for the EDG testbed was maintained at Marseille as one of the tasks of EDG WP6 [4].

Also worth noting is the local batch scheduler implementations, as some of the problems encountered were also related to the local scheduler software (a different local scheduler was also tried during the Stress Test using special CMS resources added to the testbed, as reported afterwards):

CERN used the PBS [20] local scheduler (with several different queues)

CNAF used the PBS local scheduler (with several different queues)

CC-IN2P3 used the BQS [21] local scheduler (with many different queues, also shared with non-grid Applications)

NIKHEF used the PBS local scheduler (with several different queues)

RAL used the PBS local scheduler (with several different queues)

3.2.2 Procedures and methods of coordination

Many procedures were adopted to maintain consistency in the testbed. We only discuss here the ones used during the CMS Stress Test.

The EDG Integration Team (EDG-IT) played an important role in testing patches and making bug corrections in a very short timescale. A close connection with the CMS "stressers" gave the ability to promptly decide on new implementations and to identify the correct suite of RPMs to be used.

The EDG weekly WPs Manager meeting gave the ability to decide strategies, timings and required support to perform the Stress Test.

The LCFG [22] team also had an important role to produce immediately the necessary list of "objects" to be installed at the sites with automated procedures.

The "site-administrators" mailing list was a valuable method of keeping the different sites up-to-date and aligned with the use of short advice messages.

The set-up of an EDG/CMS "CVS" [23] repository at CERN turned out to be valuable in achieving consistency among the different EDG/CMS software environments on the UIs. The availability of such a repository and its extensive use during the Stress Test guaranteed both an effective sharing and a fast deployment of the frequently modified code.

EDG WP8 group played an important role in coordinating access to the testbed among the different applications and the other activities that were taking place concurrently (for example demos and tutorials).

While many of the procedures (including the official EDG bug reporting and solving line, Bugzilla [24]) implemented by EDG people turned out to be very useful, there was a clear indication that some additional methods for coordinating the work were also needed. In particular identifying the right person at the right moment to solve a potential show-stopping problem was possible only through personal contacts.

4 Description of the test

4.1 Scope and goals

The test consisted of running several CMS production assignments on a few virtual Regional Centers. More details about CMS productions may be found in [11]. It is worth stressing that from the CMS point of view this test used all of the components of the official CMS production environment. The modifications that were needed to interface to the EDG middleware did not compromise the functionality of any of the basic components such as the job creation, job monitoring and bookkeeping mechanisms. This granted that the quality of the data produced was at the same level of that of the official CMS productions.

There were three main goals involved in this test:

- Verification of the portability of the CMS production environment into a Grid environment;
- Verification of the robustness of the EDG middleware in a production environment;
- Production of data for physics studies of CMS, possibly at the level of 1 million simulated events, which corresponds to (assuming 250 events per job):
    - 8000 jobs (4000 CMKIN and 4000 CMSIM) submitted to the system;
    - 8000 files created and registered to the Replica Management System (4000 ntuples and 4000 FZ files);
    - About 11 CPU-years on a PIII 1GHz CPU;
    - About 1.8 TB of data in FZ files.

The verification has been achieved by measuring efficiencies, classifying job crash reasons and identifying bottlenecks. In fact several problems were identified and fixed during the test. Nevertheless only a quarter of the foreseen one million events were produced. Detailed statistics about the data really produced may be found in the following sections.

4.2 Description of the workflow

The test was limited to the execution of the first two steps in the CMS production chain, i.e. CMKIN and CMSIM, thus the deliverables of the test were the CMSIM FZ files. "Ntuple only" (MC_Runjob in CMS jargon) current CMS Production was discarded because only about 20%-30% of the total number of sites in the testbed had the Objectivity/DB license. The solution of running the non-ORCA steps on sites that didn't have the Objectivity/DB license and the ORCA steps on those that had the license was also discarded, because the *added-value* to the test would have been small compared with the effort of integrating the ORCA steps in the Grid environment, given the fact that Objectivity/DB has been recently dismissed by CMS. On the other hand, running the CMKIN and CMSIM programs in two steps gave the opportunity to test the automatic input data location features of the EDG software.

The basic flow of the test was:

- IMPALA declaration of CMKIN jobs;
- IMPALA creation of CMKIN jobs;
- Submission of CMKIN jobs;
    - The output CMKIN ntuple is saved on the SE close to the CE where the job runs (*close SE*) or, if this is not present, to a default SE, and registered using the Replica Management System;
    - The Logical File Name of the ntuple is also recorded in the BOSS-DB;
- IMPALA declaration of CMSIM jobs; discovery of the input ntuples is done through the BOSS-DB;
- IMPALA creation of CMSIM jobs;
- Submission of CMSIM jobs;
    - The output CMSIM FZ file is either stored in the *close SE* or on a predefined SE (see below). In any case it is registered to the RC;
    - The Logical File Name of the FZ file is also recorded in the BOSS-DB;
- The Logical File Names of FZ files for successful CMSIM jobs are retrieved from the BOSS-DB. The files are replicated to the final destination, if they're not already there.

4.3 Job preparation and submission

Submission of both CMKIN and CMSIM jobs was done through the BOSS interface. A few more job types were defined in the BOSS system in order to monitor EDG specific information.

The job submitting sites (Regional Centers in the terminology of CMS Production) were constituted by a UI on which IMPALA/BOSS was running, and by a MySQL server where the BOSS database was hosted. The jobs created by IMPALA were submitted to the Grid, i.e. the EDG official test bed. The reason for which it was decided to have several UIs instead of a single one was that we thought, from the experience of the official CMS productions, that the BOSS MySQL server current implementation couldn't scale to the size we were going to test. In addition, the use of different UIs would make best use of the available human resources and also test a distributed submitting environment.

To overcome the intrinsic limitation of the Job Submission System to manage up to 512 simultaneous jobs from a single RB[2], each UI submitted to a dedicated RB. Each RB had its own EDG "Logging and Bookkeeping" service. II services were also set up.

Update of the RefDB was done at the UI sites, to keep track of the progress as in a normal CMS Production.

4.4 Data management

A single RC was used so as to provide a single source of information about the files produced in the whole system.

Several solutions for the storage of the output FZ files were tested. Some of the jobs were built to store the FZ file in the most convenient place at runtime (i.e. the *closeSE*), others to send the output to the final location (a SE in one of the Regional Centers that will perform the ORCA steps).

In order to avoid filling up the disk space in the generic SE, a job performing the following operations was planned to be executed periodically on a dedicated UI:

1. Find out from the BOSS databases which FZ files have been created at any SE by the Production;
2. Replicate these FZ files to the CERN SE;
3. Delete these FZ files from the SE of origin;
4. Add the information about the file replication and deletion operations in the BOSS.

Since some of the SEs had disk-based storage while others had Mass Storage System based storage, we could test different kinds of file transfers:

- Disk to disk replication;
- Disk to MSS replication;
- MSS to MSS replication.

Two different MSS configurations were used during the test. At CERN, the SE close to the CERN CE was not directly interfaced to the Castor MSS; instead a second SE interfaced to Castor was used. Files produced at CERN were not directly stored in the MSS, but had to be replicated from the SE close to the CE to the Castor enabled SE, thus testing disk to MSS replication. Only a few users were allowed to write into Castor for security and organizational reasons.

At Lyon, the HPSS was directly interfaced to the SE close to the CE. Files produced there were automatically migrated to HPSS, thus testing the disk to MSS interface. This required a dedicated UI configuration and it was decided that only Poly UI (Ecole Polytechnique) would run production in this mode. Moreover, as the data produced at Lyon were finally copied to the CERN SE Castor enabled, a MSS to MSS replication was also tested (even if the actual copy was from disk-cache to disk-cache that sit in front of the tape archive systems).

4.5 System monitoring

The sources of information were:

- The BOSS database;
- The Logging & Bookkeeping service of EDG;
- The Replica Management System of EDG;

[2] The limitation to 512 simultaneous jobs for a single Resource Broker was given by the limitation to 512 simultaneous users in Condor-G. The problem was already known by WP1 people and by the Condor developers, so the Condor Team provided a fix. However it was decided to postpone the deployment of this patch since it was not considered a showstopper problem.

- The Information Service of EDG (MDS).

While the BOSS, the Logging & Bookkeeping and the Replica Management information are persistent, the MDS information is volatile and needed to be archived for later analysis. A dedicated program was developed to save the MDS information. The program was running on a UI (a second instance of the program was running on a second UI for backup) and the information was saved on flat files. The *mysqldump* of the BOSS databases for all the UIs and the MDS flat files data were converted into ROOT [10] trees and used for the analysis. More details about the ROOT trees production system may be found later.

The overall picture of the testbed status was provided by Nagios [25], a web-based tool developed by the EU DataTAG Project to collect and display graphically the information on grid resources. More information on Nagios may be found in section 5.1.4.

5 Stress Test preparation

In order to perform the Stress Test a number of activities were needed in advance. These activities were both technical and organizational.

The CMS production software had to be adapted to the EDG provided middleware tools and some accessory tools (scripts) were created to aid the job submission and control processes.

The resources and the services to access them via the EDG Middleware had to be deployed. This preparation activity also included the identification of additional computational resources.

The operation of the Stress Test had to be organized in advance. Contact people were identified and the organization of work and meetings were planned in advance.

Finally a number of (mainly) technical choices had to be made in advance, so as to be prepared for the Stress Test start.

5.1 Software deployment

5.1.1 Modifications to BOSS

In this section the modifications to BOSS 3.0, which is the last version used for official CMS productions, are described. More information about BOSS may be found in [7]. This version was already able to manage jobs submitted to the EDG, relying on a MySQL server to access the information of the job from the worker nodes and to update it.

A problem that was experienced with BOSS during official CMS productions was related to the number of connections to the MySQL server in big Regional Centers (above ~100 CPU's). To overcome this problem the real-time flow of BOSS has been modified so that connections to the server are done at most every 30 seconds instead of for each new update. In case of multiple updates of the same variable in this time interval, only the last one was sent to the server.

The main problem in the use of BOSS in a distributed environment is related to the fact that it relies on the MySQL daemon on the server for updates. Thus the system is not fault tolerant against network or server interruptions. Two main modifications were made to BOSS. First of all the execution of the user job and the filtering of the standard input/output/error of the job are decoupled from the updating of the BOSS database. The execution can continue on the worker node even if the process that updates the database crashes. All the updates are recorded in a journal file that is shipped back to the user via the *EDG output sandbox* for off-line update of the BOSS DB. The second modification is that all the information needed by the job to run on the worker node is not retrieved from the MySQL server but shipped to the job using the *EDG input sandbox* so that the job can start on the WN even if the connection to the server is down. The long-term solution to this problem is the introduction of a fault tolerant layer between the running job and the database. A prototype that uses R-GMA [26] is being developed but we were not able to deploy it in this test.

The scripts to interface BOSS to the EDG scheduler have been modified to support multiple configuration files. This allowed easy switching of RB in case of unavailability of one of them.

New job types were registered to BOSS to manage EDG related information. Details may be found in the next sub-section.

5.1.2 Modifications to IMPALA

The modifications to IMPALA (version 3.3.1) to allow CMS production jobs to run within EDG system are described in this section. More details about the "EDG-enabled" IMPALA version can be found in [27].

5.1.2.1 Job preparation and submission

The modified IMPALA software [28] was installed on each UI, where the job preparation and job submission took place. IMPALA creates the job scripts that then need to be submitted to a job scheduler.

In order to set-up the CMS environment appropriate to run the job, IMPALA prepared the job relying on VO-specific scripts that had to be available in the WN to determine where the CMS software and environment scripts are.

For each job a JDL file was prepared to manage job submission to EDG. A tar file with all the scripts and parameters needed by the running job were sent through the *input sandbox*. It's worth mentioning that the IMPALA created jobs specified the requirements and thus defined the JDL content. This turned out to be quite inflexible during the test, as it forced the UI submitter to either modify the JDL by hand or create small bunches of JDL jobs to better match the status of the testbed.

The configuration file of the CMS Replica Catalog used by the replica-manager commands to register the output data was also sent in the *input sandbox*. The log files and the BOSS journal file were shipped back via the *output sandbox*. The requirements in the JDL could be customized, although as a default the only requirement was that the CMS software used for the Stress Test was installed at the execution site. For CMSIM jobs the additional request of CPU time above a given threshold was made based on the typical processing time of these jobs.

The ntuples produced in the CMKIN step had to be processed during the CMSIM step. In order to declare CMSIM jobs a "soft" discovery of the ntuple files that have been produced was performed by looking for their Logical File Names (LFN) stored in the BOSS DB. Using the BOSS DB allowed a CMSIM job to be submitted for each successfully completed CMKIN job. "Soft" discovery is better suited to a grid environment than a "hard" discovery that leads to site-specific functionality (i.e. "ls -l"). Moreover the "soft" discovery allowed the UI submitter to limit the search to their own created files, as the RC was common to the whole CMS-VO (although it was possible to use a logical collection per UI). The LFN of the ntuples were used as input data in the creation of the JDL for CMSIM jobs.

Jobs were submitted to the EDG scheduler through the BOSS interface. The registration of the EDG scheduler in BOSS allowed switching of the RB to be used. The RB sends jobs to suitable CEs, i.e. matching a given set of requirements or being close to the SE that holds the required input data.

5.1.2.2 Data management

IMPALA was also modified so as to use the EDG data management tools. The transfer of the output data from the WN to the SE and its registration in the RC was done using the EDG RM (edg-replica-manager-copyFile and edg-replica-manager-registerEntry commands). The transfer and registration options were specified using a configuration file sent with the job from the UI. One of the UIs used the edg-replica-manager-copyAndRegisterEntry commands so as to trigger the use of the MSS migration scripts for the sites having an MSS. This was the only specific UI command needed and was due to the limited maturity of the MSS software interface. No other site-specific commands were used in the IMPALA code. The edg-gridftp commands (edg-gridftp-mkdir and edg-gridftp-exists) were used to check or create a new directory structure on the SE from the job script running on the WN. A modification was also included to check/skip the RM copy and registration of the output file if its LFN had already been registered. This prevented cases where the WMS would resubmit jobs.

As previously described, we also tried in the test two strategies for the management of the output data. Two of the four used UIs were sending jobs in which the produced data were stored in the SE close to the CE where the job ran, while jobs submitted from the other two UIs transferred the output data to a specified SE. For the latter case, this operation required a dedicated function in IMPALA to force job transfer and registration to the "defaultSE" specified in the impala configuration file.

The transfer of input files from the SE to the WNs in the CMSIM step was activated by querying the BrokerInfo file and using globus-url-copy to construct the TFN (Transient File Name). The possibility to have some of the SEs not supporting the "file" access protocol was also taken into account. The supported protocols were determined by querying the BrokerInfo file and the "file" access protocol was used when available, while the "gridftp" protocol was used in the other cases.

5.1.2.3 Job tracking

Job tracking was performed using BOSS and a job type (*edg_job*) created. This job type had a schema and monitoring scripts that collected EDG related information:

- LFN of input/output files;
- Status of the input file copy from their SEs to the job working directory;
- SE hosting the output file;
- Output file size and checksum CRC;
- RM copy and registration status of the output file.

Another job type (*edg_transfer*) for CMSIM jobs only has been defined. The corresponding BOSS table is filled from a post-script, in order to book keep the data replication to a destination SE, as described in the following sub-section.

In a Grid environment the directories-based job monitoring in IMPALA, updating a central tracking directory on the UI did not permit job tracking in real time, as a job running on a WN cannot access files on the UI. A job tracking system based on BOSS was developed within IMPALA, as real time job information was available from BOSS. The BOSS interface to EDG job status allowed the determination of the job's status (idle, running, aborted, ended…) and access to the BOSS database allowed the retrieval of job specific information, such as the return code of the application itself or the RM copy and register status, needed to distinguish between a properly finished job and a problematic one (from the CMS point of view). These two kinds of information were used to set the job status in IMPALA, i.e. a job ended for BOSS was considered as properly finished only if its return code was zero and the output file had been successfully copied to a SE and registered into the RC. For each production step of a given dataset it was possible to:

- Check the status of all jobs as retrieved from BOSS based job tracking;
- Retrieve log files via the output sandbox and copy them to the IMPALA logs dir;
- Retrieve the BOSS journal file via the output sandbox and exploit the off-line recovery procedure in case of a failure in the BOSS database update process.

The information concerning the produced data was sent to the RefDB, thus keeping track of the progress of the production. The make-summary scripts to update the RefDB were modified to correctly reflect the production status, picking up only jobs that ended with data successfully copied and registered into the RC. The output file size was stored in the BOSS database by the running job and it was extracted from there, because the assumption of produced data accessible with "ls" or "rfdir" was not valid.

5.1.3 Tools for data replication

Some scripts were written to replicate the FZ files from the SE near to the CE where they were generated to the CERN SE, as outlined in section 4.4. Such FZ files were generated by jobs submitted from the UI at Imperial College and Ecole Polytechnique. In addition, the status and outcome of the replication was written in the BOSS database.

When the main script is run, the BOSS database is searched for jobs that have successfully copied an FZ file in a SE and registered it in the RC, with the FZ file not yet replicated. A maximum of 10 replication processes is allowed to run at the same time from a single SE. The replication processes are started with a rate of one every two minutes to minimize the load on the RC.

When a file replication starts, it first checks that the file is: a) in the source SE and b) registered in the RC. If only condition b) is not true, the file is duly registered in the RC. This operation became necessary when new logical collections were introduced in the RC, in order to replicate files registered in an old collection. After replication, the registration of the new replica in the RC and the destination SE is checked and the file size is also compared to the original one. If any of these checks fail, the RC and the SE are restored to the original status. If the checks are successful the original replica is deleted from the original SE.

5.1.4 Monitoring via CMS tools and EDG tools

Monitoring tasks within the CMS Stress Test can be classified into online and offline monitoring activities.

A set of "cron" scripts regularly collected relevant information (about the CEs and SEs) from the MDS. An extended *boss2root* [29] (see below) was used to import this information in ROOT trees for further processing.

5.1.4.1 Online monitoring

The online monitoring was mainly done using Nagios® [25]. Nagios is a host and service monitor originally designed to diagnose network problems. The monitoring daemon performs periodic checks on declared hosts and services using external plug-ins that return status information to the Nagios service. When problems are encountered, the daemon can send notifications to administrative contacts in a variety of different ways (email, instant message, SMS, etc.). Current status information, historical logs, and reports can all be accessed via a web browser.

Some modification of Nagios was necessary for the Stress Test needs. Nagios has a flexible interface that allows the creation of plug-ins that can add powerful features to the Nagios core. A plug-in represented the various sites and some site specific measurements were represented on a map image that could display a worldwide or specific country view. Different colours were implemented to represent the actual status of the measurements. For example if a site had heavily loaded CPUs, the corresponding icon for that site would turn red on the map. The developed "plug-ins" represented: CPU occupancy, disk load and memory occupancy. A per site collection of data was performed so as to represent the global status of each farm on the map.

Another plug-in on the Nagios server side was able to generate a graphical history of measurements. This gave the ability to graphically display various metrics.

Other variables were also displayed. For example, jobs submitted and status of the CPUs on the testbed. The user could therefore see the jobs that were running or queued and also the total free CPUs, per site.

5.1.4.2 Offline monitoring

Offline monitoring tasks were distinguished into BOSS-based analysis activity and EDG resource monitoring. The EDG monitoring was mostly related to the middleware dynamic check of availability and status of resources and the recording of this information. The BOSS offline monitoring, together with the EDG recorded information, gave the opportunity to develop a special tool to analyse and make a final report on the Stress Tests results.

A Perl/C++ tool called *boss2root* was developed for the CMS Stress Test, to perform analysis on the information saved in the BOSS MySQL databases of each UI involved in the testing activities. The basic idea was to read from the BOSS databases all the information relevant for monitoring purposes, store it in a ROOT tree and then further process this tree to produce user-defined histograms. The reasons for such an approach lay both on BOSS and on ROOT approaches. The contents of the BOSS database can be decided "a-priori", so that it can store as much of the information required for monitoring as possible, with a wide coverage. On the other hand, it was desirable to exploit the pre-existing and flexible analysis tools provided by the ROOT framework. In this respect, the developed package just provided the missing link from BOSS to ROOT.

The monitoring operations with *boss2root* were not restricted to UI "stressers". Although the package was mainly deployed on the four UIs involved in the Stress Test, it could have also been used from any machine (provided it had a MySQL client installed), thus potentially allowing everybody to perform their own CMS Stress Test analysis. This tool was used from the start of the test activities to regularly monitor the content of the UIs' BOSS databases and to perform off-line analysis, finally ending in the production of summary statistics and plots, including the ones shown in this report.

Although the basic purpose of the off-line monitoring was the one outlined above, the *boss2root* framework could be easily extended and adapted for EDG resource monitoring as well. The tool was available from the ramp-up days, 26-27 November 2002.

5.1.5 Accessory tools

In order to help manage the production, a few scripts were developed by each UI submitter, both for job submission and retrieval of information.

Some scripts were used to give a slow submission rate both for CMKIN and CMSIM jobs.

Some other scripts were written to collect statistics of job status and failure reasons from the EDG point of view, extracting the information from the Logging and Bookkeeping Service, and from the CMS point of view, accessing information in the BOSS database.

5.2 Resource deployment

5.2.1 Replica Catalog and top MDS

An empty RC was set up at CNAF before the beginning of the Stress Test with a single logical collection. During the Stress Test, when the responsiveness of the LDAP server became too low, it was decided to create a different logical collection for every UI. Eventually, when this solution proved to be insufficient, a second RC, running on the same host, was set up and used during the last part of the Stress Test.

The top level MDS was set up at CERN. The Italian resources (i.e. Legnaro and Padova) were not directly registered to the top level MDS but were visible through the CNAF MDS. During the upgrade of the system to EDG release 1.4.0, where the work-around to solve the MDS instability was deployed, two Information Indexes (dbII) were set up. The one at CERN used by the CERN and Imperial College RBs and the one at CNAF used by the CNAF RBs.

5.2.2 Resource Brokers and Information Indexes

The II is the Grid component that is most prone to suffer from heavy load and therefore low response times when the amount of concurrent jobs increases. It was decided to use several RBs, each one with its own II, to avoid this possible bottleneck.

Four RBs have been used: one fully dedicated to CMS at CERN (an additional one was used at CERN in some occasions, shared with all the other experiments and used as a backup solution), two at CNAF (one of them reserved for CMS and the other one shared with other Applications), and one at Imperial College. Each UI tended to use a particular RB, but could use a different one in the event of problems occurring.

Table 1 lists the location and names of the machines providing the EDG Services and in particular the UI to RB association.

EDG Service	Used by	Site	Name	Comment
Resource Broker	CNAF UI	INFN/CNAF	grid010g.cnaf.infn.it	CMS Stress Test dedicated
Resource Broker	PD UI	INFN/CNAF	grid004f.cnaf.infn.it	Shared with other Applications
Resource Broker	POLY UI	CERN	lxshare0383.cern.ch	CMS Stress Test dedicated
Resource Broker	IC UI	Imperial College London	gm03.hep.ph.ic.ac.uk	Shared between CMS and BaBar
Resource Broker	All UIs on unavailability of others RBs	CERN	lxshare0380.cern.ch	Shared with other Applications
Replica Catalog	-	INFN/CNAF	grid011g.cnaf.infn.it	CMS dedicated
Top MDS	-	CERN	lxshare0373.cern.ch	Shared with all EDG Testbed
DbII	-	CERN	lxshare0225.cern.ch	Shared with all EDG Testbed
DBII	-	CNAF	dell04.cnaf.infn.it	Used by CNAF RBs
Virtual Organization (VO)	-	NIKHEF	grid-vo.nikhef.nl	CMS dedicated

Table 1 – EDG Server location and names.

5.2.3 User Interfaces

The Stress Test was performed using four UIs. This choice was motivated by: 1) The goal to explore the possibility of distributed submission, 2) The foreseen scalability limitation of the Boss database and 3) The available human resources and sharing of work between them. Consequently a UI was deployed at CNAF, at Ecole Polytechnique, at Imperial College and at Padova, respectively on nodes testbed009.cnaf.infn.it, polgrid1.in2p3.fr, gw26.hep.ph.ic.ac.uk, and grid002.pd.infn.it. For some tests, a UI deployed on the CERN site was also used. In addition, in order to work-around RB scalability limitations, each UI used predominantly one of the four RBs mentioned above.

The deployment of the UIs was done using the UI *rpms* distributed from the EDG release repository [4]. As well as the EDG installation, the CMS production tools IMPALA and BOSS were required at each UI. IMPALA and BOSS general modifications have been described previously. BOSS installation needed the setting up of a *mySQL* server and the configuration of a dedicated SQL DB on each UI node. In order to be write accessible from all the Grid WNs, the configuration used was granting write access from every possible node. IMPALA configuration at each UI necessitated very few modifications, being packed in a single file named "EDG_Defaults.rsc" in which the RB configuration had to be specified declaring the corresponding BOSS scheduler. In case of change of the RB being used, a single line had to be modified. This configuration also defined a UI dependent path from which LFNs and PFNs of output files were built. The monitoring using *boss2root* also necessitated having ROOT software installed on the UI.

5.2.4 Computing Elements and Storage Elements

The deployment of computing and storage resources started using the EDG Testbed resources from the CC-IN2P3, CERN, CNAF, NIKHEF and RAL sites.

CMS software installation was validated by using the standard CMS software installation and validation procedures used in CMS production, i.e. producing 10 files of 10 events each for both CMKIN and CMSIM steps for a reference sample and comparing the checksum of the produced output data files with the reference ones accessible on the CERN AFS area. In order to validate one site, the RB was forced to schedule execution on that site. Each UI was assigned to validate one of the main sites, with Ecole Polytechnique UI validating the 5th site (NIKHEF). It is worth mentioning that CMS software installation at NIKHEF, which was not a CMS site, was easily validated, showing that the procedure for installation of CMS software was simple enough to be executed by non-CMS experts. Once validated, the sites were asked to publish the Stress Test tag in their local information system, so that the corresponding resources would be available for CMS jobs and matching of the CMS JDL's.

One main goal of the test was to assess the ability of new resources to join the testbed and so increase the available resources for CMS production. Four additional CMS sites at Legnaro, Imperial College, Padova and Ecole Polytechnique also deployed the EDG software, in parallel with the main sites and/or after that they were validated and operational. The installation and configuration of EDG CE and SE software on the new sites was checked with the help of the Integration Team and with the collaboration of EDG experts using the edg-site-certification test suite, provided by the Integration Team [30]. The CMS software installation at each site was checked using the validation procedure described earlier. The installation and special configuration of the EDG software was an onerous task for the sites involved but nevertheless they managed to cope with a change of release in the middle of the test in about three days.

The resources finally deployed and used for the Stress Test are summarized in Table 2

Site	CE	Number of CPUs	SE	Disk Space (GB)
CERN	lxshare0227	122	lxshare0393 lxshare0384	100 1000(=100*10)*
CNAF	testbed008	40$^+$	grid007g	1000*
RAL	gppce05	16	gppse05	360
NIKHEF	tbn09	22	tbn03	35
LYON	ccgridli03 ccgridli08	120$^\wedge$ 400$^\wedge$	ccgridli07	200
Legnaro*	cmsgrid001	50	cmsgrid002	513(+513)
Padova*	grid001	12	grid005	680
Ecole Polytechnique*	polgrid1	4	polgrid2	220
Imperial College*	gw39	16	fb00	450

Table 2 – Computing and storage resources used during the Test.
$^+$*Some of the CPUs (~20) were added from CMS dedicated resources of the CNAF-Tier1.*
**CMS resources added to the EDG Testbed.*
$^\wedge$*These resources were shared with all French NUHEP ongoing productions.*

5.3 The CMS/EDG Task Force

The planned test (see chapter 4) required a careful coordination of human resources and needed deep knowledge of various fields. As well as the already established methods and procedures within EDG, there was a recognized need for a larger and more cohesive operational structure: a Task Force for the Stress Test.

A similar structure had already been set-up during a previously performed Test by the ATLAS Experiment (mid 2002), and proved to be a very good means to attack (and eventually solve) the encountered problems [31].

A joint Task Force between CMS and EDG was therefore established to perform the Stress Test.

5.3.1 Composition of the CMS/EDG Task Force

The composition of the Task Force was chosen taking into account both the person-availability during the planned period and the necessary skills on key issues (see Appendix A), and included:

- The UIs responsible persons that were already experts in CMS-production procedures and tools.
- The participating sites' responsibles (both for EDG official sites and for CMS added-on sites) were chosen among the empowered people at the corresponding site (EDG WP6). Most of the participating sites to the Test were also distinguished sites for CMS traditional production, thus granting local CMS specific knowledge.
- The experts of the many CMS production tools were asked to be part of the Task Force. (BOSS, RefDB, IMPALA, etc.).
- The key persons of CMS Grid effort and Production effort were also involved in the Task Force. (CMS GRID and Production Teams).
- Most of the CMS persons involved in the Test were also already aware, if not experts, in the EDG Grid environment and procedures.
- The key persons of the relevant EDG middleware were identified and agreed to be part of the Task Force. In particular the WP1, WP2, WP3 and WP4 contacts (mostly the WP managers) were part of the effort.
- The EDG WP8 Experiment Independent Persons ("Loose Cannons") were part of the Task Force to contribute their invaluable experience and effort. The "liaison" with WP8 was also a key element of the composition of the Task Force.
- The EDG main experts of packaging and site set-ups agreed to be promptly available for upgrades or problem solving (EDG Integration Team).
- The technical managers of the EDG Project (at CERN) and the main sites managers (where the most important Grid services resided, namely CERN, CNAF, RAL and NIKHEF) agreed to be part of the Task Force.
- Finally, the management of the EDG Project was also involved.

5.3.2 Task Force and pre-Stress-Test activities

A detailed plan of activities was developed before the actual Stress Test deployment [32]. The plan included a deep discussion of technical details and the identification of the goals. Also a careful evaluation of risks and an analysis of possible successes were performed.

All the possible scenarios of running environment of the Test, including the evaluation of possible fallback positions, were discussed and planned. This work allowed advanced knowledge, whenever possible and foreseeable, of how to proceed and which actions should be undertaken (or from whom to require urgent technical help).

In order to assess the detailed plan and to boost the process, a two-day working session was organized at CNAF and CERN, with video connection between the two sites as well as with other participants, before the start of the real Stress Test. Its goal was to set-up a smooth ramping-up of the CMS Stress Test. Most members of the Task Force attended it; many others participated remotely, and were available on-call in the event of any problem arising that would require their help. Each day of the work sessions was divided into a short plenary-like meeting in the morning (setting the plan of work for that day, agreeing on the activities and subsequent check points), several long parallel working sessions throughout the day (concerning job submission, BOSS-based monitoring, EDG resource monitoring, etc.) and a wrap-up session in the late afternoon. The experience gained through this brainstorming between the many competent people (both from EDG and CMS) working together in a single room allowed a clear plan to be made for the Stress Test start-up over a well-defined timescale. Furthermore the start-up of the four UIs and the validation of the first testbed sites were completed.

Another activity worth mentioning was the participation of CMS people in the EDG-organized Tutorials, in order for them to gain a deeper knowledge of the delivered EDG Middleware.

5.3.3 Operation and meetings

Operation of the Task Force (and therefore of the Stress Test) required close collaboration with the established method of operating of the EDG Project and with the established CMS Production procedure. However it is worth mentioning a few key points:

- The small size and composition of the Task Force ensured good communication and efficient problem solving.
- Meetings of the Task Force were called weekly. The meetings were focused on current technical issues and on the decision taking process. Besides the specific meetings, there were also other coordination meetings attended by many components of the Task Force during the period, for example the EDG WPs meetings, the CMS meetings, the EDG WP8 meetings, etc.
- An "ad-hoc" mailing list was created (cms-stress-test@cern.ch). The list was "open", so as to allow a broader audience than the Task Force itself. The mailing list became much more important than foreseen as it proved to be a very quick means to communicate widely and to ask for suggestion/operation/problem-solving responses. For example, in a time period of about four weeks there were more than a thousand mails of which at least half of them were of importance and contained technical suggestions. All the mails making a request were answered.
- A Web page for fast communications and mail archive was also created [32]
- The distributed nature of the Test and the distributed knowledge of the different components proved to be a key strength for the Test. There was no "central" driving-point for the day-to-day activities, but the corresponding distributed experts naturally drove the daily activities.

The Task Force composition and operation allowed for rapid decision making, driven by expert people on technically effective grounds.

5.4 Summary of pre-Test decisions

Before the actual start of the Stress Test (and awaiting the delivery of the EDG version 1.3 on the Testbed) a number of choices were made, to ease and better control the Stress Test itself.

The decisions taken were described above in some detail, what follows is a summary of the final choices:

- The use of four different UIs geographically distributed within the EU.
- The use of an "allocated" RB for each UI. The RB was either CMS-dedicated or EDG-Testbed shared. The possibility to switch to a different RB (including the "spare" CERN one) was always possible, and proved to be useful.
- Re-submission feature of the EDG-RB was enabled only for CMKIN jobs, being disabled for CMSIM jobs. The last choice was dictated mainly by the long duration (~12 hours) of the CMSIM jobs and by their larger complexity (read existing ntuples via RC, check BOSS/DB, write on selected SEs, etc.).
- The default RB ranking to choose the CE for load balancing was adopted on the basis of the *EstimatedTraversalTime* published by each CE.
- The CEs were identified and classified for use by the different UIs' submitted jobs. Firstly each UI was charged to validate specific sites (official EDG Testbed site or added CMS side). Secondly a potential plan for the distributed use of the CE resources was agreed by the responsible of each UI in case it was required (for example if the RBs were not able to correctly distribute the jobs).
- The SEs were identified for use by the different UIs, and a strategy was defined for each UI. Two UIs forced the final results of the simulation (CMSIM output) to a given SE (Legnaro and CNAF SEs), no matter which CE the jobs were finally run on, and the two other UIs used the "close SE" to the CE where the jobs were finally run. The produced final files of the simulation of the latter two UIs were then replicated to a given SE destination (CERN)

"offline" (via the RM tools).

- Two SEs were designated to also use the MSS storage systems at two different sites (CERN and CC-Lyon), both for jobs which happened to run at the "near-to-the-MSS-SE" CE and for the "offline" replication of the files produced in the EU Testbed distributed sites.
- The choice to use a unique RC was dictated by the latest delivered middleware and by the uniqueness of the "simulation assignment" for the CMS event production ("Dataset" production). Producing the same "type" of simulated data (Dataset) from different submitting sites and to different distributed resources requires using a single Catalog for file name archiving and retrieval. A different scheme (and flexibility) for CMS production assignments (and the following analyses) and a different architectural approach for the RC of the EDG Grid, could have allowed for better shared solutions of RC between the UIs and CEs.
- The BOSS scheme of DB information had to be frozen before the start of the Stress Test (as for any relational DB). Identification of all the possible information had to be made from both the EDG middleware provided tools and from CMS internal checks, in order to be able to perform a careful analysis of the results during and after the end of the Stress Test.
- A smooth and controlled plan for the introduction of sites and resources was developed, in order to avoid possible conflicts of bug introduction and debugging activities.
- The CMS-VO members allowed to access the Stress Test were identified and agreed. Moreover the access to every resource in the Testbed was defined using the map of user certificates as appropriate (CMS dedicated resources were only allowed to be accessed by CMS VO users, while all the shared resources were open to all the EDG Testbed users belonging to any VO).
- Finally a definitive choice had to be taken from the beginning of the Stress Test for the number of events to be simulated by each submitted job. The "normal" number of events per job during the "traditional" CMS Production of this kind of DataSet was 250 events, corresponding to a single CMSIM job running for about 24 hours. It was decided that it was safer to reduce the mean execution time of the jobs when submitted over the EDG Testbed, resulting in a choice of 125 events per job and therefore to an execution time of about 12 hours per CMSIM job.

6 Phenomenology of problems

This chapter describes the phenomenology of the problems we found during the Stress Test and the way that they were solved or overcome.

The discussion of them is a valuable hint both for the EDG Project and for CMS software development. In the end most of the problems were solved with a reasonable and acceptable solution, even if it required a lot of effort and caused some limitation in the efficiency and manageability of the Test.

No criticism is implied in what is described below; all the involved persons were aware that the EDG middleware was a middle-time deliverable of the EDG Project and also that the CMS software was what was currently available.

6.1 Detailed description of problems and work-around

Some of the major issues discovered during the Stress Test are described in more detail in the following subsections, including suggestions for possible evolution of the middleware together with the adopted work-around.

6.1.1 Problems related to the Information System

The top level MDS was highly unstable and several MDS restarts were needed. The problem was due to too many accesses: the top MDS and the II slowed down dramatically once the query rate increased above a certain level and eventually hung indefinitely. Since the RB relies on the II to discover available resources, the MDS instability caused jobs to abort due to lack of matching resources.

In order to reduce the effect of the problem, a lower rate of launched jobs in the job submission process was adopted, in particular for CMSIM jobs where the RB matchmaking is more complex, since it requires the querying of both the II and the RC.

To alleviate the problem, after a suggestion from the Globus team, the thread limit for OpenLDAP was increased to 512 on the II while a work-around proposed by EDG people was being tested. The deployed solution was to replace the II with a simple OpenLDAP server connected to a BerkeleyDB back-end, the so-called dbII. A new release of EDG software (1.4.0) adopting this solution and including some minor bug fixes from EDG-WP1 (WMS work package) was deployed on the Application Testbed.

With EDG 1.4.0 the situation was more stable and the rate of aborted jobs due to information system problems was highly reduced. Problems with the dbII showed up when one of the GRISes hung, thus causing the II to hang as well. In this case the RB's matchmaking fails and all jobs abort. The work-around is to remove the offending GRIS from MDS and "clean up" the dbII.

The problem of stuck GRISes and site GIISes seems to be the same problem observed on the top level MDS, for which further investigations with Globus people are needed.

6.1.2 Problems related to Replica Catalog limitations

The RC implementation on the EDG released software displayed some limitations when coupled with the requirements of CMS Production (or a typical HEP simulation programme). While in the design of the middleware it was possible to have many RCs used by the same VO, this multiple RC approach did not fit the simulation of many HEP events of the same Physical process performed from different sites. This limitation of the current CMS job-generation process forced the use of the same and unique RC. Clearly the "single point of failure" problem could arise.

The only possibility was to keep the RC under close attention and monitoring, keeping the monitoring queries at a level that did not augmented the risk of RC overload.

Too many concurrent jobs writing into the RC overload the LDAP server, thus slowing down its performances and eventually causing it to stick. A side effect of having concurrent jobs is the increased failure rate and odd behaviour of RM commands. Having the RC stuck appears to be a very difficult situation to deal with, the jobs resulting running forever and thus requiring manual intervention from the local system administrators to remedy the situation.

Another limitation, though this time not structural, was found during the Stress Test. Even though the RC capability of registering a large number of entries (more than tens of thousands) was tested and proved before the EDG delivery, we faced a limitation due to the number of lengthy named entries. Logical file names (LFN) to be registered in the RC associated with the physical file names (PFN) happened to be long strings of characters, thus causing the premature saturation of the capabilities of the RC. LFN and PFN are composed with combinations of network name of hosts, name of the directories where the files reside and by the actual name of the files (in turn a combination of CMS labels of the physics process to be simulated, of the running Regional Centre (UIs in this Test) and of the ID of the jobs). The limit was hit at the level of about 2000 Entries.

A work-around for this limitation was adopted creating different RC entries for the different UIs under the same RC. As a consequence the UIs had to modify their CMS tools for submission of jobs (IMPALA) during the Test, producing modified JDL files that take care of the different logical location of their own RC entry. Beside the coordination required to perform that modification (replacing a configuration file in the UI software), this work-around implies a different configuration among the UIs. Clearly this solution has scalability problems.

The EDG WP2 knew of this problem before the Test (even if it became largely critical only during the test, as explained above), and had already planned a more scalable and hierarchical delivery of the RC due for EDG release 2.

6.1.3 Problems related to job submission chain (RB, JSS, GK/PBS)

Several problems at various levels of the job submission chain were found during the Stress Test.

- All jobs stuck in "Ready" status never being scheduled. This behaviour was due to jobs stuck in CondorG queue.
- RB/Logging & Bookkeeping: Jobs in "Done" status never reaching the "OutputReady" status, due to LB Interlogger down. The LB Interlogger was restarted. The debug option was not enabled so it was not possible to understand the reason for the crash. After restarting the Interlogger, the jobs correctly reached the OutputReady status, thus not losing any data/information concerning the job.
- JSS (Job Submission Service) refusing jobs because of a crash of the CondorG *schedd* process due to

use of too low value for the *file-max* and *inode-max* parameters. These parameters were increased to their proper values and the RB restarted

- Jobs aborted because the submission done via CondorG to Globus didn't work (e.g. the gatekeeper was unreachable due to the CE being disconnected because of a network crash) (*Globus down/failed submission*)
- Jobs aborted with reason "*Failure while executing job wrapper*". Mainly two kind of problems were behind this behaviour:
    - The globus-url-copy issued from the WN to the RB node, to download/upload the sandboxes files didn't succeed. This can be due to problems with certificates and/or CRLs (Certificate Revocation List) in the WN, or some other problems in the gridftp client, or to problems with the gridftp server on the RB machine, or network failures, etc.
    - The standard output of the script which wraps around the user job (and which also includes the transfer of the input and output sandboxes) and which is transferred via Globus GASS from the CE node to the RB machine in order to check if the job reached the end successfully, happened to be empty. It was hard to identify the real cause of this problem since there could be many possible reasons, including:
        - Home directory (where the GASS cache resides) not available from a WN (i.e. NFS problems between CE-WN, failing disk)
        - Exhausted resources on the CE head node. The size of the cached standard output file is evaluated by opening the file repeated times, and the transfers are implemented by opening new sockets. If no files can be opened on the CE node, or no free "inode" structure is available, or no non-privileged socket is available, the transfer fails, and therefore the job is lost. Increasing the *sysctl* values for *file-max* and *inode-max* on the CE highly reduce the rate of this failure.
        - Race conditions for file updates between the WN and the CE node when PBS was used was considered too. PBS removes a job from the queue on the server node before requesting the transfer (copy) of standard output and standard error on/from the WNs. The latter is sensitive to temporary load on the WN, the network, the "/home" directory being NFS server mounted and the RB node. Globus determines that a PBS job is done when a periodically executed "qstat-command" reports that the job disappeared from the queue, then it immediately executes a stat() on the standard output and error files, and if these happen to be still empty in the GASS cache at that time, it will remain. A modified version of the "jobmanager", provided by WP1, was deployed in EDG 1.4.1 in order to reduce this effect.
        - Glitches in the "gass_transfer". The final commit of the GASS transfer for standard output and error can occasionally take about 30 minutes and then die. This case could have been dependent on broken connectivity between the CE and the RB node at any time during the lifetime of a job.
        - PBS doesn't remember, through the "qstat" interface, about terminated jobs and Globus considers a job that is not found by "qstat" as a "done" job. This, however, also applies if "qstat" has a transient failure and causes the standard output of an otherwise successful job to be lost. These problem showed up for example on a CE where, probably due to a DNS overload, the "qstat" command failed for all jobs and they were all aborted. A new GRAM PBS script, provided by WP1, was included in the EDG 1.4.1 release.
        - Besides the above-mentioned fixes/work-arounds, in order to reduce the rate of failures, a modification was introduced in the WMS software (the so-called JSS-Maradona), in order to transfer the standard output of job wrappers additionally via *gridftp*, if necessary.

Sometimes there were problems of timed out connections during submission due to Globus FTP failure while sending the input sandbox to the RB. With the slow submission rate adopted within the Stress Test, the effect was negligible for the UIs that had fast network connectivity to their RBs. The effect was reduced but still a potential problem for the UIs having low speed connectivity to their RBs.

A side issue for the job submission chain was also created by the *EstimatedTraversalTime* parameter mechanism. The RB, in order to estimate the load of the CEs and thus balance the job submission on the available resources, uses this parameter, which is published by each CE.

During the Test load balancing the production was a significant problem. Part of this difficulty came from the fact that the CMKIN jobs were very short in duration, of the order or even shorter than the submission rate, so that only one free machine per site could cope with CMKIN job submission in some cases. Moreover, several sites were publishing a value of *zero* for the *EstimatedTraversalTime* parameter, causing jobs to be scheduled mostly at those sites. Finally, since the CMSIM jobs were driven by the input data location, even a correct load balancing of CMKIN jobs would have lead to incorrectly load balanced CMSIM jobs. To solve this issue there should have been the combined requirements of: 1) not too short computing time, 2) reliable MDS information on which to base rank expression, 3) a single step for production, 4) or a way to globally optimize load balancing for several consecutive Production steps.

6.1.4 Other middleware and Testbed problems

Varieties of problems, mostly uncorrelated and with little impact on the results, were also found during the Stress Test. However they caused some inefficiency and had to be either understood or at least tracked before proceeding.

- When accessing a SE to create a new directory to store the output data, the "edg-gridftp" command sometimes failed. This odd behaviour can be due to many reasons and a clear cause could not be identified.
- Some SE appeared to be inaccessible or unavailable for a running job. The problem was traced to be either a hardware failure of the SE itself or a local NFS glitch/failure.
- Problems in the configuration file of one RB, where the default "MyProxyServer" was missing, caused all the jobs submitted to that RB to be aborted.
- Grid-map-file could not be updated because the NIKHEF VO server was disconnected from the network. It was a foreseen possibility, but the update procedure failed to leave the "old" grid-map-file and an "empty file" was produced. Since the machine affected by this problem was a SE, all running jobs didn't succeed to write their output data due to not being able to find the "authorization" match.
- Hardware failures were experienced during the Stress Test, including some network disconnections (as also just described). They had to be traced and recorded for the final analysis. However we had also complex behaviours of some pieces of hardware (namely a home disk in a CE), and the identification of them was not trivial.
- Local WN directories may become messed-up when more than a single job is running on the machine (being mostly dual processors, this was the normal case) or when a previous job has crashed. One of the reasons is that the Stress Test jobs are all of the same kind and the "temporary directory" can accidentally get the same name as another job. Another reason is that when a job crashes, the local temporary files are not always correctly cleared. This was indeed very worrying and was fixed in EDG release 1.4.2.
- The "/home" of a CE can be filled-up by the many jobs simultaneously running there. This potential behaviour was increasingly likely for CEs having many WNs. If this happens (and it did unfortunately happen), all the jobs coming to that CE will fail.

6.1.5 Problems related to CMS production tools

Some of the problems faced with the CMS Production tools during the Stress Test are listed here, however none of them was severe. Some of them were already known as being a limitation of scalability or flexibility (and were taken into account in the planning of the Test), others were discovered to slow the usability of the Testbed during the Stress Test itself.

- The BOSS database schema had to be defined from the beginning (as already said), as once started it would have been very difficult to change. Changes in the entries recorded would imply the creation of a new database with obvious difficulties in matching the "two steps" of the simulation production.

  The BOSS commands that were interfaced to "dg-job-status -all", which retrieve from the Logging & Bookkeeping Service the status of all the jobs, were extremely slow when the number of submitted jobs started to become considerable.

The scalability problem of the BOSS DB was known and the decision of having one BOSS DB per UI was decided in the original plan of the Test to partially overcome this problem. However many DB queries over a large number of entries (order of thousands) proved to be time consuming, ending up in some cases taking several minutes.

The four BOSS databases (one per UI) could have been merged at the end of the Stress Test, thus recovering the uniqueness of the information for the single Physics channel to be studied, however that would have implied the loss of the detailed information on the "two-steps" simulation process (CMSKIN and CMSIM) and on the submitting UI.

- IMPALA CMS Production tool (modified as described above to adhere to the EDG middleware) as was installed at each UI showed some limiting efficiency while submitting jobs. It should be noted that IMPALA has already gone under a major revision within CMS and is now deprecated (MCRunJob has superseded it).

  IMPALA was designed to submit bunches of jobs to simulate a "complete" CMS simulation assignment and therefore is not tuned to submit jobs in "sub-bunches". We were forced to slow down the time frequency of submission of jobs (mainly because of the problems related with the middleware services access) and also to have small bunches (order of tens) of jobs at a time. When processing the IMPALA step of "job creation", it looked through all the already "declared" jobs, including the submitted ones. The status of all the "declared" jobs was checked through a "boss query" that was interfaced to a "dg-job-status -all", retrieving from the Logging & Bookkeeping service the status of all jobs submitted to EDG by the submitting user. When we reached a considerable number of already declared jobs this process took a long and annoying (waste of) time. Moreover IMPALA creates a myriad of directories and sub-directories to keep the jobs in a safe space. This proved to be a problem when trying to trace back failures or the status of a single job.

- Daily management of the UIs turned out to be much more demanding than expected. The UIs were customized for CMS Stress Test purposes and could not benefit from the site system managers' support. The CMS responsible for each UI needed "root" access to relevant machines to properly configure the software. As a consequence, the use of the UI by the submitting person generally required good support, with fast response from the machine administrator. The UIs and the BOSS database sitting on them were single points of failure (as a lost UI or a lost BOSS DB would have caused the loss of all the work done by that UI), and therefore frequent backups and careful continuous attention were needed.

6.2 Summary of problems and adopted solutions

A summary of the problems and possible reasons of them is given in Table 3. The solution and/or the work-around for each problem are also presented with a qualitative estimation of the frequency of occurrence.

Symptom	Cause	Solution	Frequency
No matching resources. (Despite the apparently missing suitable resources matching the specified JDL, the resources were available at that moment)	Information Index stuck because of too many accesses	Solved by dbII in 1.4.0 A slow job submission was adopted for CMSIM jobs, also because they needed more complex queries to II and RC to satisfy “InputData” constraint	Very high before 1.4.0
"dg-job-list-match/submit" hang	Stuck GRISes and GIISes This seems to be related to problem observed on top level MDS ⇒ further investigation with Globus	Work-around: remove offending GRIS from MDS and “cleanup” dbII. A fix in the RB to address the problems when querying a stuck GRIS was implemented and deployed (EDG 1.4.2). Not however solving the basic GRISes problem, but preventing hung of RB.	2-3 times If it happened during a week-end no submission was possible
Connection timed out during submission of jobs	Globus-url-copy failure when transferring input "sandbox" from the UI to the RB machine		Negligible with the slow submission rate adopted within the Stress Test
Generic Failure: MyProxyServer not found in JDL expression	Problems in the RB configuration	Properly configure the RB (rb.conf file)	Vary rare, only once with noticeable effect (~100 aborted jobs).
Globus down/failed submission	Gatekeeper unreachable		Mainly when CERN CE was highly loaded by tutorial activities and when Legnaro was disconnected because of a network crash. Not so often for the rest of the Stress Test.
Cannot download input sandbox	Failure in globus-url-copy issued from WN to RB		Low: Few percent
Condor Failure	CondorG schedd crashing because the file-max and inode-max parameters were too low	Increase file-max and inode-max parameters	Once at CNAF after upgrade to 1.4

Table 3 (Part 1 of 3) - Problems encountered and solution adopted during the Stress Test

Symptom	Cause	Solution	Frequency
Standard Output of job wrapper does not contain useful data: Many possible reasons.	Home dir not available on WN	Identify the cause (i.e. NFS problems between CE-WN, failing disk)	A failing disk with "/home" at CNAF caused most of the jobs running there to be aborted
	Exhausted resources on CE	Increase file-max and inode-max parameters	Hard to detect by the user
	”qstat” failure	New script for PBS in EDG 1.4.1	Hard to detect by the user. Once identified at CNAF and probably due to DNS overload; the ‘qstat’ failed and all jobs were aborted
	Race Conditions for file updates between the WN and CE	"jobmanager" modification in EDG 1.4.1	Hard to detect by the user
	Glitches in the Gass transfer	Job wrapper "stdout" also transferred via Gridftp	Hard to detect by the user
Jobs in “Done” status forever	LB Interlogger went down. (The debug option was not enabled so it was not possible to understand the reason for the crash)	Restart the LB Interlogger	Couple of times
Failure in "edg-gridftp-exists/mkdir"	1. Can’t open data connection 2. No local mapping for Globus ID		1. At least once from all RAL and LNL WNs creating a dir on CNAF SE. 2. Few times from CERN WN creating dir on CNAF SE
SE unavailable	NFS, hardware problems, no grid-map-file	Identify and fix the problem	3 times
Failure in "grid-mapfile" update	The LDAP server at NIKHEF was not reachable and the failsafe mode of the updating procedure failed, thus producing an empty "grid-mapfile"		Once

Table 3 (Part 2 of 3) - Problems encountered and solution adopted during the Stress Test

Symptom	Cause	Solution	Frequency
Replica Catalog client problem	Replica Catalog client is not able to handle lots of files in one collection	Work-around: create and use multiple logical collections, one for each UI	See text
Replica Catalog stuck due to server problems: Cannot connect to RC server Can't contact LDAP	LDAP server overloaded due to concurrent jobs writing into the RC (short jobs lasts 60-100 sec and all write into RC)	Create new RC, restart the server, slow down the submission rate of short jobs (rather painful since they are short jobs)	Twice. Rather messy recovery after these crashes. All the jobs running will hang forever.
Replica Manager "RegisterEntry" failure: 1. Filenames don't exist in collection 2. The server sent an error response: "530 No local mapping for Globus ID"	Behaviour that seems to be related to concurrent jobs writing into the RC. Having concurrent jobs increases the failure rate in registration and the job execution time	Slow down the submission for short jobs.	High when there are concurrent jobs
Replica Manager "RegisterEntry" odd behaviour: Sometimes it tries to create a collection for an SE even if the collection already exists. It sounds like it can't figure out which SE collections are in the RC. The files are usually properly registered.	Behaviour that seems related to concurrent jobs writing into the RC		Rare

Table 3 (Part 3 of 3) – Problems encountered and solution adopted during the Stress Test

7 Results and Achievements

The analysis of the behaviour of submitted jobs is the final aim of the Stress Test and therefore was accurately planned from the beginning.

A detailed analysis of the job's success or reason of failure was done using both the information of the CMS BOSS DB and of the EDG Logging and Bookkeeping.

The results can be presented in many different ways thus giving different kinds of information.

- The analysis of the EDG logging gives the relative importance of the different pieces of middleware, while the analysis of the CMS logging gives the rate of successes for CMS data production.
- A chronological analysis of the successes and failures can give information on the stability of the Testbed, provided that also the periods of inactivity (no job submitted because of many reasons, including meetings and demos) and system upgrade are considered.
- A detailed breakdown and classification of job failures can give information on the rate and the importance of each problem.
- The number of events produced as a function of time can give an idea of the usability of the system.
- A cross analysis of the final status of jobs between the two monitoring systems can give detailed information on the correlation of failures and successes.

A total of about 10500 jobs were submitted during three weeks in December 2002, using four different submitting UIs, over nine Testbed sites.

In the following we will go through some of the possible analyses.

7.1 Achievements and limitations of the Test

The Test was able to identify many problems and bugs, tracking them via the job monitoring system and providing a hook to the EDG experts to develop a solution. Chapter 6 discussed the details of the adopted solutions and work-arounds. The ability to manage and improve the EDG Testbed performances was a major result.

The lessons learned both by CMS and EDG people on the necessary modification of their tools are also an important result, as it can provide suggestions for future development. Identification of bottlenecks and missing or poor functionality could also influence the plans of new releases and eventually suggest new studies.

Limitations in delivered functionality, fragility of the system and modifications of the tools (software and middleware) posed many problems for the running of the Stress Test, however ultimately the Stress Test ran successfully.

The test also demonstrated that CMS software and EDG middleware could cooperate to build a distributed environment where CMS Monte Carlo production can be performed. The addition of extra resources by new sites was successful, thus proving the ability to rapidly scale up in a dynamic way. The use of shared resources and sites where the CMS experiment has no direct local support was also proved.

Major problems were identified in the Information System area, in the RC performances area and in some components interfaced with the RB (Globus, local scheduler systems, Condor-G, Job Submission System, etc).

A limitation on the possible results of the Test and in the evaluation of the EDG Testbed came also from the Monte Carlo production procedure of CMS. Two steps were needed to complete a simulation of the events (125 events per job): CMKIN followed by CMSIM. Only the CMKIN jobs could be distributed over the sites as the CMSIM were forced to run where the output of the previous step was performed. The CMSIM are the "long-CPU-time" ones and it would be a better evaluation if those jobs could have been distributed via the RB mechanism. The replication of CMKIN output files was not considered during the Test, however it could have been used to improve the distribution of the CMSIM jobs. This option proved to be useful during further testing after the end of the Stress Test.

As a final point, it should also be said that it was not possible to load completely the Testbed and therefore saturate the available resources. This was due to many reasons, among them being the forced slow rate submission of jobs.

7.2 EDG Testbed Performance

The careful inspection of the logging and trace of jobs submitted gave much information about the status and performance of the Testbed from the point of view of the EDG Middleware. The Logging and Bookkeeping information was processed, checking the status of every submitted job. There were mainly two possibilities: *outputready* status (successful termination for EDG) and *abort* status (abnormal termination for EDG). Other possible statuses of the jobs are symptoms of misbehaviour: *done*, *scheduled*, *running*, *ready* and *waiting*. These are normal statuses if they are transient, however if these kinds of statuses persisted for a long time (or forever) the job had to be classified as failed. A particular treatment had to be applied to the *done* status, as in this case the job correctly terminated with there being only some problem in making available the job-log to the UI.

Finally the EDG Logging reported also another status (*cleared*) that applies to the *outpuready* jobs whose logs had already been retrieved. According to the above statuses obtained by the "dg-job-status" (and similar EDG commands), the following classification was defined:

1. *Output ready* or *Cleared*: successful job termination (labelled "finished correctly" in the following).
2. *Done*: jobs terminated but with a problem in the log retrieval (reported in the following as "finished without logs").
3. *Scheduled* forever: jobs still to be run or that finally stay in the queue indefinitely.
4. *Running* forever: jobs still running or that cannot reach an end (stalled).
5. *Ready* forever: jobs holding and that cannot proceed further.
6. *Waiting* forever: jobs waiting to be scheduled by the RB and not proceeding further.

7. *Aborted*: jobs that crashed for various reasons related to middleware problems.

Category 1 enumerates the jobs that are to be considered to have gone through the Grid and that produced the output correctly as for the EDG evaluation.

Category 2 enumerates the jobs that did not complete the last step of the EDG middleware for some reason; normally they are jobs that completed their computation and registration of the output. However the "log" of the job could not be made available to the submitting user (retrieval of the output sandbox not possible).

Categories from 3 to 7 represent the failed jobs, "crashed or bad status" for different reasons. A special class of aborted jobs is represented by the *no matching resources* jobs: jobs that could not find any available resource matching the requirements, a condition of the Testbed in the moment of initial processing.

The other principal sources of *abort* status will also be reported in the following analyses.

7.2.1 CMKIN jobs

Table 4 gives the classification of the CMKIN-type jobs submitted to the Testbed during the Stress Test according to the above categories. The table sums all the jobs submitted by the four UIs and it shows the CE where the jobs submitted by the RB went to run (and where sometimes they were forced to run by the UI submitter in order to overcome partial inefficiencies of job load balancing).

"Finished Correctly" lists the number of jobs of category 1. "Finished without logs" lists the number of jobs of category 2 above. "Crashed or bad status" lists the number of jobs from category 3 to 7. "No match" lists the number of jobs from special *aborted* category: *no matching resources* (clearly no CE can be listed for this case).

A total of 6336 CMKIN jobs were launched into the Grid, and a total of 5518 were successful.

The overall efficiency during the whole Test for CMKIN jobs for EDG evaluation of the Testbed and middleware performances turned out to be 87%.

Details of the rates of successes and failures per executing CE are also given in Table 4. CEs' efficiencies appear to be higher than average because the "no matching resources" category can only be attributed to all the CEs together.

Table 4 also gives the details of the abort status reason. There is no evidence of any peculiar behaviour, given the problems already discussed in Chapter 6. The only exception is a high number of jobs failed at CERN for "Generic Failure: MyProxyServer not found in JDL expr" which can probably be considered a transient Testbed condition (mis-configuration of the RB.conf file).

It's worth also mentioning that a bunch of very fast CMKIN jobs ends up writing concurrently into the RC increasing the risk to overload it. Having the RC jammed, the jobs hung on RM commands and stayed in "*Running*" status forever.

CMKIN jobs		CE									
Status	**Totals**	**CNAF**	**IN2P3**	**LNL**	**CERN**	**NIKHEF**	**PD**	**POLY**	**RAL**	**IC**	**No match**
Finished Correctly	**5474**	1090	559	796	2085	223	148	18	325	230	
Finished without logs	**44**	0	17	1	14	0	1	0	11	0	
Crashed or bad status	**818**	52	39	50	147	17	1	0	3	0	509
Total number of jobs	**6336**	**1142**	**615**	**847**	**2246**	**240**	**150**	**18**	**339**	**230**	**509**
Details of "Crashed" jobs											
Scheduled forever	**11**	2	2	0	2	3	0	0	2	0	
Running forever	**74**	1	13	48	4	7	1	0	0	0	
Ready forever	**63**	4	23	0	30	6	0	0	0	0	
Generic Failure: Cannot open input sandbox directory	**1**	0	0	0	1	0	0	0	0	0	
Generic Failure: MyProxyServer not found in JDL expr.	**102**	1	0	0	100	1	0	0	0	0	
Condor failure: Submitting job(s) - condor command failed	**19**	18	1	0	0	0	0	0	0	0	
Failure while executing job wrapper	**37**	26	0	0	10	0	0	0	1	0	
Globus Failure:	**2**	0	0	2	0	0	0	0	0	0	
No matching resource found	**509**										

Table 4 – CMKIN type of Jobs: classification for EDG status.

7.2.2 CMSIM jobs

Table 5 gives the classification of the CMSIM jobs submitted to the Testbed during the Stress Test according to the 7 categories discussed above. The table sums all the jobs submitted by the four UIs and it shows the CE where the jobs run, submitted by the RB according to the location of the required input data.

As for CMKIN jobs, "Finished Correctly" lists the number of jobs of category 1. "Finished without logs" lists the number of jobs of category 2 above. "Crashed or bad status" lists the number of jobs from category 3 to 7. "No match" lists the number of jobs aborted for no matching resources.

A total of 4340 CMSIM jobs were launched into the Grid, and a total of 1678 were successful.

The overall efficiency during the whole Test for CMSIM jobs for EDG evaluation of the Testbed and middleware performances turned out to be ~39%.

Details of the rates of successes and failures per executing CE are also given in Table 5. CEs' inefficiencies seem to fluctuate among the CE sites, giving a visible idea of the fragility of the system when "watched" by the jobs for a 12-hour time of run. Again the "no matching resources" category can only be attributed to all the CEs together.

Table 5 also gives the details of the *Abort* status reasons. There is no evidence of any peculiar behaviour over the CE sites. However the effects of the major sources of problems encountered during the Stress Test are evident. The "no matching resources" problem is mainly due to Information System instability and the "failure while executing job wrapper" problem is related to the RB components relying on other basic middleware pieces. As it was explained before in section6.1.3, the "failure while executing job wrapper" error was mainly due to:

a) Problems when downloading/uploading the input/output sandbox;

b) *Standard output of job wrapper does not contain useful data* message.

Case a) means that a globus-url-copy didn't work, while case b) means that a "condor_submit" command was issued and the standard output after this CondorG submission (done via Globus) was empty, instead of containing the stdout of the job wrapper script.

In addition jobs could also be aborted because the submission done via CondorG to Globus didn't work (e.g. the gatekeeper was unreachable due to a CE disconnected because of a network crash). The resulting classification being either “Globus Failure: Globus down/Submit to Globus failed” or “Globus Failure”.

It’s also worth to mentioning that the category of jobs aborted for “administrative request” were caused by the requirement of local site manager intervention to restart misbehaving services and then killing running jobs

CMSIM jobs		CE									
Status	**Totals**	**CNAF**	**IN2P3**	**LNL**	**CERN**	**NIKHEF**	**PD**	**POLY**	**RAL**	**IC**	**No match**
Finished Correctly	**1552**	349	2	226	760	24	33	1	93	64	
Finished without logs	**126**	10	0	0	36	69	0	1	10	0	
Crashed or bad status	**2662**	357	169	227	866	117	112	11	57	24	722
Total number of jobs	**4340**	**716**	**171**	**453**	**1662**	**210**	**145**	**13**	**160**	**88**	**722**
Details of "Crashed" jobs											
Scheduled forever	**52**	1	10	1	4	35	0	0	1	0	
Running forever	**116**	0	17	2	54	8	0	5	13	17	
Ready forever	**7**	0	0	0	0	7	0	0	0	0	
Condor failure: Submitting job(s) - condor command failed	**1**	0	0	0	0	1	0	0	0	0	
Failure while executing job wrapper	**1476**	305	135	198	638	38	112	6	38	6	
Globus Failure:	**90**	3	6	0	68	13	0	0	0	0	
Globus Failure: Globus down/Submit to Globus failed	**144**	24	1	26	76	15	0	0	1	1	
Terminated by administrative request.	**54**	24	0	0	26	0	0	0	4	0	
No matching resource found	**722**										

Table 5 – CMSIM type of Jobs: classification for EDG status.

7.3 CMS view of Testbed performances

The CMS view of EDG evaluation was obtained by analysing the CMS BOSS/DB and cross checking the location and dimension of the final ntuple files (for CMKIN jobs) and of the fz files (for CMSIM jobs) in the relevant SEs. The results were extracted by querying the BOSS DB for job entries matching a number of requirements including: correct Dataset ID, time of execution start, registration of output into the SE and the RC, etc.

The criteria of success and/or failure are therefore quite different to the ones of the EDG status and logging. The important driving issue here was that the final simulated events produced by every job were available for further processing and analysis. One (among many other) of the requirements that the CMS production of a job imposes to be successful was agreed to be less stringent for the Stress Test: a job could be considered successful even if its log could not be retrieved (provided that the fraction of these jobs was reasonably low). Normally these kind of *semi-successful* jobs were classified by EDG as “*done*”. In addition, some of the EDG “aborted” jobs belonging to the “Standard output of job wrapper does not contain useful data” category may have been successfully completed.

It should also be noticed that the total numbers of submitted jobs are different from the ones quoted for EDG evaluation. The differences came from the different logging procedure and mainly from the requirement that a job should have had a chance to run to be considered for CMS evaluation.

7.3.1 Events produced

During the CMS Stress Test, a total of 268375 events were produced (with both CMKIN and CMSIM steps completed) with the corresponding fz files properly copied and registered into the RC.

The contribution of every UI involved in the Stress Test to the total amount of produced data is shown in Table 6, where the number of events (for both CMKIN and CMSIM type of jobs) produced at each site is given.

In summary, the percentage contribution to the CMKIN production step were: CNAF 43%, IC 12%, POLY 19%, PD 26%, whereas the percentage contribution for the CMSIM step were: CNAF 48%, IC 9%, POLY 12%, PD 31%.

	Total number of CMKIN events	**% of total**	**Total number of CMSIM events**	**% of total**
CNAF	253625	43	130250	48
IC	73125	12	23375	9
POLY	114250	19	32125	12
PD	151750	26	82625	31
Total	***592750***		***268375***	

Table 6 – Total number of successfully produced CMKIN and CMSIM events (i.e. with correct copy and register operations to the RC). (The four UIs were using respectively the listed RBs: CNAF-UI the CNAF RB for CMS, IC UI the IC RB, POLY UI the CERN RB for CMS, PD UI the CNAF RB)

In Figure 1 and Figure 2, a chronological summary of the integrated total amount of CMKIN and CMSIM events produced in the Stress Test is shown. The x axis covers the whole Stress Test period, and contributions from each UI are superimposed using different colours; the total amount of successfully processed data on a given production step at a given site (UI) can be obtained by selecting the time (date) of interest on the x axis and measuring the width of the coloured band corresponding to the UI of interest. Note that in the plot only the events that have been produced within jobs that correctly copied and registered their output fz files into the RC are shown.

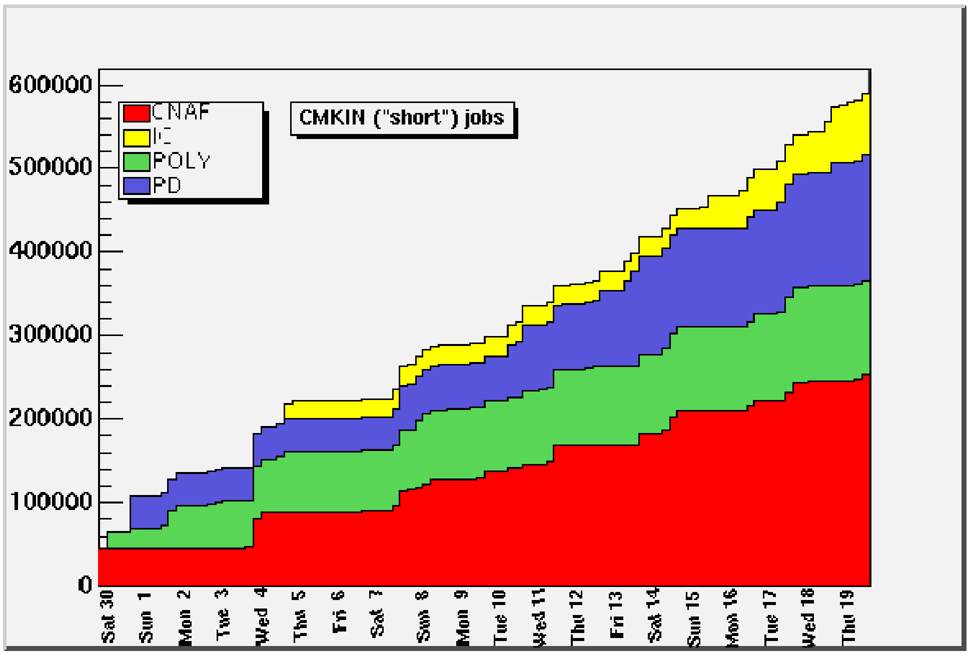

Figure 1 – Chronological summary of the integrated total amount of CMKIN events produced. The contributions from different UIs are stacked on each other, and distinguishable by their colour.

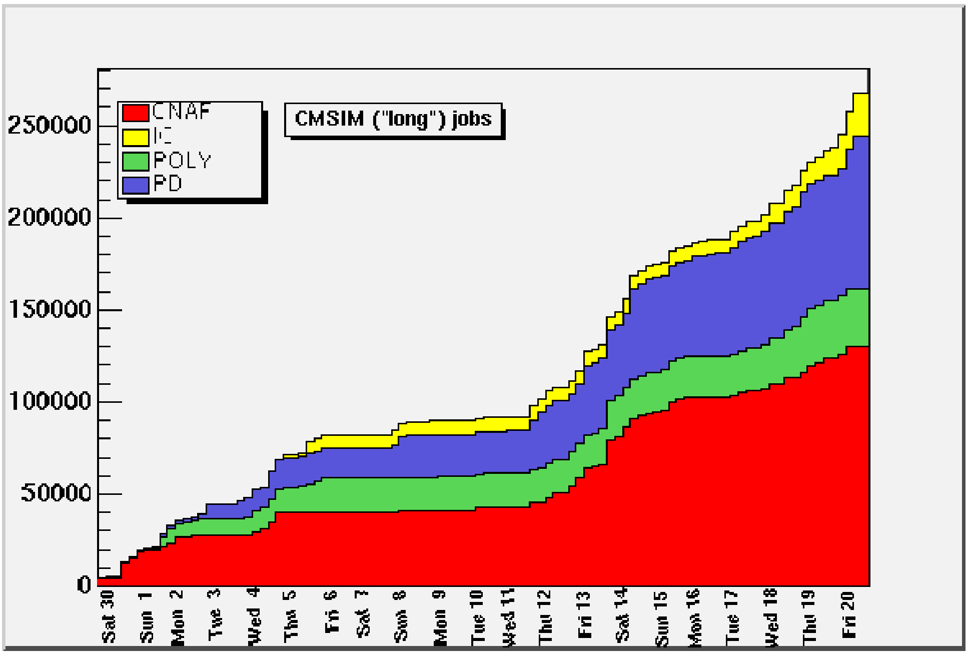

Figure 2 – Chronological summary of the integrated total amount of CMSIM events produced.

7.3.2 Jobs processed

During all Stress Test activities, a total (UI-integrated) number of 8782 jobs - belonging to the official *eg02_BigJets* dataset and processing 125 events each of both the CMKIN and CMSIM processing phase - were started.

The number of jobs contributing to the CMS classification is the number of jobs that were registered into the BOSS DB as belonging to the *eg02_BigJets* dataset. Since the name of the dataset is updated into the BOSS DB filtering on the standard output of the running job, these numbers represent the jobs that started the run, and therefore the total (and partial) number of jobs is different from the one(s) registered by EDG logging.

The breakdown of the total number of jobs submitted by each UI - regardless of their exit status - is shown in Table 7, together with the fraction of jobs that ended successfully, with also the Replica Catalog operation properly executed.

It can be seen that a general efficiency of about 83% for CMKIN jobs and about 70% for CMSIM jobs could be achieved.

	Total number of started CMKIN jobs	**Number of successful CMKIN jobs**	**Total number of started CMSIM jobs**	**Number of successful CMSIM jobs**
CNAF	2430	2029	1412	1042
IC	647	585	290	187
POLY	1327	914	473	257
PD	1296	1214	906	661

Table 7 – For both production steps and for each UI, total number of "started jobs" and number of jobs ending successfully (i.e. with correct copy and register operations to the RC).

Part of the jobs classified as not successful are due to inefficiencies related to the RM Copy and Registration commands, the remaining jobs were aborted before reaching the copy of the produced output to the SE. In addition, for CMSIM jobs there are about 1-2% of jobs with "globus-url-copy" failures in retrieving the input ntuple from the SE.

The efficiencies of the copy of output data into a SE and their registration into the RC are reported in Table 8 and Table 9, for CMKIN and CMSIM jobs respectively.

These efficiencies are computed on the basis of the information extracted from BOSS database. The failure rate while copying data into a SE is higher for CMSIM jobs where the size of the output file to be copied is about four times higher with respect to CMKIN output. The failure rate in registration is higher for CMKIN jobs where the probability of having more CMKIN concurrent jobs writing into the RC was higher.

As described in section 5.1.2.2, the jobs submitted from all UIs but POLY used the commands *edg-replica-manager-copyFile* and *edg-replica-manager-registerEntry*, while the jobs from POLY UI used the command *edg-replica-manager-copyandRegister*, that should be equivalent to the previous ones with in addition the possibility to interface to mass storage. Performing the copy and registration altogether showed a worst behaviour than expected (for unknown and not investigated reasons).

UI	Nb of started CMKIN jobs	Jobs aborted before the copy	Failed copy	Failed register (ok copy)	Successful copy and registration	ε (copy)	ε (register)	ε (copy and register)
CNAF+IC+PD	4373	2	28	515	3828	99%	88%	87%
POLY	1327	0	137	276	914	90%	79%	71%

Table 8 – Efficiencies of RM commands (copy into SE and registration into RC) for started CMKIN jobs, for jobs submitted from CNAF, Padova and IC UIs. Statistics from POLY UI are reported separately since the jobs were submitted using different RM command to allow the use of the MSS interface as explained in section 5.1.2.2.

UI	Nb of started CMSIM jobs	Jobs aborted before the copy	Failed copy	Failed register (ok copy)	Successful copy and registration	ε (copy)	ε (register)	ε (copy and register)
CNAF+IC+PD	2608	206	387	125	1890	84%	95%	79%
POLY	474	115	47	55	257	87%	85%	72%

Table 9 – Efficiencies of RM commands (copy into SE and registration into RC) for started CMSIM jobs, for jobs submitted from CNAF, Padova and IC UIs. Statistics from POLY UI are reported separately since the jobs were submitted using different RM command to allow the use of the MSS interface as explained in section 5.1.2.2.

The CEs and SEs usage statistics are shown in Figure 3 and Figure 4 respectively, for CMSIM Jobs. Figure 5 and Figure 6 show the same information for CMKIN jobs. The CE plots clearly show the adopted strategy of job submission by the UIs and also the use of available resources. The SE plots for CMKIN shows the pseudo-random distribution of jobs writing the output in the local SE (close to the CE where they ran). The CMSIM SE plot shows the choice of forcing the jobs' output to predefined SE, for the CNAF and Padova UIs.

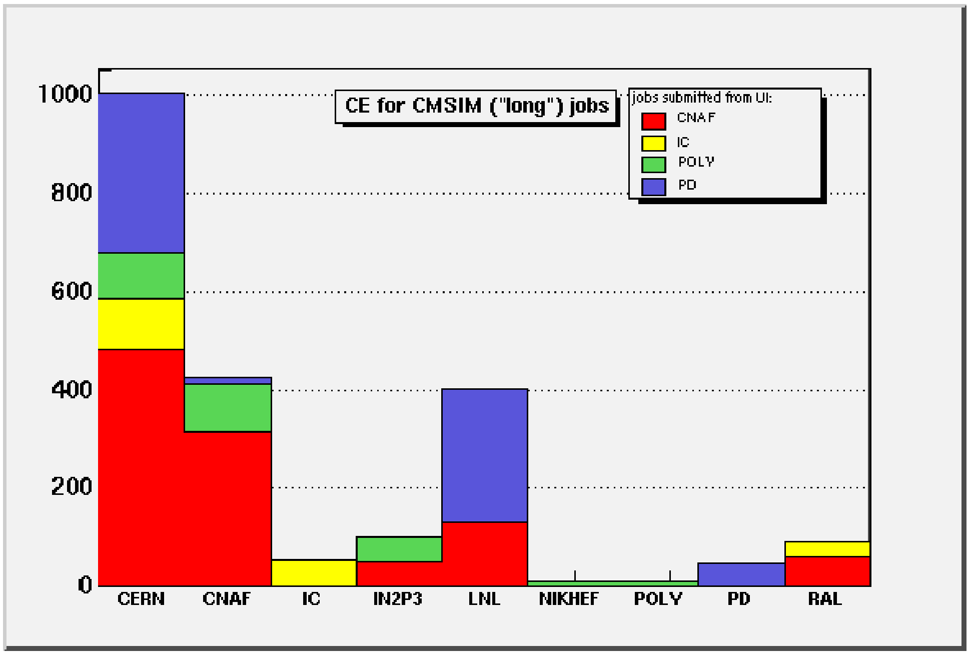

Figure 3 – Number of CMSIM jobs that were executed on each CE.

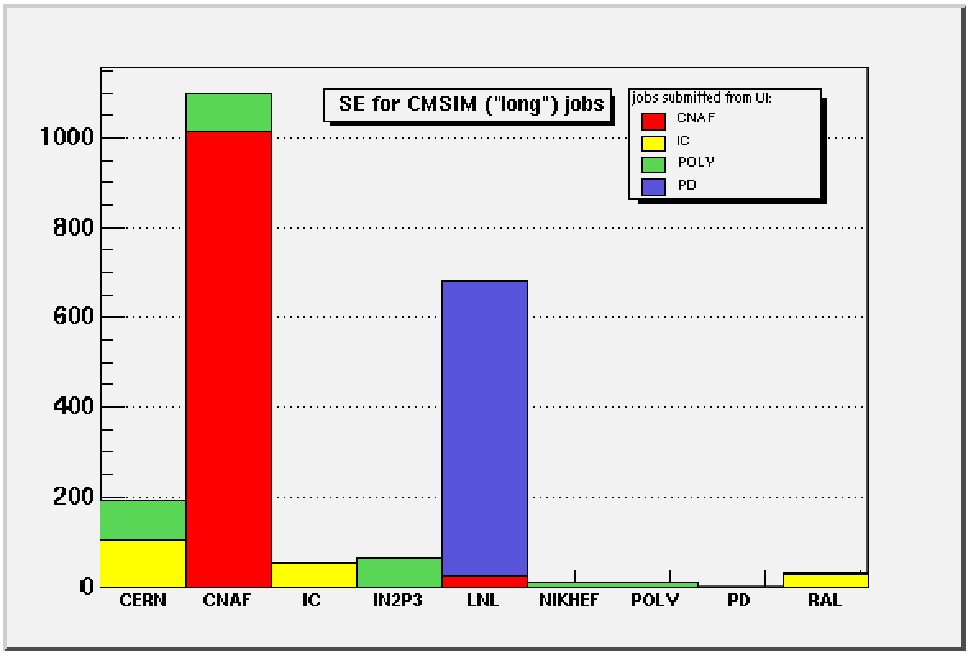

Figure 4 *– Number of CMSIM jobs that ended to write on each SE.*

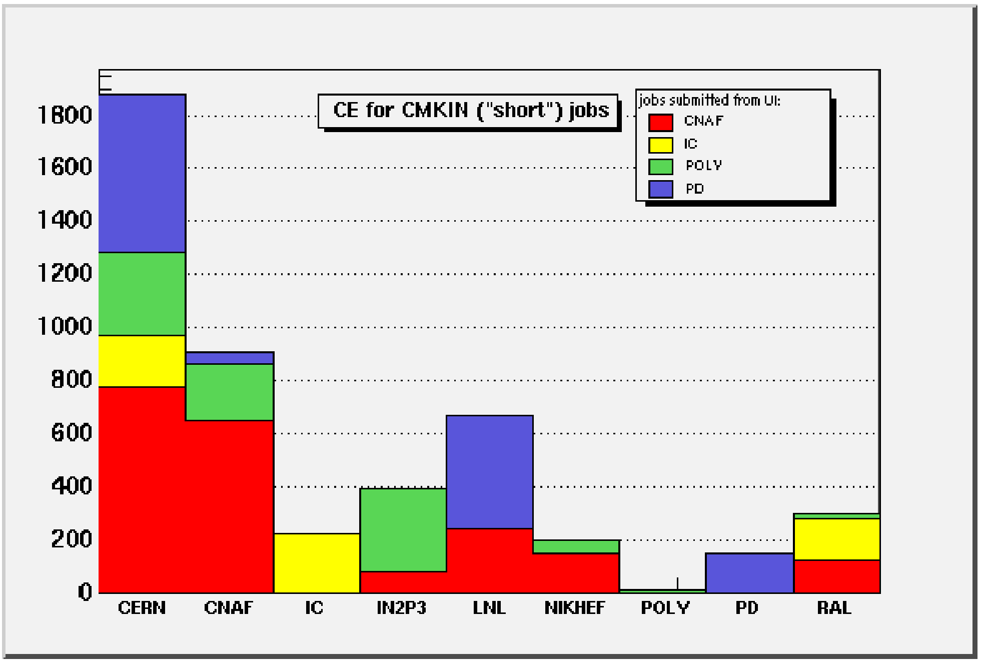

Figure 5 – Number of CMKIN jobs that were executed on each CE

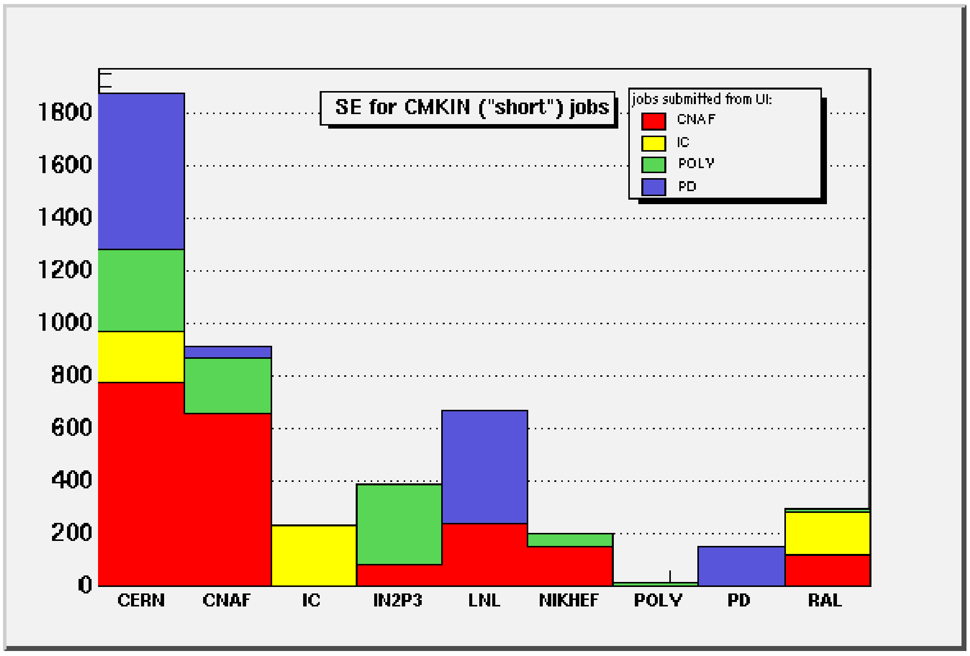

Figure 6 - Number of CMKIN jobs that ended to write on each SE

Figure 7 and Figure 8 shows respectively the distribution of the processing time and of the output fz file size for CMSIM jobs submitted from the CNAF UI. The average CPU time was approximately 12 hours (as expected); the average size was about 232 MB (each job generating 125 simulated events), consistent with “normal” CMS productions.

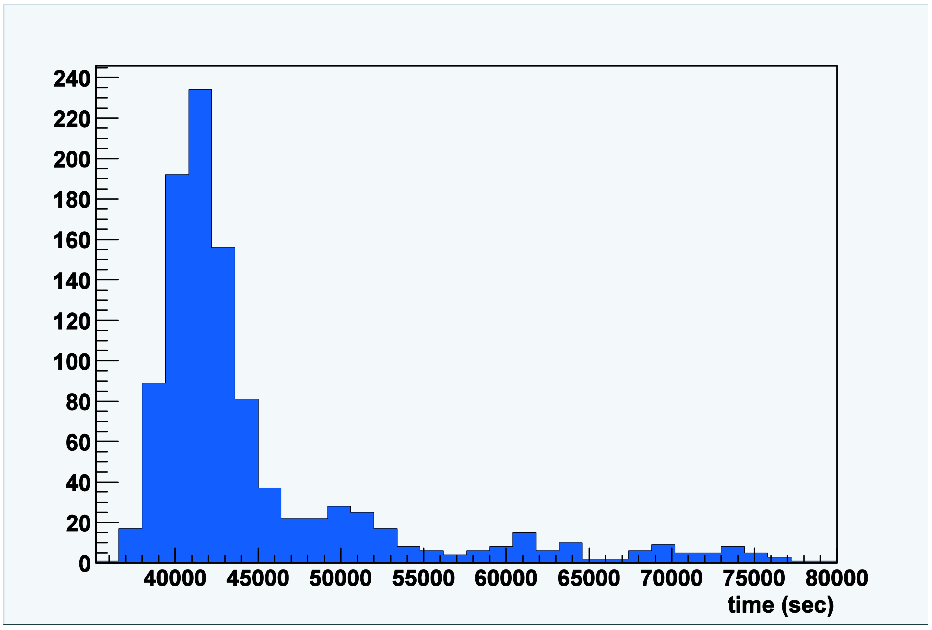

Figure 7 – Distribution of CPU time for CMSIM jobs submitted from CNAF UI.

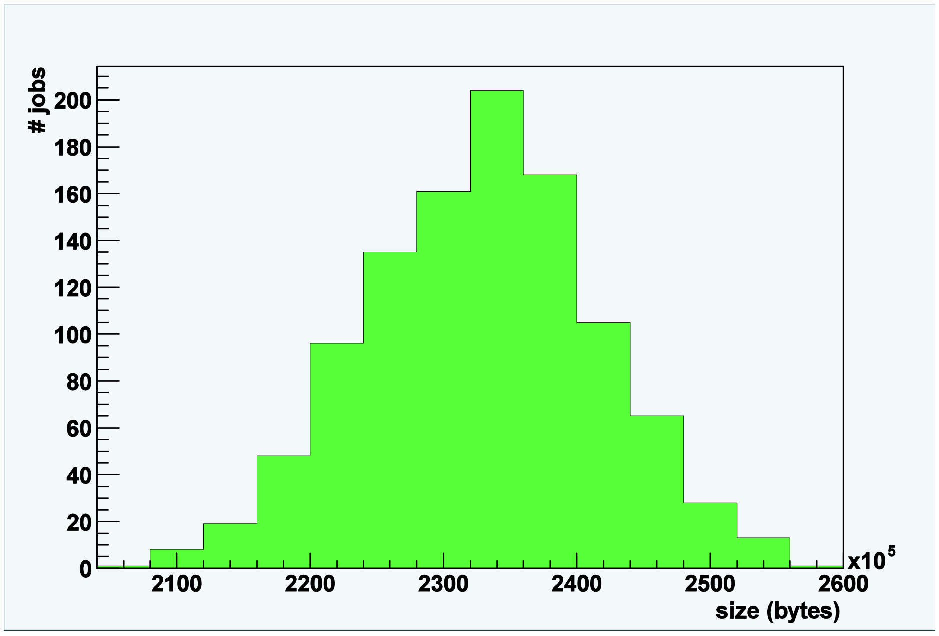

Figure 8 – Distribution of the size of CMSIM output fz files submitted from CNAF UI.

7.4 CMS successful jobs and EDG classification

An attempt to understand the correlations between the different categories of job classification according to EDG and CMS analysis was performed.

The main approach here was to understand both from an EDG and CMS perspective, where the jobs classified by CMS to be successful came from. An alternative approach, though not followed here, would have been to ask the reverse question – to understand from which categories of EDG the "crashed" CMS jobs belong to. Table 10 shows the number of CMKIN jobs submitted by each UI and their total, categorized between the correctly finished and the unusable ones for CMS. The total number of jobs is the same as has already been shown in Table 7.

The same table lists the breakdown of EDG categories for the CMS "good" jobs. The noticeable feature is that the majority of jobs are in the class of EDG "good" jobs. Few jobs come from the "no log" category (as explained above in the paper) and some come from the "aborted" EDG jobs category. Jobs within this last category are the ones that EDG claims have not finished correctly (the "Output does not contains useful data" message), but are indeed finished and the output is correctly registered and stored.

CMKIN jobs	UI				
Status	**All**	**CNAF**	**POLY**	**IC**	**PD**
Finished Correctly	**4742**	2029	914	585	1214
Crashed or bad status	**958**	401	413	62	82
Total number of jobs	**5700**	**2430**	**1327**	**647**	**1296**
Details of Correctly Finished Jobs: classification against EDG Status					
EDG Finished Correctly	**4691**	2020	893	568	1210
EDG Scheduled	**1**	1	0	0	0
EDG Running	**3**	3	0	0	0
EDG Finished without logs	**7**	0	0	6	1
EDG Aborted for various reasons	**40**	5	21	11	3

Table 10 – CMKIN type of jobs: classification for CMS status

Table 11 shows the same analysis presented in Table 10 but for the CMSIM jobs. Again the total number of jobs is the same as in Table 7. Few jobs come from unexpected categories, like "running" or "scheduled", again showing that the EDG status is only related to some internal middleware mismatch, leaving the possibility that the jobs ended correctly. The most noticeable feature is that the number of "aborted" category is very high. This large number of CMS "good" jobs could only be identified via the CMS Production tools (namely BOSS in this case), and the final efficiency for CMS production is highly influenced by this rescue operation (in fact this had been planned for in advance of the Stress Test).

CMSIM jobs	UI				
Status	**All**	**CNAF**	**POLY**	**IC**	**PD**
Finished Correctly	**2147**	1042	257	187	661
Crashed or bad status	**934**	370	216	103	245
Total number of jobs	**3081**	**1412**	**473**	**290**	**906**
Details of Correctly Finished Jobs: classification against EDG Status					
EDG Finished Correctly	**1335**	817	103	81	334
EDG Scheduled	**2**	0	0	0	2
EDG Running	**36**	17	3	0	16
EDG Finished without logs	**47**	47	0	0	0
EDG Aborted for various reasons	**727**	161	151	106	309

Table 11 – CMSIM type of jobs: classification for CMS status

7.5 Time analysis of activities

In this section a general time analysis of the produced events is described. A detailed chronological list of events during the Stress Test is reported in Appendix B.

In Figure 2 the total number of produced CMSIM events as a function of time is shown, adding the contributions of jobs submitted from the various UIs. The flat distribution around 3 December for CNAF and POLY UIs reflects the attendance of the submitters at the CMS Week meetings. The production of about 50 000 events was reached on the 4th December when the submission from Imperial College UI had also started.

The plateau of produced events in the period from 6 to 11 December was due to several problems affecting the EDG Testbed:

- The overload of the RC LDAP server caused it to stick, so all the jobs writing into the RC were hanging and no further job submission was possible until the server was restarted.
- The limitation of the RC to deal with large collections was found and overcome by creating and using multiple collections.
- The MDS was highly unstable during the weekend, causing jobs to be aborted because of no matching computing resources. There were also jobs submitted from the Padova UI, since it was upgraded to the EDG release 1.4.0, in order to test the new functionality before its official deployment on the Application Testbed.
- Testbed System upgrade to release EDG 1.4.0 (where a work-around for the MDS problem was deployed) was installed on all the Testbed sites in one day. The rate of aborted jobs due to “no matching resource” was reduced from 12% to 3% for CMKIN jobs and from 26% to 9% for CMSIM jobs (the overall reduction went from 17% to 6%).
- The RC got stuck again and a new RC was set up. A reduced rate in job submission for short jobs (CMKIN) was adopted in a coordinated way between all the UIs. The file replication over the SEs also required coordination with the job submission activity.

After this period, there was a significant increase in the number of events produced: the threshold of 100 000 events was reached on the 12th of December and about 50 000 events where produced in the following 2 days. During the following weekend there was a problem with the Information System, resulting in most of the submitted jobs being aborted. From 16 to 20 December about 85 000 events were produced and a total of 200 000 events was achieved on the 18th. During the last days of the Stress Test the CNAF site was updated with a new PBS script for the CE and also with the so-called “JSS-Maradona” patch (including other WP1 fixes for the RB).

A total of more than 250 000 events were simulated in 3 weeks, corresponding to about 500GB data written to SEs. More than 10 000 jobs were submitted producing about 8 000 files that were registered into the RC.

7.6 Post Stress Test activities

A few important activities had to continue after the end of the Stress Test. Apart from analysis of the results, the produced data was moved to their final destination and some further tests were performed in January 2003.

7.6.1 Final delivery of simulated data

The Monte Carlo simulated data for CMS physics analysis had to go through the remaining steps of the CMS production chain (reconstruction of events and ntuple creation via the CMS ORCA program). Because of the license necessary for Objectivity/DB, only a few selected sites could perform this data treatment. It was decided to perform this final step at CERN. In any case, the analysis is not currently performed on the Grid, hence a central repository for processed data (CERN) had to be used.

Some of the produced data had already been moved to CERN during the Stress Test. In particular the jobs submitted by the POLY and IC UIs were stored (as planned) in the local SE where the jobs happened to run. Asynchronously, but during the Stress Test, such dispersed data were replicated using the RM functionality, to a “special” CERN CE. The “special” refers to the CERN CE interfaced with the MSS Castor System, so that the data were stored into its tape facility. Finally the original files had to be cleared from the remote site. This procedure had been planned for at the start of the Stress Test.

A total of 402 FZ files, corresponding to 96GB of data, were replicated and the average replication time per file was about 360 sec. A few files transferred over a 2Mb connection to the SE from Ecole Polytecnique affected this averaged transfer rate value. The failure rate during replication was about 1%, due to problems relating to the RC.

It is worth noting that the data being written at the IN2P3-Lyon site were also stored in the local MSS system (HPSS), therefore the final step of replication via the RM was a MSS to MSS movement (though not a “direct” MSS to MSS movement, as a cache disk system was active, as usual). Concerning the RM interface to MSS, it worked at a very basic level with low flexibility, i.e. it was not possible from the user command to specify which files had to go on MSS and which not. The switching on/off of the MSS interface had to be done in a configuration file sent with the job. In order for the files stored on tape that were incorrectly registered to be replicated, they needed to be re-registered and therefore they had to be re-fetched from tape to disk.

The produced data by the other two UIs (CNAF and PD) were stored at the CNAF and LNL SEs, no matter where the jobs happened to run. Therefore a movement to the CERN “special” SE and clearing of the original files was again necessary. Those data were moved to CERN “CASTOR enabled SE” without replication using the *globus-url-copy* EDG command.

7.6.2 Evaluation of the last EDG release: version 1.4.3 (End of Dec 2002 – Jan 2003)

During the last days of the Stress Test (from Dec 18 to Dec 20) a pre-release of the patched version 1.4.x was installed at the CNAF site and some quick tests showed a promising improvement in performances.

At the beginning of January 2003, a new EDG release was tagged as version 1.4.3, including some notable bug correction and new performance enhancements.

Jobs were submitted to the newly updated Testbed in order to try a measurement of the efficiency of EDG 1.4.3. The test-jobs were of the same kind as those used for the Stress Test and were submitted by three UIs (CNAF, Padova and POLY). However the submitted jobs were not part of the CMS physics production, but just for testing.

On the basis of limited statistics of submitted jobs (about a thousand) the efficiencies were found to range from 95% for CMKIN jobs to about 71% for CMSIM jobs.

The above efficiencies were an estimation of EDG middleware performances. These evaluations were very preliminary and indeed should be considered as just an indication, even though they were promising.

Table 12 and Table 13 gives the details of the successful and failed EDG jobs submitted by all the UIs involved, respectively for CMKIN and CMSIM types of jobs, with the details of the ones failed. It’s worth mentioning that most of the failures classified as “Globus failure” and “Failure while executing job wrapper” were due to specific CEs (as indicated) during a short (and identified) period of time. They are symptoms of unexpected hardware failures or temporary unavailability.

CMKIN jobs		**CE**									
Status	**Totals**	**CNAF**	**IN2P3**	**LNL**	**CERN**	**NIKHEF**	**PD**	**POLY**	**RAL**	**IC**	**No match**
Finished Correctly	**1013**	257	52	174	255	73	80	0	53	69	
Finished without logs	**1**	0	0	0	0	0	1	0	0	0	
Crashed or bad status	**57**	0	20	0	2	1	1	0	10	0	23
Total number of jobs	**1071**	**257**	**72**	**174**	**257**	**74**	**82**	**0**	**63**	**69**	**23**
Details of "Crashed" jobs											
Scheduled forever	**2**	0	0	0	0	0	0	0	2	0	
Running forever	**9**	0	0	0	1	0	0	0	8	0	
Failure while executing job wrapper	**2**	0	0	0	1	0	1	0	0	0	
Globus down	**4**	0	3	0	0	1	0	0	0	0	
Globus Failure:	**17**	0	17	0	0	0	0	0	0	0	
No matching resource found	**23**										

Table 12 – CMKIN jobs submitted on the EDG Testbed with EDG version 1.4.3 of middleware

CMSIM jobs		**CE**									
Status	**Totals**	**CNAF**	**IN2P3**	**LNL**	**CERN**	**NIKHEF**	**PD**	**POLY**	**RAL**	**IC**	**No match**
Finished Correctly	**626**	217	34	27	167	16	69	0	41	55	
Finished without logs	**27**	0	0	0	0	0	0	0	0	0	27
Crashed or bad status	**264**	44	5	3	59	41	13	0	15	1	83
Total number of jobs	**917**	**261**	**39**	**30**	**226**	**57**	**82**	**0**	**56**	**56**	**83**
Details of "Crashed" jobs											
Scheduled forever	**1**	0	0	1	0	0	0	0	0	0	
Running forever	**10**	0	1	0	0	0	2	0	7	0	
Globus down	**10**	9	0	1	0	0	0	0	0	0	
Globus Failure:	**72**	10	2	0	58	1	1	0	0	0	
Failure while executing job wrapper	**88**	25	2	1	1	40	10	0	8	1	
No matching resource found	**83**										

Table 13 – CMSIM jobs submitted on the EDG Testbed with EDG version 1.4.3 of middleware

The CMS-like efficiencies (using BOSS) are less significant as the test was performed just to evaluate the EDG release, with poor attention to the usability of data produced (not used for physics studies). However they showed good results, with the CMKIN efficiency at the level of 98% and the CMSIM one at the level of 76%. Table 14 reports the details of the started and successfully ended jobs from the UIs involved, classified with the CMS logging.

The same table gives the analysis of the operation of copying and registering in the RC, suggesting an improvement from the previous versions of EDG middleware (tables 8 and 9 above in the text).

All UIs jobs (EDG vers 1.4.3)	**Started**	**Aborted before the copy**	**Failed copy**	**Failed register (ok copy)**	**Successful**	**Efficiency**
CMKIN	1032	0	3	20	1009	0.98
CMSIM	690	104	34	26	524	0.76

Table 14 – CMS classification for started and successful CMKIN and CMSIM jobs.

8 Conclusions

The submission of more that 10 000 jobs during a period of three weeks over the EDG Testbed proved to be a *real stress* for the whole system: hardware, software, middleware and personnel.

The results show that most of the stepped activities planned for the Test were met with success, even if the final goals of the Stress Test were not completely achieved.

The planned steps for the Stress Test activity were:

1) Preparation of CMS software and tools for installation on the EDG testbed environment.
2) Installation and check of CMS software and tools in a controlled testbed environment with the new

EDG release.

3) Installation of EDG release on “new” CMS testbed resources at CNAF and validation of them.
4) Run moderate production statistics in a reduced and well-controlled testbed configuration. One confining factor would be the limitation of the involved sites and another would be the limitation of the number of jobs (and/or events/job).
5) Progressive addition of the other UIs into the Stress Test, with a growing number of jobs.
6) Addition of the other CMS sites’ resources to the Testbed. Judgement of readiness for this would happen on the basis of the results of the previous phases.
7) According to success of the previous phases, move to production mode with a stable configuration of sites and software (including tools).

All of the above steps were met with success, thanks to the availability of the EDG middleware and support, except for the final one (step 7) that was met partially. As described in this paper this result was not achieved without difficulties and some re-planning.

The initial goals of the Test were mainly:

- Verification of the portability of CMS production environment into a Grid environment;
- Verification of the robustness of the EDG middleware in a production environment;
- Production of data for physics studies of CMS, possibly at the level of 1 million simulated events.

All of the three goals were met partially, though by differing amounts.

Portability of the CMS production environment into Grid implementation was demonstrated to a high degree, giving however some good hint for possible modifications of global design architecture and/or implementation of it.

Verification of the EDG robustness in a production environment demonstrated a lack of software maturity. Though this was to be expected, given the fact that this is only the second year of the EDG R&D program, getting a robust, or at least stable, environment proved to be quite difficult during the test.

The production of CMS simulated events at the level of 1 million could not be obtained. However, more than 250 kevents were produced successfully in a period of three weeks (compared to the four weeks planned). Taking into account the many interruptions of the test due to meetings, unavailability of resources and personnel, hardware failures and software changes, the result can be considered a reasonable success (even compared to “traditional” CMS productions over dedicated Farms).

Therefore the Test finally demonstrated that CMS software could be run on the EDG environment, provided that some modification of the CMS Production tools are developed to account for the different interfaces between the software/middleware components.

There was also the indication that the EDG middleware was close to being mature enough to be used for CMS Productions, though stability is still a major concern and some key functionalities are still missing.

Dynamic addition of new sites and resources to the testbed was demonstrated to be possible without disruption to the system as a whole, which is promising for the future as the system scales up in complexity, size and use.

8.1 Outcomes of the Stress Test

Among the many lessons learn during the Stress Test, some major outcomes can be extracted and summarized as follows:

- No serious “show stopper” problem was found. However the efficiency of use required a large amount of effort around the clock by many people
- Many bugs and limitations were found by stressing the system. It would not have been possible to find them without such a Stress Test.
  - Bugs (coming from many pieces of software provided by many authors) were promptly and iteratively corrected and new versions of the middleware were installed “on the fly”.
  - A large EDG effort was also spent in addressing issues related to the basic Globus components like the Information System MDS, the RC, the GRAM protocol and Gass mechanism for job submission and the Globus interface to OpenPBS.

    - Limitations were correctly identified and workarounds were found in close collaboration between CMS and EDG personnel, whenever possible. Many of the limitations were already planned to be overcome with EDG version 2 (and Globus version 2.2), thus confirming the real necessity of these developments.
- The testbed implementation proved to be
    - Flexible enough to allow for prompt reconfigurations;
    - Able to quickly accept new sites with new resources (provided by CMS for the Stress Test). In addition the use of resources where CMS personnel were not present was also successful;
    - Fragile for a number of reasons ranging from hardware failures to wrong configurations. Also some intrinsic fragility of components was highlighted. Identifying all of them proved to be a major undertaking;
    - Coordinated enough to permit synchronous operations in daytime. However many "personal" direct contacts were needed for this to be successful.
- The measured and final efficiencies (both for CMS and for EDG evaluation) were found to be:
    - Substantially different for jobs requiring small CPU time (few seconds) and for jobs lasting longer (order of 12 hours). The range was from 95% to 40% depending also on the kind of analysis applied. The "short" jobs showed better efficiencies than the "long" ones. This can be explained by the larger complexity of "long" jobs (input/output loads, larger requests to the services, etc.);
    - Overall, about 60% of successful jobs attained EDG completion;
    - Overall, about 70% of successful jobs resulted in correct and available CMS files of simulated events;
    - Major sources of inefficiencies were: Information System (MDS) instabilities, RC performances, hardware failures and mis-configurations, fragility of the GRAM-GASS Globus mechanism for job submission and output retrieval;
    - A much more improved situation was experienced during the last days of the Stress Test and during the follow-up, at the beginning of 2003, when EDG release 1.4.3 was deployed. That release incorporated some Globus work-around and bug corrections for the problems quoted above. Even though a small sample of submitted jobs (~1,000) were submitted, a preliminary estimation of efficiency indicated a value of about 80% (or even better if the trivial errors are excluded) for the EDG successful jobs.
- Due to the necessity of slow submission of jobs to prevent misbehaviours of the system and the "two-step" procedure of CMS simulation forcing the second step in the same CE where the first step ran, it was not possible to completely balance the system load and saturate the resources.

8.2 Recommendations for GRID Projects

Some key points identified for consideration for Grid Projects (in addition to the EDG) that CMS Stress Test results would recommend are listed here:

- Addition of new functionality to overcome the "work-around" solutions that normally results in inefficiencies and consumes effort.
- Bug solving and solution propagation should be improved.
- Better procedures for coordination should be implemented, so as to allow a larger scale of sites and a "production like" environment.
- Monitoring at a deeper level (all kinds of mis-configurations, hardware failure identification and job tracking) is needed. Redundancy here is also required.
- Robustness to possible failures of services and fail-safe solutions, including the duplication of key services, are needed.
- Close collaboration between the Applications and the Middleware developers has to be strengthened, also during the Testing periods.

8.3 Issues to be addressed by CMS

From the point of view of CMS, one important outcome of the Test was identifying the need to better consider the integration of the CMS software architecture with the Grid middleware architecture. Specifically the definition of the interfaces and the instantiation of the required functionality have to be better understood.

The current tools for simulation production have to be better tuned to the capabilities offered by the Grid middleware, thus giving better flexibility and scalability to a widely distributed environment. The process towards this achievement has already started within CMS (new tools' architecture and implementation) and will continue in the future.

The Data management over a distributed Grid environment of resources and services is a key goal of CMS software, and the lessons learned from the test over the EDG testbed improved this process.

Modularity, robustness, clearly defined interfaces and responsibilities of different pieces of software/middleware, procedures of validation and maintainability, coordination of actions and distributed knowledge and commitments are the key goals to be pursued.

Acknowledgments

We would like to acknowledge the invaluable help and support of the CMS Production Team and of the EDG Integration Team.

Without the friendly collaboration and prompt effort of the EDG Testbed people the present work would have not been possible.

Particular thanks is due to all the site managers of the participating sites, and also of some sites not involved, for their assistance and continuous excellent work.

The collaboration and support of the European DataTag Project (EDT) and of the LHC Grid Project (LCG) were invaluable.

Many colleagues gave their constant help and advice and in particular we would like to thank Fabrizio Gagliardi (CERN), Frank Harris (Oxford), Tony Wildish (Princeton), Bob Jones (CERN), Steve Traylen (RAL), Michele Michelotto (INFN-Padova), Flavia Donno (INFN-CERN LCG), Mirco Mazzucato (INFN-Padova), Nikolai Smirnov (INFN-Padova), Oxana Smirnova (Lund), John Gordon (RAL), Daniel Cano (ICFA), Rafael Marco de Lucas (ICFA), Luca Dell'Agnello (CNAF), Pietro Matteuzzi (CNAF) and Pier Paolo Ricci (CNAF).

A final thank is due to CNAF people for having hosted and supported the CMS/EDG Task Force "ramp-up" meeting.

References

[1] CMS Experiment: http://cmsdoc.cern.ch/cms/outreach/html/index.shtml

[2] LHC Project: http://lhc.web.cern.ch/lhc/general/gen_info.htm

[3] CERN (Geneva, CH): http://public.web.cern.ch/public/

[4] EDG Project http://eu-datagrid.web.cern.ch/eu-datagrid/, EDG Testbed http://marianne.in2p3.fr/datagrid/giis/giis.html

[5] CMS Internal Note 2002/044, V. Lefebure and J. Andreeva, *"RefDB"*

[6] http://computing.fnal.gov/cms/Monitor/docs/cms_documents/ProductionSoftware/CMS-PSOFT-002/cms-psoft-002.html: CMS IMPALA tool for event Simulation

[7] CMS Note 2003/005, C. Grandi, A. Renzi, *"Object Based System for Batch Job Submission and Monitoring (BOSS)"*.

[8] Zebra Files: http://wwwinfo.cern.ch/asdoc/zebra_html3/zebramain.html

[9] Objectivity/DB Main Web site: http://www.objectivity.com/

[10] http://root.cern.ch ROOT Project Main Web site

[11] CMS Note 2002/034, The CMS Production Team: T. Wildish, V. Lefebure, et al.,*"The Spring 2002 DAQ TDR Production"*.

[12] EDG Architecture: http://edmsoraweb.cern.ch:8001/cedar/doc.info?document_id=333671

[13] Globus Main Web site: http://www.globus.org/

[14] http://www.cnaf.infn.it/ CNAF Main Web site

[15] http://webcc.in2p3.fr/ CC-IN2P3/Lyon Main Web site

[16] http://www.nikhef.nl/ NIKHEF Main Web site

[17] http://www.rl.ac.uk/ RAL Main Web site

[18] CERN Castor MSS Project: http://castor.web.cern.ch/castor/Welcome.html

[19] HPSS by IBM: http://www4.clearlake.ibm.com/hpss/index.jsp

[20] http://www.openpbs.org/ PBS Main Web site

[21] http://webcc.in2p3.fr/man/bqs: BQS Main Web site

[22] http://www.lcfg.org/ LCFG Main Web site

[23] http://www.cvshome.org/ CVS Main Web site

[24] http://www.bugzilla.org/ Bugzilla Main Web site

[25] NAGIOS monitoring page: http://datatag-monitor.lnl.infn.it:58080/ ; NAGIOS Main Web site: http://www.nagios.org/

[26] R-GMA, EDG WP3 page: http://hepunx.rl.ac.uk/edg/wp3/

[27] IMPALA/EDG modified installation and details Web page: http://home.cern.ch/fanfani/grid/impala-edg/

[28] CMS CVS repository EDGIntegration/impala_EDG: http://cmsdoc.cern.ch/swdev/viewcvs/viewcvs.cgi/PROD/EDGIntegration/impala_EDG/?cvsroot=PROD

[29] *Boss2root* Web page: http://home.cern.ch/bonacors/boss2root.htm

[30] EDG Integration Team Web site: http://marianne.in2p3.fr/

[31] ATLAS Data Challenges: http://atlas.web.cern.ch/Atlas/GROUPS/SOFTWARE/DC/dc_page.html

[32] CMS Stress Test Web home page: http://sciaba.home.cern.ch/sciaba/stresstest/

Appendix A

Task Force Composition

Massimo Biasotto – CMS – INFN/Legnaro, Italy
Daniele Bonacorsi – CMS – INFN/Bologna, Italy
Paolo Capiluppi – CMS – INFN/Bologna, Italy
Claude Charlot – CMS – IN2P3/Ecole Polytechnique, France
David Colling – CMS – PPARC/Imperial College London, United Kingdom
Marco Corvo – CMS – INFN/Padova, Italy
Alessandra Fanfani – CMS – INFN/Bologna, Italy
Sergio Fantinel – CMS – INFN/Legnaro, Italy
Federica Fanzago – CMS – INFN/Padova, Italy
Anne-Marie Gaillac – CMS – IN2P3/ Ecole Polytechnique, France
Claudio Grandi – CMS – INFN/Bologna, Italy
Veronique Lefebure – CMS – CERN, Switzerland
Barry MacEvoy – CMS – PPARC/Imperial College London, United Kingdom
Owen Maroney – CMS – PPARC/University of Bristol, United Kingdom
Henry Nebrensky – CMS – PPARC/Brunel, United Kingdom
Igor Semeniouk – CMS – IN2P3/Ecole Polytechnique, France
Heinz Stockinger - CMS/EDG – CERN, Switzerland

Hugh Tallini – CMS – PPARC/Imperial College London, United Kingdom
Marco Verlato – CMS – INFN/Padova, Italy

Ingo Augustin – EDG – CERN, Switzerland
Jean-Jacques Blaising – EDG – IN2P3/LAPP, France
Stephen Burke – EDG – PPARC/RAL, United Kingdom
Andrea Chierici – EDG – INFN/CNAF, Italy
Alessandro Cavalli – EDG – INFN/CNAF, Italy
Vincenzo Ciaschini – EDG/EDT – INFN/CNAF, Italy
Laurence Field – EDG – PPARC/RAL, United Kingdom
David Groep – EDG – NIKHEF, The Netherlands
Fabio Hernandez – EDG – IN2P3/CC Lyon, France
Alessandro Italiano – EDG – INFN/CNAF, Italy
Peter Kunszt – EDG – CERN, Switzerland
Nadia Lajili – EDG - IN2P3/CC Lyon, France
Erwin Laure – EDG – CERN, Switzerland
Emanuele Leonardi – EDG – CERN, Switzerland
Charles Loomis – EDG – IN2P3/LAL, France
Francesco Prelz – EDG – INFN/Milano, Italy
Mario Reale – EDG – INFN/CNAF, Italy
Markus Schulz – EDG – CERN, Switzerland
Andrea Sciaba' – LCG/EDG – CERN, Switzerland
Massimo Sgaravatto – EDG – INFN/Padova, Italy
Jeffrey Alan Templon – EDG – NIKHEF, The Netherlands
Gennaro Tortone – EDG – INFN/Napoli, Italy

Appendix B

Chronology of events during the Stress Test

- 26 Nov: CMS validation of IN2P3 site from Ecole Polytechnique UI
- 26 Nov: CMS validation of RAL site from Imperial College UI
- 26 Nov: CERN CE configuration modified in order to allow jobs writing to local disks of the WNs
- Since 27 Nov: EDG release 1.3.4 deployed at all the sites
- 27 Nov: CMS validation of CNAF site from CNAF UI
- 27 Nov: CMS validation of CERN site from Padova UI
- 28 Nov: CMS validation of Legnaro site from Padova UI
- 29 Nov: Job submission from CNAF and Ecole Polytechnique UIs
- From 29 Nov: Few percent of failures in file registration into the RC with many concurrent jobs
- From 30 Nov: Many jobs (~30-40 jobs) asking for matchmaking with InputData seems to "trigger" a higher rate of aborted jobs due to II/MDS problems. Low submission rate adopted
- 30 Nov: Job submission started from Padova UI
- 30 Nov: Added more resources (WNs) at Legnaro site.
- 1 Dec: Problems with CMS CNAF RB: jobs always in the "Done" status never reaching the "OutputReady" status and then all jobs stuck in the "Ready" status
- 2 Dec: NIKHEF and Lyon SEs missing from the Information system, top level MDS was restarted
- 2 Dec: NFS problems with CNAF SE
- 2 Dec: High load of CERN CE related to tutorial activities, giving many aborted jobs due to Globus failure
- 3 Dec: The GARR (National Italian Network) disconnected CNAF for a couple of minutes
- 4 Dec: Job submission started from Imperial College UI

- 4 Dec: Produced a total of ~50000 CMSIM events
- 5 or 6 Dec: Restart of MDS top level server
- 5 Dec: SE CNAF unavailable because of a stale ethernet link
- 5 Dec: Padova UI operating system had to be re-installed
- 6 Dec: Coordinated upgrade of Padova UI and CMS CERN RB to EDG 1.4.0, in order to test the new release before its official deployment on the Application Testbed
- 6 Dec: RC got stuck
- 6 Dec: A limitation of the Globus RC to deal with large collections was found, so New RC collections, one for each submitting UIs, were created to avoid too many entries in a single collection
- 7 and 8 Dec: MDS instability; only NIKHEF and CERN CEs were visible and a restart of the top level MDS was needed. CNAF server was not responding and it was removed from the policy file
- 8 Dec: CMS validation of NIKHEF site from Ecole Polytechnique
- 9 Dec: The whole Application Testbed was upgraded to EDG 1.4.0
- 10 Dec: Ldap at CNAF was stuck and no CNAF, Legnaro and Padova resources were available from dbII
- 10-11 Dec: CMS CNAF RB was refusing jobs because of a crash of the CondorG *schedd* process due to a too low value for the *file-max* parameter. This parameter was properly increased and the RB restarted
- 10-11 Dec: RC connection problems: when submitting 20-30 concurrent jobs the RC started responding slowly and then it got stuck. An Ldap restart on the machine hosting the RC was needed.
- 11 Dec: A new RC (an exact clone of the first one, on the same machine on a different port, but completely empty) was set up at CNAF
- 11 Dec: LB Interlogger down on CMS CERN RB
- 11 Dec: CMS validation of Padova site from Padova UI
- 11 Dec: High rate of aborted jobs due to "Standard out..." at CNAF at the end it turned out that the disk with /home in CNAF CE was not properly working
- 11 Dec: Top-level MDS got stuck because IN2P3 GRIS was not responding, basically hanging all job submissions. The IN2P3 CEs were removed from MDS and dbII at CERN cleaned up. It was too late to ask CNAF admins to cleanup their dbII so CNAF RBs weren't usable
- 12 Dec: Reached a total of ~100000 CMSIM events
- From 12 Dec: Slow submission of "short" jobs in order not to have the RC stuck. Increasing the submission rate usually yields to an increase of registration failures into the RC
- 12 Dec: Gatekeeper mysteriously not running at CNAF
- 12 Dec: Imperial College RB, CE, SE upgraded to EDG 1.4.0
- From 12 to 16 Dec: Produced ~85000 CMSIM events in these days
- 14-15 Dec: II stuck thus aborting most of the submitted jobs
- 16 Dec: Failure in grid-mapfile update for CNAF SE. The script to create the grid-mapfile was hanging because Ldap server at NIKHEF was not reachable and the failsafe mode of the updating procedure, preventing to replace the grid-map-file with an empty one failed
- From 16 to 20 Dec: about 85000 CMSIM events produced in these days
- 17 Dec: All jobs running at CNAF CE failed with "Standard out...", probably related to major problems with DNS overload at CNAF
- 18 Dec: CNAF CE updated with a new PBS script and CMS CNAF RB updated with “JSS-Maradona” and other WP1 fixes (pre release of 1.4.1)
- 18 Dec: Reached 200000 CMSIM events
- 19 Dec: Legnaro was disconnected because of a network crash
- 20 Dec: Total of about 270000 CMSIM events produced; end of the Stress Test