The CMS Remote Analysis Builder (CRAB)

D. Spiga, S. Lacaprara, W. Bacchi, M. Cinquilli, G. Codispoti, M. Corvo, A. Dorigo, A. Fanfani, F. Fanzago, F. Farina, M. Merlo, O. Gutsche, L. Servoli, and C. Kavka

University of Perugia and INFN of Perugia
06100 via pascoli 10
Perugia, Italy

Abstract. The CMS experiment will produce several Pbytes of data every year, to be distributed over many computing centers geographically distributed in different countries. Analysis of this data will be also performed in a distributed way, using grid infrastructure. CRAB (CMS Remote Analysis Builder) is a specific tool, designed and developed by the CMS collaboration, that allows a transparent access to distributed data to end physicist. Very limited knowledge of underlying technicalities are required to the user. CRAB interacts with the local user environment, the CMS Data Management services and with the Grid middleware. It is able to use WLCG, gLite and OSG middleware. CRAB has been in production and in routine use by end-users since Spring 2004. It has been extensively used in studies to prepare the Physics Technical Design Report (PTDR) and in the analysis of reconstructed event samples generated during the Computing Software and Analysis Challenge (CSA06). This involved generating thousands of jobs per day at peak rates. In this paper we discuss the current implementation of CRAB, the experience with using it in production and the plans to improve it in the immediate future.

1 Introduction

The CMS experiment (Compact Muon Solenoid) [1] is one of the four physics experiments that will collect data at the Large Hadron Collider (LHC) [2] located at CERN.

The expected rate of event to disk will be about 150 Hz, so few PBytes of data per year will be stored and processed. At the same time, the experiment needs also to use the computational resources for the simulated data generation. The choice of CMS to cover all these needs is a distributed architecture and the use of the grid middleware components.

A hierarchy of computing regional centers, called Tiers, is defined in the CMS Computing Model. The system is geographically distributed and includes a single Tier-0 center at CERN, a CMS Analysis Facility also at CERN, few Tier-1 centers and many Tier-2 centers with different level of resources. In order to manage the data and optimize the use of the distributed resources, a combination of generic Grid tools, provided by the LCG (LHC Computing Grid) [3] and OSG (Open Science Grid) [4] projects, as well as specialized CMS tools are used together.

CMS has also to provide a single interface to physicists, capable to operate with all grid components and different back-ends in a transparent way. In this paper the CMS analysis model and the tools used to perform it are discussed.

2 CMS Data Model

The CMS computing model defines that the data collected by the CMS online data acquisition system is sent to the Tier-0 center at CERN where raw data is archived. A prompt reconstruction is performed and a first version of the Analysis Object Data (AOD) is produced.

All the high-level physics objects are stored in the AOD together with the information sufficient to support typical analysis. Raw and first pass reconstructed events are distributed from the Tier-0 to a Tier-1 centers which takes custodial responsibility for those while the AOD are also transferred to all Tier-1. The Tier-1 centers provide services for data archiving, re-processing, calibration, skimming and other data-intensive analysis tasks. All AOD and a fraction of the first pass reconstructed events and RAW data are transferred to Tier-2 centers which provide resources for physics analysis.

The computing model foresees that all CMS users must use the Grid in order to perform its own analysis.

3 User Analysis and CRAB

CRAB has been designed to allow physicists to access efficiently distributed data hiding the complexity of Grid infrastructure. Following the analysis model, the user runs interactively over small data samples in order to develop and test his code, using CMS analysis framework (CMSSW). Once ready, the user uses CRAB from a User Interface,

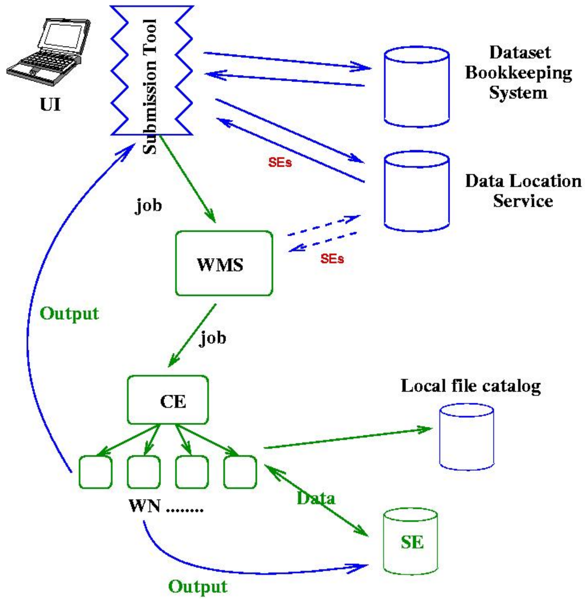

Fig. 1. CRAB work flow and interaction with the Grid and Data Management

where grid client middleware is available, to access to all data available on all remote sites. The work flow covered by CRAB can be factorized on three main steps:

- The Data Discovery step, interacting with the CMS data management infrastructure, to know if required data are found and where are located;
- The interaction with CMSSW on local machine, so that the very same environment can be reproduced on the remote resources;
- The Grid specific step, where all actions, from submission to output retrieval, are performed;

A user interacts with CRAB through simple configuration file which is arranged in several specific sections. Here all job specific parameters are defined, such as: the dataset that the user wants to access, the name of CMSSW specific configuration file, the job splitting parameters, how to manage the produced output, etc.

It is important to note that the very same CMSSW configuration, which the user has used interactively can also be used for remote data access. CRAB will take care to apply any changes needed to run on selected dataset on remote sites. The typical work flow (figure 1) is the following:

- *the input data discovery* to determine the Storage Elements (SE) of sites storing data. They are found interacting with CMS specific services, Data Bookkeeping and Location Services (DBS and DLS);
- *the packaging of user code*, to create a tar archive with user code which contains executable, library and user data files, as found on user local environment on User Interface;
- *the job preparation*, which consists in creation of a wrapper script. It sets up the running environment, performs integrity check on remote resources (WN), launches the executable and finally handles the output;
- *the creation of grid job configuration* (jdl), used by the Resource Broker (RB) the requirements needed for resources matchmaking and job running;
- *the job splitting* according to user requests and data distribution;
- *the job submission* to the Grid performed via BOSS [6];
- *the monitoring of the jobs*, in order to check the status of jobs;
- *the output retrieval* and the handling of user output. Currently, CRAB supports the copy of users output to an UI or to a generic Storage Element (SE) or to any host with a gsiftp server (e.g. CASTOR).

Other useful functionalities are the job killing, the job resubmission and the postmortem analysis, for debugging purpose, etc.

3.1 Experience Using CRAB

CRAB has been used with success for more than two years by CMS physicists, to perform data analysis. The first intensive usage of the tool by a large number of users from different places was during the Spring 2006 for the Physics TDR [7] preparation.

Moreover, CRAB has been used to access data during the CMS data challenges. The last one was CSA06, in that case millions of simulated events were analyzed, reaching peaks of 100’000 submitted jobs per month.

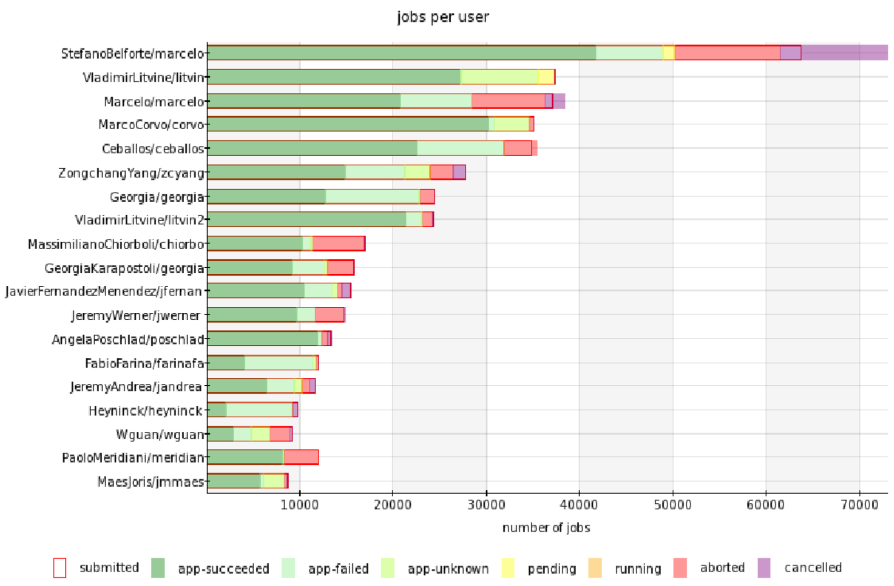

Fig. 2. The top 20 users during the first four 2007 months

During the Magnet Test Cosmic Challenge (MTCC), CRAB was even successfully used to access the real data distributed among several sites, the first real data for CMS. There are about $\sim$ 300 CRAB users distributed around the world, that means a daily rate of submitted jobs which reaches peaks of 10'000 jobs. In figure 2 the top twenty CRAB users in the first part of 2007, sorted by number of submitted jobs, are shown. The different color of the bars represents the different job status. In figure 3 the submission rate is shown, referred to the same period. In this plots each color represents a different site where the jobs run. The total efficiency is currently of order of 80%. The most important causes of the failure rate are related to the input data and to the middleware infrastructure. So the resulting inefficiency is not directly dependent on CRAB which indeed doesn't introduce a relevant fraction of jobs failures.

3.2 CRAB Improvements: Motivations and Implementation

The actual work flow of CRAB is based on a direct submission from the UI, where the user is working, to LCG and OSG via RB. This *standalone* model has the advantage of simplicity, but it lacks some features, which can be provided by a more advanced architecture *client-server* where a server is placed between the user and the Grid to perform a set of actions for him. The main goals of the client-server architecture is to automate as much as possible the whole analysis work flow and to improve the scalability of the system. The aim of the project is also to create a tool which is easy to use for physicists and easy to maintain for administrators.

The client-server implementation is transparent to the end users: the interface, the installation, the configuration procedure and the usage remains exactly the same as for standalone. The general CRAB client-server architecture is shown in figure 4.

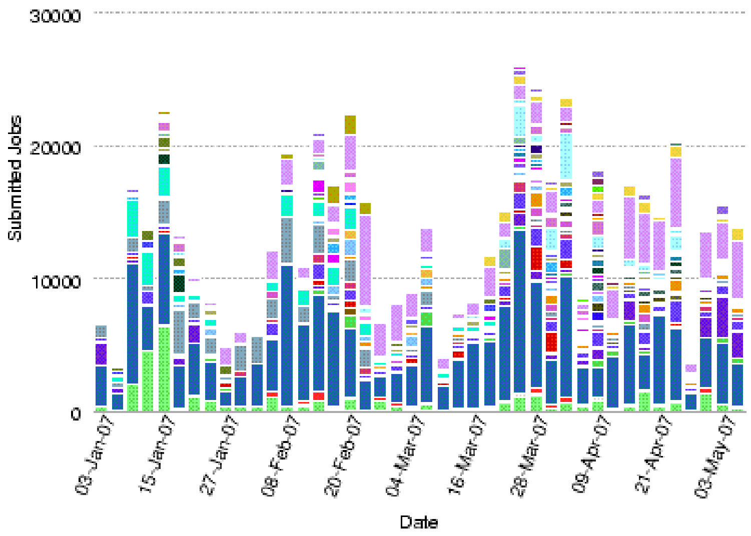

Fig. 3. The submission rate during the first four 2007 months. Each color represents a different CMS site where the jobs run.

The server architecture is based on components implemented as independent agents communicating through an asynchronous and persistent message service, plus a gridftp server which allows all communication from client to server. The user proxy is shipped using a WS-Delegation compliant service for proxy delegation.

The server core is a MySQL [8] DB of which the distinct components publish and subscribe messages to communicate. This architecture is as similar as possible to the CMS production system ProductionAgent [9] sharing components where possible, allowing an easier maintenance of the WorkLoad Management tools.

The role of the client is to interact with DBL/DLS for the data discovery, to prepare the jobs reading the local environment and finally to send the user proxy and the prepared task to the server.

The server manages the project interacting with the Grid from the job submission to the output retrieval on the name of the user. The actual server implementation provides the following components:

DropBoxGuardian: to check the dropBox, which is the container where the client put the user tasks, for new stuff to be managed. It also monitors the delegation service to check new proxy arrivals;

ProxyTarAssociator: Associates the task to the right user proxy and localizes the task configuration files w.r.t. the specific server instance;

CrabWorker: Submits jobs to the Grid in the CRAB style. It fully reuses the submitter components of CRAB (which supports: EDG, glite, glite-bulk, condor_g) It has some task-level resubmission features;

TaskTracking: Keeps general informations about all tasks under execution (e.g. status of the task, percentage of completed jobs in the task, etc);

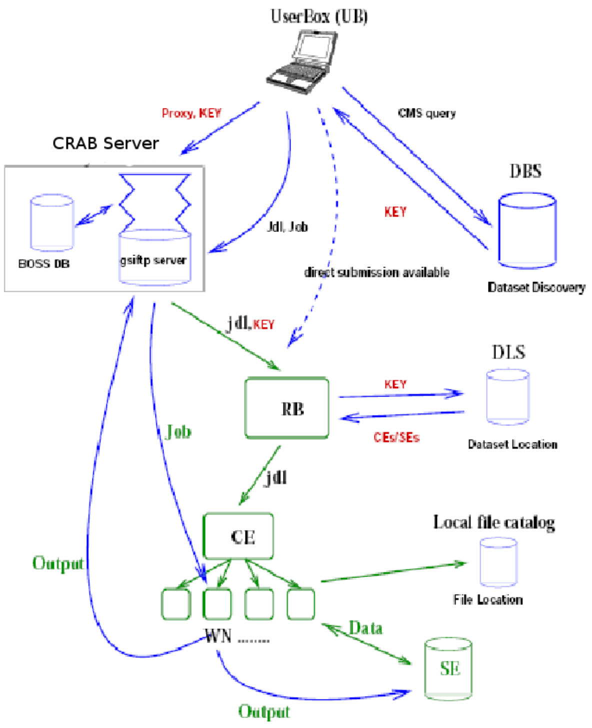

Fig. 4. CRAB server work flow and interaction with the Grid and Data Management

Notification: Notifies via e-mail the users when their tasks are ready/failed;
JobTracking: Tracks the status for every single job querying the grid and caching the infos on the local DataBase;
ErrorHandler: Handles errors of jobs. Depending on the type of error (e.g. job run error) it will initiate the appropriate error handler and update the job state;
JobSubmitter: Resubmits single jobs if needed;
RSSFeeder: Provides multiple RSS channels which can be used to forward important informations to the server administrator.

4 Conclusions

CRAB has been used since summer 2005 by several hundreds of end users distributed all over the world, it has reached more then 100’000 jobs/month with a daily record of 10’000 jobs. CRAB was also extensively used by hundreds of physicists to access data for the Physics TDR preparation, which was published during spring 2006. The CRAB project is still under development in order to satisfy the users and the whole CMS computing requirements for the data analysis.

At the moment the key effort of the development activity is devoted to the client-server implementation. The first version is already released and it is now under commissioning to prepare the next CSA07 challenge (Computing, Software and Analysis CMS challenge). Today the work flow implements the whole client interaction with the server, the actual submission from the server to the Grid, Job and Task tracking, simple error handling, retrieval of the produced output, and the user notification of the task status.

The next development steps are dedicated to cover other use cases still missing, such as job killing, disk space management and a web interface in order to show the status of jobs submitted to the server.

References

1. The CMS experiment, `http://cmsdoc.cern.ch`
2. The Large Hadron Collider Conceptual Design Report CERN/AC/95-05
3. L.C.G Project: (June 2005), `http://lcg.web.cern.ch/LCG and LCG Technical Design Report, CERN-TDR-01 CERN-LHCC-2005-024`
4. OSG Project: `http://www.opensciencegrid.org`
5. The Computing Project CERN/LHCC, 2005-023 CMS TDR 7
6. Batch Object Submission System, `http://boss.bo.infn.it/`
7. Physics Technical Design Reports, CERN/LHCC/, 2006-021
8. `http://www.mysql.com/`
9. `https://twiki.cern.ch/twiki/bin/view/CMS/ProdAgent`