PROCEEDINGS OF SCIENCE

FERMILAB-CONF-08-691-CD

Distributed analysis with CRAB: the client-server architecture evolution and commissioning

G. Codispoti[*] [†]
INFN and Università di Bologna
E-mail: Giuseppe.Codispoti@bo.infn.it

M. Cinquilli
INFN Perugia
E-mail: Mattia.Cinquilli@cern.ch

A. Fanfani
Università di Bologna
E-mail: Alessandra.Fanfani@bo.infn.it

F. Fanzago
CERN and INFN CNAF
E-mail: fanzago@cern.ch

F. Farina
CERN and INFN Milano Bicocca
E-mail: fabio.farina@cern.ch

S. Lacaprara
INFN Legnaro
E-mail: Stefano.Lacaprara@pd.infn.it

V. Miccio
CERN and INFN CNAF
E-mail: Vincenzo.Miccio@cern.ch

D. Spiga
CERN and INFN and Università di Perugia
E-mail: Daniele.Spiga@pg.infn.it

E. Vaandering
FNAL
E-mail: ewv@fnal.gov

CRAB (CMS Remote Analysis Builder) is the tool used by CMS to enable running physics analysis in a transparent manner over data distributed across many sites. It abstracts out the interaction with the underlying batch farms, grid infrastructure and CMS workload management tools, such that it is easily usable by non-experts.
CRAB can be used as a direct interface to the computing system or can delegate the user task to a server. Major efforts have been dedicated to the client-server system development, allowing the user to deal only with a simple and intuitive interface and to delegate all the work to a server.
The server takes care of handling the users jobs during the whole lifetime of the users task. In particular, it takes care of the data and resources discovery, process tracking and output handling. It also provides services such as automatic resubmission in case of failures, notification to the user of the task status, and automatic blacklisting of sites showing evident problems beyond what is provided by existing grid infrastructure.
The CRAB Server architecture and its deployment will be presented, as well as the current status and future development. In addition the experience in using the system for initial detector commissioning activities and data analysis will be summarized.

XII Advanced Computing and Analysis Techniques in Physics Research
November 3-7 2008
Erice, Italy

---

*Speaker.
†Presented on behalf of the CMS collaboraton.

1. Introduction

The Compact Muon Solenoid (CMS)[1] experiment at CERN LHC[2] is going to collect up to 2 PetaBytes of data every year. Data will be spread over the several sites all around the world: the CMS collaboration counts 183 institutes from 38 countries. Data will be accessed for the whole experiment lifetime for reprocessing and analysis through a distributed system based on the Grid middleware. The system is geographically distributed, organized in a hierarchical order of regional computing centres: a single Tier-0 centre at CERN for prompt data reconstruction; few Tier-1 centres having custodial responsability for raw and reconstructed data and providing services for data-intensive analysis tasks; several Tier-2 centres replicating a fraction of reconstructed data and high level analysis object and providing also Monte Carlo simulation facilities.

A Cern Analysis Facility (CAF) is also foreseen for low latency activities requiring access to all raw data: it combines flexible CPU resources with rapid access to the entire CMS dataset for fast analysis. The tiered architecture is represented in figure 1.

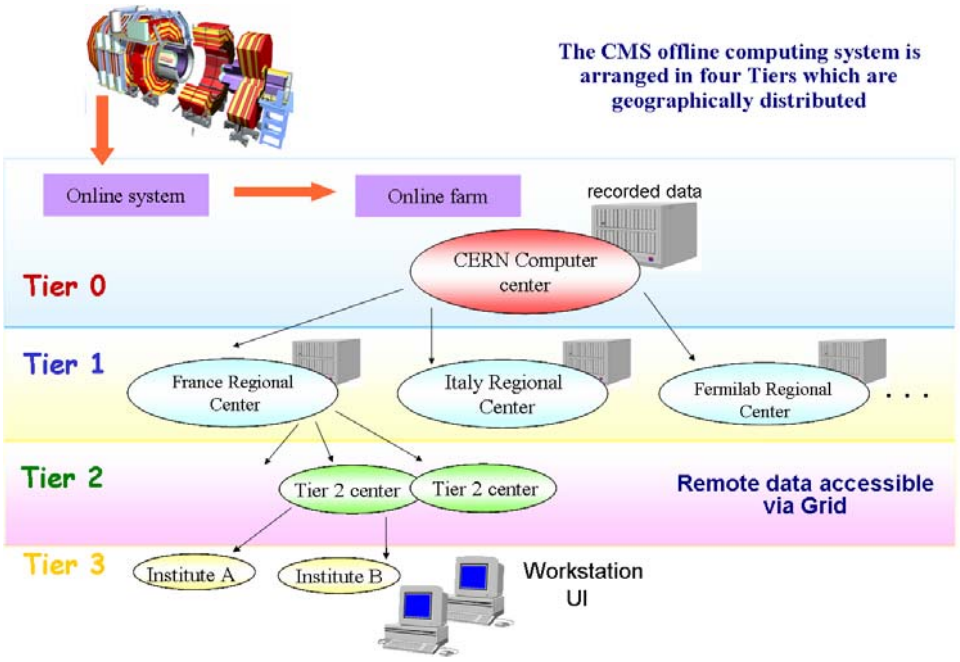

Figure 1: CMS Computing Model: the tiered architecture.

In order to allow the user analysis, CMS developed a set of tools to hide the complexity of the system to the final user. CMS Remote Analysis Builder (CRAB)[3][4][5] is the tool designed to abstract out the interaction with the underlying batch farms, grid infrastructure and CMS workload management tools, such that it is easily usable by users non-expert users.

Being the CMS computing model data location driven, the user analysis runs where data are located. CRAB takes care of interacting with the data management infrastructure to retrieve data availability and location.

To make an optimal use of the distributed environment, the user task is split in several jobs, accessing each a part of the dataset to be analyzed, so that they can run in parallel and reduce the solar time needed to extract high statistic results.

The user may develop its own code for the analysis using the framework provided by CMS (CMSSW). Once tested over a local small sample, it has to configure few parameters, specifying its code area, the dataset to be accessed and optionally a destination for the jobs output. Few more parameters may be also needed for particular configurations. Then all the work should be delegated to the CRAB installation, sitting on top of a Grid User Interface (UI). A simple and intuitive batch-like command line allows the user to submit, check the job status and eventually retrieve the output but also to track grid failures and cancel jobs.

CRAB implementation is able to deal both with the WorldWide LHC Computing Grid (WLCG)[6] and Open Science Grid (OSG)[7] through a plug-in system. A plug-in is also provided for local LSF submission, enabling easy access to the CAF facilities using the very same, well defined interface.

The interaction with the underlying system can be direct, leaving to the user tasks such as submission, status check and output retrieval, or using an intermediate Analysis Server. Indeed, even if the set of operation is already very limited, most of the workflow can be still automated, leaving to the user just the preparation of the configuration file and notifying him for the output availability. A system developed with this main purpose has the advantage of being integrated with the Grid infrastructure and the CMS data and workflow management system. This allows any new feature or automation to be easily implemented in a single tool, providing at needing new services to the user while keeping transparent the usage of the resources. Indeed the main reasons for an analysis server are:

- automate as much as possible the whole analysis workflow;
- reducing the unnecessary human load, moving all possible actions to server side, keeping a thin client as user interface;
- automating as much as possible the interaction with the Grid, performing submission, resubmission, error handling, output retrieval, post-mortem operations
- and allowing better job distribution and management.

Such a centralized system represents a collection point for user job logging, error detection and resolution, grid and site monitoring and in general a single point to be accessed by the expert to debug and improve the whole system. At the same time it does not represent a single point of failures, since many CRAB Server instances will be deployed at different sites. The user chooses the server through the CRAB configuration file. No interactions are foreseen among different CRAB servers.

2. Server Architecture

The server architecture adopts a modular software approach, made of independent components (Agents) implemented as daemons. In some cases a multithreading approach has been implemented to avoid bottlenecks in the most intensive operations such as submission, job status poll and output rearrangement.

The communication among Agents is performed through an asynchronous and persistent message service based on a "publish & subscribe" model. The message service is provided by a MySQL Database, which also provide job logging and bookkeeping facilities.

Another important external component is represented by a Storage Element, storing user input and output data. The default implementation is GridFTP based, but other flavours are also supported.

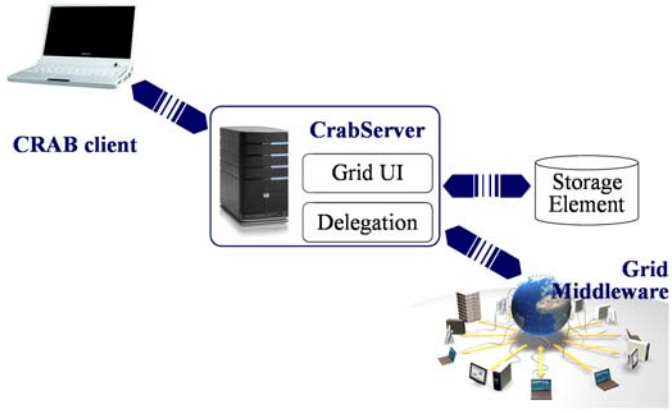

Figure 2: The CRAB Server architecture and role.

The communication between the client and the server is implemented by using the web services technology, based on SOAP.

The implementation shares many components and libraries with others CMS Workload Management Tools, reusing code and sharing efforts for improvements, in particular the Agents are partially shared with the Monte Carlo Production System (ProdAgent)[8]. The following components are used by a CRAB Server instance:

- CommandManager: the endpoint of the SOAP services which handles commands sent by the CRAB client;
- TaskRegister: registers task entities in the database;
- CrabWorker: performs direct job submission to the Grid;
- TaskTracking: collects all the general information about tasks under execution and organizing them for user internal logging and user notification;
- Notification: notifies the user by an e-mail when his task is ended and the output has been already retrieved; it is also used to notify the server administrator for special warning situation;
- TaskLifeManager: manages the task life on the server cleaning ended tasks;
- JobTracking: tracks the job status interacting with the underlying middleware or batch system;

- GetOutput: retrieves and/or organize output generated by finished jobs and log grid failures for grid aborted jobs;
- JobKiller: on demand service for job cancellation;
- ErrorHandler: performs a basic error handling that allows resubmitting jobs; it can implement more complex logic to handle specific failures: in this phase we are collecting information to understand which behaviours can be safely implemented.
- RSSFeeder: provides RSS channels which can be used to forward information about the server status;
- AdminComponent: executes specific server maintenance operations;
- HTTPFrontend: provide a web interface for the user to monitor their task and the site administrator to monitor resources usage, dataset accessed, destination sites.

3. CRAB Server dedicated scale tests

The CRAB Server implementation needs to sustain high job submission rate and to be very reliable in the functionalities it provides. The most challenging aspects of the CRAB Server implementation are the concurrency among components and threads of the same component, user's certificate handling and consistency of the whole job flow, including recovery procedures in case of failures.

Dedicated commissioning tests are foreseen to probe the server's overall functionalities and to provide useful feedback to the development cycle in case of bottlenecks or misbehaviour of the server under stressing conditions.

During the Common Computing Readiness Challenge (CCRC08) in May 2008 the CRAB Server was used to simulate physics group activities submitting realistic user's analysis jobs lasting for approximately 4 hours. The aim was to test the readiness of sites filling their resources, reading datasets available at all sites and producing an output file that was staged-out remotely to a single site [9]. More than 100 kjobs on about 30 sites were submitted in two weeks and the CRAB Server provided the needed functionality for job submission and tracking.

In order to test the actual server scalability and reliability up to the expected CMS operational rates a dedicated test environment was setup. To better understand the Server load, the GridFTP based Storage Element was installed on a different machine, while a 4 CPU was used for CRAB Server components an dMySQL. The test was performed in the WLCG context using two dedicated gLite WMS. Monitoring information were collected from various sources like the CRAB Server database tracking the job flow and the CPU usage of its components, as well as from underlying services like MySQL and GridFTP server and gLite WMS monitoring [10]. The kind of jobs submitted were very short jobs running for less than few minutes, not reading a dataset and without stage-out. This choice was made to provide a harder environment for job handling due to continuous outputs and to limit the resource usage at sites.

The test was split in two phases: a controlled submission pattern originated by a single user and an emulation of a multi-users environment.

For the first phase, a single user performed a controlled job submission pattern, reaching almost 200 kjobs were submitted spread over more than 50 sites with a rate of about 50 kjobs/day, as shown in Figure 3. The CRAB Server was able to cope with that rate with no indication of reaching a breaking point. No significant backlogs were accumulated. The main contributors to the load of the system were the MySQL and GridFTP usage.

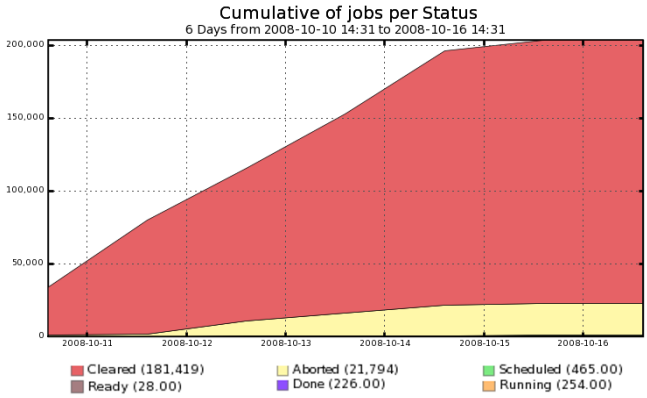

Figure 3: Cumulative distribution of jobs submitted to CRAB Server during the single user test phase.

In the multi user environment phase of the test the job submission were from different users certificates, thus emulating the CRAB Server usage by more users. The needs to further stress the multi-user environment arise from its higher complexity. Single threads of the very same component may be handling different user's jobs, requiring different certificates. Any interaction among them may bring to strong failures stopping the component work and accumulating backlog.

Different submission patterns were adopted by the 12 users involved producing scheduled activity peaks and including also random submission. No CRAB Server misbehaviour was identified due to the multi-user environment and 120 kjobs were successfully handled in 48 hours with peaks of 2-3 Hz, as shown in Figure 4. During the test indications for future improvements in the job retrieving form the Grid has been obtained.

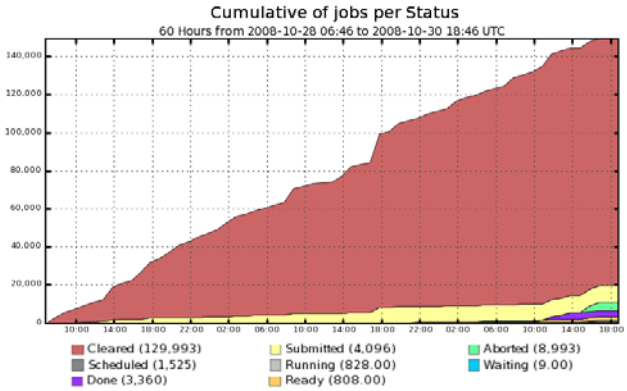

Figure 4: Cumulative distribution of jobs submitted to CRAB Server during the multi-user test phase.

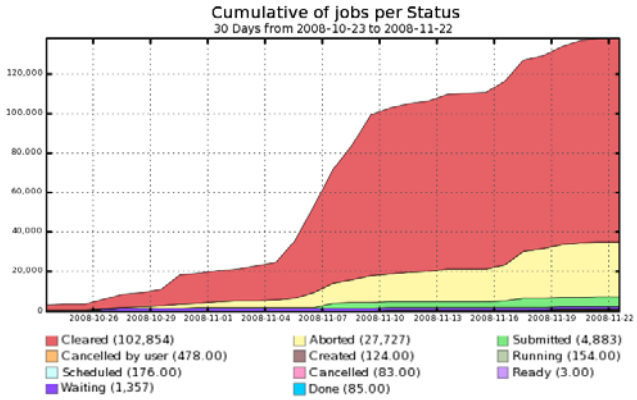

Figure 5: Cumulative jobs handled by a single CRAB Server instance in one month.

4. Usage and experience

In 2008 the number of distinct CRAB users is more than 700, which is approaching the 40% of the whole CMS community. About 100 users per day submit through CRAB analysis jobs over simulated dataset and cosmic ray samples.

The CRAB Server it's becoming more popular given the extra features it provides. Few server instances have been deployed in several countries and are being used by local community or whole CMS users. One of the latest instance deployed handled more than 120000 jobs in less than one month (figure 5).

An instance of CRAB Server is deployed also at CAF where it's used mainly for low latency calibration activities.

5. Future plans

Using the CRAB Server, the user analysis in CMS reached a high level automation level. The increasing usage by real user will give indication about the next steps to be taken in that direction. Among the possible usage, we can already identify the merge of all the job outputs; but also system fo moving and distributing the user results to make them available for the community. Furthermore, typical analysis workflows can be embedded in the server so that they can be used on demand by the user.

A good improvement in the tool usability can came from the extention of the user interfaces. In particular the web monitoring and in general the graphic views exposed to the user. This will lead automatically to an improvement to the monitoring tools also for the administrator.

References

[1] S. CHARTRCHYAN et al., The CMS Experiment at CERN LHC Jornal of Instrumentation, vol 3, pp.s08004 (2008)

[2] LHC Homepage, http://cern.ch/lhc-new-homepage/

[3] F. FARINA et al., Status and evolution of CRAB, ACAT 2007 - XI International Workshop on Advanced Computing and Analysis Techniques in Physics Research, April 23-27, 2007, Amsterdam, Netherlands, proceedings at POS (ACAT) 020

[4] CRAB homepage, https://twiki.cern.ch/twiki/bin/view/CMS/SWGuideCrab

[5] D. SPIGA et al., The CMS Remote Analysis Builder (CRAB) 14th Int. Conf. on High Performance Computing (HiPC 2007). Goa, India. Dec 18-21 2007. (vol. 4873, pp. 580-586). ISBN/ISSN: 978-3-540-77219-4.

[6] LHC Computing Grid (LCG), Web Page, http://lcg.web.cern.ch/LCG/ and LCG Computing Grid - Technical Design Report, LCG-TDR-001 CERN/LHCC 2005-024, (2005)

[7] OSG Web Page, http:/opensciencegrid.org/

[8] D. EVANS et al., The CMS Monte Carlo Production System: Development and Design, 18th Hadron Collider Physics Symposium 2007 (HCP 2007) 20-26 May 2007, La Biodola, Isola d'Elba, Italy. Published in Nuclear Physics B - Proceedings Supplements,ÂăVolumes 177-178 (2008) 285-286

[9] A. SCIABÀ et al., The Commissioning of CMS Computing Centres in the WLCG Grid N29-5, IEEE NSS 2008 Proceedings

[10] D. CESINI et al., WMSMonitor: a Monitoring Tool for Workload and Job Lifecycle in Grids, Proceedings of the 9th IEEE/ACM International Conference on Grid Computing (Grid 2008)