IL NUOVO CIMENTO Vol. 32 C, N. 2 Marzo-Aprile 2009
DOI 10.1393/ncc/i2009-10399-2

Colloquia: CSFI 2008

The Grid-distributed data analysis in CMS

F. Fanzago[1][2], F. Farina[1][3], M. Cinquilli[4], G. Codispoti[5][4], F. Fanfani[6], S. Lacaprara[7], E. Miccio[1][3], S. Spiga[1][4] and E. Vaandering[8]

([1]) *CERN - Genève, Switzerland*
([2]) *INFN-CNAF - Bologna, Italy*
([3]) *INFN, Sezione di Milano - Milano, Italy*
([4]) *INFN, Sezione di Perugia - Perugia, Italy*
([5]) *Università di Perugia - Perugia, Italy*
([6]) *Università di Bologna - Bologna, Italy*
([7]) *INFN, Sezione di Legnaro - Legnaro (PD), Italy*
([8]) *FNAL - Batavia, Illinois, USA*

(ricevuto il 22 Giugno 2009; pubblicato online il 10 Settembre 2009)

Summary. — The CMS experiment will soon produce a huge amount of data (a few PBytes per year) that will be distributed and stored in many computing centres spread across the countries participating in the collaboration. Data will be available to the whole CMS physicists: this will be possible thanks to the services provided by supported Grids. CRAB is the CMS collaboration tool developed to allow physicists to access and analyze data stored over world-wide sites. It aims to simplify the data discovery process and the jobs creation, execution and monitoring tasks hiding the details related both to Grid infrastructures and CMS analysis framework. We will discuss the recent evolution of this tool from its standalone version up to the client-server architecture adopted for particularly challenging workload volumes and we will report the usage statistics collected from the CRAB community, involving so far almost 600 distinct users.

PACS 07.05.Kf – Data analysis: algorithms and implementation; data management.

1. – Introduction

The CMS experiment is one of the four high-energy-physics experiment that will start to collect data in the 2009 at LHC accelerator at CERN. The large quantity of data foreseen to be produced per year (approximately 2PB) and to be stored, the big computing power required for analysis and data simulation and the need for data access to all the physicists of the collaboration throughout the world induced CMS to base its compunting model on the Grid concept.

The CMS computing model 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 re-processing, calibration, skimming and other data-intensive analysis tasks; several Tier-2 centres where a fraction of reconstructed data and high-level analysis object are replicated for user analysis 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 provides a rapid access to the entire CMS dataset for fast analysis.

The Grid infrastructure ensures remote-resources availability and assures remote-data access to authorized users, belonging to the CMS virtual organization. Grid services are provided by WorldWide LHC Computing Grid (WLCG) [1] and Open Science Grid (OSG) [2] to manage the distributed resources.

The data analysis in a distributed environment is a complex task. It assumes to know where data to analyze are located, how to access them, in which remote site the analysis software is installed and how to use the Grid infrastructure. In order to make the analysis on remote resources feasible, the CMS Collaboration is developing some tools interfaced with the already available Grid services, including a tool for data distribution among tiers (Phedex), specific catalogues for remote data discovery (DBS) and a tool to help users to create, submit and manage analysis job in the Grid anvironment: CRAB (CMS remote Analysis Builder) [3, 4].

2. – CRAB, the CMS Remote Analysis Builder

The purpose of CRAB is to allow users not knowing Grid infrastructure to run their analysis code on data available only at remote sites as easily as in a local environment, hiding Grid architecture details.

It consists of a userfriendly tool, written in python, to be installed on a UserInterface (the entry point for the Grid), where the software for the analysis (CMSSW) is available. The user has simply to develop its analysis code, to decide which data he want to analyze and how to manage the produced analysis output. This means either to retrieve it in the UI or to copy it to a remote storage element when the analysis job has finished. The user interacts with CRAB via a configuration file, by changing some parameter values such as the name of the dataset to analyze, the total number of events, the name of its CMSSW configuration file and by flagging the copy or retrieving of outputs and their publication.

Once CRAB has read its configuration file, its flow can be factorized in the following steps:

- Data discovery: it queries the DBS catalogue for data location. The query returns the list of sites where data are located and how they are distributed. This information is used in the job creation step.

- Job creation: during this step, CRAB performs the packaging of the analysis user code including executable, library and user data. Then it prepares the wrapper of the user analysis code, script to be executed on the remote farm. The wrapper sets up the correct running environment depending on the Grid middleware, launches the user executable and handles the output. CRAB creates also the jdl file, specific

script to submit job to the Grid whose requirements are the CMSSW version used by user and the site name hosting the data.

– Job submission and management: the submission and monitoring are performed using BossLite (Batch Object Submission System). The management includes the check of jobs' status, the possibility to kill and resubmit them. CRAB is able to submit jobs not only on Grid environment but also using local batch system as LSF(Load Sharing Facilities) for CAF.

– Output retrieval and publication: When jobs have finished, CRAB takes charge either of retrieving analysis outputs into the UI or copying them to a remote SE, specified by user in the crab configuration file. In order to make the analysis results available to physics groups, the user can decide to publish them into a "local" DBS. So published data can be re-analysed via CRAB.

CRAB provides the user with a simple command line, whose options are the CRAB functionalities, *i.e.* crab -create; crab -submit.

3. – CRAB evolution: the server

The major effort of the development activity during the last year is focused to the client-server implementation. The aim of this architecture is to automatize as much as possible the interaction with the Grid by reducing the unnecessary human workload demanding all possible actions to the server side, improving also the scalability of the whole system.

The client is on the User Interface, while the server could be located somewhere over the Grid. The CRAB client is directly used by physicists to perform the operations involved in the task/job preparation and creation, such as the data discovery, the input-sandbox preparation, the job preparation (included the job splitting) and the requirement definition. Then the client sends the CRAB server a request completely transparent to the user. The server manages: job submission to the Grid, automatic job monitoring, automatic job output retrieval, re-submission of failed jobs following particular rules for different kinds of job failures, every specific command requested by the user as the job killing, and e-mail notification when the percentuage of finished jobs of a task reachs a specified level to the user. The communication between the client and the server is implemented by using the web services technology, based on SOAP which is standard in the Grid service development community. It uses HTTP protocol and provides interoperability across application languages. The internal CRAB server architecture is based on components implemented as independent agents communicating through an asynchronous and persistent message service based on a MySQL database. Each agent takes charge of specific operations:

– CommandManager: the endpoint of the SOAP services which handles commands sent by the CRAB client;

– CrabWorker: component that performs direct job submission to the Grid;

– TaskTracking: keeps all the general information about tasks under execution, polling it from the database by using threads;

- Notification: notifies the user by an e-mail when his task has ended and the output has already been retrieved; it is also used to notify the server administrator about special warning situation;
- TaskLifeManager: manages the task life on the server by cleaning ended tasks;
- JobTracking: tracks the status of every job;
- GetOutput: retrieves the output of ended jobs;
- JobKiller: kills single or multiple jobs when required;
- ErrorHandler: performs a basic error handling allowing jobs resubmission;
- RSSFeeder: provides RSS channels which can be used to redirect information about the server status;
- AdminComponent: executes specific server maintenance operations;
- HTTPFrontend: provide a web interface to monitor submitted tasks, resources usage, dataset accessed, and destination sites.

Many of the above-listed components are implemented following a multithreading approach, that allows to manage many tasks at the same time shortening the delay time for a single operation that has to be accomplished over many tasks. Two important entities in the CRAB server architecture are the WS-Delegation and a specific area on an existing Storage Element. The WS-Delegation is a service for the user proxy delegation from the client to the server allowing the server to perform each Grid operation for a given task with the corresponding user proxy. The SE allows to transfer the input/output-sandboxes between the User Interface and the Grid, working as a sort of drop-box area. The server interacts with any associated storage server independently of the protocol allowing to have a portable and scalable object, where the Storage Element hosting the job sand-boxes is completely independent of the server.

4. – CRAB usage

CRAB has been in production since Spring 2004. The client-server architecture is born at the end of 2007. CRAB was used by CMS physicists to analyze data for the Physics Tecnical Design Report and during some LCG and CMS challenges such as, CSA06, CSA08, and CCRC08 with the aim to test the maturity of Grid infrastructure and CMS framework and to demostrate the feasibility of distributed analysis. Furthermore, CRAB is daily used to check the availability of remote resources, through an automatic tool to spread job continuously over all published data. Currently about 100 users per day submit through CRAB analysis jobs, reaching a peak of 70000 jobs per day and mantaining an average of about 10000 jobs per day.

5. – Conclusion

CRAB is the official distributed analysis tool of CMS. The aim of CRAB is to simplify the creation and the management of analysis jobs in the Grid environment. It was born as simple tool to be installed in the UI, but the need for scalability and robustness to face the foreseen number of jobs going to be submitted when real data are available

has driven its evolution to a client-server architecture. Currently, CRAB is able to perform three main kinds of job submission, totally transparent to the user: the direct submission to the Grid interacting directly with different middlewares (LCG and OSG); the direct submission to local resources and relative queues, using the batch system; the submission to the Grid using the CRAB server as a layer responsible of the interaction with the middleware and the task/job management.

CRAB is being extensively using by the CMS community, that provides ideas and use cases for new functionalities and improvements.

REFERENCES

[1] *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).
[2] *OSG* Web Page, `http:/opensciencegrid.org/`.
[3] Codispoti G. *et al.*, *CRAB: a CMS Application for distributed analysis*, preprint 2008 Nuclear Science Symposium.
[4] Spiga D. *et al.*, *The CMS Remote Analysis Builder (CRAB)*, in *Nucl. Phys. B*, **4873** (2007) 580.