LcgCAF: CDF access method to LCG resources

To cite this article: Gabriele Compostella et al 2011 J. Phys.: Conf. Ser. 331 072009

View the article online for updates and enhancements.

Related content
- Eurogrid: a new glideinWMS based portal for CDF data analysis
  S Amerio, D Benjamin, J Dost et al.
- CDF way to Grid
  Donatella Lucchesi and Cdf Collaboration
- CDF software distribution on the Grid using Parrot
  G Compostella, S Pagan Griso, D Lucchesi et al.
LcgCAF: CDF access method to LCG resources

Gabriele Compostella¹, Matteo Baice, Simone Pagan Griso¹, Donatella Lucchesi¹, Massimo Sgaravatto¹ and Marco Cecchi²

¹ INFN Padova, via Marzolo 8, 35131 Padova, ITALY
² INFN CNAF, viale Berti Pichat 6/2, 40127 Bologna, ITALY
E-mail: compostella@pd.infn.it

Abstract. Up to the early 2011, the CDF collaboration has collected more than 8 fb⁻¹ of data from pp collisions at a center of mass energy \( \sqrt{s} = 1.96 \) TeV delivered by the Tevatron collider at Fermilab. Second generation physics measurements, like precision determinations of top properties or searches for the Standard Model higgs, require increasing computing power for data analysis and events simulation.

Instead of expanding its set of dedicated Condor based analysis farms, CDF moved to Grid resources. While in the context of OSG this transition was performed using Condor glideins and keeping CDF custom middleware software almost intact, in LCG a complete rewrite of the experiment’s submission and monitoring tools was realized, taking full advantage of the features offered by the gLite Workload Management System (WMS). This led to the development of a new computing facility called LcgCAF that CDF collaborators are using to exploit Grid resources in Europe in a transparent way.

Given the opportunistic usage of the available resources, it is of crucial importance for CDF to maximize jobs efficiency from submission to output retrieval. This work describes how an experimental resubmission feature implemented in the WMS was tested in LcgCAF with the aim of lowering the overall execution time of a typical CDF job.

1. Introduction

The Collider Detector at Fermilab (CDF) is an experiment at the Tevatron collider where protons and antiprotons collide at an energy in the center of mass \( \sqrt{s} = 1.96 \) TeV. Thanks to the continuous improvements in the luminosity of the collider, currently CDF has more than 8 fb⁻¹ of data available for physics analyses, corresponding roughly to \( 10^{10} \) events. Such a large dataset allows to perform high precision physics measurements and wide ranges of searches for new phenomena: for example, the CDF collaboration has recently released a very accurate top quark mass measurement with an overall uncertainty on the top mass lower than 1 GeV [1], comparable in precision to the most recent combination for the top quark mass at the Tevatron [2]; as another example, the large amount of data collected allowed in Summer 2010 the CDF and D0 collaborations to combine their results on the search for the not yet observed Standard Model higgs boson and exclude a wide range of values for the mass of this particle, giving substantial hints on the possibility for the Standard Model higgs to be light [3].

In order to perform these kind of precision measurements, it’s necessary not only to process, analyze and produce all the data recorded by the detector, but also to produce a similar or even larger amount of simulated Monte Carlo events for each of the different physics processes under evaluation. To cope with the increasing computing power requirements for data analysis
and event simulation, CDF had to evolve its computing model, based on dedicated analysis farms (CAF) built on top of Condor pools [4], towards the Grid, following the example of many other experiments in High Energy Physics.

In the context of the OSG Grid, this transition was accomplished using Condor glideins as a natural way to expand the local pool towards Grid sites. Glidein jobs can be seen simply as properly configured Condor Startd daemons that allow to add new virtual machines to the CDF pool once they start running on the target Grid site. This solution allowed to keep the CDF custom software used for job submission and monitoring completely unchanged and to exploit a large amount of resources in the OSG Grid in an opportunistic way.

In the context of the LCG Grid, a different solution was employed. All the custom software tools used by CDF in its dedicated farms were completely rewritten and adapted to take advantage of the features offered by the gLite Workload Management System (WMS).

2. LcgCAF overview

The structure of LcgCAF as a job submission portal and its typical workflow are shown in Fig. 1. Users can submit their jobs using a custom command line tool or a Graphical User Interface (GUI) distributed with the CDF software base and available to all collaborators. These tools allow job submission to any of the CDF analysis facilities, and let users specify the type of job they want to submit, the executable they want to run, the tarball containing the files needed by the executable as well as the desired number of parallel segments of the job (sections): each section of a job runs the same executable, but uses a different range of input parameters. For example, this is useful in Monte Carlo jobs, where the same simulation code can be parallelized and different sections of the same job can simulate a chosen process for different run conditions of the detector, given as inputs. The LcgCAF portal headnode hosts the daemons responsible for job submission, monitoring and user notification. The submitter daemon is the service responsible for accepting the job submission requests, reading the submission parameters and creating the proper Job Description Language files (JDL) to be submitted to the WMS. The submitter also accepts and stores the user tarball in a disk location that can be accessed via a Web server. Once jobs are successfully accepted, the JDLs corresponding to their sections are stored in a local “First In First Out” (FIFO) queue and then each of them is submitted as a single Grid job to the WMS by another daemon running on the headnode, called job manager, using the WMS submission client commands available in the gLite Grid User Interface package. After the submission, the job is delegated to the WMS which is then responsible of finding computing elements with free resources available to run the submitted sections, choosing them from a list of sites that support CDF, as will be described in Sec. 3.

The job manager submits to the WMS a custom executable, called the LcgCAF wrapper, instead of the user specified executable. When the wrapper starts on the worker node, it forks...
Figure 2. Parrot implementation in LcgCAF. CDF code is distributed using a single code web server located at CNAF, and squid caches are used close to sites with a large number of worker nodes, like Gridka. Parrot supervises the running code, intercepts system calls and pauses the job when it tries to access files belonging to the CDF code distribution, expected to be located under the mount point /home/cdfsoft, and not available on the worker node. Parrot retrieves the required files via http from the code server and then releases the execution of the job.

additional monitoring processes on the machine, downloads the user tarball from the headnode web server and finally supervises the execution of the job. The monitoring processes are used to retrieve directly on the worker node, at fixed time intervals, informations on the job state, like the list of active processes or cpu and ram usage, and to communicate back to the portal headnode, where a monitoring daemon collects these data and summarizes the status of the system and of the different jobs in a set of web pages. Moreover, the job wrapper starts the execution of the file system virtualization software Parrot [5, 6], whose implementation in LcgCAF is shown in Fig. 2. This tool allows to distribute CDF specific software without any specific requirement on the worker node: Parrot is a user space program that supervises the job execution and, by trapping system calls, can retrieve from a code server the files needed by the application at run time, using the standard http protocol. This way, there is no need of installing CDF software on the worker nodes, as files required by the user application are downloaded when they are necessary. The usage of squid proxy caches close to large sites allows to speed up the data transfers when many jobs require the same files. A detailed explanation of Parrot implementation within LcgCAF can be found in Ref. [7].

When a job is complete, the wrapper copies its output to CDF data servers and the monitoring daemon on the headnode sends an email notification to the user.

3. WMS matchmaking
As discussed in the previous section, LcgCAF relies on the WMS to distribute jobs sections among the sites supporting CDF. Because of this, the performances of the matchmaking process are crucial and can have a sizable impact on users' jobs. The WMS uses informations published by the different sites to rank them according to rules that can be specified at submission time. Currently LcgCAF requires the WMS to rank sites starting from the one promising the lowest expected execution time of the job, using a custom ranking expression that takes into account the number of running and waiting jobs on the site and the number of available worker nodes. When informations published by the sites are not accurate or not up to date, or when the activity of other virtual organizations is heavy on a particular site, the ranking of the available resources can become far from optimal. Typically what happens in these situations is that different sections belonging to the same job are spread into multiple sites, with largely heterogeneous waiting times. Since a job is considered complete when all its sections have terminated successfully and its output is ready to be used in physics analyses, the overall completion time of a job corresponds to the completion time of its longest running section.
Figure 3. The plots show the number of running (top histogram in blue) and waiting sections (bottom histogram in red) versus time in a time span of two days, for the standard WMS configuration (a) and for the test WMS configuration with automatic resubmission (b). It can be seen in (a) that many sections keep waiting for several hours in busy sites, while this effect is removed in (b) thanks to the automatic resubmission. See text for details.

The qualitative result of this setup is that if a single section of a job is submitted to a very busy site, or stuck in a long waiting queue, the overall job completion time can become very large, even if most of the other sections of the job completed in a short time. This effect can be seen in Fig. 3(a). We submitted through LcgCAF 10 jobs each made of 100 sections, using the standard WMS configuration. The plot shows the number of running (top blue histogram) and waiting (bottom red plot) sections versus time in a time window of two days. It can be seen how, after their submission, many sections wait in queue at the site where the WMS matched them for a long time before being executed. The completion time of those sections is shown in Fig. 4(a), and the overall completion time of the submitted jobs to which they belong is shown in Fig. 4(c).

At the same time, we performed an identical submission experiment of 10 job of 100 sections, using the same list of supported sites, with an experimental WMS implementing an automatic resubmission feature. Thanks to this feature, the WMS automatically resubmits a section, after a new matchmaking, if it doesn’t run within 30 minutes after the previous submission. The number of running and waiting sections versus time during the same period of two days is shown in Fig. 3(b). We can see from the plot how, thanks to the resubmission feature, there is no section stuck waiting in a site queue. The effect of the resubmission on the completion time
Figure 4. Top plots show the completion time of the sections submitted through LcgCAF during a 10 jobs with 100 sections submission test using a standard WMS (a) and a WMS with an experimental resubmission feature (b). Bottom plots show the overall completion time of the jobs submitted through LcgCAF, calculated as the time needed for the longest section of a job to complete, using a standard WMS (c) and a WMS with an experimental resubmission feature (d).

of the sections is shown in Fig. 4(b), while the overall completion time of these jobs is shown in Fig. 4(d). We can note how the resubmission feature, in this preliminary test, helps in lowering the overall execution time of a typical CDF job, meant as a collection of sections.

4. Conclusions
LcgCAF represents a powerful tool for CDF to access LCG Grid resources in Europe, that allows a substantial expansion of the computing power available to the collaboration, fundamental for the production of high quality physics results. Thanks to its setup and to products like Parrot, LcgCAF maintains the same functionalities of the other dedicated and glidein based CDF analysis facilities, making the access to the Grid completely transparent to users.

Issues with WMS matchmaking and ranking features can have a sizable impact on the completion time of the jobs managed by LcgCAF, but the effect of the few longest sections on typical CDF jobs can be mitigated using an experimental WMS automatic resubmission feature. This paper describes a preliminary test of this feature showing promising results, further tests and more fine tuning will be necessary to understand if this feature alone can be sufficient to solve LcgCAF job submission issues.
References

[1] Aaltonen T et al. (CDF Collaboration) 2010, Top Quark Mass Measurement in the lepton+jets Channel Using a Matrix Element Method and in situ Jet Energy Calibration, Phys. Rev. Lett 105 252001.

[2] The Tevatron Electroweak Working Group 2010, Combination of CDF and D0 Results on the Mass of the Top Quark, FERMILAB Report No. FERMILAB-TM-2466-E

[3] Tevatron New Physics, Higgs Working Group 2010, Combined CDF and D0 Upper Limits on Standard Model Higgs-Boson Production with up to 6.7 fb$^{-1}$ of Data, arXiv:1007.4587v1

[4] Litzkow M J, Livny M and Mutka M 1988 Condor - a hunter of idle workstations Eighth International Conference of Distributed Computing Systems

[5] Thain D and Livny M 2003 Parrot: Transparent user-level middleware for data-intensive computing, Workshop on Adaptive Grid Middleware, New Orleans

[6] Parrot official Web site: http://www.cse.nd.edu/~ccl/software/parrot/

[7] Compostella G, Pagan Griso S, Lucchesi D, Sfiligoi I and Thain D 2010, CDF software distribution on the Grid using Parrot J. Phys.: Conf. Ser. 219 062009.