FERMILAB-PUB-08-552-E

International Conference on Computing in High Energy and Nuclear Physics (CHEP'07) IOP Publishing
Journal of Physics: Conference Series **119** (2008) 072025 doi:10.1088/1742-6596/119/7/072025

CDF experience with Monte Carlo production using LCG Grid

S. Pagan Griso[1], D. Lucchesi[1], G. Compostella[2], I. Sfiligoi[3], D. Cesini[4]

[1] INFN and University of Padova
[2] University of Trento and INFN Padova
[3] Fermilab
[4] INFN-CNAF

E-mail: simone.pagan.griso@pd.infn.it

Abstract. The upgrades of the Tevatron collider and CDF detector have considerably increased the demand on computing resources, in particular for Monte Carlo production. This has forced the collaboration to move beyond the usage of dedicated resources and start exploiting the Grid. The CDF Analysis Farm (CAF) model has been reimplemented into *LcgCAF* in order to access Grid resources by using the LCG/EGEE middleware. Many sites in Italy and in Europe are accessed through this portal by CDF users mainly to produce Monte Carlo data but also for other analysis jobs. We review here the setup used to submit jobs to Grid sites and retrieve the output, including CDF-specific configuration of some Grid components. We also describe the batch and interactive monitor tools developed to allow users to verify the jobs status during their lifetime in the Grid environment. Finally we analyze the efficiency and typical failure modes of the current Grid infrastructure reporting the performances of different parts of the system used.

1. Introduction

The Collider Detector at Fermilab (CDF) [1] is an experiment at the Tevatron collider where protons and antiprotons collide at an energy in the center of mass of 1.96TeV. The Tevatron instantaneous luminosity has reached 2.9×10^{32} cm^{-2} s^{-1}, the highest luminosity reached by a hadronic collider. This has provided CDF an integrated luminosity of about 2.7 fb^{-1}, corresponding to almost 4×10^9 events that have to be processed and made available to the collaboration for physics analysis in a fast and efficient way. At least the same amount of Monte Carlo data is also needed to perform high precision physics measurements or to search for new phenomena.

The problem of being able to process, analyze and produce such a large amount of real and simulated data was first addressed in 2001, when the CDF computing model was designed. It was based on a dedicated farm, CDF Analysis Farm (CAF) [2], hosted at Fermi National Laboratory (FNAL), to which were soon added some dCAFs (distributed CAFs) located at several CDF institutions around the world.

In order to cope with the increasing demand, CDF decided to adapt its computing architecture to the Grid, instead of multiplying the number of dedicated resources. In this proceeding we will briefly summarize the current status of the CDF computing model, and will then describe the approach currently adopted for exploiting European Grid resources.

1.1. The CAF model

After having developed and debugged their analysis software on their own desktops, users can submit them to the CAF using standard CDF software; the authentication is performed by a server, the headnode, using Kerberos [4], which is also in charge of parallelizing each job into multiple parametric copies according to the user specification. The output of the job can be sent to any user-specified location. Three classes of daemons run on the portal: submitter, monitor and mailer. The submitter deals with accepting the user tarball and submitting each job segment to the batch system. The monitor has two components: a web-based batch monitoring and an interactive monitoring. The latter allows real time interaction with jobs: in addition to listing running, pending or completed jobs, users have also the possibility to display the content of the directory on the worker node where the job is running, to look at error and log files, or to hold the job execution, release it, or kill the job while running. The mailer collects each segment status and sends a summary email to the user upon job completion. Finally the job wrapper runs on the worker nodes: it deals with the job execution and allows communication among the interactive monitoring and the running job.

1.2. CDF toward the Grid

CDF users were used to select where to submit their jobs among several different farms. The experiment is now migrating toward a more Grid-compliant computing model for all the jobs which do not require direct data access, constituting by only three different access points that will process user jobs using Grid resources; these three portals, already running and available for every CDF user, are:

- NamCAF
- PacCAF
- LcgCAF

NamCAF is designed to access OSG resources mainly located in North America; PacCAF, instead, accesses both Lcg and OSG resources located all around the Asian continent. Finally LcgCAF[5], that is the main topic of this proceeding, has been designed to access European Grid resources using the Lcg/Egee middleware.

Up to now real data from $p\bar{p}$ collisions are accessed by users mainly from the central analysis farm located at Fermilab; in case of small datasets these data can also be transferred in an automatic way to the CNAF farm, located in Bologna (Italy), allowing users to process some data also from there.

2. LcgCAF architecture

The basic idea is to provide users a transparent way to access Grid resources, i.e. using the same tools that were available for the submission to dedicated farms, in order to produce Monte Carlo simulation and to execute all user analysis jobs that do not require access to the data stored at Fermilab.

To adapt the CAF model to the Grid, a number of different issues had to be solved; we can group them in 6 points:

- Submission mechanism
- Job execution
- Authentication
- Code distribution and remote database access
- Monitoring
- Output retrieval

2.1. Job submission and execution

The submission mechanism implemented in LcgCAF can be schematically represented as in Figure 1.

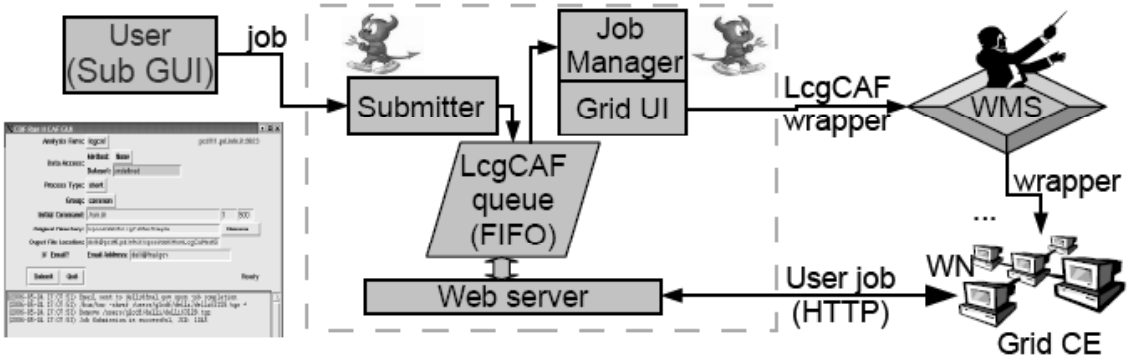

Figure 1. Logical schema of the submission mechanism of LcgCAF. The dashed line contains the logical elements running on the LcgCAF headnode.

Every user submits jobs using the standard CDF software, contacting the *submitter* daemon of the LcgCAF headnode. The submitter processes the request and stores the user job in a local queue, also preparing all the necessary files and wrappers for the submission to the Grid. The *Job manager* daemon periodically checks the local queue (just a FIFO) and for every job to be submitted sends a wrapper to the Workload Management System (WMS) using a standard gLite User Interface (UI)[3].

LcgCAF currently uses a WMS hosted at CNAF (Bologna, Italy) which is in charge of accepting the request, matching available resources and submitting it to the selected Computing Element (CE). The match-making algorithm, in particular, is one of the most sensitive parts for the global performance, as we will discuss in more detail later. Other advanced features are also available, like an automatic resubmission of the job when Grid specific failures arise, which is particularly important to keep the global efficiency high. Finally, part of the WMS is in charge of keeping track of the job status during its lifetime and providing some debugging information when problems occur.

Once the job arrives on the Worker Node (WN), the LcgCAF wrapper is executed: it adapts the environment to the expected one, gets the needed support software and the real user job, runs it and forks some monitoring processes, and finally retrieves the output when the job is completed.

Support software and user jobs are not submitted via the standard Input/Output Sandbox Grid mechanism, but are delivered in "pull" mode directly on the WN by the job wrapper: this wrapper contacts the LcgCAF headnode and asks for the user tarball to an apache server via the HTTP protocol. The user job is then transferred directly to the WN and some proxy servers are used as caches near bigger sites to store it temporarily, in order to reduce transfer time when executing multiple copies of the same jobs[1].

2.2. Authentication and output retrieval

Users authenticate to the headnode using the standard CDF authentication method: *Kerberos V*[4]. Some special kerberos tickets are issued by Fermilab for each user which are

[1] This is obviously the case of most jobs at CDF, since every user submits usually hundreds of copies of the same job in order to parallelize it.

unique for each different CAF and user. These tickets are used to identify the user to the Kerberized Certification Authority (KCA) hosted at FNAL in order to get a valid user X509 Grid certificate. Finally this certificate is used to ask for a Grid proxy to submit the user job, contacting a VOMS server hosted at CNAF and adding this way also the CDF VO extension.

We'd like also to underline that this mechanism has the important feature that each job is owned by the submitting user all the time during its lifetime on the Grid.

Once the job is on the WN, the user must also have the ability to authenticate to a user-specified machine in order to store the output. Two mechanisms are available for this purpose: the Grid proxy that has been forwarded to the WN and a Kerberos V ticket, to ensure a full compatibility with the old style CAFs, and to reduce to the minimum the impact for the users during this transition to the Grid world. The Kerberos V ticket has a lifetime limited to one day, often much less than the lifetime of a typical CDF job; for this reason a mechanism has been implemented to securely renew this ticket directly on the WN: when the ticket has to be renewed, the job wrapper contacts the *KDispenser* daemon running on the headnode, gets authenticated using the Grid proxy and establishes a secure connection. Then the renewed kerberos ticket is issued from the headnode and transferred using this channel to the WN, where it is stored, ready to be used by the job.

Upon completion, the wrapper sends the job directory content to a user-specific location (specified at submission time). Although authentication and rcp-like tools provided by the Kerberos package can be used to perform this transfer, users are strongly encouraged to use a storage element that is accessible via a GridFTP server and is available at CNAF, accessible by the use of globus[6] utilities which are included on every Grid compliant worker node environment.

The more the CDF Monte Carlo production is moving to the Grid, the more these output copying methods become inefficient; moreover they waste much CPU power, since many users copy their output files directly from Europe to the United States; this mechanism is also really sensitive to failures due to peak in the data transfer bandwidth usage. For these reasons a new mechanism of output retrieval is currently under design.

A new intermediate layer will be added, constituted by a "local" Storage Element (SE), which is physically near the site in order to reduce the transfer time; only after that the output will be transferred to FNAL, optimizing the available resources. Many solutions for the implementation of this idea are still under evaluation, and some tests are starting right now from CNAF to FNAL using a SAM[7] catalog, which is the standard CDF catalog system, interfaced with an SRM compatible SE.

2.3. DB access and code distribution

A CDF Monte Carlo job needs access to external resources from the WN: mainly it needs to access a FNAL central database which stores the different run conditions of data we want to simulate, and specific CDF software developed by the collaboration which implements experiment specific functionalities and configuration settings.

In order to ensure a scalable and efficient DB access, CDF adopted a central Oracle database with a set of Squid[8] proxies to distribute the load, used in association with Tomcat[9], and the FroNtier[10] library in order to translate DB queries into HTTP requests; this architecture allows an efficient cache of requests by the Squid proxies and it has been shown to improve performances in speed more than 60% for usual CDF MC jobs[11].

In the High Energy Physics context, LcgCAF exploits a quite unique way to distribute the huge experiment software to the WNs. The goal is to require just a minimal standard Linux environment on the Grid sites, and to deliver only the necessary pieces of software to the WNs to avoid useless bandwith usage. These ambitious objectives have been successfully achieved using the architecture schematically depicted in Figure 2.

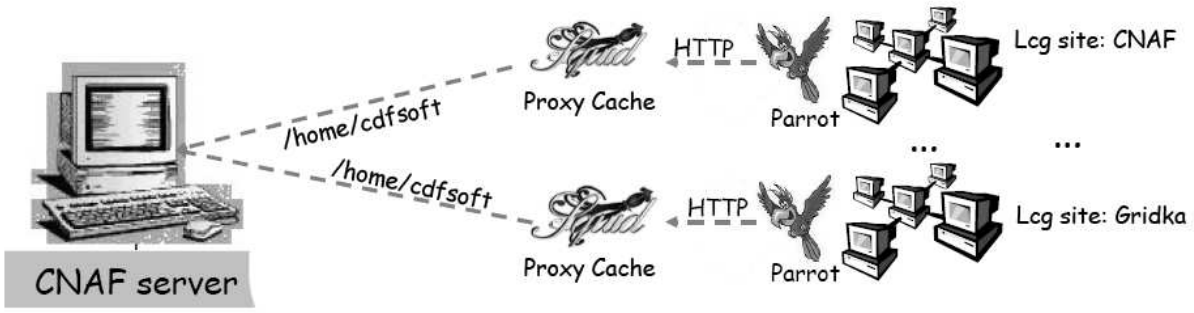

Figure 2. Schematic architecture for CDF software distribution using Parrot. */home/cdfsoft* represents the cdf software root directory.

The solution adopted exploits the functionalities of Parrot[12], which is able to virtually mount a remote filesystem using different protocols. This software is able to run at user level (i.e. does not require any root privileges to be executed): it traps the user program's system calls in order to establish if the requested file is local or not and, if it is not, it is transferred and stored in a local cache in a completely transparent way for the user program. We chose the HTTP protocol in order to transfer the software, which is exported by an Apache server hosted at CNAF and updated whenever a new CDF software version is released. This solution allows the use of squid proxies near the bigger Grid sites in order to cache frequently asked pieces of code, with a big improvement in the timing performance; it also allows the transfer of only the requested pieces of code, and does not require any particular experiment-dependent software installation by the site administrators. Recently a cache-coherence algorithm has been implemented in Parrot in order to assure the coherence of the different squid caches all around Europe; this algorithm is now in the testing phase and will be soon included in the new Parrot releases.

2.4. Monitoring

One of the most important parts of the LcgCAF architecture is the monitoring of the jobs during their lifetime. Users access this information to verify job status, to see if everything is going as expected and eventually they have to be able to kill an already running job if something is not working properly.

In order to give to the user the same information they were used to in the local dedicated farms, LcgCAF collects information about job status from two different sources: the WMS and a monitoring daemon running on the worker nodes with the user jobs, forked as soon as the job starts. The WMS provides information about the job status on the Grid: if it is waiting in the queue of a Computing Element (CE), if it is running, or if it has been completed and the output is ready to be retrieved[2]. The monitoring process running on the WN sends periodically some sensitive information about the job status directly to the headnode: the memory and CPU usage, the name of the machine where the job is running, all the processes tree of that WN, the content of the working directory and finally also the content of some sensitive log files in order to monitor the job execution.

This information is stored, for each submitted job, in a file-based database on the headnode and is accessible by the user with two different methods. A daemon running on the headnode

[2] Actually, as stated in section 2.2, the main job output is retrieved by the LcgCAF wrapper directly from the WN; the residual output is LcgCAF log files.

called *xml_monitor* is in charge of exporting this information in XML format and a web server publishes it in a user-friendly way, as can be seen from a screenshot in Figure 3. Job status for all users as well as specific information about a single submitted job can be easily retrieved using this interface, giving a good and quick overview of the status of the portal and the submitted jobs. A history of managed jobs is also available for analysis purposes. Another way to access similar information is via a command-line tool, which queries directly another daemon running on the headnode, namely *monitor*, and implements unix-like commands like *ls, ps, tail,..* in order to give access to specific information about the submitted jobs. These queries are, unlike what is happening for the web interface, authenticated by a kerberos certificate, so that only the owner of the job can ask for specific job information and take actions on the job itself. A *kill* command is also implemented in this command-line utility to allow the user to kill the running jobs, in case the actual behavior is different from the expected one. Only using this command-line tool the user has the possibility of killing his/her running jobs or looking at their log files, since it happens after a Kerberos authentication process; the web interface gives instead a more general overview of the job status accessible to all the users.

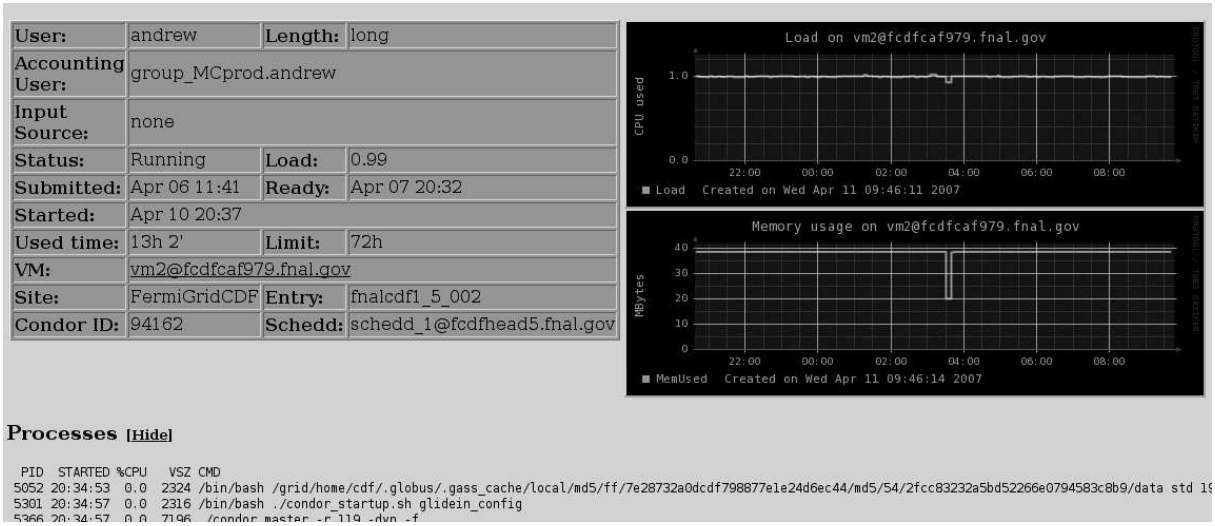

Figure 3. Screenshot of some of the web monitor information available in the web interface for the user.

This structure is really powerful and gives the user a good control on the running jobs. Unfortunately the design has the bad flaw that it doesn't scale very well with the number of the jobs, as we'll discuss in more detail in section 3.2.

3. LcgCAF performance

LcgCAF is in production since almost one year at CDF and all CDF users can submit using this portal.

We now turn to analyzing the performance achieved with such an architecture to outline its main merits and flaws, also connecting it to the existing Grid tools that are implicitly used to process a user job.

3.1. Usage

In order to understand the usage of the Grid resources by this portal, in comparison with the other major VOs, we can look at figures 4 and 5, where a pie chart shows the usage of European and Italian Grid resources from January to the end of August 2007 for each major VO.

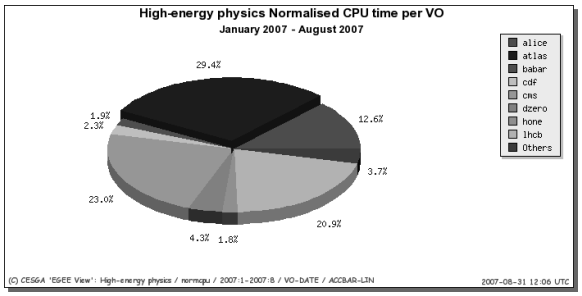

Figure 4. Usage of European Grid resources by LcgCAF (CDF VO).

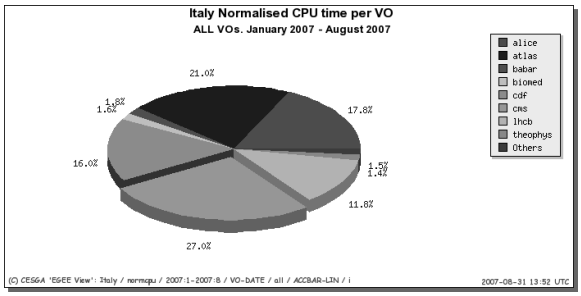

Figure 5. Usage of Italian Grid resources by LcgCAF (CDF VO).

Only 2.3% of European resources are currently used by CDF VO, much less than each LHC experiment currently does; on the other hand, if we look at the Italian usage we see that CDF VO uses a comparable amount of Grid resources like other LHC experiment: about 16% of all the resources. Actually LcgCAF accesses 10 different European Grid sites, and eight of them are Italian; this apparent preference for Italian sites is only due to the early stage of the production age of this portal, and we plan that in a short timescale the amount of European resources accessed outside Italy will exceed the Italian ones.

Looking closely at this situation, the CNAF-T1 Grid site is the most used, since CDF has a big amount of CPUs reserved in this site, processing almost 90% of the jobs that go to italian Grid sites. In figure 6 we can see the usage of the italian sites, excluding CNAF-T1. It can be noticed that jobs exploit many different resources, even if bigger sites (like INFN-PISA) have a comparable number of jobs processed as smaller ones (like INFN-BARI); this behaviour can be linked to the resource matching algorithm, and will be discussed further in section 3.2.

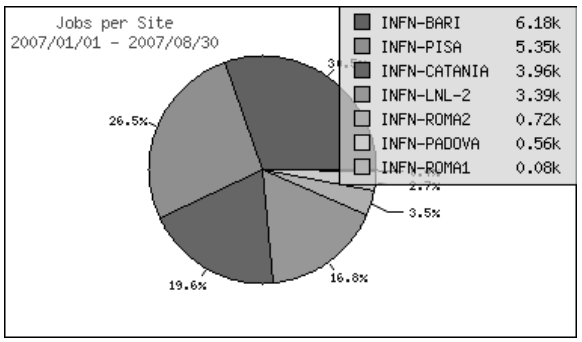

Figure 6. Number of jobs submitted for each Italian site, excluding CNAF-T1.

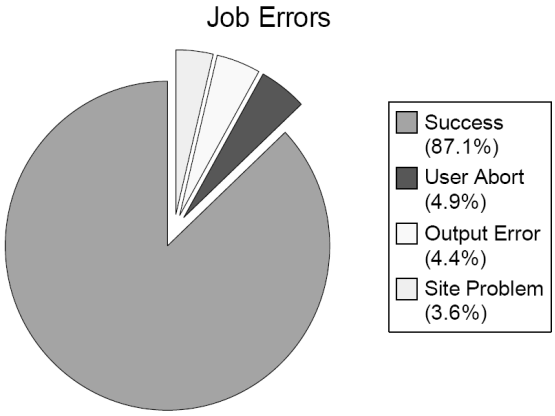

Figure 7. Pie chart of the exit status of the jobs, grouped into 4 different main categories.

During the last year, LcgCAF managed about 70000 jobs, 50000 in the year 2007, and we're expecting this number to grow very fast in the near future[3];

[3] From the time of the conference to now, about 19000 jobs were submitted by CDF users in less than 2 months!

3.2. Performance

Such a complex architecture allows a big flexibility, but it has also many sensitive points that can easily degrade its performance. In figure 7 we analyzed the return codes of an unbiased sample of the submitted jobs, in order to understand the global efficiency and the main problems that were responsible for a failure; the global efficiency (number of jobs successfully executed plus the number of jobs aborted by the user, divided by the total number jobs analyzed) is 92%, while the main failure reasons can be grouped into:

- site specific problems;
- output retrieval problems.

Site specific problems are mainly related to misconfigurations of CEs or WNs, to middleware updates executed at different times between sites and to expired certificates. This kind of problems are related to the pure Grid-based components of this architecture and the only way to improve them is to actively collaborate with site administrators in order to be able to act immediately when a problem occurs, limiting any job loss for the user; in this process ticketing systems are a fundamental and often efficient way of recording such problems, also for future improvements and statistics.

Output retrieval problems are mainly due to temporary unavailability of the destination host, or to a misconfiguration of the WN clock when using Kerberos utilities[4]; we're working actively, as mentioned in section 2.2, to improve the current output transfer mechanism and reduce this source of inefficiency.

Beside the current error rate of submitted jobs, there are also other really sensitive points that can lead to hidden sources of inefficiencies, not detected with the analysis described. The WMS is a crucial piece of this architecture, and its performance directly reflects on the LcgCAF performance. In the last months the WMS version 3.1 entered in production, leading to a big improvement in stability that has been the key to high efficiency. Beside that, we had a hint from figure 6 that jobs are not submitted proportionally to the resources available on each site; this is unfortunately the case, since the match-making algorithm, which is in charge of deciding to which CE the job is going to run, is not yet properly optimized (i.e. does not show good performances in the long term). This is due both to the specific algorithm implemented and to a lack of correct information published by each specific site. Currently this problem is under investigation and will probably improve in the next months. Finally, the design and the current implementation of the LcgCAF monitor will give some stability problem when the number of jobs managed will scale by about one order of magnitude; this is due both to the really simple file-based database where the information is collected and to the high CPU consuming implementation in case of many user requests. One of the future projects will be to migrate to a more Grid-compliant monitoring tool, since many products are already available.

4. Conclusions

During the last years, CDF decided to adapt its computing model to Grid in order to answer the increasing computing power demand. This task is almost accomplished, and results are really promising: a new portal, named LcgCAF, has been designed to access European Grid resources using the LCG/EGEE middleware, with a negligible impact for the users, since the architecture mimics the behavior of CDF standard computing model for dedicated local farms.

LcgCAF is working for all CDF users since almost one year, and has good prospects for growing in CPU power with a design that minimizes experiment-specific requirements for Grid sites. An extensive use of caching allows to minimize latencies for software delivery, DB requests and job transfers as well as to scale the system to a large number of jobs that can be efficiently managed simultaneously.

[4] Kerberos requires clocks to be synchronized within 5 minutes to work properly while transferring files.

This project has also many upgrades undergoing; among them the output storage strategy is being re-designed in order to improve scalability and efficiency of the portal and will soon replace the existing one.

References

[1] CDF Collaboration, The CDF II technical design report; 1996 *FERMILAB-Pub* 96/390-E.
[2] Casarsa M, Hsu S C, Lipeles E, Neubauer M, Sarkar S, Sfiligoi I, Wuerthwein F, The CDF analysis farm; 2005 *AIP Conf. Proc.* **794** 275.
[3] Laure E, *et al.*, Programming the Grid with gLite; 2006 *EGEE-TR-2006-001.*
[4] *Kerberos Web Site*, http://web.mit.edu/Kerberos/
[5] *LcgCAF web site*, http://webserver.infn.it/cdf/LcgCAF/lcgcaf.html
[6] *Globus Web Site*, http://www.globus.org/
[7] Terekhov I, *et al.*, Distributed data access and resource management in the D0 SAM system; *FERMILAB-CONF-01-101.*
[8] *Squid Web Site*, http://www.squid-cache.org
[9] *The Tomcat project*, http://tomcat.apache.org
[10] *The Frontier homepage*, http://lynx.fnal.gov/ntier-wiki/
[11] Kosyakov S,*et al.*, Frontier: High performance database access using standard web components.; Sep 27-Oct 1 2004 Presented at CHEP04 Interlaken Switzerland, **204**.
[12] Moretti C, Sfiligoi I, Thain D, Transparently Distributing CDF Software with Parrot, Feb 13-17 2006 Presented at CHEP06, Mumbay, India, **26**.