Abstract. The improvements of the peak instantaneous luminosity of the Tevatron Collider require large increases in computing requirements for the CDF experiment which has to be able to increase proportionally the amount of Monte Carlo data it produces and to satisfy the computing needs for future data analysis. This is, in turn, forcing the CDF Collaboration to move beyond the used dedicated resources and start exploiting Grid resources. CDF has been running a set of CDF Analysis Farm (CAFs), which are submission portals to dedicated pools. In this paper will be presented the CDF strategy to access Grid resources. GlideCAF, a new CAF implementation based on Condor Glide-in technology, has been developed to access resources in specific Grid Sites and is currently in production status at CNAF Tier-1 in Italy. Recently have been configured GlideCAFs also in San Diego (US), Fermilab and Lyon Tier-1 Center (France). GlideCAF model has been used also to implement OsgCAF, which is a Fermilab project to exploit OSG resources in US. LcgCAF is basically a reimplementation of the CAF model in order to access Grid resources by using the LCG/EGEE Middleware components in a total standard Grid way. LcgCAF is constituted by a set of services each of them responsible for accepting, submitting and monitoring CDF user jobs during theirs lifetimes in the Grid environment. An overview of the Grid Environment and of the specific Middleware services used will be presented; GlideCAF and LcgCAF implementations will be discussed in detail. Some details on OsgCAF project will be also given.

1. Introduction
The Collider Detector at Fermilab (CDF)[1] is an experiment at the Fermi National Accelerator Laboratory (FNAL or Fermilab), which investigates the structure of the matter exploiting the proton-antiproton collisions provided by the Tevatron Hadron Collider. The CDF Collaboration consists of about 800 Physicists affiliated to 59 different Institutions around the world. Up today the Experiment has collected an integrated luminosity of about 1.23 $fb^{-1}$; during the next year it will collect about 2 $fb^{-1}$ and much more in the future. Figure 1 shows the Tevatron performance in term of luminosity projections in the coming years. CDF has now more then 1 $PBytes$ of data under management and the amount of data is expected to grow up in the future.
This implies that an increased amount of simulated data will be also necessary to perform high precision measurements. In order to fulfill new requirements for computing resources, CDF will move from its computing model based on dedicated PC farms to a distributed Grid compliant model.
2. CDF Computing Analysis Architecture

CDF has a farm dedicated to the events reconstructions (Production Farm) and several farms where users can analyze these events. The Data Production flow can be summarized as reported in Figure 2. Physical events coming from the Collider are filtered by a 3 Level Trigger which can select events with a Trigger rate of 100 Hz. Events passing the requirements are written in Raw Data and stored on tape. All the Raw Data are processed to reconstruct physical objects by a procedure performed on the dedicated Production Farm. After the production process data are split in dataset on the basis of physical requirements and stored on tape.

Figure 3 shows the CDF typical Data Analysis flow: the SAM Data Handling System[3] retrieves the requested input data files from the tape Robot in Fermilab and copies them to the disk cache locations near the CAF farm in order to grant a fast access to the user job. At the end of the processing procedure the output files can be stored to CDF specific file servers and then copied back to the Tape Robot. Also Monte Carlo production files or user ntuples can be stored in the Tape Robot.

All the farms rely on the CDF Analysis Farm Model (CAF)[2]. CAF was designed and assembled in 2002 and originally was a PC cluster localized at FNAL; successively the model was exported offsite and now many Decentralized CDF Analysis Farms (dCAFs) exist in many sites worldwide. Currently two CAFs are located at FNAL and nine dCAFs are hosted in offsite Institutions. Onsite and offsite CAFs are basically identical but, since Data are stored in the Tape Robot at Fermilab, onsite CAFs are mainly used for Data Analysis while offsite resources are basically devote to Monte Carlo production. The only exception is CNAF dCAF in Bologna.
Figure 2. CDF Data Production flow.

(IItaly) where some dataset replicas allow users to run also data analysis jobs.
At the moment about 50% of the computing power of CDF resides outside Fermilab. Table 1 shows the current CAF’s resources around the world.

The next sections a general description of the CAF Architecture will be given and an overview of the new Grid Environment, in particular LHC Computing Grid, will be shown. Consequently two new dCAF’s implementations to exploit the Grid Resources (GlideCAF and LcgCAF) will be analyzed in detail. Finally an overview about OsgCAF project to exploit the OSG Grid Resources will be given.

3. The standard CAF Architecture
The CAF is basically a portal which allows CDF users to easily perform Physic Analysis and Monte Carlo Production by exploiting cluster resources such as CPU power, scratch disk and massive data access. CAF is based on a large Batch Manager Cluster and is constituted by a series of command line tools (CAF Clients) which allows user to submit, monitor and interact with the remote jobs. CAF has also a Web Monitor interface to display the job status and many other useful information.

The rest of the Section describes more in detail the architecture of the CAF system.
Figure 3. CDF Data Analysis flow.

Table 1. Current CAF’s resources around the world.

| Cluster Name | Location               | CPU (KspecInt2000) | Disk (TB) |
|--------------|------------------------|--------------------|-----------|
| FNAL CAF     | Fermilab               | 2600               | 370.0     |
| TORCAF       | Toronto (Canada)       | 250                | 10.0      |
| CNAF CAF     | CNAF (Italy)           | 1500               | 32.0      |
| MITCAF       | MIT (USA)              | 140                | 3.2       |
| SDSCCAF      | San Diego (USA)        | 170                | 4.0       |
| KORCAF       | KNU (South Corea)      | 80                 | 5.1       |
| JPCAF        | Tsukuba (Japan)        | 140                | 10.0      |
| ASCAF        | Academia Sinica (Tw)   | 60                 | 3.0       |
| HEXCAF       | Rutgers (USA)          | 50                 | 4.0       |
| CANCAF       | Cantabria (Spain)      | 20                 | 1.5       |

3.1. CAF components

The batch manager adopted by the CAF is the Condor[4] Batch Manager: it is configured with just a submission node, which plays the role of manager node (Head Node) of many Worker Nodes: for each Worker Node six Virtual Machines (VM) are allocated and for each Virtual Machine an user account is created.[5]. All nodes run Linux Operating System and have access to the CDF Analysis Framework via NFS file system.

On top of the Batch Manager lies the CAF software, which is basically an interface between the user and the cluster.
Figure 8 shows an overview of the CAF structure. User Desktops host the CAF Clients which are basically the front-end applications for job submission (*CafGui* or *CafSubmit*) and for job monitoring of the submitted jobs (*CafMon*). The Head Node hosts not only the Condor Manager process but also the CAF services: the submission request are satisfied by the *submitter* daemon, the *monitor* daemon is in charge for providing all the job information to the user and the *mailer* daemon sends a mail to the user as soon as the job finish. On the Head Node runs also the *CafRout* daemon which provides the communication between the monitor daemon and the user job: it relies on Condor CoD technology to interact with the job. The submission process sends the CDF Job Wrapper (*CafExe*) to the Worker Node: *CafExe* is in charge for preparing the job environment before the end-user job start.

### 3.2. Job submission

CDF jobs can be submitted to any dCAF from any desktop or laptop running the CAF clients software; user authentication is performed via a Kerberos V ticket. User has to provide an executable, all the necessary shared libraries, a startup shell script and any configuration files needed by the job; at submission time all the needed files are packed into a tarball and sent to the CAF cluster. Job parameters can be sent to the system via both a Graphical User Interface (GUI) and a Command Line Interface (CLI) for the submission process. Figure 5 shows an example of job submission via the GUI. User has to specify the following input parameters: the Analysis Farm, the access method to read input data, the process type (*short*, *medium*, *long* or *test*) to set up the max lifetime of the job (6h, 12h, 72h or 2h), the group (to have use of the institution member privileges), the local directory that contains the job files, the location to copy job output files and the mail address for the final notification; user has also to specify the initial shell script command of the job and the section range: many instances of the same job (*segments*) may be submitted at the same time and the differentiation of each parallel segment...
is realized by passing an integer number as parameter of the initial command.

3.3. Job execution

The Condor Batch Manager takes into account the user job and sends it to the Worker Node together with the user tarball: the CafExe Job Wrapper unpacks the tarball and forks the initial command; it also performs some monitor tasks such as the creation of a job summary file and the forking of an interactive CafMon callback server on the Worker Node. As soon as the job finish the CafExe tarups the working directory and copies the output files to the specified output location via a kerberized connection.

3.4. Job monitoring and management

User can monitor the running job both in interactive and web based way. Interactive Monitoring relies on the CafMon mechanism which allows to perform some job management tasks such as killing jobs and segments, holding and release jobs, changing the priority in the Condor queue and the group of a running job. CafMon allows also to list the submitted jobs and segments, to show the status and the log files of a specific section and to inspect the content of the working directory or of a specific file on the Worker Node; it can also perform some debugging tasks attaching a debugger session to a remote running process.

4. The Grid world

In order to develop, build and maintain a distributed computing infrastructure for the Storage and Analysis of Data from the four LHC experiments [6] (CMS, LHCb, Atlas and Alice), the LHC Computing Grid Project (LCG) [7] was approved by the CERN Council on 20 September 2001 [8]. The project was defined with two distinct phases. In Phase 1, from 2002 to 2005, the

Figure 5. The CDF CAF Graphical User Interface.
necessary software and services would be developed and prototyped, and in Phase 2, covering the years 2006-2008, the initial services would be constructed and brought into operation in time for the first beams from the LHC machine. The second phase is just started so significant fraction of the computing resources are now available not only to LHC experiments but also for the other Experiments that are currently running (CDF[1], Babar, Virgo,...).

The LCG Project collaborates and interoperates with the other major Grid development projects around the world. The EGEE (Enabling Grids for E-Science) Project[9], partially funded by the European Union, has the goal of deploying a robust Grid infrastructure for science. It interconnects a large number of sites in 34 countries around the world, integrating several national and regional initiatives in Europe, such as INFN Grid[10] in Italy, DutchGrid in the Netherlands and GridPP in the UK.

The LCG Project has also relationships with the other Grid operations environments, such as OSG (Open Science Grid)[11] in the US and NDGF (Nordic Data Grid Facility)[12]. Basically the LCG Project has relationship with the US Experiments Projects whose resources are connected with OSG, but not with OSG itself.

Another project is started to build the next generation Middleware for Grid Computing: the gLite Project[13]; gLite is born from the collaborative efforts of more than 80 people in 12 different academic and industrial research centres as part of the EGEE Project and provides a bleeding-edge, best-of-breed framework for building grid applications tapping into the power of distributed computing and storage resources across the Internet.

The EGEE Middleware consists of a packaged suite of functional components providing a basic set of Grid services including job management, information and monitoring and data management services. The LCG Middleware is currently deployed in over 100 sites worldwide and is now evolving to include some functionalities of the gLite Middleware provided by the EGEE Project.

All EGEE middleware services rely on the Grid Security Infrastructure (GSI). Every user has a certificate from an accredited Certificate Authority (CA); to access Grid services the user has to create or renew a short-term proxy which will be used throughout the system for authentication and authorization. Users are grouped into Virtual Organizations (VOs); in this way every Experiment has its own VO. The short-term proxies are annotated with VO membership and group information so it is possible to manage access to site services ans resources. Every site maintains a Certificate Revocation Lists (CRLs) to invalidate unauthorized usage for a revoked Grid user.

The LCG and gLite middleware components and services are described below.

- **Computing Element (CE)**. The Computing Elements, also called Head Nodes, provides the Grid Interfaces to a Local Resource Manager, that is the site Batch Systems. It handles job submission (including staging of the files required by the user job), job cancellation, suspension and resume, job status inquiry and notification. It only works in push mode where a job is sent to the CE by a Resource Broker (RB). The Computing Elements exports a Web Service interface (called Globus Gatekeeper) to accepts incoming jobs and then it submits the accepted jobs to the Local Resource Manager (LRMS). It also interfaces to a Logging and Bookkeeping Service to keep track of the jobs during their lifetime. The LCG CE can interface with many LRMS: such as Condor[4], LSF, PBS[14]

  The gLite Project is developing a new type of CE which make use of the new Condor-C[4] technology but at the moment in the Production Grid around Europe only LCG CEs are deployed.

- **Worker Node (WN)**. The Worker Node provides the CPU power to process end-user jobs.
• **Storage Element (SE).** The Storage Element provides the Grid interfaces to site Mass Storage system or Disk Locations. An SE can be a classic SE or an SRM SE. The classic SE provides a GridFTP (efficient FTP functionality with GSI security) interface to Disk Storage while an SRM SE provides the GridFTP interface to a Storage Resource Manager (SRM) such as CASTOR[15] or dCache/Enstore[16].

• **Monitoring and Accounting Services.** The monitoring service is based on information providers which collect the status of Grid services and publish their data into the LDAP based BDII (Berkely Database Information Index) System. The BDII System contains information about the whole LCG Production GRID.

• **Virtual Organization Membership Service (VOMS).** The Virtual Organization Membership Service manages User Certificate Subjects and annotates short-term proxies with information on VO and group membership, roles and capabilities (VOMS Extension Informations). It is in particular used by the Workload Management System. A single VOMS server can serve multiple VO's. A VOMS Administrator Web interface is available for managing VO membership through the use of a Web browser.

• **Workload Management System (WMS).** The Workload Management System essentially provides the facilities to manage jobs (submit, cancel, suspend/resume, signal) and to inquire about their status. It makes use of Condor technology and relies on GSI security. It dispatches jobs to appropriate CEs, depending on job requirements and available resources. BDII is used for retrieving information about the resources. The user interfaces to the WMS is based on a Job Description Language to specify the parameters of the end-user jobs.

Also gLite Project has developed a Workload Management System that is an evolution of the one in LCG. It relies on BDII as Information System and can interoperate with LCG Production CEs. The gLite Workload Management System (WMS) operates via many components.

The Workload Manager (WM) or Resource Broker, is responsible for accepting and satisfying job management requests coming from its clients and pass job submission requests to appropriate CEs for execution, taking into account requirements and preferences expressed in the job description. The decision of which resource should be used is the outcome of a matchmaking process that depends on the state of resources and on policies on the CEs.

The WMPProxy component provides a Web Service Interface to the WMS and allows user to submit not only single jobs but also Collection of jobs, Parametrized jobs and DAG jobs that means jobs constituted by many single jobs whose execution order depends on a Direct Acyclic Graph.

• **Information Services (IS).** Information services publish and maintain data about resources in Grids. The IS at the moment in Production on the LCG sites is the Berkeley Database Information Index (BDII).

• **User Interface (UI).** The User Interface is basically the door to the Grid: it is constituted by a set of commands and clients to allow end-user job submission specifying the Job Description Language (JDL) files.

5. **GlideCAF: a late-binding approach to the Grid**

The natural way for CDF to extend its Computing Model to access Grid Resources is the Condor glide-in Mechanism[4], which is basically a generalization of the Standard Condor Mechanism. Condor glide-in can dynamically add Grid Worker Nodes to a regular Condor pool keeping all the advanced features of the Condor Batch Manager. This section describes more in detail the
Standard Condor Mechanism and the Condor glide-in extension. The architecture of the new CDF dCAF implementation built on top of glide-in (GlideCAF[20]) will be explained in detail.

5.1. The Condor glide-in mechanism
A standard Condor pool is composed by a set of daemons to manage the different part of the system and which communicate each other with UDP protocol. The pool is defined by the collector daemon, which is in charge for gathering informations about all the other daemons. The Worker Nodes are controlled by the starter daemons and the user jobs are managed by one or more schedd daemons. When a user job is submitted another daemon, the negotiator, assigns the job to the starters which execute them; a master daemon overlooks all the other daemons and takes care of starting, stopping and restarting them. Figure 6 shows the interaction between the different Condor daemons that take part of a regular Condor pool.

The Condor glide-in mechanism is an extension of the standard Condor pool model which allows to create a virtual pool using Grid Resources. A Condor glide-in is basically a regular starter daemon submitted to a Grid Computing Element. After the submission Authentication the CE delegates the glide-in to the underlying Batch Manager. As soon as this job starts on the Worker Node, it contacts the Condor collector daemon and joins the Condor pool as a new Virtual Machine. From the Condor point of view this new resource is not distinguishable from a dedicated resources; it will be matched in the same way to a user job with the best priority. After this join process is terminated, an standard user job submitted to the Condor Batch Manager can eventually run on the new Grid Resources in a transparent way. The Condor glide-in mechanism is independent by the Local Resource Manager used in the Computing Element so the collecting of the Grid Resource is rather easy.

5.2. The GlideCAF Architecture
GlideCAF is an extended CDF dCAF which relies on a Condor pool extended using Condor glide-in Mechanism. The Condor virtual pool is managed by the glide-in factory (called glidekeeper) which is a process that keeps submitting glide-ins to the Condor schedd daemon as soon as new user jobs arrive at the queue. Figure 7 shows the GlideCaf architecture and gives a view of how
the virtual pool is created by submitting the Condor glide-ins to the Grid. From the point of view of the CDF, GlideCAF is basically a Standard dCAF without any changes. Since also dynamic Resources are part of the Condor pool, GlideCAF preserves all the power and the flexibility of the Condor fair share Policies. Moreover Policies are managed both at Computing Element level, by the VOMS Service policies to access Grid Sites, and at user level, by the Condor Batch Manager.

The glide-ins approach allows also to protect from black holes nodes: since glide-in starts before the user job is sent, it is possible to add some sanity check to discard Worker Nodes that don’t meet the needs of the Virtual Organization and to prevent job failures.

Another important feature of GlideCAF is the late binding approach to the Grid: since user jobs are pulled only by those Sites where the glide-ins has started and resource are available, the risk of long delay at the Computing Element of busy Sites is eliminate.

5.3. The GlideCAF implementations
At the moment four GlideCAF system are deployed in Production Status. The largest GlideCAF is currently running at CNAF Tier-1 in Bologna (Italy): it can easily scale up to the full size of the Tier-1, which means about 1.5 M$SpecInt$2000 of CPU power.

Recently have been configured GlideCAF’s also in San Diego (US), Fermilab and Lyon Tier-1 Center (France). Alpha test systems are going to be deployed at University College of London (UCL) and Karlsruhe (Germany).

GlideCAF approach constitutes the first successfully step in the direction of extent CDF Computing Model to the Grid Computing Environment; a more decisive approach, offered by the LcgCAF Architecture, is shown in the next section.

6. LcgCAF: a totally GRID based CDF dCAF
LcgCAF[21] is a purely LCG/gLite based dCAF which allows CDF users to access LCG Production Grid Sites performing the submission via the gLite Workload Management System. LcgCAF has been developed in order to provide a Monte Carlo Production dedicated farm outside Fermilab to CDF users: it doesn’t support CDF Data Analysis at the moment.

Figure 7. The GlideCAF architecture.
LcgCAF is a totally rewrite of the CDF CAF software and it is basically a portal responsible for accepting, submitting and monitoring the CDF user jobs during theirs lifetimes in the Grid Environment. It is constituted by a set of services each of them performing a specific task. The rest of the sections describes more in detail the LcgCAF architecture.

6.1. The User Submission

LcgCAF Architecture shown in Figure 9 is based on a User Submission Point or Head Node. This is a Grid User Interface where the major part of the services, delegated to manage user job, are running. The job can be submitted from any normal desktop machine with access at the Standard CAF Clients. CDF user needs a valid Kerberos V ticket to be authenticated, in this way the communication between CAF Clients and Head Node is crypted and secure.

The submitter daemon is the service responsible for accepting the incoming submission requests, reading the submission parameters from the CAF Client and creating the correct Job Description Language files for the Grid Environment. After the CDF Kerberos Authentication process, the CAF Client connects to the submitter daemon which retrieves and stores the user tarball and in a specific disk location. This tarball contains all the files needed by the user job (except the CDF executable which LcgCAF distributes with another mechanism). A Web Server, also running on the Head Node, is responsible for distribute the tarballs when required by the Worker Nodes. In order to parallelize the job execution LcgCAF splits the user job into many segments; this can be done using two different Grid submission mechanisms, which differ in the way JDL files are created: DAG and Single submission. The DAG submission mechanism allows to create a bunch of jobs with execution dependences, which means that the execution of a segment has to wait the successfully completion of another segment before starts. It is possible to create complex dependences paths...
Figure 9. The LegCAF architecture.

Figure 10. The DAG structure for a LegCAF job.

on the basis of a Direct Acyclic Graph description. Figure 10 shows the DAG job adopted by LegCAF: a job with a start segment and a stop segment, that will run at the beginning and at the end of the bunch, can be set in order to prepare and clean the environment of the CDF job (for example this will be necessary for the data analysis job support). The Single submission instead creates bunches of single grid jobs without any dependences: for Monte Carlo job this mechanism can be successfully adopted.

The submission to the LegCAF Head Node finishes when the submitter daemon associates a progressive number (Job ID) to the job and returns it to the user through the CAF Client; in this way the job can identified for further monitoring tasks. The job is not submitted to Grid at this moment but it is enqueued in a submission queue and prepared for the real submission.

6.2. Grid authentication

The whole CDF Virtual Organization and every CDF user are managed by the Production VOMS server. LegCAF needs to have a valid X509 proxy with a valid VOMS Extension for each user, in order to successfully perform the submission to the Workload Management System.
A cron job on the Head Node every day translates the Kerberos V ticket into a valid Grid proxy by contacting the VOMS server. The proxy obtained is allowed to access the LCG Production resources.

The lifetime of this proxy depends by the VOMS configuration parameters and currently allows the permanence of the job in the Grid Environment for at maximum one week, enough to guarantee successfully completion of the user jobs.

6.3. Job submission

Since the submission to the Grid takes a long time to be performed, this task is done in asynchronous way. Another service running on the Head Node (job_manager) looks for not submitted jobs in the submission queue and performs a submission to the gLite Workload Management System using the Standard User Interface command and the JDL files prepared by the submitted daemon: the grid submission is done contacting the WMProxy service, which allows to submit both DAG and Single Grid jobs.

The WMS takes into account every job segments and dispatches them to the more convenient resource; basically the choice of the more convenient Computing Element is evaluated looking at the free CPUs of the Computing Element Farms.

The WMS submission model includes specific functionalities to manage the job related input and output files transfers from Head Node to Worker Nodes, called InputSandbox and OutputSandbox. Since the CDF user application tarball typically reaches size of hundreds MBytes, the InputSandbox is not a proper way to manage its transfer: it contains only few files useful for managing job execution and job monitor: the CDF Job Wrapper (CafExe) and the JobMon Daemon[17].

6.4. Job execution

As soon as the job reaches the Worker Node, the CafExe Job Wrapper is executed; it takes care of the user job during all its lifetime. It retrieves the tarball from the Head Node Web Server via HTTP protocol and prepares the environment for the user job. To guarantee scalability of tarball retrieval a Squid Cache/Proxy System [19] can be easily configured near the Computing Element.

Subsequently the CafExe Wrapper forks the job in the form of a start script provided by the user tarball and a psMonitor process which keeps track of the processes running on the Worker Node during the whole execution time of the user job.

When the job finishes it packs the working directory into an output tarball and copies it to the user specified output location. LcgCAF allows user to copy the outputs both to Grid specific Storage Elements and to CDF specific fileservers. Transfer to the SEs are authenticated by GSI using the Grid user proxy; transfers to CDF fileservers must be authenticated using Kerberos V mechanism so it necessary to have a valid Kerberos ticket also on the Worker Node. LcgCAF has developed a mechanism to take care of this, called KDispenser: KDispenser in a service that basically creates on the Head Node the user ticket and transfers it to the Worker Node via a GSI authenticated channel.

6.5. CDF code distribution

At the moment the CDF Analysis Framework is accessible by all the supported Worker Nodes via AFS protocol: an AFS cell is configured and maintained in Bologna (Italy) where the releases of the CDF Analysis Software (AC++ Framework) are kept updated respect with the Fermilab installation. In this way new CDF Software updates are easily available for every Computing Element of every LCG sites accessed.
6.6. Job monitoring and Information Cache System

LcgCAF allows users to monitor submitted jobs both in an Interactive and Web-based way. Interactive monitor (CafMon) provides a way to view job’s processes status and log files by executing simple commands on the Worker Nodes. It has been implemented using JobMon Grid Interactive Monitor Tool[17] which is the official Job Monitoring Project supported by the Open Science Grid, inserted in the Clarens Service Framework[18]. JobMon cannot query information of all jobs status such as running, queuing, waiting or complete status but can only provide ability for interactive job debugging: it is possible to see only jobs which start running. JobMon is constituted by three separate components: a persistent Clarens Web Service located near the execution site, a JobMonD daemon running on the Worker Node and a Client Code hosted by the Head Node. Figure 11 shows the architecture of the JobMon tool.

As soon as the CafExe Wrapper reaches the Worker Node, the JobMonD daemon is forked:

This component tries to connect to the Clarens Service in order to authenticate the user and register job information, such as job name, hostname and port of the Worker Node where the JobMonD daemon is running; these information are collected into a database (file based) hosted on the Clarens Service node. JobMonD daemon set up an outgoing TCP connection from the Worker Node by connecting to a TcpServer component also hosted in the Clarens Service node: incoming connection to the Grid Worker Nodes are not granted by all the Grid Sites so the outgoing connection is the only way to set up a communication channel.

When the user tries to monitor the running job, the CafMon client connects to the monitor daemon which delegates to the JobMon Client the monitor request; then the JobMon Client searches into the JobMonD database the TCP connection to use to communicate with the proper JobMonD daemon; the monitoring request is solved by executing a shell command (ps, top, dir, cat, tail,...) on the Worker Node and the standard output is sent back to the User Desktop.

Another service hosted on the Head Node, the data_collector, is delegated to collect all the job information and to cache them on a file database easy and fast to access by the other LcgCAF components. This service groups together the information from many sources such as LCG Logging and Bookkeeping Service and the JobMon System (during the running time); when the job finished the OutputSandbox is retrieved and the CafExe and psMonitor log files are parsed to collect also CDF job specific information. The Information Cache is kept updated inspecting this information sources every few minutes. The LcgCAF Web-based Monitor[22] relies on this Information Cache: the xml_monitor reads the information from the Cache and translates them...
in XML format; then communicates the updates to Web Server node. Figure 12 shows how a job can be seen on the Web Monitor Page.

**Figure 12.** The Web Monitor Page.

### 6.7. Automatic Job Resubmission Mechanism

Since the Grid Model is heavily distributed, many problems could occur during the submission or the execution time of the job: misconfiguration of some Computing Elements, lack of some necessary services such VOMS server or Logging and Bookkeeping service or some temporary bugs in the system may cause the job to abort before it is properly delegate to a Local Batch Manager or before the OutputSandbox is correctly copied back to the WMS. In these cases the user job should be resubmitted.

To prevent user to manually resubmit Aborted jobs because of Grid problems, LcgCAF implements an automatic Job Resubmission Mechanism: every few minutes the job_manager daemon looks for Aborted segments and manage the resubmission to Grid in a transparent way for the user. The old Aborted segment is ignored and the new one is managed. The resubmission mechanism tries to resubmit a single segment for a configurable number of retrials (typically 2 are enough to reach 100% of efficiency); if the segment aborts again after all the resubmission trials, it is flagged as a Failed segment and is not submitted again.

### 6.8. The final mail

As soon as the job finish, which means that all the segment are Completed (or Failed after retrials), a mail is sent to the specified mail address. The mailer daemon on the Head Node
queries the Information Cache and composes a mail with some useful data such as succeeded and failed number of segments (with eventually the failing reasons), CPU and Real Processing Time, Input Processed Data and Output Produced Data. This mail notification is very useful for recovery and bookkeeping of the job.

6.9. CDF Production Database access
In order to properly simulate the CDF Detector, a typical CDF Monte Carlo job, needs to access the necessary physics informations such as geometrical detector information, trigger configurations, calibration and luminosity tables. These data are kept by an Oracle Production Database which resides at Fermilab. Since the CDF Computing Architecture has a widely distributed environment the access of this Central Database is critical: in order to manage the accesses from remote sites around the world a FroNTier System [23] is deployed. FroNTier basically abstracts the Oracle Client Database Interface to significantly reduce the load and provide a scalable deployed model. FroNTier is implemented by a Squid Proxy/Caching[19] service present on each site and the SQL queries are abstracted via HTTP protocol.

6.10. LcgCAF resources
On order to be successfully accessed and utilized by the LcgCAF portal, LCG Production Sites has to satisfy some requirements: the AFS Client Installation has to be deployed on every Worker Nodes, to provide access the CDF Analysis Framework, and the Computing Elements has to be totally VOMS compliant, which means that the Sites has to be configured to inquiry also VOMS Proxy Extensions Informations of the user proxy in the Authentication process. The LCG Grid Production Sites in Italy, that are INFN Grid Sites, satisfy these requirement by default so LcgCAF can access transparently these Grid Resources. The LCG European Resources are not VOMS compliant by default so they need to be configured by hand to became exploitable by LcgCAF: at the moment just some European Tier-1s Site access is under investigation and configuration.

The total LCG CPU power in theory accessible from LcgCAF is about 75 $MSpecInt2000$.

6.11. Queues and policy management
The current LcgCAF implementation provides an easy Queues Management based on limits of the job lifetimes: it is basically implemented by the CafExe Wrapper without any Grid Mechanism support; CafExe is in charge for killing running jobs when they run for more than a configurable time limit. At the moment in LcgCAF are present four queues: in the long queue job can be running on the Worker Node at maximum for 72 hours, in the medium queue for 12 hours, in the short for 6 hours and in the test queue at maximum for 2 hours.

Currently LcgCAF hasn’t a Policy Management Mechanism because Grid Environment itself hasn’t a Policy Mechanism deployed in Production Status. However LCG Project is going to implement a new component, called GPBOX (Group and Policy BOX)[24], which allows or denies users to access Computing and Storage Resources around the Grid at submission time; a LcgCAF Policy Management Mechanism will be implemented on the basis of this new Grid service as soon as it will be in Production Status.

7. OsgCAF
In order to access also OSG Grid Resources in US, Fermilab has some plans based on the GlideCAF model.

The intrinsic limits of this model, such as the UDP based communication that does not work over the Wide Area Network and the required bidirectional traffic that fails over the firewalls,
has been solved using the Generic Brokering Connection (GCB[25]) mechanism. GCB can de-
couple the decision on what direction a connection must be made from who the client/server is;
the direction of a connection is determined based on the relative network situation of commu-
nicating parties so GCB can guarantee communication between Condor daemons basically in
every situation encountered exploiting OSG Sites. In order to solve scalability problems, more
then one GCB service can be easily deployed.
At the moment San Diego, MIT e Fermilab OSG resources has been successfully accessed via
GlideCAF portal while there are short term plans to exploit also Florida and Toronto resources.
Recently NAmCAF (North American CAF) has been inaugurated: it is basically a single point
of submission to match all the deployed GlideCAF s around US.

8. Acknowledgment
I'm very pleased to thank all the CDF collaborators, whose cooperation has been fundamental
to realize this project. I would like also to thank all the EGEE, LCG and gLite collegues who
helped to realize this work.

9. Conclusion
The increase of computing requirements for Data Analysis and Monte Carlo Production for
the CDF experiment can be satisfied exploiting the Grid Resources available around the world
before LHC era starts.
The Standard CDF model has been preserved in order to retain all the virtues of the dCAF like
ease of use and distributed architecture.
The GlideCAF, an extension of the Condor based CAF implemented on the basis of the Condor
Glide-ins technology has been deployed around the world to access both EGEE/LCG resources
in Europe and OSG Resources in US.
The LcgCAF, a re-engineering of the CAF model to use the EGEE Middleware in a Standard
Grid way, has been implemented and is at the moment running Monte Carlo Production jobs.
10. Bibliography

[1] CDF Homepage, http://www-cdf.fnal.gov/
[2] M. Casarsa, S.-C. Hsu, E. Lipeles, M. Neubauer, S. Sarkar, I. Sfiligoi and F. Wurtwein, "The CDF Analysis Farm", AIP Conference Proceedings (2005) 794
[3] The SAM Grid Project Homepage, http://projects.fnal.gov/samgrid/WhatisSAM.html
[4] Condor High Throughput Computing Homepage, http://www.cs.wisc.edu/condor/
[5] I. Sfiligoi et al., "The Condor based CDF CAF", CHEP04 Proceedings, Interlaken, Switzerland (2004)
[6] LHC Homepage, http://lhc.web.cern.ch/lhc/
[7] LCG Homepage, http://lcg.web.cern.ch/LCG/
[8] LHC Computing GRID Technical Design Report, LCG-TDR-01, CERN-LHCC-2005-024, 25 June 2005, http://cern.ch/lcg/tdr
[9] EGEE Homepage, http://www.cern.ch/egee
[10] INFN Grid Homepage, http://grid.infn.it/
[11] Open Science Grid (OSG) Homepage, http://www.opensciencegrid.org/
[12] Nordic Data Grid Facility (NDGF) Homepage, http://www.ndgf.org/
[13] gLite Homepage, http://glite.web.cern.ch/glite/
[14] Portable Batch System Homepage, http://www.openpbs.org/
[15] The CASTOR Homepage, http://castor.web.cern.ch/castor/
[16] The dCache Homepage, http://www.dcache.org/
[17] JobMon Interactive Grid Job Monitoring, http://jobmon.sourceforge.net/
[18] Clarens Grid-Enabled Web Services Framework, http://clarens.sourceforge.net/
[19] Squid Web Proxy Cache, http://www.squid-cache.org/
[20] S. Sarkar et al., "GlideCAF: a late binding approach to the Grid", CHEP06 Proceedings, Mumbai, India (2006)
[21] LcgCAF a Grid based CDF dCAF, http://www.pi.infn.it/cdf-italia/public/offline/lcgcaf.html
[22] LcgCAF Web Monitor, http://wn-04-04-26-a.cr.cnaf.infn.it:8081/lcgcaf/index.html
[23] S. Kosyakov et al., "FroNTier: high performance database access using standard web component in a scalable multi-tier architecture", CHEP2004 Conference Proceedings (2004)
[24] A Policy Management Framework for Grid Environments, http://infnforge.cnaf.infn.it/gpbox/
[25] Generic Connection Brokering, http://www.cs.wisc.edu/ sschang/firewall/gcb/