CDF Monte Carlo Production on LCG Grid via LcgCAF portal

Gabriele Compostella *University of Trento and INFN Padova*,
Donatella Lucchesi *University and INFN of Padova*,
Simone Pagan Griso *University and INFN of Padova*,
Igor Sfiligoi *Laboratori Nazionali di Frascati, Roma*.

Abstract **The improvements of the 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. This is, in turn, forcing the CDF Collaboration to move beyond the use of dedicated resources and to exploit Grid resources. CDF has been running a set of CDF Analysis Farm (CAFs), which are submission portals to dedicated pools, and LcgCAF is basically a reimplementation of the CAF model in order to access Grid resources by using the LCG/EGEE Middleware components. By mean of LcgCAF CDF users can submit analysis jobs with the same mechanism adopted for the dedicated farms and at the same time the Grid resources are accessed without any speci c software requirements for the sites. This is obtained using Parrot for the experiment code distribution and Frontier for the run condition database availability on the worker nodes. Currently many sites in Italy and in Europe are accessed through this portal in order to produce Monte Carlo data and in one year of operations we expect about 100,000 Grid jobs submitted by the CDF users. We review here the setup used to submit jobs and retrieve the output, including the Grid components CDF-speci c con guration. The batch and interactive monitor tools developed to allow users to verify the jobs status during their lifetimes in the Grid environment are described. We analyze the ef ciency and typical failure modes of the current Grid infrastructure reporting the performances of different parts of the used system.**

Index Terms **CDF, computing, Grid, WMS.**

I. Introduction

The Collider Detector at Fermilab (CDF) [1] is an experiment at the Tevatron collider where protons (p) and antiprotons ($\bar{p}$) collide at an energy in the center of mass of 1.96 TeV.

CDF is taking data with an upgraded detector since 2001 but the Tevatron is fully efficient since 2003. The current instantaneous luminosity is greater than 2×10^{32} cm^{-2} s^{-1}, the highest luminosity reached by a hadronic collider as today. This has provided the experiment with an integrated luminosity of about 2 fb^{-1} with the prospect of doubling it in the next year.

Such integrated luminosity corresponds to 3.4×10^9 events that have to be processed and made available to the collaboration for physics analysis as quick as possible . The same or even larger amount of Monte Carlo data are needed to perform high precision physics measurements or to search for new phenomena.

To be able to process, analyze and produce this large amount of real and simulated data CDF evaluated a cpu need of about 3900 KspecInt2000 corresponding to about 9 THz in 2006 which becomes around ∼8600 KspecInt2000 for an amount of 20 THz in 2007 .

The CDF computing model, designed in 2000, was based on one dedicated farm, called CAF [2], hosted at Fermilab National Laboratory (FNAL) . This evolved in the dCAFs (distributed CAFs) located at several CDF institutions around the World. Instead of keeping this model and growing the number of dedicated farms to cope with the increased cpu request, CDF decided to adapt the computing architecture to the Grid structure. This has the advantage of exploiting the LHC resources before it starts to take data with the addition also that the support needed is reduced only to the maintenance of CDF specific code, while farms and middleware are maintained by the Grid employees.

For these reasons LcgCAF has been designed to access the LCG sites.

II. CDF Computing Model

The result of $p\bar{p}$ interactions is passed to a three levels trigger which, based on increasing number of information, makes the final decision as to whether the event is interesting enough to record it. Raw data is logged to tape via an intermediate cache disk at an average of 60 MB/sec.

These events are then reconstructed to high level objects like electrons, muons and jets using a dedicated farm and then written to tape.

Raw and reconstructed data is catalogued using SAM [3], which provides distributed data access as well as dataset and files history.

The reconstructed data is analyzed by users running on CAFs, the basic unit on which the computing model is based upon.

Currently CDF has a three CAFs at FNAL, one dedicated to raw data reconstruction and two open to users for data analysis, these CAFs have also a "SAM station", a computer system that steers the data requests to the catalog, necessary to get files from the tape robot and to store files to tape.

Around the world in several sites where CDF has representatives there are other dCAFs (distributed CAFs) used mainly for Monte Carlo data production, since data access over the network is not efficient. The only exception is CNAF, in Bologna, where some datasets are replicated so that data analysis can be done also there.

0-7695-3064-8/07 $25.00 © 2007 IEEE
DOI 10.1109/e-Science.2007.39

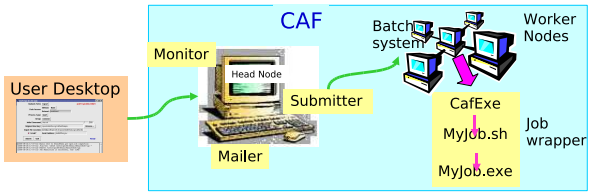

Fig. 1. The CAF architecture. Several daemons run on the head node while the job wrapper executes the users job on the worker nodes.

A. CAF overview

The CAF idea is that users develop and debug their analysis jobs on their desktops, where they have access to the CDF code while user authentication is performed using Kerberos [4]. Each job is then submitted to CAF via a software custom legacy interface and executed in multiple parametric copies in order to parallelize it. If the submission is successful a job ID is returned to the users for monitoring. The output of the job can be sent to any user-specified location. Figure 1 shows the portal: the head node, where several daemons run, and the local farm worker nodes where the job wrapper acts. Three classes of daemons run on the portal: submitter, monitor and mailer.

The submitter accepts the user tarball and stores it on local disk. It submits each job segment to the batch system after having created the submission files necessary for the specific batch system.

The monitor has two components: an interactive monitoring and a classical batch monitoring. The former is very important and somehow unique to CAF, is not common on other experiment-dedicated farm especially on Grid since it allows the users to interact directly with the job to check its status. Each user can look at the list of his/her jobs running, pending or completed and at the status of the machine running a specific job. Users have also the possibility to display the content of the directory on the worker node where the job is running and to look at the error and log files of these jobs to standard output. Moreover, users can interact with the job holding the execution, releasing it, and killing it while running. Rarely used but possible also is the job interactive debug. The web-based monitor displays on a web page some useful information for all the jobs of all users on the selected farm and keeps an history record up to 2 years.

The mailer collects each segments status and when the job is finished sends an email to the user-specified email address communicating the job summary.

The job wrapper (CafExe) runs on the worker nodes, unpacks the user tarball and forks the initial job command. It also performs some monitoring tasks such as the creation of a job summary file. When the job finishes the CafExe tars up the working directory with logs and errors files and any other leftover and copies it to a user-specified location.

III. CAF evolution toward Grid

The CAF success is mainly due to the fact of having separated the user interface from the CAF portal itself. Since the 2002 when the first CAF was deployed the portal has interfaced to different batch systems without the users having noticed it. The first CAF used FBSNG [5], then a big step forward was done implementing Condor [6], [7] batch system. Condor allows to manage dedicated pools and also access Grid resources. As discussed in the introduction, CDF cpu needs increase year per year and dedicated farms are not anymore sufficient to satisfy the experiment's requirements. Moreover, even if CDF could have the possibility of having large dedicated farms, this is not convenient in terms of manpower, while moving to a distributed environment allows the experiment to exploit resources supported by the Grid community. The first approach to the Grid was done via the glide-in mechanism. The Grid worker nodes are dynamically added to a condor pool keeping all the advanced features of the Condor batch system and building up a so-called GlideCAF [8]. This farm has served and is serving CDF in an excellent way with very high performances in terms of efficiency and reliability. But CDF has to move toward a more distributed computing model in order to exploit the Grid resources and starting from what was already in use two projects were pursued: LcgCAF [9], a portal based on gLite Workload Management System (WMS) [10] and NAMCAF [11]. NAMCAF is based on GlideCAF having solved the issue of the communication over WAN by using Generic Connection Brokering (GCB) [12], a tool which allows cross-firewall communication. This paper is focused on LcgCAF and NAMCAF will be described in a future work.

IV. LcgCAF

LcgCAF is a total rewrite of the CDF CAF software to have a portal responsible for accepting, submitting and monitoring the CDF users jobs during their lifetime in the Grid environment. Since CDF has to cope with the Grid infrastructure and middleware as it is only its fundamental elements are exploited to access the resources. Moreover, CDF unlike the LHC experiments, does not require to have any support nor for middleware nor for hardware but that the site is VOMS compliant.

The general LcgCAF architecture, shown in figure 2 is based on a submission point which is basically a head node. This is a Grid User Interface (UI) where the major part of the services responsible for accepting, submitting and monitoring the users job are running. The users can submit a job from any desktop with access to the standard CAF legacy submit mechanisms.

The Grid authentication is done via a cron job running on the head node that every day translates Kerberos V ticket of each user into a valid Grid proxy by contacting the Kerberos Certification Authority (CA). The lifetime of this proxy depends on LcgCAF settings and on VOMS configuration parameters. Currently Grid proxy allows the job to survive for a maximum of one week, evaluated to be enough for a long Monte Carlo job to complete. After the users are authenticated, the CAF clients connect to the submitter daemon which then delegates the job to the gLite Workload Management System [10], the gLite facility to submit jobs to the resources matching users requirements. A this point the Resource Broker, one of the elements of the WMS, dispatches jobs to the Grid Computing Element (CE), the interface to the

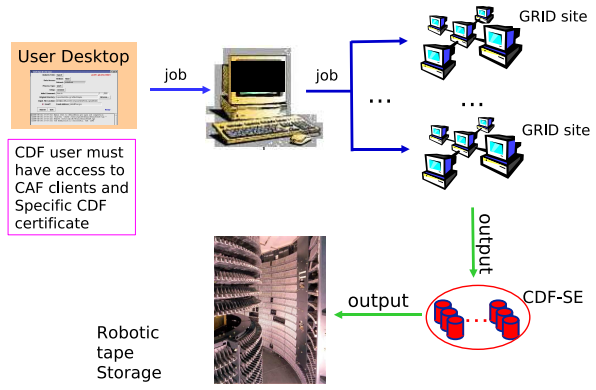

Fig. 2. General LcgCAF architecture. The user job goes from the portal to the Grid submission point and then to Grid sites where they run. The job output is collected on CDF Storage Element and then shipped to Fermilab.

local resource manger. When user jobs finish on a Grid site, their output is stored on a CDF Storage Element (SE) or copied to a user defined location.

A. Job submission and execution

As for the CAF, LcgCAF portal hosts many daemons and in particular those that are responsible for the job submission, monitoring and user notification. The structure is the same already seen for the CAF but with different functionalities. Figure 3 shows the LcgCAF with the three major classes of daemons and then the path to the Grid site submission.

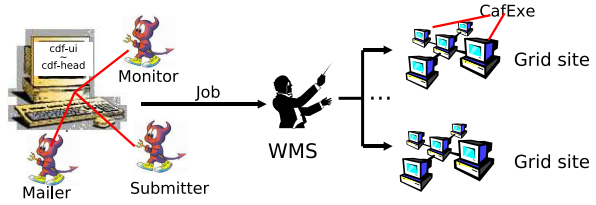

Fig. 3. LcgCAF submission and execution cartoon.The portal with the necessary daemons and the job wrapper running on worker nodes.

The submitter is the service responsible for accepting the incoming submission requests, reading the submission parameters from CAF client and creating the proper Job Description Language files (JDL). The submitter retrieves and stores the user tarball in a defined disk location which can be accessed via a Web server. User tarball are then pulled from the Web server once jobs are running on working nodes. The job parallelization, done in the standard CAF by splitting the job in several segments, is reproduced in Grid using different submission mechanisms implemented in the JDL file at creation time: DAG (Direct Acyclic Graph), and single submission. The actual version of the code uses the single submission which is good enough for Monte Carlo production. If LcgCAF will be used also for data analysis a DAG submission will be implemented. The job is not directly submitted to the Grid, but enqueued in a submission queue, since the current version of the WMS for Grid submission takes a too long time. When the job is enqueued a job ID is returned to the user and later matched to the Grid job ID for monitoring purposes. Another service on the head node looks for jobs on the submission queue and submits them to the gLite WMS. This, then, performs the real job submission to the most convenient CEs, where the most convenient CEs mean those with the shorter time response. Studies of other ranking methods are in progress in order to optimize the Grid resources available and the experiments requests.

The InputSandbox, one of the functionalities of the Grid submission used to bring files to worker node, is exploited to transfer the job wrapper (CafExe) and monitoring daemon.

Once each job reaches the worker node, CafExe, the job wrapper, is executed. It retrieves the user tarball from the head node web server via HTTP protocol and prepares the environment for the user job. The job wrapper forks the job in the form of a user script and the monitoring process which keep track of the process running on that worker node (see section IV-B for the detailed description). When the job finishes CafExe packs the working directory into an output tarball and copies it to the user specified location.

When all the parallel segments forming the user job are completed (or failed after retrials) an email is sent to the email address specified by the user at submission time. The mailer daemon queries CDF Information System (see section IV-B) and collects useful information like segments succeeded and failed, reason of failure (codified), cpu and real time processing. This tool demonstrated to be very useful for users recovery and bookkeeping.

B. Monitor

CDF users always had the possibility of monitoring their jobs status using command line and Web monitoring as described in section II-A. In LcgCAF this is achieved in a naive way, for the moment, by collecting information from WMS and worker nodes as schematically shown in figure 4.

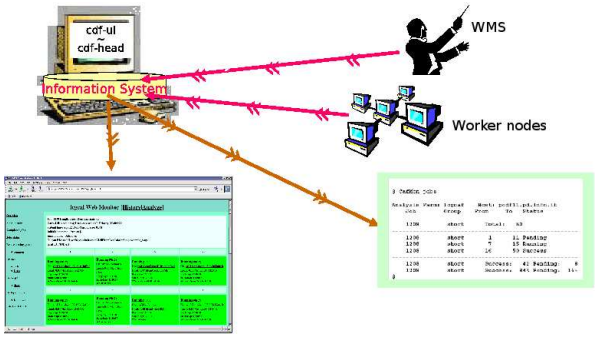

Fig. 4. LcgCAF monitors architecture. Job information are collected from WMS and worker nodes and stored in CDF Information System. An example of the interactive monitoring is shown on the right while on the left a screen snapshot of the WEB monitoring page is displayed.

The interactive monitor (CafMon) provides a way to view the status and log files of running jobs based on users queries as for the CAF. But the query mechanism is changed, now it is based on the *WNCollector*, part of the LcgCAF code.

It has a server, *WNCollDaemon*, running on the head node and a client *WNColl.py* executed on worker nodes and active during all the job lifetime. The client periodically collects several information related to user, working directory, cpu and memory usage, errors and logs of the job and passes them on to the CDF Information System (CDF-IS), a file-based database on the head node, where they are stored.

These information are then retrieved on demand by the users. Implemented commands are only a subset of those available on CAF the most used: *CafMon jobs* gives the list of the jobs and segments submitted by the user ,*CafMon dir* shows the working directory contents of the specified segment, *CafMon ps* and *CafMon top* are equivalent to running ps and top commands respectively on the worker node, *CafMon tail* mimics the tail command on a given file.

When user sends a request, the information system is accessed and the user gets the cached information as shown on the right of figure 4. In this way the interactive monitoring has a delay among the current status of the job and what is shown which can be tuned depending on the number of jobs on the system to avoid an overload on the head node. *CafMon kill* is the only command to allow users to interact with the job for killing it and it is implemented using *glite-wms-job-cancel*.

The Web monitoring is implemented using other tools developed for LcgCAF. Another service hosted on the head node, *data_collector*, is delegated to collect job information inspecting the Logging and Bookkeeping of the WMS for the job status, the *WNCollector* cache for worker nodes information and the log files of the CafExe for CDF framework specific status. All these information are cached in the CDF Information System, easy and fast to access by other LcgCAF components. These information are translated in XML format by the *xml_monitor* which reads the CDF-IS and communicates the updates to the Web server node. On the left of figure 4 is shown an example of the Web based monitor, where for each user a summary of each job and segment is displayed.

C. CDF Code distribution and CDF database access

In order to be executed a Monte Carlo job needs both CDF code and Run Condition database where detector and trigger configurations which change run by run are stored. Both must be available to worker nodes at runtime and since this can not be expected to be granted in all the Grid sites, alternative solutions are needed.

The Run Condition database contains physics information such as detector geometry, trigger configurations, calibrations and luminosity tables necessary for Monte Carlo production which has to reproduce as much as possible the data. These information are kept in a Oracle database at Fermilab and access to this database, even if in read-only, is one of the most critical points due to the limited number of allowed simultaneous connections. In order to satisfy queries from all the remote sites around the World, a FroNTier client on the local site translates users requests into Web requests that are then transformed to database queries from the local FroNTier server at FNAL that returns data to the client. With the use of Squid Proxy caching layers deployed near the servers, as well as close to the clients, the load on the central database is significantly reduced and CDF has a scalable deployment model [13].

The CDF software was designed to be executed in dedicated environment with an easy access to a large set of executables, shared libraries and configuration files. In a general distributed computing model the assumption that this is available on each worker node does not hold anymore and the fastest way to reproduce the CDF filesystem around the World goes through Parrot [14], a virtual filesystem for performing POSIX-like I/O on remote data services. When the program to which Parrot is attached attempts a system call, it is halted, and Parrot is notified by the kernel. Parrot then interprets the arguments to the system call, and then implements a remote call exploiting standard protocols like HTTP, FTP, GridFTP, etc. In CDF, Parrot (see figure 5) uses HTTP protocol in a such a way that HTTP calls can be cached using the Squid Proxy, already available close to big sites for database access. This is useful because the CDF code server is hosted or at Fermilab or at CNAF (Bologna) and Parrot retrieves the libraries from there introducing a latency time which is reduced by factor ~ 25 adding Squid.

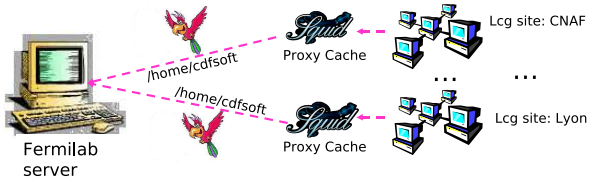

Fig. 5. LcgCAF code distribution cartoon. Each site sees the CDF filesystem as locally mounted.

D. Output storage

LcgCAF allows the users to copy the output of their job either to a Grid specific Storage Element or to a CDF dedicated fileserver. Output transfers to a Grid SE are authenticated by GSI using the Grid user proxy. The output copy to a CDF dedicated fileserver must be authenticated using Kerberos V mechanism and hence it is necessary to have a valid Kerberos ticket also on the worker node. To accomplish that a new service mechanism has been created, the *KDispenser*. This service creates the user ticket on the head node and transfers it to the worker node via GSI authenticated channel.

In case of intensive Monte Carlo production files are preliminarily stored into a SE or a dedicated fileserver and then via an automatic procedure these files are pulled from Fermilab and copied to another dedicated fileserver from where they are eventually catalogued and transferred to tape.

E. Re-submission Mechanism

Since the Grid Model is heavily distributed, different problems could occur during the submission or the execution of the job: misconfiguration of CEs, lack of a necessary service such VOMS server or Logging and Bookkeeping service or

some temporary bugs in the system. The result is that the job abort before it is properly delegated 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 uses "Job Shallow Resubmission Mechanism": when a job does not reach a worker node, ie the CE fails to accept it, another CE is tried immediately. The resubmission mechanism tries to resubmit a single segment for a configurable number of retrials, three is the default for LcgCAF and typically 2 resubmissions are enough to reach 100% of efficiency. If the segment aborts again after all the resubmission tentatives, it is flagged as a Failed segment and is not submitted again. The WMS has also the "Deep Resubmission Mechanism" that can be used when a job fails after it started running but this is more problematic to use and it is not used yet.

V. LCGCAF USAGE AND PERFORMANCES

LcgCAF is in production since October 2006 and is used by normal CDF users for Monte Carlo production or other simulation jobs and by the official Monte Carlo production groups for intensive Monte Carlo simulations. Using the web based monitor it is possible to count the number of jobs submitted and also to analyze the cpu usage. Figure 6 shows the number of running segments as a function of time starting from October 2006 up to mid-July 2007. The analysis of this data shows that 42,000 jobs have run on LcgCAF with a temporal structure which depends on international conferences (users need more resources close to conferences) and Grid infrastructure downtimes and middleware deployment.

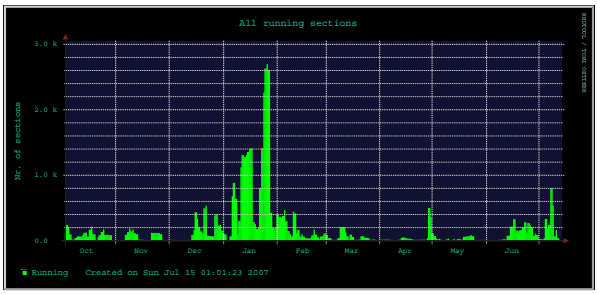

Fig. 6. Number of running jobs on LcgCAF in the period of time October 2006 - mid-July 2007, as monitored by the LcgCAF.

The Grid tool GridIce [15] allows to monitor CDF jobs on Grid. Unfortunately this is not possible for the all European sites since GridIce sensors are not installed everywhere yet but can be done for the Italian ones. Figure 7 shows the usage of Italian resources by all VOs in the last two months. CDF is represented as two VOs: CDF and CDFCAF for technical reasons. CDF appears to be one of the major users.

The performances of LcgCAF have been evaluated in two different cases. The first one during a massive Monte Carlo production where something of the order of 800 jobs and 1000 segments at the same time were running. The efficiency, defined as the number of successful jobs divided by the total

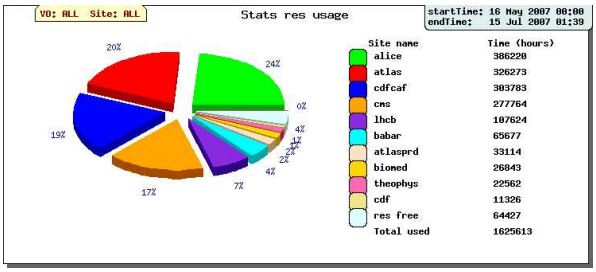

Fig. 7. Usage of the Italian resources for each VO as monitored by the GridIce in the the last two months.

number of the submitted jobs, is 94%. With one recovery done by the user the efficiency is almost 100%.

The second method uses GridIce information which contains the job exit status. This is zero, i.e. successful jobs, in 89% of the cases, 3% of jobs have been cancelled the users and 2% have been cancelled by "root" due to local problem on the sites. The remain 6% of faillures are due mainly to gLite middleware misconfigurations and a very small fraction to LcgCAF itself. The difference in the two efficiencies can be due to several reasons. The Grid infrastructure and middleware is more stable when only few "power users" are using them for a short period of time and everybody knows that is in usage. During the year there have been several downtimes for infrastructures and middleware which are not well coordinated yet. For a running experiment the performances must be almost equal to those reached with dedicated farm and Grid is approaching but has not reached them yet.

The major contributors to the LcgCAF Grid resources are GridKa and T1 at CNAF, these two are not displayed in figure 8 where contributions of different Italian sites are shown. The major part of the jobs are processed by 4 sites out of 7 currently accessible by CDF. This is caused mainly by the differences in cpu power of the sites.

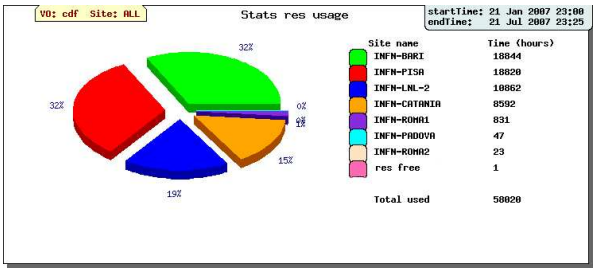

Fig. 8. Contribution of different Italian sites to the resources used by CDF as monitored by the GridIce in the the last two months.

VI. FUTURE IMPROVEMENTS

Two major upgrades are foreseen for LcgCAF, one related to policies and queues management and the other one connected to the interface with SAM, the CDF catalog.

Currently LcgCAF hasn't a Policy Management Mechanism yet because LCG Grid Environment itself does not have a

policy mechanism deployed in production status. However, the current LcgCAF implementation provides an easy queue management system based on limits on job lifetimes: it is basically implemented by the CafExe wrapper without any Grid mechanism support; CafExe is in charge of 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. LCG Project is going to implement a new component, called GPBOX (Group and Policy BOX) [16], 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.

If CDF users want not only to produce Monte Carlo data on the Grid but also to analyze data, an interface between Grid tools and SAM is needed, since SAM can not become a catalog supported by LCG middleware. The basic concept of the integration is to apply SAM data handling policies over the generic storage system managed by SRM (Storage Resource Manager) protocol. The premise of the concept is similarity between SAM managed disk storage and SRM managed storage in terms of many supported operations like transferring files, removing files, retrieving files metadata etc. which allows a one-to-one mapping of several functions like "copy" , "delete" ect. This project is currently in progress at the Fermilab Computing division.

VII. Conclusion

The CDF computing model has been very successful in giving the possibility to the whole CDF collaboration around the World to analyze the data almost at the same time as the collaborators in Fermilab. This was achieved thanks to the CAF idea, a very flexible portal that kept the user interface the same during the years with major changes in the underlying structure. One example is LgcCAF, a portal to LCG resources which has all the CAF characteristics adapted to the gLite middleware. The porting of the CAF code to LCG was relatively fast, after a year of work of an expert person LcgCAF was ready with all the functionalities, but CDF started to use it with a reasonable efficiency for a running experiment only this year after the release of WMS Version 3.1. This portal is now in production at CDF and performing quite well, but it still needs to improve to run on data and to have policy management mechanism. The major issues at the moment are on the Grid side. The most important is the fact that the sites are not stable, having not working or misconfigured worker nodes cause failure of jobs that users have to recover by hand. The other problem is the jobs matching to the sites, the procedure used is based on the time the job has to wait in queue and the Grid Information System evaluated that using the information published by the sites and these are not so timely updated.

Acknowledgment

The authors would like to thank Daniele Cesini and WMS team for the continuous support together with all CNAF personnel.

References

[1] CDF Collaboration "The CDF II Thecnical Design report" *FERMILAB-Pub-96/390-E* (1996)

[2] M. Casarsa, S. C. Hsu, E. Lipeles, M. Neubauer, S. Sarkar, I. Sfiligoi, F. Wuerthwein, "The Cdf Analysis Farm," *AIP Conf. Proc.* **794**, 275 (2005).

[3] I. Terekhov *et al.*, "Distributed data access and resource management in the D0 SAM system," *FERMILAB-CONF-01-101 Presented at 10th IEEE Internat'l Symposium on High Performance Distributed Computing, San Francisco, CA, Aug 7-10, 2001* (2001).

[4] "Kerberos Web site" *http://web.mit.edu/Kerberos/*.

[5] "FBSNG Web site" *Next Generation of FBS http://www-isd.fnal.gov/fbsng/*.

[6] D. Thain, T. Tannenbaum, M. Livny, "Distributed computing in practice: the Condor experience.", *Concurrency - Practice and Experience* **17, 2-4**, 323 (2005).

[7] I. Sfiligoi et al. "The Condor based CDF CAF.", *Presented CHEP04 , Interlaken Switzerland, Sept. 27-Oct. 1, 2004* , **390**, (2004).

[8] S. Sarkar, I. Sfiligoi, *et. al.*, "GlideCAF - A Late binding approach to the Grid", *Presented at Computing in High Energy and Nuclear Physics, Mumbay, India, Feb 13-17, 2006*,**147**, (2006).

[9] LcgCAF F. Delli Paoli,A. Fella,D. Jeans, D. Lucchesi, *et al.*, "LcgCAF - The CDF portal to the gLite Middleware", *Presented at Computing in High Energy and Nuclear Physics, Mumbay, India, Feb 13-17, 2006*, **148**, (2006).

[10] E. Laure *et. al.*, "Programming the Grid with gLite", *EGEE-TR-2006-001* (2006).

[11] S. C. Hsu, E. Lipeles, M. Neubauer, M. Norman, S. Sarkar, I. Sfiligoi, F. Wuerthwein, "OSG-CAF - A single point of submission for CDF to the Open Science Grid", *Presented at Computing in High Energy and Nuclear Physics, Mumbay, India, Feb 13-17, 2006*, **140**, (2006).

[12] M. Sechang Son Livny "Cluster Computing and the Grid,", *Proceedings. CCGrid 2003. 3rd IEEE/ACM International Symposium* , **542- 549**, (2003).

[13] S. Kosyakov, et. al., "Frontier: High Performance Database Access Using Standard Web Components", *Presented CHEP04, Interlaken Switzerland, Sept. 27-Oct. 1, 2004* , **204**, (2004).

[14] C. Moretti, I. Sfiligoi,D. Thain, "Transparently Distributing CDF Software with Parrot", *Presented at Computing in High Energy and Nuclear Physics, Mumbay, India, Feb 13-17, 2006*,**26**, (2006).

[15] S. Andreozzi, C. Aiftimiei, G. Cuscela, N. De Bortoli, G. Donvito, S. Fantinel, E. Fattibene, G. Misurelli, G.L. Rubini, A. Pierro, G. Tortone. "GridICE: Requirements, Architecture and Experience of a Monitoring Tool for Grid Systems", *Presented at Computing in High Energy and Nuclear Physics, Mumbay, India, Feb 13-17, 2006*, **359**, (2006).

[16] C. Aiftimiei, D. Andreotti, S. Andreozzi, S. Bagnasco, S. Belforte, D. Bonacorsi, A. Caltroni, S. Campana, P. Capiluppi, A. Cavalli, D. Cesini, V. Ciaschini, M. Corvo, F. Dellipaoli, A. De Salvo, F. Donno, G. Donvito, A. Fanfani, S. Fantinel, T. Ferrari, A. Ferraro, E. Ferro, L. Gaido, D. Galli, A. Ghiselli, F. Giacomini, C. Grandi, A. Guarise, S. Lacaprara, D. Lucchesi, L. Luminari, E. Luppi, G. Maggi, U. Marconi, M. Masera, A. Masoni, M. Mazzucato, E. Molinari, L. Perini, F. Prelz, D. Rebatto, S. Resconi, E. Ronchieri, G. Rubini, D. Salomoni, A. Sciab, M. Selmi, M. Sgaravatto, L. Tomassetti, V. Vagnoni, M. Verlato, P. Veronesi, M. C. Vistoli; " Prototyping production and analysis frameworks for LHC experiments based on LCG/EGEE/INFN-Grid middleware", *Presented at Computing in High Energy and Nuclear Physics, Mumbay, India, Feb 13-17, 2006*, **334**, (2006).