LcgCAF: a CDF Submission Portal to Access Grid Resources

Gabriele Compostella *University of Trento and INFN Padova*,
Francesco Delli Paoli *INFN of Padova*,
Daniel Jeans *INFN National Center for Telematics and Informatics, Bologna*,
Donatella Lucchesi *University and INFN of Padova*,
Subir Sarkar *INFN National Center for Telematics and Informatics, Bologna*,
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 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, and LcgCAF is basically a reimplementation of the CAF model in order to access Grid resources by using the LCG/EGEE Middleware components. 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. This paper presents an overview of the LcgCAF architecture within the Grid environment. The performances and future improvements are also discussed.**

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

I. Introduction

The Collider Detector at Fermilab (CDF) [1] is an experiment on the Tevatron collider where protons and antiprotons 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 full efficient since 2003. The current instantaneous luminosity is grater 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 about 900 TBytes of raw data and a total amount of about 2 PBytes is reached when reconstructed data and Monte Carlo data are considered.

To be able to process, analyze and produce this large amount of data CDF evaluated a cpus need of about 9 THz corresponding to about 3900 KspecInt2000 which will became around 20 THz in 2007 for an amount of $\sim$8600 KspecInt2000.

The CDF computing model, designed in the 2000, was based on dedicated farm hosted at FNAL and in other CDF institutions around the World, maintained by CDF personnel. Instead of keeping this model and growing the dedicated farms 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 cataloged using SAM [2], which provides distributed data access as well dataset and files history.

The reconstructed data is analyzed by users running on CDF Analysis Farms (CAFs) [3], the basic unit on which the computing model is based.

CDF has a CAF at FNAL dedicated to raw data reconstruction and one open to users for data analysis, these 2 CAFs have also a SAM station 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 and the data analysis can be done also there.

A. CAF overview

The CAF idea is that users develop and debug their analysis jobs on their desktop, where they have access to the CDF code and to the authentication tools. Then each job is split in hundred of parallel sections in order to parallelize it and then submitted to CAF via a custom interface. 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,

1-4244-0561-0/06/$20.00 ©2006 IEEE.

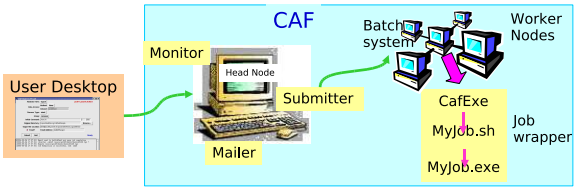

Fig. 1. The CAF architecture. Several daemons run on the head node while the job wrapper executes the users job on the worker nodes. nodes.

submitter, monitor and mailer. The submitter accepts the user tarball and store 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 allows the users to check the jobs status, each user command is translated in batch-system-specific request, the result is translated back to the user in a CDF specific format. 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 monitor tasks such as the creation of a job summary file. When the job finish the CafExe tarups 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 system but the user did not even noticed. The first CAF used FBSNG, then a big step forward was done implementing Condor [4], [5] batch system. Condor allows to manage dedicated pools and also access Grid resources. As discussed in the introduction, CDF cpus needs increase year per year and dedicated farms are not anymore sufficient to satisfy the experiment requirements. Moreover, even if CDF could have the possibility of having large dedicated farms, this is not convenient in term 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 is dynamically added to a condor pool keeping all the advanced features of the Condor batch system and building up a so-called GlideCAF [6]. From the users point of view GlideCAF is a standard CAF, while behind there is an access to the Grid distributed resources. One of the most valuable feature of GlideCAF is that the policies are managed both at Computing Element (CE) level by the VOMS service policies and at user level by the Condor batch manager. This feature is missing for the time being in the Grid LCG middleware.

GlideCAF has on the other hand, some limitations. There is a privacy issue related to the fact that all the glide-in job runs under a single VO specific UID, that can be solved using a very recent version of a tool, *glExec*, to change unix credentials based on Grid identity to match the UID with the real user.

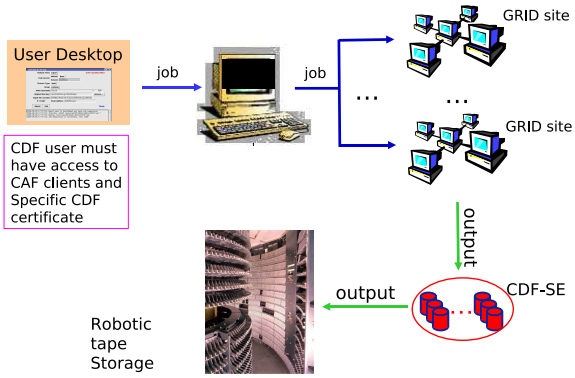

Fig. 2. General LcgCAF architecture . The portal with the necessary daemons and the job wrapper running on worker nodes.

An other limitation is due to the usage of the UDP protocol by Condor which does not work over Wide Area Network (WAN) while GlideCAF needs a bidirectional mechanism. This has the consequence that a GlideCAF has to be deployed in each site.

These limitations pushed to the investigation of a more Grid compliant CAF. Two projects were pursued in CDF: LcgCAF [7], a portal based on gLite Workload Management System (WMS) [8] and NAMCAF [9]. NAMCAF is based on GlideCAF having solved the issue of the communication over WAN by using Generic Connection Brokering (GCB) [10] a tool which allows cross-firewall communication. The software distribution could be done with Parrot(see IV-C for a discussion) but for the moment the self-contained tarball procedure is pursued.

IV. LcgCAF

LcgCAF is a totally 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. The general architecture, shown in figure 2 is based on a submission point which is basically an 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 CDF CAF clients. The users also need a valid Kerberos V ticket to be authenticated, in this way the communication between CAF clients and head node is crypted and secure. A cron job on the head node every day translate the Kerberos V ticket into a valid Grid proxy by contacting the VOMS server. The lifetime of this proxy depends on the VOMS configuration parameters and currently allows the job to survive for a maximum of one week, evaluated to be enough for a long Monte Carlo job. After the users are authenticated the CAF clients connect to the submitter daemon which then delegates the job to the WMS. A this point the submission is a Grid gLite submission to the Grid Computing Element. When the user job finish on a Grid site the output is stored on a CDF Storage Element (SE) or copied to a user defined location.

A. Job submission and execution

The CDF portal hosts several daemons which are responsible for the job submission, monitoring and user notification. The structure is the same already seen for the CAF with different functionalities. Figure 3 shows the CDF portal with the three major classes of daemons and then the path to the Grid site submission. The submitter is the service responsible

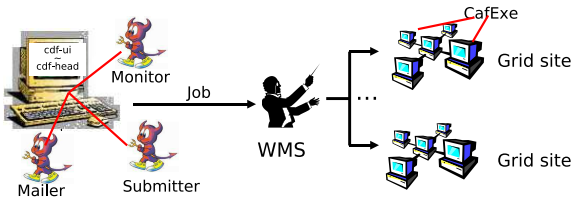

Fig. 3. LcgCAF submission and execution cartoon.

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. A Web server, also running on the head node, distributes these tarballs to the worker node when required. As for the former CAF the job execution can be parallelized splitting the user job into several segments. This can be achieved in Grid using different submission mechanisms which differ in the way the JDL file is created: DAG (Direct Acyclic Graph), and single submission. DAG submission allows to create a bunch of jobs with execution dependences, which means that the execution of a given segment has to wait for the successfully completion of an other. The dependencies paths can be very complicated but for CDF only a start segment before all the sections and a stop segment at the end are needed if data are processed and hence the CDF catalog has to be access. The actual version of the code uses the single submission with no dependencies which is enough for Monte Carlo production. The job is not directly submitted to Grid, but enqueued in a submission queue, since the 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. An other service on the head node looks for jobs on the submission queue and submit them to the gLite WMS.

The WMS perform the job submission to the more convenient CEs, where the more convenient CEs mean those with highest number of free cpus. The InputSandbox, one of the functionalities of the Grid submission used to bring files to worker node, is used 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. The output of the job can be copied or to a SE or to a CDF specific as discussed later.

When all the parallel jobs forming the user job are completed (or failed after retrials) an email is sent to the email address specified by the user. The mailer daemon queries the 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 monitor the job status with two tools: a web based and interactive monitor. A schematic view of how these monitors work is shown in figure 4. The interactive monitor (CafMon) provides a way to

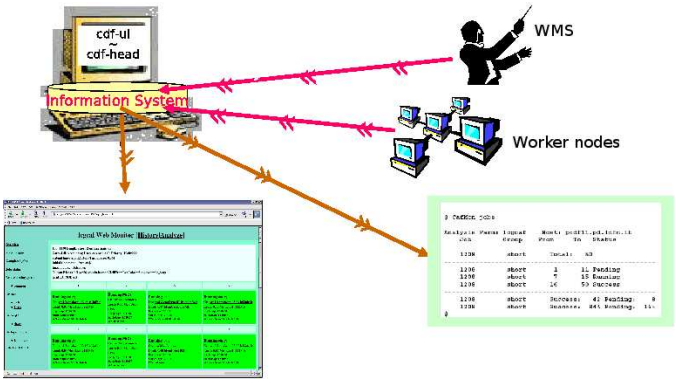

Fig. 4. LcgCAF monitors architecture.

view the status and log files of running jobs based on users queries. The query mechanism 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 for 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 pass on them to the CDF Information System (CDF-IS), a file-based database, on the head node.

Among he available commands the more used are: *CafMon list*, *CafMon dir*, *CafMon ps*, *CafMon kill*, *CafMon tail* and *CafMon top*. When the users send a request the information system is access and the user get 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.

An other 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 run a job needs CDF code and Run Condition database available to workers nodes even if it is a Monte Carlo production job. Since this can not be expected to be available 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 the access to this database even if in read-only is one of the most critical point. In order to satisfy queries from all the remote sites around the World a FroNTier system has been deployed [11]. The implementation consists of a middle tier that translates client requests into database specific queries and returns the data to the client as XML datagrams. Squid Proxy caching layers are deployed near the servers, as well as close to the clients, to significantly reduce the load on the database and provide a scalable deployment model.

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 it 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 [12], a virtual filesystem for performing Unix-like I/O on remote data services. Each time the application attempts a system call, the application 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 the 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 at Fermilab and Parrot retrieves the libraries from there introducing a latency time which is reduced by factor 25 adding Squid.

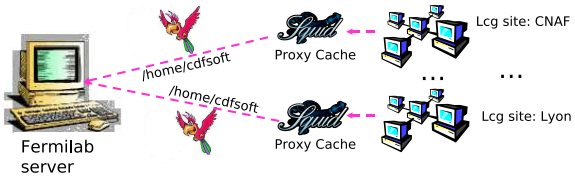

Fig. 5. LcgCAF code distribution.

D. Output storage

LcgCAF allows the users to copy the output of the job either to a Grid specific Storage Element and 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 ticked also on the worker node. To accomplish that a new mechanism is created, the *KDispenser*. This service creates on the head node the user ticket and transfers it to the worker node via GSI authenticated channel. In case of intensive Monte Carlo production files are preliminary stored into a SE or dedicated fileserver and then via an automatic procedure these files are transfered to Fermilab to an other dedicated fileserver from where are eventually cataloged and transfered 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 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 aborted segment is ignored and substituted by the new one. 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 tentatives, it is flagged as a Failed segment and is not submitted again.

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 Monte Carlo production groups for intensive Monte Carlo simulations. Figure 6 shows the number of running segment during the time mid-October, mid-November. A similar plot can be produced also looking at the

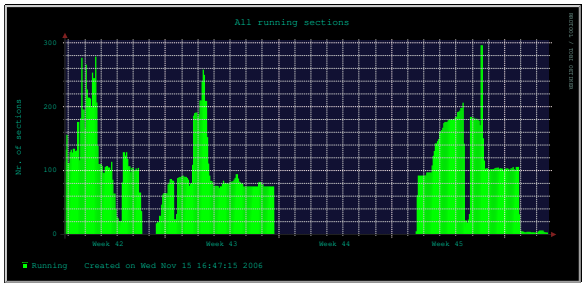

Fig. 6. Number of running jobs on LcgCAF in the period of time mid-October, mid-November.

Grid monitoring tools. The analogous plot in GridIce [13] is shown in figure 7. This represents the Italian resources usage by LcgCAF as seen by the Grid in one week of October.

Data Samples	Events	ϵ	ϵ_{1R}
B_s	6.2×10^6	94%	100%
Toy MC	-	100%	not needed

TABLE I
SUMMARY OF LCGCAF EFFICIENCIES.

This is important not only for a merely monitoring of the jobs but also for a future resources accounting that can be done if only completely compliant with the Grid requirements. The performances have been evaluated in two different cases.

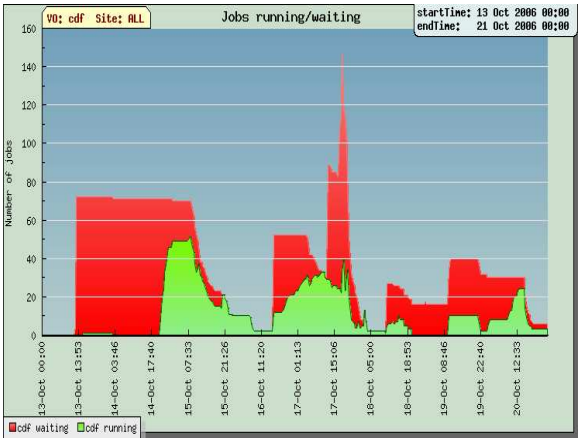

Fig. 7. Number of running jobs on LcgCAF has seen by GridIce in the period of time mid-October, mid-November. This monitors only Italian resources.

The first one during a massive B_s mesons Monte Carlo production ($B_s \rightarrow D_s\pi$ with D_s in $\phi\pi$, K^*K and K_sK), where something the order of 800 jobs at the same were running. The second one monitoring for a short period of time (a week) a user who was running toy Monte Carlo on LcgCAF. The results are summarized in table V. The efficiency ϵ, is defined as the number of successful segments divided by the total number of the submitted segments and ϵ_{1R} is the efficiency after one recovery. With one recovery all the job end successfully in the case of B meson production and no recovery is needed for toy Monte Carlo. The toy Monte Carlo jobs do not need access to CDF code or Database which reduce a lot the sources of errors. Moreover the toy Monte Carlo exercise is more recent than the B Monte Carlo production and as the times goes the Grid becomes more and more stable.

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 because LCG Grid Environment itself does not have a policy mechanism deployed in production status. However, 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. LCG Project is going to implement a new component, called GPBOX (Group and Policy BOX) [14], 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 the CDF users want to not only produce Monte Carlo data on the Grid but also analyze data an interface between Grid tools and SAM is needed, since SAM can not become a catalog supported by the 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 term of many supported operations like transferring file, removing file, retrieving file metadata etc which allows a mapping one-to-one 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 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 resource which has all the CAF characteristics adopted to the gLite middleware. This portal is now in production at CDF and performing quite well but it needs to improve as the World computing is moving to Grid.

ACKNOWLEDGMENT

The authors would like to thank Armando Fella for his contribution to LcgCAF code development at the first stage of the project. Daniele Cesini and WMS team for the continuous support together with all the CNAF personnel.

REFERENCES

[1] CDF Collaboration "The CDF II Thecnical Design report" *FERMILAB-Pub-96/390-E* (1996)
[2] 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).
[3] 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).
[4] D. Thain, T. Tannenbaum, M. Livny, "Distributed computing in practice: the Condor experience.", *Concurrency - Practice and Experience* **17, 2-4**, 323 (2005).
[5] I. Sfiligoi et al. "The Condor based CDF CAF.", *Presented CHEP04 , Interlaken Switzerland, Sept. 27-Oct. 1, 2004* , **390**, (2004).

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

[7] 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).

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

[9] 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).

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

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

[12] 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).

[13] 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).

[14] 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).