CDF Computing

Igor Sfiligoi [a,1,2],

[a] *INFN Frascati, Italy and Fermilab, Batavia, USA*

Abstract

The Collider Detector at Fermilab (CDF) is a detector on the Tevatron, currently the world's highest energy particle collider. Since the start of Run II in 2001, more than $1.4^f b{-}1$ of physics data have been collected, resulting in 600TBytes of raw data. To analyze all that data, CDF has deployed an efficient distributed computing system that serves both production needs and user analysis. This paper gives both an overview and a detailed description of the most prominent pieces.

Key words: CDF; computing; Grid; glideins, WMS

1. Introduction

The Collider Detector at Fermilab (CDF) is an experiment on the Tevatron, currently the world's highest energy particle collider. Although an older experiment, the CDF detector was thoroughly upgraded, and began its Run II phase in 2001. To date, Run II has gathered more than $1.4fb^{-1}$ of data, equivalent to $2x10^9$ events or 600 TBytes of raw data, and this is expected to double during the next year.

CDF computing needs are composed of raw data reconstruction and data reduction, event simulation, and user analysis. Although very different in the amount of resources needed, they are all naturally parallel activities. Any problem can easily be split in several independent sections, each processing a subset on the needed events. This makes it very suitable for distributed computing. As a matter of fact, CDF is currently using approximately 5000 CPUs to convert the collected data in physics results.

The CDF computing model is based on the concept of CDF Analysis Farms (CAFs)(1). In this model, the user develops and debugs on his own desktop, and when ready, submits the same script to one of the CAFs, splitting it in hundreds of parallel sections. The submission uses a custom protocol to submit to a CAF portal, with kerberos being used for authentication and integrity checking. In case of success, the user receives back a Job ID. The submitted job has no further contacts with the submit machine and the user output will be sent to any user-specified location. The job can be monitored from anywhere, provided the user knows the Job ID.

The data handling is managed by the SAM(2) system. In this model, a central database system manages all the experiment files and delivers them to the execute nodes. The use of caching and optimization make the file delivery in a very efficient one.

2. CAF overview

The CDF Analysis Farm (CAF) was developed as a portal. A set of daemons accept requests from the users, via kerberized socket connections and a legacy protocol. Those requests are then converted into commands to the underlaying batch system that does the real work.

[1] e-mail: sfiligoi@fnal.gov

[2] for the CDF Computing group

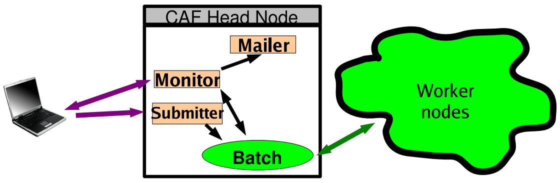

Fig. 1. The CAF portal

The CAF portal hosts three classes of daemons, as shown in Fig.2:

– The submitter daemon - Accepts the user tarball and stores it on the local disk. Creates the batch-system-specific submission files for the job sections and submits them to the batch system. Returns the batch system ID of the cluster.
– The monitor daemons - Accept queries from users and convert them into batch-system-specific requests. Interpret the results and send them back in a CDF-specific format.
– The Mailer daemon - Monitors the status of the jobs and send a mail every time a job finishes.

On top of that, each section has a wrapper associated with it. The task of the wrapper is to setup the security envelope and prepare the environment before the actual user code starts. When the user code finishes, it will also copy whatever is left to a user specified location.

The second task of the wrapper is to monitor the job while it is running. Such information is used both to log information not captured by conventional batch systems, and to provide pseudo-interactive access to the job. This way, CDF users can monitor the jobs almost like they run in interactive mode.

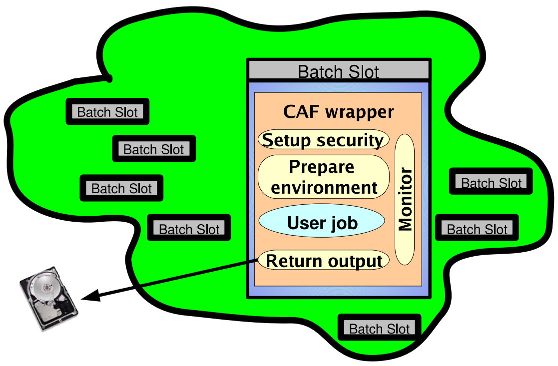

Fig. 2. The CAF wrapper

A schematic view can be seen in Fig.2.

2.1. *CAF evolution*

Separating the user interface from the CAF portal has proved to be very useful. Over the years, the CAF portal has undergone several reimplementations, while the user interface remained essentially the same. The CAF portal has gone from interfacing to a FBSNG-managed pool to a Grid based implementation, without the need for users to learn new interfaces.

The current main production CAF portal is based on the Condor(3) batch system. It can be used to both manage a dedicated pool and to access Grid resources. The Grid option is based on the Condor glide-in concept, where Condor daemons are submitted to the Grid, effectively creating a virtual private batch pool, known also as the pull model. This model is serving us well and has given us a smooth transition to the Grid.

Unfortunately, not all Grid sites like the pull model, so we have a second implementation of the CAF portal based on the gLite Workload Management System (WMS)(4). Although less flexible than the Condor glide-in option, it is sufficient for most of our organized activities and has allowed us to use a substantial amount of CPU hours.

2.2. *Condor based GridCAF*

The Condor batch system is based on the marketplace philosophy. Submit nodes advertise the jobs, while execute nodes advertise the available resources. An arbitration body then matches jobs to resources.

In a dedicated pool, the execute nodes have Condor pre-installed and those nodes become permanently part of a Condor pool. However, the same Condor processes can be started on any other node, and join the same pool, if authorized.

The Condor cased GridCAF(5; 6) uses the above idea to create a virtual private Condor pool out of Grid resources. An additional daemon, called the glidekeeper, submits Condor processes to all the willing Grid sites. A job containing Condor daemons is also known as a glidein job.

When a glidein job gets to a worker node, the Condor processes start, join the pool, and start pulling jobs from the job queue. A single glidein job can execute several jobs from different users, although a CDF glidein exits after the first job finishes. Although this introduces a small inefficiency, it makes the administrators of Grid sites happy, since it increases the Grid job turnaround.

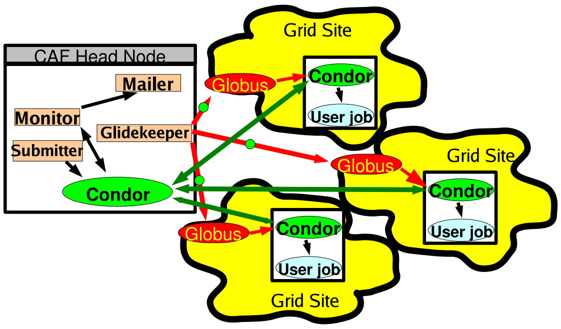

Fig. 3. The Condor based GridCAF

The big advantage of this approach is that all the Grid infrastructure is hidden by the glideins. The jobs themselves can run both on local, dedicated worker nodes and on opportunistic Grid resources, without any change. Moreover, all the advanced features of Condor, like its fine grained fair share mechanism, are preserved.

A second advantage comes from the pull model itself. Given that the user job is sent to a worker node only once it is available and tested, the CAF does not have to guess in advance to which queue and on which Grid site should it go. Plus, any miss-configured worker node, or even Grid site, will kill only the glideins, not user jobs, further raising the success rate.

2.3. *gLite WMS based GridCAF*

Unfortunately, the pull model has some disadvantages, the two major being the use of a single proxy for all the glidein submissions and the reliance on the availability of outgoing connectivity.

The gLite WMS based GridCAF(7) is based on the push model. In this implementation, the Grid itself replaces the local batch system, and the CAF portal talks directly to the gLite WMS, also known as the Resource Broker, that does the real work.

Although on paper a much more elegant solution, this approach still does not deliver all the features we come to expect from the Condor based CAF. In particular, the management of fair share is very rudimentary if at all present, and the user job success rate is slightly lower.

However, it does allow us to use Grid sites where the Condor based GridCAF would not work at all, and is adequate for MC production needs.

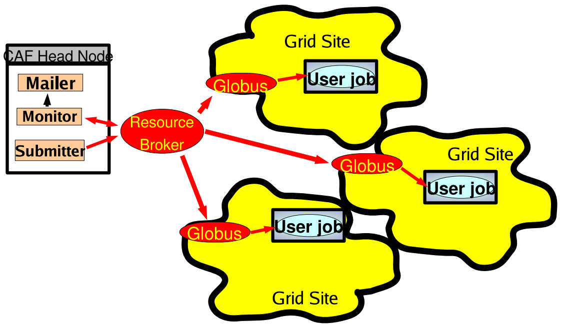

Fig. 4. The gLite WMS based GridCAF

3. Software distribution overview

The CDF software distribution has historically always been available on all the worker nodes. This was necessary since several production, and most user analysis jobs were using it. On the dedicated worker nodes at Fermilab, the software distribution was exported by means of NFS mounts. Unfortunately, in the Grid world this is not an option anymore.

To address this problem, CDF went trough all of its production jobs and found out all the dependencies on the software distribution. Once found, those dependencies were removed, and all the production jobs are nowadays self contained, i.e. all the necessary binaries, scripts and shared libraries are sent to to the worker node in the tarball.

Unfortunately, this is not that easy for generic user jobs, given the wide variety and nested structure of user jobs. So for now, those jobs are restricted to our dedicated resources only. However, we are looking at new technologies, like Parrot over HTTP(8), to make the software distribution once again seamlessly available to all the worker nodes. These efforts are not yet mature for production use, but have been demonstrated in test runs.

4. Data Handling overview

The CDF data is managed by the SAM(2) system, centered around a central database system that contains the metadata and the location of all the experiment files. The files at any location are managed by a set of services, called a SAM station. Each SAM station has a storage area associated with it that is manages autonomously.

Files are grouped in datasets, that are generated both by the production tools and by final users themselves. A whole dataset can be used to create a SAM project, that will be used to access data via a specified SAM station. Once a project is created, processes can ask for files, and the SAM station will serve files starting with the ones already in the local storage, and fetching the others from other SAM stations in parallel, thus minimizing the wait times.

Using this mechanism, CDF users are accessing over 1.5PBytes of data. Fermilab hosts both the tape library that contains all the data and a 290TBytes disk cache. The second largest SAM station is hosted at the Italian CNAF, with a 70TBytes disk cache.

While CDF has done well with delivering files to the jobs, the handling of job output was left to the users. The CAF itself does send back user log files and smaller data files, but users are still required to manage the bulk of their data by themselves.

As CDF is moving more computing on the Grid, this has proved to be inefficient and new mechanisms will need to be put in place.

5. Summary

The CDF computing has been very successfull so far. We were able to reconstruct, simulate and analyze all of our data in a timely fashion and publish high quality papers about the physics at the Tevatron. In the past year, CDF needed an excess of 4000 CPUs to reach that goal.

The CAF concept have been very helpfull. The portal philosophy allowed us to rapidly and painlessly scale from a small in-house cluster to the Grid world. In this effort, the choice of mature and trustfull partners, like the Condor group, was especially important and allowed to succeed where many others have found significant problems.

The SAM data handling have proved to work well for the data serving needs of CDF. However, we have delegated for too long the problem of user output to the users themselves, and are only just now looking for a global solution. Although we are still doing reasonably well, we do need to improve it as more CDF computing is moving to the Grid.

Acknowledgements

CDF owes a lot of gratitude to the Condor team for all their help in setting up and scaling up of both the regular Condor CAF and the glide-in based GridCAF.

References

[1] 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).

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

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

[5] 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).

[6] 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).

[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] 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).