Using Condor glide-ins and Parrot to move from Dedicated Resources to the Grid

Stefano Belforte – INFN Trieste, Italy

Matthew Norman – University of San Diego, California, USA

Subir Sarkar – INFN-CNAF, Bologna, Italy

Igor Sfiligoi – INFN Frascati, Italy & Fermilab, Chicago, USA

Douglas Thain – University of Notre Dame, Indiana, USA

Frank Wuerthwein- University of San Diego, California, USA

Abstract: In order to meet future needs of data analysis and MC production, the CDF experiment is reorganizing its computing model from dedicated Condor pools towards the Grid computing paradigm. The direction chosen has been to augment the existing model with the Condor glide-in mechanism which can dynamically extend an existing Condor pool into Grid resources.
The main problems we found in deployment were restrictive firewalls, so we had to add another piece of infrastructure, a Condor proxy mechanism called Generic Connection Brokering (GCB), that allows us to bridge them. The second problem we faced was related to software distribution. Given the number of CDF users and the amount of existing applications, it was impractical to impose new ways of software access on the end-users. The solution we have adopted instead is to use Parrot, a user level application that allows a generic executable to access remote files as if they were local.

Introduction

CDF is a high energy experiment at the Tevatron collider located at the Fermi National Accelerator Laboratory in Chicago, USA. The experiment acquires data at a sustained rate of 60 MB/s and needs an ever increasing amount of CPU power for analysis and simulation. The data are composed of independent pieces, called events, that can be analyzed and/or simulated in parallel; nevertheless, the total amount of CPU power required is huge. The current requirement is of the order of 5M SPECint2k which is expected to climb to almost 15M SPECint2k by the end of 2007.

The current model of event reconstruction, simulation and analysis has been based on dedicated Condor pools with a CDF-specific submission infrastructure, called CDF Analysis Farms (CAFs)[LNSW04]. In order to meet future needs, the CDF experiment is reorganizing its computing model and is moving from dedicated resources towards the Grid computing paradigm. This has been dictated mainly by the higher instantaneous luminosity of the Tevatron collider, which requires large increases in computing resources, and the realization that any further expansion of dedicated resources is practically impossible.

The path chosen has been to augment the existing model with the Condor glide-in mechanism which can dynamically extend an existing Condor pool into Grid resources. Moreover, since moving from dedicated to Grid resources eliminates the possibility to have CDF specific software on the worker nodes, Parrot has been used to access such software over HTTP.

Condor Glide-Ins

Overview

A regular Condor pool is composed of a set of daemons managing different parts of the system. The pool is defined by the **collector** daemon, which gathers information about all other daemons. The worker nodes are managed by **starter** daemons, while user jobs are managed by a process called the **schedd**. Finally, the **negotiator** daemon assigns jobs to starters, which execute them. A simple overview is given in Figure 1; for more information have a look at [LNSW04].

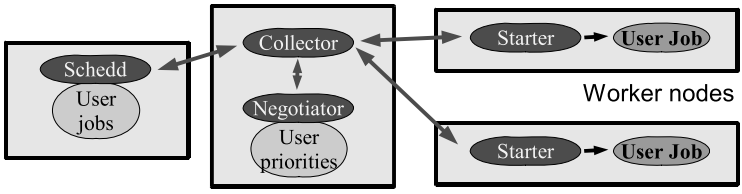

Figure 1: A dedicated Condor pool

Condor glide-ins are just regular Condor starter daemons, properly configured and submitted as jobs to Grid Computing Elements (CEs). Once such a job starts on a worker node, it contacts the designated Condor collector and joins the Condor pool as a new virtual machine (VM). From the Condor point of view, such a resource is indistinguishable from a dedicated one, and it will be matched in the same way to a user job with the best priority. Such a job will then be sent to that VM, effectively starting on the Grid worker node. See Figure 2.

The original CAF we started from was Condor based, so the glide-in mechanism was the easiest way to extend it. A simple glide-in factory was all that was needed, and from the user point of view, nothing changed.

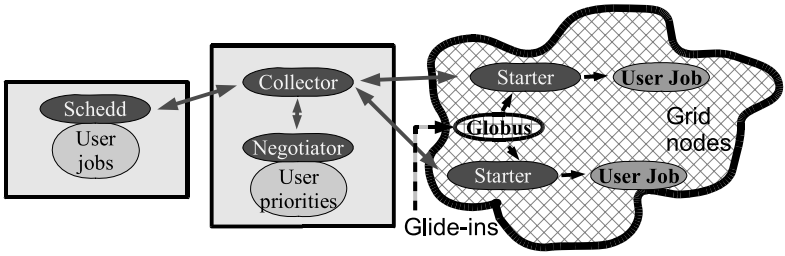

Figure 2: A glide-in based Condor pool

Moreover, the Condor glide-in mechanism has several advantages over direct submission of user jobs to the Grid CEs:

- **Finer grained policy management**: user priorities are managed inside the Condor pool, preserving all the power and flexibility of the Condor fair share policies. Moreover, the policies are managed at two different levels: by virtual organization (VO) at the CE level, and among users at the Condor level.
- **Black holes are not a problem**: a defective node will kill glide-ins before they start, so no user job is compromised. Additional sanity checks can be performed, discarding worker nodes that do not meet the needs of the virtual organization before a user job is started there.
- **Late binding of resources**: in case of multiple Grid sites, just glide-ins are sent to the queues of the CE. User jobs, on the other hand, are sent to the Grid site only after a glide-in started and resources are available, eliminating the risk of long, possibly infinite waits in the CE queues.

The First Implementation

CDF has always adopted a step-by-step approach to problem solving and the present one was no exception. The first step was to extend the Condor pools at the sites that already had a CAF installed and enjoy close ties with the local Grid administrators. Several such pools have been deployed at different Grid sites, most of which are in production mode.

The biggest such CAF is installed at CNAF, the Italian Tier-1 center in Bologna. It was deployed in September 2005 and has since run ~700k jobs from approximately 150 users. The system is behaving very well, with virtually no job loss, and has proved to scale up to the full size of the present Tier-1, i.e. ~2k slots, as can be seen from Figure 3. The only major complaint from users were related to long directory path names due to the multiple middleware layers.

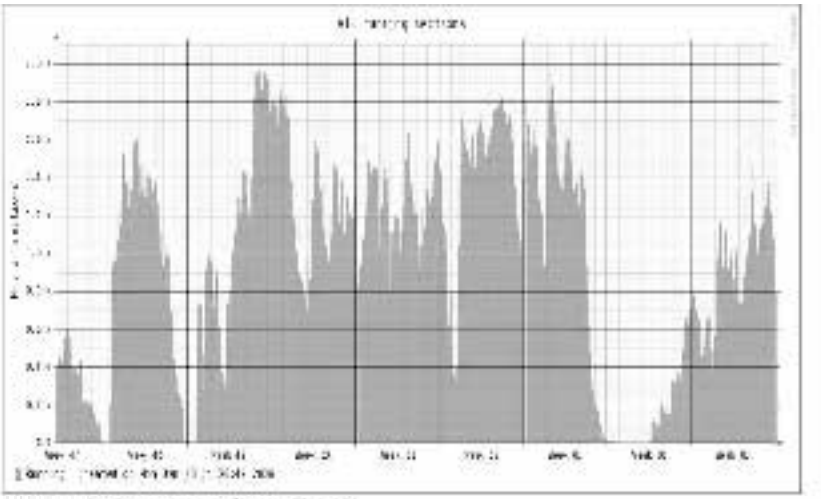

Figure 3: Running jobs at CNAF

The glide-in solution was easy to implement, but it is not very generic. In order for it to work, no firewalls are allowed, a fast and reliable network is needed between the Condor submission node and worker nodes, and the CDF software distribution needs to be served to the worker nodes by means of a shared file system.

To remove the above limitations, three tools have been identified: the newly released Generic Connection Brokering (GCB) to bridge firewalls and work over wide area networks, Parrot for software distribution, and HTTP caches for limiting the required network bandwidth.

All three are planned to be present in the final incarnation of the Grid-based CAF. However, each separately improves the functionality of the system and has been tested and deployed independently.

Condor On The Wide Area Network

Overview

The Condor system was developed with a local area network in mind; Condor daemons continuously exchange messages by means of UDP packets. Unfortunately, such a mode is not suitable for the wide area networks where UDP packets are often lost and proxy services are needed to bridge firewalls. To solve these problems, the Condor team has developed the Generic Connection Brokering (GCB) service[SFL05] and has released the first production version in November 2005.

GCB is a proxy server, used by the starters to route all their communication, as described in Figure 4:

- a starter establishes a permanent TCP connection with GCB,
- communicates the GCB node and the obtained port to the other parties,

- from this point on, any Condor daemon can connect via TCP to the specified GCB, that will route the traffic to and from the correct open TCP connection.

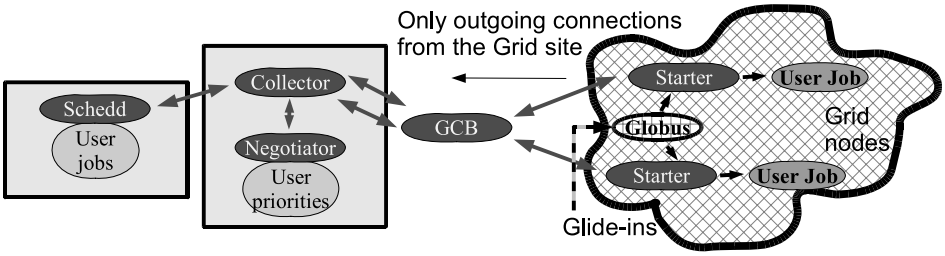

Figure 4: A Condor pool using GCB

The GCB is very flexible:

- It can be installed anywhere in the world, as long as the hosting machine accepts incoming network traffic. In a very restrictive environment, when nothing can enter nor exit the site boundaries, the GCB would need to be installed on a machine that can see both the site network and the WAN.
- Several GCB servers can be installed, both for scalability reasons and to accommodate restrictive sites.
- On the other side, when outgoing connections are available, GCB can be used just for notification, and the starter will directly connect to the other daemons using TCP, for efficiency reasons.

GCB supports strong authentication, currently by means of Grid Security Interface (GSI). Kerberos should be supported soon, too.

The First Implementation

The first successful test system has been assembled in early January 2006. This system was configured to:

- run the Condor collector and schedd at Fermilab
- the GCB run at Fermilab, too
- the glide-ins were submitted to the CE of the OSG site at the San Diego Supercomputing Center (SDSC), so they run on SDSC worker nodes, on a private network

Note that this OSG site is already used for a regular glide-in pool, so software distribution was not a problem. No problems were found, and user jobs were able to run unmodified.

Work is now beginning on the assembly of an OSG wide Condor pool, which will use GCB to communicate with glide-ins running throughout all the OSG sites. As a preliminary to this, glide-ins are being submitted to multiples sites, with all communications routed back to the head node via GCB. Successful submissions and communications have been confirmed at OSG sites at both the University of Wisconsin, and Brookhaven National Laboratory, although scaling tests are currently impaired due to lack of CDF priority.

Deployment of the above pool, with beta-testing with production jobs, is slated to begin in February of 2006.

Software distribution on the Grid

Overview

The CDF software model was developed with dedicated resources in mind. One of the main assumptions is the availability of a large set of executables, shared libraries and configuration files on a shared file system. As long as CDF ran on dedicated resources distributed over only a few sites, this was easy to implement. In the Grid world, although possible, installing and maintaining a copy of the CDF software distribution at each and every Grid site is not a very attractive solution.

One obvious step was to create self contained tar balls, including the user executable and all the used shared libraries, for as many use cases as possible. We were indeed quite successful for most organized activities and some user analysis, but given the amount of both users and existing applications, it proved impractical to force all the users to change their way of work and stop relying on the CDF software distribution completely.

Instead, we decided to keep only one copy of the software at Fermilab and access it directly from the worker nodes. Our jobs make heavy use of scripts, so modifying just the analysis binary to access that was not an option. Luckily, using Parrot solved the problem for the general case.

Accessing remote files from a generic executable

Parrot[TKWBSI05] is a user level application that allows a generic executable, or script, to access remote files as if they were local. It works by trapping a program's system calls through the ptrace debugging interface, and replacing them with remote I/O operations as needed. It follows the complete lifetime of the executable, serving also any child that is spawned from the original process.

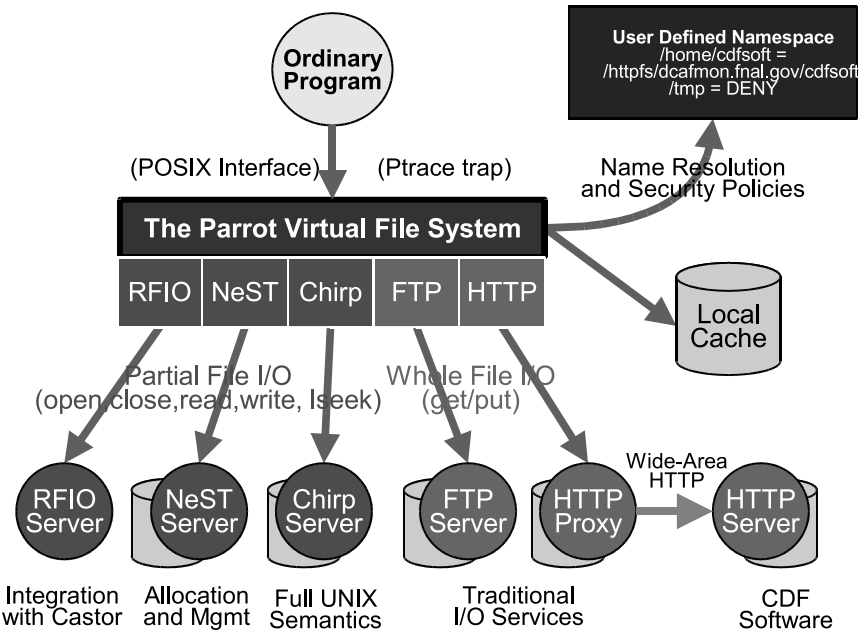

Figure 5: Overview of Parrot

The flexibility of Parrot does come with a price tag:

- A small CPU overhead is imposed on every I/O operation, no matter if it is local or remote. The local overhead seems to be minimal and becomes noticeable only for throughputs in excess of 10MB/s or 150k ops/s. Remote access overhead strongly depends on the protocol involved. Testing on real CDF jobs, using the HTTP protocol, showed an average overhead of about 5%.
- Suid executables cannot be run under Parrot; the operating system will non allow Parrot to attach to the ptrace debugging interface, for security reason.

Neither of the above seem to be too high a price for the service we get.

Parrot supports several remote I/O protocols, including HTTP, FTP, GridFTP, Chirp, Nest, rfio and dccp. CDF is committed to use only HTTP, because this is the most mature non-authenticated protocol that allows for easy proxy caching and load balancing.

Using proxy caching

HTTP proxy caching is an established mechanism used in a variety of fields, so it will not be described here. We just want to stress that for the purpose of distributing CDF code it is extremely useful, since virtually all information we serve via HTTP does not change and is accessed by a large number of jobs.

Installing one or more Squid servers at or near a Grid site can drastically reduce the bandwidth needed between the central information repository and the site. Moreover, using a Squid server on a machine on the border of the Grid site, can allow us to work on sites without outgoing connections.

The First Implementation

Parrot has been tested on real user jobs and the results are promising. Two simulation packages, one based on fully compiled C++ code and one based on interpreted ROOT code, have been tested within Parrot. No problems have been found when running in interactive, but they refused to run in batch mode. The later is still under investigation.

We tested Parrot both with and without an intermediate Squid server and as expected, the results were the same. When running on remote sites, HTTP caching greatly reduced the wall clock time needed to run user jobs after the first one.

The above tests were done in mid January 2006, so there were no time to test the scalability of the system.

Putting everything together

Due to bugs found in both GCB and Parrot, we were able to run with these pieces of infrastructure just weeks before the due date of this paper, so there was no time to put everything together.

Everything seems to work fine now, so we expect to assemble a fully working system, integrating all the components, by early February 2006.

Conclusions

CDF is committed to changing its computing model from dedicated Condor pools towards the Grid computing paradigm, and to do it with minimum changes to the way the physics analysis is done by physicists. Extending the existing Condor-based infrastructure with the addition of a glide-in factory, paired with the virtualization of the of the software distribution access, seems to achieve this goal.

CDF has been successful in exploiting Grid resources on several friendly sites, and glide-in based resources now account for almost a third of the total CDF CPU resources.

However, we do want to use any available Grid resource we are allowed to use, so we are extending our infrastructure to remove any limit that would prevent us to get there. The two most limiting factors, working over WANs and bridging firewalls, and the distribution of CDF software, are being addressed and the needed technical solutions, namely GCB and Parrot, are available. The preliminary results are very promising, and we expect to have a working system available before the end of February 2006.

References

[LNSW04] Lipeles, E.; Neubauer, M.; Sfiligoi, I.; Wuerthwein, F.: The Condor based CDF CAF, CHEP 2004 proceedings, Interlaken 2004.

[SFL05] Son, S.; Farrellee, M.; Livny, M.: A Generic Proxy Mechanism for Secure Middlebox Traversal, Cluster 2005 proceedings, Boston 2005.

[TKWBSI05] Thain, D; Klous, S.; Wozniak, J.; Brenner, P.; Striegel, A.; Izaguirre, J: Separating Abstractions from Resources in a Tactical Storage System, Proceedings of Supercomputing 2005, Seattle 2005.