JobMon: A Secure, Scalable, Interactive Grid Job Monitor

C. D. Steenberg, Caltech, Pasadena, CA, USA
S.-C. Hsu, E. Lipeles, F. Würthwein, UCSD, San Diego, CA, USA

Abstract

We present the architecture and implementation of an interactive system for monitoring long-running jobs on large computational clusters. JobMon comprises an asynchronous intra-cluster communication server and a Clarens web service on a head node, coupled with a job wrapper for each monitored job to provide monitoring information both periodically and upon request. The Clarens web service provides authentication, encryption and access control for any external interaction with individual job wrappers.

INTRODUCTION

The massive data storage and processing demands of modern high energy physics is driving the use of an ever-increasing number of distributed, but interconnected computing centers of various sizes. This distributed heterogeneous environment presents an extra layer of complexity to both users and administrators who want to maximize the use of the available resources.

Grid technology has proven to be a useful tool in providing a scalable distributed authentication, authorization and execution environment on the scale of computing clusters. In general, however, management of individual jobs has been left to the job scheduler and other tools on on each cluster.

In the high energy physics community jobs are not run as single applications that span multiple compute nodes, but rather as one or more individual jobs per compute node. This increases the complexity of managing and debugging jobs because there is potentially orders of magnitude more jobs than is the case for problem domains where large message passing jobs are the norm.

We will describe a system that aids in the monitoring and interactive debugging of individual computing jobs running on compute nodes in a Grid environment. This interactive approach presents additional technical and organizational challenges compared to passive monitoring systems where information flows from a sensors (jobs in this case) to one or more aggregators. These challenges include job tracking over multiple sites, security, private network configuration at various Grid sites, and probably most importantly, security.

Other monitoring systems in use in HEP today that include job monitoring to a lesser or greater extent include MonALISA [3], BOSS [1] and R-GMA [5]. The most advanced and complex of these is the MonALISA, which is built upon a distributed agent-based architecture and is able to consume information from a large number of sensors into multiple aggregators.

JobMon implements an architecture that addresses these problems, generalizing an earlier implementation developed as part of the CDF experiment.

ARCHITECTURE

The JobMon architecture, shown in Figure , consists of a components on the client a server normally at the execution site, as well as on the worker node where the job is running.

The client issues web service requests to the Clarens server instance over the WAN or LAN using XML-RPC over https or any of the other protocols supported by both server and client, e.g. SOAP or JSON, and is encrypted. The server in turn issues commands to either a UDP or TCP wake up module via IPC over a named Unix pipe. The server is responsible for access control to any communication with the worker node, so that clients can only communicate with their own jobs.

Clients that can interact with the system can be web browsers, Python scripts, ROOT [6] analysis scripts, compiled C/C++ code, or any other web service client that can be authenticated using an X509 certificate.

On the worker node, a process named *JobMonDaemon* is started which handles communication and executes commands on the worker node. The *JobMonDaemon* process persists for as long as the job is running, and communicates with the Clarens server via web service calls, as well as being contacted by the appropriate wake up client. For security reasons this process can not be used as a server itself, and for this reason many sites disable listening on server ports by regular users by default.

An important criterion for the design of *JobMonDaemon* was that is should be very lightweight in terms of memory and storage consumption as well as requiring no or very little installed software dependencies on the worker node. The server, wake up client and *JobMonDaemon* are written in Python, and fit this requirement very well since the interpreter is installed by default on the CDF analysis facility which was targeted as an installation site.

The Clarens server can be installed as part of the the Virtual Data Toolkit [9] as a normal user or as a set of OpenPKG [4] packages by a system administrator..

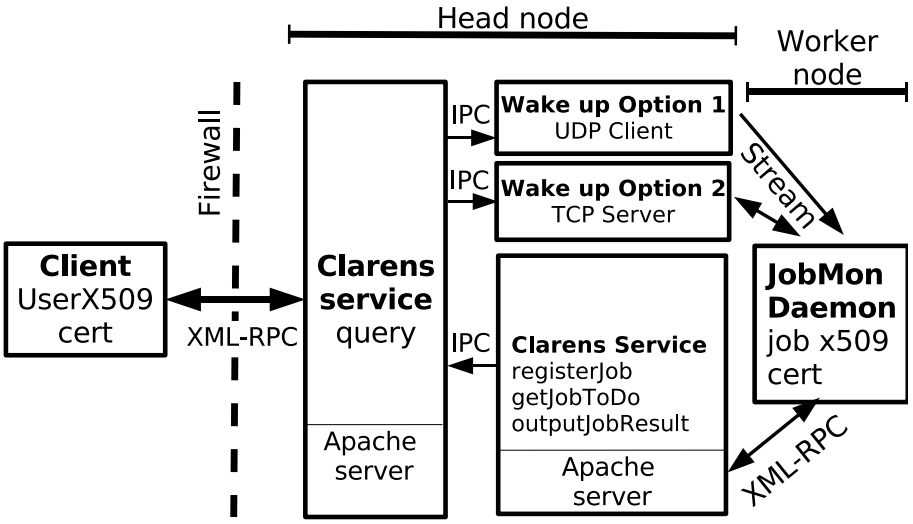

Figure 1: JobMon Architecture showing the client on the left, two Clarens services for access by the client and the job respectively, as well as two possible wake-up services, one using TCP and one using UDP, and the *JobMonDaemon* running on the worker node on the right.

COMPONENT INTERACTION

Job initialization

For a job to make use of the facilities provided by the JobMon system, it must include the *JobMonDaemon* code. Since the job must authenticate itself with the Clarens server a public/private key pair unique to the job must also be sent with the job unless it's credentials can be obtained through some other means, e.g. Kerberos.

When the job reaches the worker node and it gets executed, *JobMonDaemon* is started up and registers itself with a specified JobMon server and retrieves the configuration for that server. This makes it possible for the configuration to be both site-dependent and dynamic.

Depending on the configuration retrieved, *JobMonDaemon* will either make a TCP client connection with a TCPJobMonServer running on the server specified in the configuration, or start listening to UDP wake up messages.

User request handling

When a user accesses the JobMon web service by calling the `query` method, the request is authenticated using the standard Clarens access controls, as well as only allowing the job submitter to send commands to the job. Authentication is described in more detail in the next section.

After the request is authorized, a wake-up message is sent to the job via UDP broadcast or through the TCPJobMonServer connection. If a job gets woken up in this way it will contact the JobMon service using the `getJobToDo` method which will provide *JobMonDaemon* with information about the command it should execute.

After executing the given command, *JobMonDaemon* calls the `outputJobResult` method on the server, which in turn returns the information to the client who made the initial request.

Internally, the Clarens server processes handling the `query`, `getJobToDo`, and `outputJobResult` web service method calls communicate via Unix named pipes.

AUTHENTICATION AND AUTHORIZATION

Two aspects of authentication and authorization needs further explanation.

Firstly, it is imperative that the job key pair distributed with the job never be sent over unencrypted network connections. If the key pair does become compromised an attacker would only gain access to the `registerJob`, `getJobToDo`, and `outputJobResult` methods as access to all methods is controlled based on the identity of the caller.

Secondly, the way that a caller is authorized to send commands to a job must be clarified. Once a user has been authorized by the Clarens access control mechanism to call one of the JobMon service methods, the X509 certificate subject is used to determine whether the user is allowed to interact with a particular job.

Certain users may be given administrator privileges to send commands to any jobs, while the certificate subject of normal users must be a substring of the job certificate substring.

E.g. a user identified by the certificate subject `/DC=gov/DC=fnal/O=Fermilab/OU=People/CN=<name>` will be allowed to send commands to the job with certificate subject `/DC=gov/DC=fnal/O=Fermilab/OU=People/CN=<name>/CN=<jobid>`.

The issuing Certificate Authority is charged with ensuring that the certificate subjects are unique, and the jobid is chosen such that name clashes within one user's job namespace are assumed to be extremely unlikely.

For jobs submitted by users with Fermilab kx509 certificates subjects have the form
`/DC=gov/DC=fnal/O=Fermilab/OU=People/CN=<name>`
`/UID=<user>`
and the job certificate subjects are of the form
`/DC=gov/DC=fnal/O=Fermilab/OU=Robots/CN=cdf`
`/CN=<name>/0.9.2342.19200300100.1.1=<user>`
In this case the service could require that the < name> and <user> parts of the subjects must match.

The job key pair validity period must be chosen such that it doesn't expire while the job is running, but also such that it doesn't leave the server accessible by this certificate for an extended period of time.

PERFORMANCE

Although the performance of the Clarens implementation was shown [8] to be quite reasonable for up to 80 simultaneous clients, the JobMon service presents uncharted territory with up to 3000 nodes and tens of clients needing to contact the server.

Specifically, each *JobMonDaemon* listening in either UDP or TCP modes needs to periodically re-register itself with the JobMon service since there is no other way for the server to know which jobs are still running. This reregistration traffic should exceeds any traffic generated by external clients on average.

Tests have shown that re-registrations at a rate of 4 Hz allows the server to respond with a latency of around 1 second. In a cluster with 3000 nodes, this corresponds to a reregistration time of about 12 minute. If the registration rate is increased by a factor 2.5 to 10 Hz, the server respond time increases to $\approx$10 seconds.

CONCLUSION

The JobMon system allows individual users to interact securely with running jobs, via a two phase callback mechanism to pass queries to the *JobMonDaemon* job wrapper, which responds asynchronously with the results of these queries. and has been deployed and tested on the CDF Central Analysis Facility at Fermilab since March 2005. The major user of JobMon up to date has been the LCG Central Analysis Facility [2] with an average of 100 concurrently registered jobs.

JobMon has also been part of the OSG Monte Carlo Processing Service (MCPS) deployment since July 2005, where it can be used by jobs in the same way as in CDF.

The performance of the Clarens server was shown to limit the number of nodes that can make use of the JobMon service, depending on the desired job reregistration frequency. In practice, the average number of concurrently registered jobs has been significantly lower than the maximum number possible.

JobMon is not undergoing any further active development at this time and is a stable component of the Grid-enabled Analysis Environment used within the CDF experiment.

ACKNOWLEDGMENTS

This work supported by Department of Energy contract DE-FC02-01ER25459, as part of the Particle Physics Data Grid project.

REFERENCES

[1] http://boss.bo.infn.it/

[2] http://www.pi.infn.it/cdf-italia/ public/offline/lcgcaf.html

[3] MonALISA: An Agent based, Dynamic Service System to Monitor, Control and Optimize Grid based Applications I.C. Legrand, H.B. Newman, R. Voicu, C. Cirstoiu, C. Grigoras, M. Toarta, C. Dobre, Proceedings of CHEP 2004, Interlaken, Switzerland, September 2004

[4] http://www.openpkg.org

[5] http://www.r-gma.org

[6] http://root.cern.ch

[7] Conrad D. Steenberg, Eric Aslakson, Julian J. Bunn, Harvey B. Newman, Michael Thomas, Frank van Lingen, Proceedings of CHEP 2003, paper MONT008, 2003

[8] The Clarens Grid-Enabled Web Services Framework: Services And Implementation Conrad Steenberg, Julian Bunn, Iosif Legrand, Harvey Newman, Michael Thomas, Frank van Lingen, Ashiq Anjum, Tahir Azim Proceedings of CHEP 2004, paper 184, 2004

[9] http://vdt.cs.wisc.edu