A Grid Job Monitoring System

Catalin Dumitrescu[1], Andreas Nowack[2], Sanjay Padhi[3] and Subir Sarkar[4,†]

[1] Fermi National Accelerator Laboratory, USA [2] RWTH Aachen, Germany [3] University of California San Diego, USA [4] INFN, Sezione di Pisa & Scuola Normale Superiore, Pisa, Italy

Abstract. This paper presents a web-based Job Monitoring framework for individual Grid sites that allows users to follow in detail their jobs in quasi-real time. The framework consists of several independent components : (a) a set of sensors that run on the site CE and worker nodes and update a database, (b) a simple yet extensible web services framework and (c) an Ajax powered web interface having a look-and-feel and control similar to a desktop application. The monitoring framework supports LSF, Condor and PBS-like batch systems. This is one of the first monitoring systems where an X.509 authenticated web interface can be seamlessly accessed by both end-users and site administrators. While a site administrator has access to all the possible information, a user can only view the jobs for the Virtual Organizations (VO) he/she is a part of. The monitoring framework design supports several possible deployment scenarios. For a site running a supported batch system, the system may be deployed as a whole, or existing site sensors can be adapted and reused with the web services components. A site may even prefer to build the web server independently and choose to use only the Ajax powered web interface. Finally, the system is being used to monitor a glideinWMS instance. This broadens the scope significantly, allowing it to monitor jobs over multiple sites.

1. Introduction

Real-time job monitoring with easy access to job log files is essential to end-users who run long-lived analysis jobs and wish to follow the various stages during the lifetime of a job. Computing efficiency of a site critically depends on how fast mis-behaving jobs are found and removed. In a distributed environment it is even more crucial and affects the overall success of the Grid Computing in High Energy Physics in a significant manner. A number of monitoring tools do exist in the Grid world but none of them even comes close to real time monitoring that we are so used to with local batch systems.

The Grid Job Monitoring system discussed in this article is a web-based framework that enables users to track jobs in substantial detail in quasi-real time. This is one of the first monitoring applications where both end-users and site administrators can transparently access information about Grid jobs using an X.509 authenticated web interface. A site administrator has access to all the information, whereas to users the visibility of jobs of a Virtual Organization (VO) depends on privacy settings of the site. The framework supports LSF, Condor, and PBS like batch systems through a plug-in mechanism and can be easily extended to integrate other batch systems. The system has been extended to monitor glideinWMS [1] instances also. This broadens the scope significantly, allowing a single interface to monitor jobs over multiple sites.

† Corresponding author, email : subir.sarkar@cern.ch

The web-based presentation layer is a Rich Internet Application (RIA) that is based on a basic Asynchronous JavaScript and XML (Ajax) pattern [2]. An Ajax powered web application matches desktop applications in terms of interactivity and simplicity. The added advantage is that such an application is accessible from anywhere and requires only a modern web browser and no other client-side software installation.

2. Objective

The monitoring system aims to achieve the following primary goals for end-users and site administrators :

❑ End-users should be able to

- track jobs using global job-id and for each job find the following basic information :
    - the summary information, process list,
    - CPU, memory, and disk usage with time,
    - job and working directory listing,
    - access to job specific, user defined log files anywhere within the working directory,
    - status of the computational (a.k.a worker) node that runs a job,
- select jobs based on
    - Resource Broker/WMS,
    - site Computing Element,
    - submission/start time

  for further debugging purposes.

❑ Site administrators must easily find jobs,

- running on individual worker nodes,
- by local queues/users,
- behaving unexpectedly (e.g. 0 CPU load, expiring Grid proxy, etc.)

using local job-id, in order to spot local problems immediately.

All these should be accessible using a single, intuitive, secured web interface respecting privacy of information.

3. Description

The framework consists of several independent components : (a) a set of sensors that run on the site's computing elements and worker nodes and update a database, (b) a simple yet extensible application server framework based on Common Gateway Interface(CGI) and (c) an Ajax powered web interface with desktop-like look-and-feel and control. Figure 1 shows a simple scheme of the framework.

Modularity in the design of the framework makes it possible to support several deployment scenarios with minimum effort :

❑ The monitor can be deployed fully at a site that runs a supported batch system,

❑ Existing site sensors could be adapted to fill the database defined by the monitoring framework and reused with the web services components,

❑ Sites may build the web server independently and use only the web interface.

The sensors may be easily replaced with other source of data, e.g. Condor Quill DB [3], glideinWMS Collectors, experiment specific Dashboard etc.

Different possible use-cases prompted a number of different ways of accessing the web-based monitor :

- ❑ The site monitor interface can be accessed directly using a URL like https://gridse.sns.it/jobmon/pisa/jobmon.html in which case information of the first job in the list is displayed. One can also access a site monitor and display information for a selected job-id passed as a parameter along-with the URL as, e.g. https://gridse.sns.it/jobmon/pisa/jobmon.html?gridid=https://gridlb1.desy.de:9000/8pX2aZXpCu_y8mdEmNEf9Q
- ❑ A single job from another web page (e.g. CMS Dashboard [4] that may link user jobs using the global Grid Id) can be accessed as, e.g. https://gridse.sns.it/jobmon/pisa/jobinfo.html?gridid=https://gridlb1.desy.de:9000/8pX2aZXpCu_y8mdEmNEf9Q
- ❑ An API based access to the monitoring information as JSON, XML, ASCII, and image data by other web-based applications and possibly command line tools.

4. Client-Server Communication

The client-server communication is based on a simple Ajax pattern. The Ajax processing model, shown in detail in figure 2, leverages the full potential of JavaScript which is at the heart of all actions. The HTML page is fully loaded only initially. Subsequently, JavaScript intercepts all the events and communication with the server side. As the server sends XML, ASCII, JSON, and image data JavaScript updates relevant parts of the page programmatically. In this model, asynchronous update of the web page fragment is the key to responsiveness.

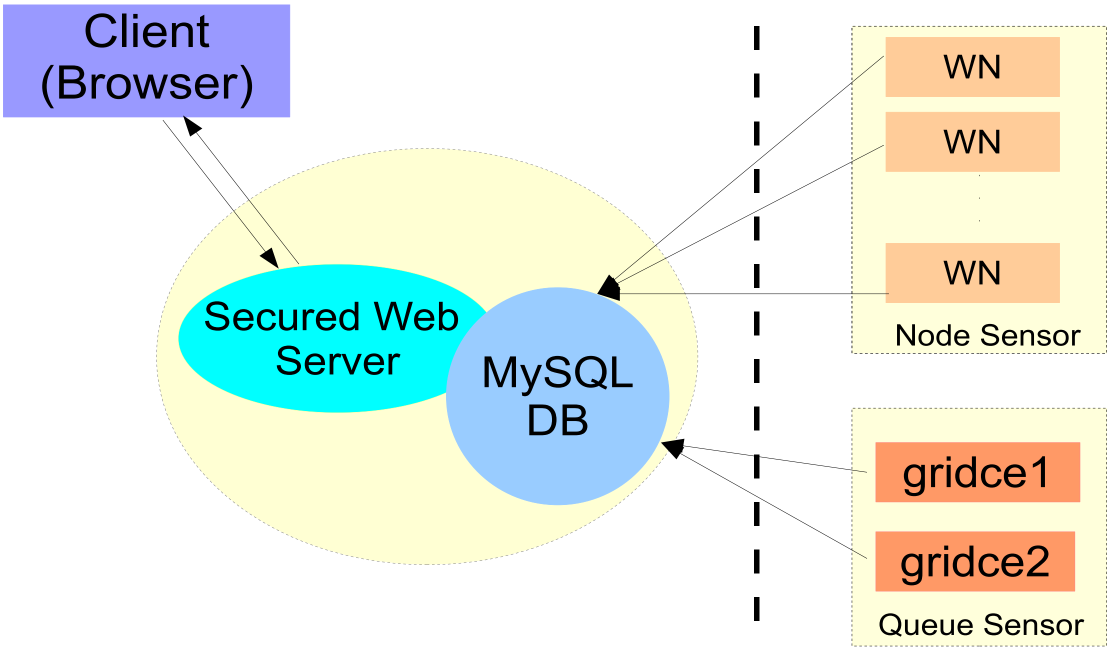


Figure 1. Schematic view of the monitoring framework. The framework consists of three distinct components: (1) a set of batch system specific sensors that must be deployed at all the computing elements (Queue Sensor) and worker nodes (Node Sensor) of a site. These sensors run and collect monitoring data at regular intervals and fill several tables of a MySQL Database which serve the monitoring information to the clients. (2) a simple, extensible CGI based application server, and (3) a modern web interface powered by the Ajax technology with can easily match interactivity and control of desktop applications.

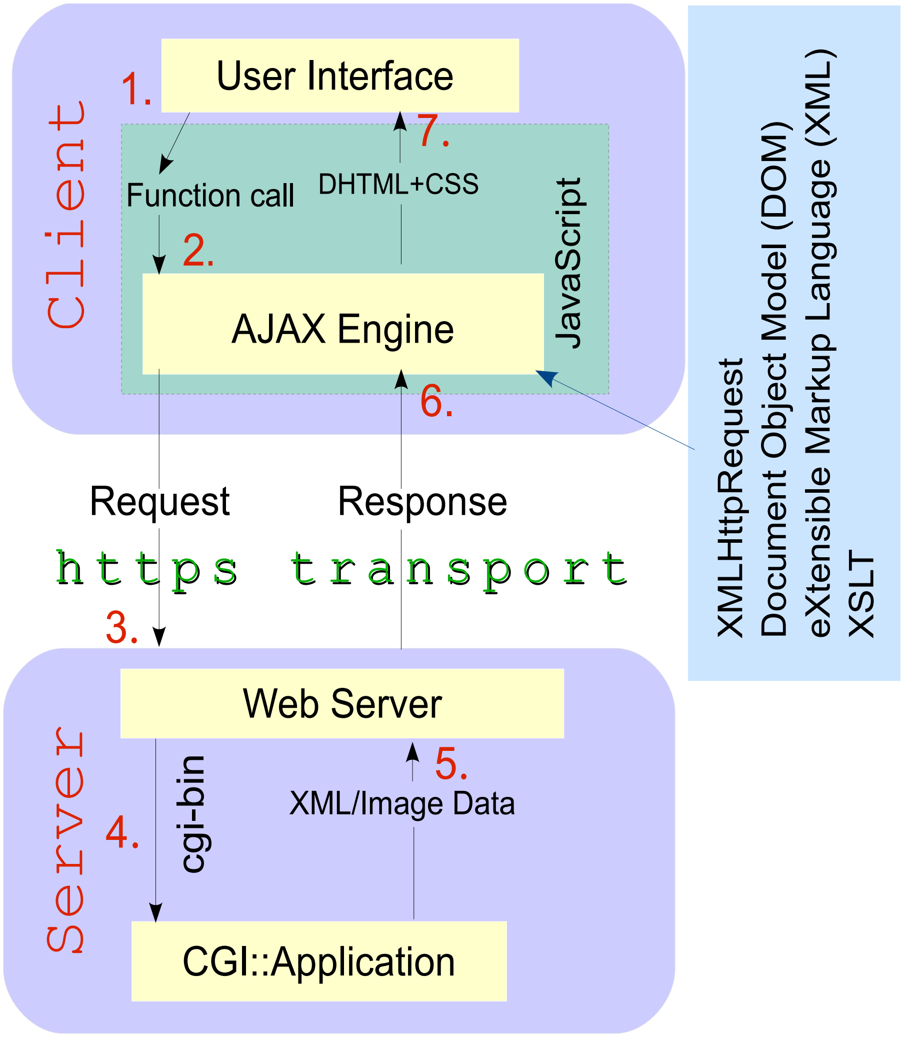


Figure 2. The Ajax Processing Model. In this model all the user actions involve JavaScript calls through an Ajax engine. An XMLHttpRequest object is at the heart of the Ajax Engine which prepares and sends the request to the server and unpacks the server response automatically which is subsequently processed by the JavaScript callback functions. In the present application the jQuery [5] library provides the Ajax capability. The present figure borrows heavily from reference [2].

5. Privacy of Information

Web access requires a valid Grid certificate of the client which is true even for the site administrators. Several distinct, configurable levels of privacy of information have been implemented to cover various use cases:

- ❑ Site administrators can access all the information,
- ❑ A list of VO administrators may access all the detail of each job of that VO,
- ❑ An end-user shall access detailed views of jobs owned by that user (i.e. the Distinguished Name (DN) of the Grid certificate of the user) and summary view of jobs for the VOs the user is a member of,

- ❑ Strict privacy can be enforced such that a client can access only those jobs that belong to the client DN.

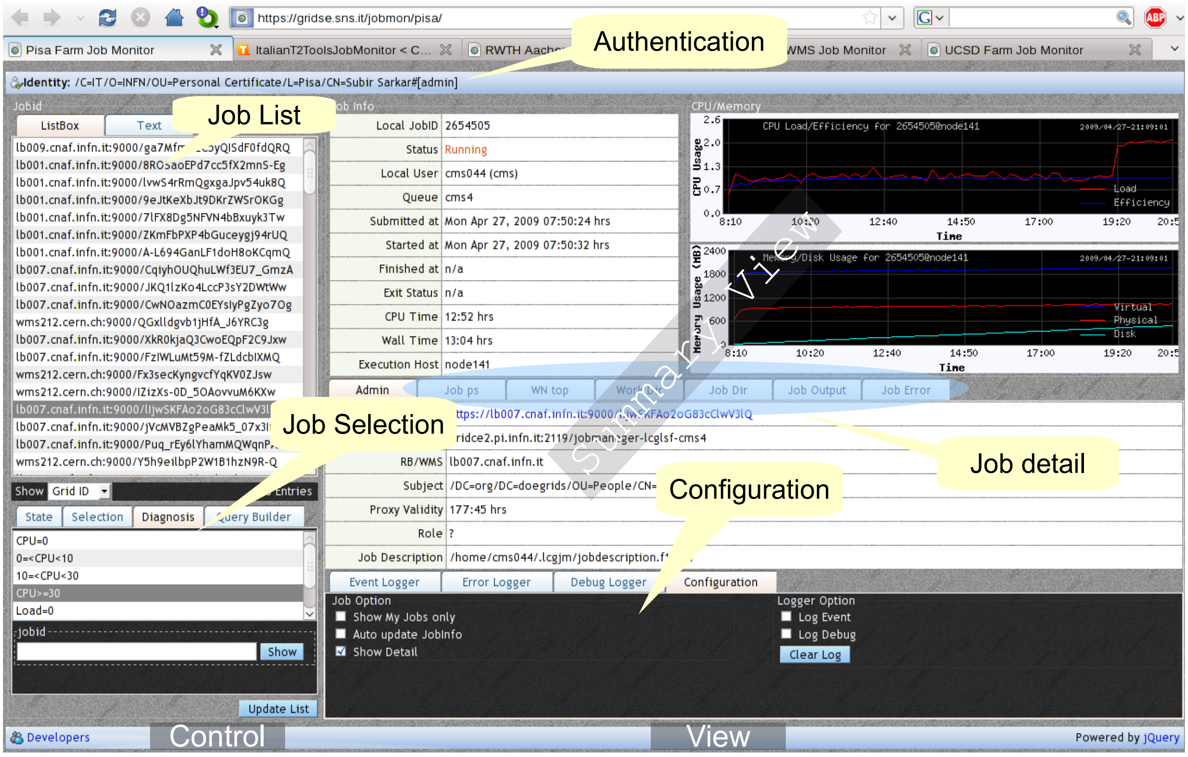
Note that for VO level access, DN level privacy has been enforced for the detailed information like directory listing and job log view.

6. Web Interface

The web interface, as shown in figure 3 is divided into two parts,

- ❑ Job Selection Panel on the left which is itself divided vertically into two parts: (a) the top containing the job-id list with options to show global/local job-id, and (b) the bottom panel consisting of four distinct tabbed panels as described below:
  - – **State**, in addition to running ones, one can also look at jobs that are queued as well as those that were finished within a time window.
  - – **Selection**, one can select jobs belonging to a
    - queue (e.g. cms, cdf),
    - local user (e.g. cmsprd, babarsgm),
    - subject or certificate DN,
    - Resource Broker / WMS,
    - worker node,
    - date when certain jobs got submitted / queued,
    - date when certain jobs started.
  - – **Diagnosis**, look for mis-behaving jobs,
    - jobs falling in different CPU Efficiency(ϵ = ratio of CPU time to Wall time used by a job) ranges (in %):
      - · $\epsilon = 0$,
      - · $0 < \epsilon < 10$,
      - · $10 \leq \epsilon < 30$,
      - · $\epsilon \geq 30$ i.e well behaved jobs.
    - Load = 0: jobs that do not add any CPU load to the node,
    - no proxy lifetime left: user proxy expired, stage-out would eventually fail,
  - – **Query Builder**, complements the Selection tab in order to combine a number of related parameters in a DB query (work in progress).
- ❑ Individual Job Information Panel on the right which is further divided into the following components:
  - – a summary table,
  - – graphical history of CPU load, memory, and disk usage,
  - – a table with Grid related information,
  - – several tabbed panels that show detailed information of a running job, namely:
    - associated processes,
    - job and work directories,
    - last few hundred lines of output and error log files,
    - worker node processes.

Figure 4 shows two job detail tabbed panels: (a) listing of the job directory and (b) the last few lines of the job error log files. A configuration panel allows to control the client-server communication and flow of information. There are also several logger panels for error reporting etc.



https://gridse.sns.it/jobmon/pisa/

Identity: /C=IT/O=INFN/OU=Personal Certificate/L=Pisa/CN=Subir Sarkar#[admin]

```
lb009.cnaf.infn.it:9000/ga7Mfr...cJyQISdF0fdQRQ
lb001.cnaf.infn.it:9000/8ROsaoEPd7cc5fX2mnS-Eg
lb001.cnaf.infn.it:9000/lvwS4rRmQgxgaJpv54uk8Q
lb001.cnaf.infn.it:9000/9eJtKeXbJt9DKrZWSrOKGg
lb001.cnaf.infn.it:9000/7lFX8Dg5NFVN4bBxuyk3Tw
lb001.cnaf.infn.it:9000/ZKmFbPXP4bGuceygj94rUQ
lb001.cnaf.infn.it:9000/A-L694GanLF1doH8oKCqmQ
lb007.cnaf.infn.it:9000/CqiyhOUQhuLWf3EU7_GmzA
lb007.cnaf.infn.it:9000/JKQ1lzKo4LccP3sY2DWtWw
lb007.cnaf.infn.it:9000/CwNOazmC0EYslyPgZyo7Og
wms212.cern.ch:9000/QGxlldgvb1jHfA_J6YRC3g
lb007.cnaf.infn.it:9000/XkR0kjaQ3CwoEQpF2C9Jxw
lb007.cnaf.infn.it:9000/FzIWLuMt59M-fZLdcbIXMQ
wms212.cern.ch:9000/Fx3secKyngvcfYqKV0ZJsw
wms212.cern.ch:9000/IZizXs-0D_5OAovvuM6KXw
lb007.cnaf.infn.it:9000/lljwSKFAo2oG83cClwV3l
lb007.cnaf.infn.it:9000/jVcMVBZgPeaMk5_07x3I
lb007.cnaf.infn.it:9000/Puq_rEy6lYhamMQWqnP
wms212.cern.ch:9000/Y5h9eilbpP2W1B1hzN9R-Q
```

```
CPU=0
0=<CPU<10
10=<CPU<30
CPU>=30
Load=0
```

jobid [ ] Show

Local JobID	2654505
Status	Running
Local User	cms044 (cms)
Queue	cms4
Submitted at	Mon Apr 27, 2009 07:50:24 hrs
Started at	Mon Apr 27, 2009 07:50:32 hrs
Finished at	n/a
Exit Status	n/a
CPU Time	12:52 hrs
Wall Time	13:04 hrs
Execution Host	node141

	ttps://lb007.cnaf.infn.it:9000/lljwSKFAo2oG83cClwV3lQ
	ridce2.pi.infn.it:2119/jobmanager-lcglsf-cms4
RB/WMS	lb007.cnaf.infn.it
Subject	/DC=org/DC=doegrids/OU=People/CN=
Proxy Validity	177:45 hrs
Role	?
Job Description	/home/cms044/.lcgjm/jobdescription.f

Job Option
□ Show My Jobs only
□ Auto update JobInfo
☑ Show Detail

Logger Option
□ Log Event
□ Log Debug
Clear Log

Figure 3. The entry point to the web-based monitoring. The main features are briefly annotated and described in detail in the text.

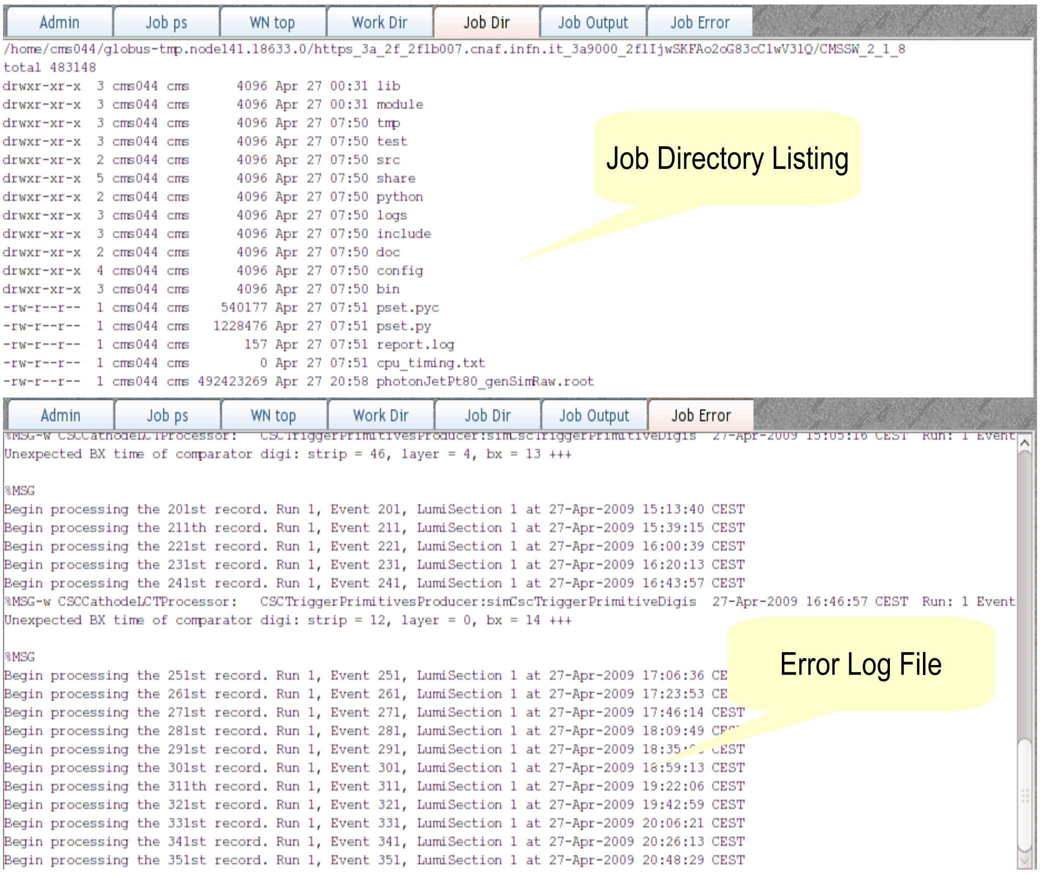
7. Site Monitors

A number of sites running different local batch systems have deployed the monitoring system, as shown in table 1 :

Condor	
T1_US_FNAL	https://cmsjobmon.fnal.gov:8443/jobmon/fnal_t1_newi
LPCCAF, FNAL	https://cmsjobmon.fnal.gov:8443/jobmon/fnal_newi
T2_US_UCSD	https://glidein-mon.t2.ucsd.edu/jobmon/ucsd
LSF	
T2_IT_Pisa	https://gridse.sns.it/jobmon/pisa
PBS	
T2_DE_RWTH	https://grid-mon.physik.rwth-aachen.de/jobmon/rwth-aachen/jobmon.html
glideinWMS	
UCSD Instance	https://glidein-mon.t2.ucsd.edu/jobmon/ucsd_g

Table 1. The monitor has been adapted and deployed at various sites, both Tier-1 and Tier-2, running different local batch systems. The monitoring instances are not yet synchronised to the latest development of the web interface.

The CMS Analysis Support Task Force (ASTF) has recommended the Analysis Operations team to deploy the monitoring tool at a few reference sites in order to facilitate debugging on



```
/home/cms044/globus-tmp.node141.18633.0/https_3a_2f_2flb007.cnaf.infn.it_3a9000_2flIjwSKFAo2oG83cClwV31Q/CMSSW_2_1_8
total 483148
drwxr-xr-x  3 cms044 cms      4096 Apr 27 00:31 lib
drwxr-xr-x  3 cms044 cms      4096 Apr 27 00:31 module
drwxr-xr-x  3 cms044 cms      4096 Apr 27 07:50 tmp
drwxr-xr-x  3 cms044 cms      4096 Apr 27 07:50 test
drwxr-xr-x  2 cms044 cms      4096 Apr 27 07:50 src
drwxr-xr-x  5 cms044 cms      4096 Apr 27 07:50 share
drwxr-xr-x  2 cms044 cms      4096 Apr 27 07:50 python
drwxr-xr-x  3 cms044 cms      4096 Apr 27 07:50 logs
drwxr-xr-x  3 cms044 cms      4096 Apr 27 07:50 include
drwxr-xr-x  2 cms044 cms      4096 Apr 27 07:50 doc
drwxr-xr-x  4 cms044 cms      4096 Apr 27 07:50 config
drwxr-xr-x  3 cms044 cms      4096 Apr 27 07:50 bin
-rw-r--r--  1 cms044 cms    540177 Apr 27 07:51 pset.pyc
-rw-r--r--  1 cms044 cms   1228476 Apr 27 07:51 pset.py
-rw-r--r--  1 cms044 cms       157 Apr 27 07:51 report.log
-rw-r--r--  1 cms044 cms         0 Apr 27 07:51 cpu_timing.txt
-rw-r--r--  1 cms044 cms 492423269 Apr 27 20:58 photonJetPt80_genSimRaw.root
```

```
%MSG-w CSCCathodeLCTProcessor:   CSCTriggerPrimitivesProducer:simCscTriggerPrimitiveDigis  27-Apr-2009 15:05:16 CEST  Run: 1 Event
Unexpected BX time of comparator digi: strip = 46, layer = 4, bx = 13 +++

%MSG
Begin processing the 201st record. Run 1, Event 201, LumiSection 1 at 27-Apr-2009 15:13:40 CEST
Begin processing the 211th record. Run 1, Event 211, LumiSection 1 at 27-Apr-2009 15:39:15 CEST
Begin processing the 221st record. Run 1, Event 221, LumiSection 1 at 27-Apr-2009 16:00:39 CEST
Begin processing the 231st record. Run 1, Event 231, LumiSection 1 at 27-Apr-2009 16:20:13 CEST
Begin processing the 241st record. Run 1, Event 241, LumiSection 1 at 27-Apr-2009 16:43:57 CEST
%MSG-w CSCCathodeLCTProcessor:   CSCTriggerPrimitivesProducer:simCscTriggerPrimitiveDigis  27-Apr-2009 16:46:57 CEST  Run: 1 Event
Unexpected BX time of comparator digi: strip = 12, layer = 0, bx = 14 +++

%MSG
Begin processing the 251st record. Run 1, Event 251, LumiSection 1 at 27-Apr-2009 17:06:36 CE
Begin processing the 261st record. Run 1, Event 261, LumiSection 1 at 27-Apr-2009 17:23:53 CE
Begin processing the 271st record. Run 1, Event 271, LumiSection 1 at 27-Apr-2009 17:46:14 CEST
Begin processing the 281st record. Run 1, Event 281, LumiSection 1 at 27-Apr-2009 18:09:49 CE
Begin processing the 291st record. Run 1, Event 291, LumiSection 1 at 27-Apr-2009 18:35:   CEST
Begin processing the 301st record. Run 1, Event 301, LumiSection 1 at 27-Apr-2009 18:59:13 CEST
Begin processing the 311th record. Run 1, Event 311, LumiSection 1 at 27-Apr-2009 19:22:06 CEST
Begin processing the 321st record. Run 1, Event 321, LumiSection 1 at 27-Apr-2009 19:42:59 CEST
Begin processing the 331st record. Run 1, Event 331, LumiSection 1 at 27-Apr-2009 20:06:21 CEST
Begin processing the 341st record. Run 1, Event 341, LumiSection 1 at 27-Apr-2009 20:26:13 CEST
Begin processing the 351st record. Run 1, Event 351, LumiSection 1 at 27-Apr-2009 20:48:29 CEST
```

Figure 4. Job detail panels. Here the listing of the job directory and the last few lines of the job error log files are shown.

part of the end-users [6]. We expect more sites to deploy the monitor in the near future. We also hope that the CMS Dashboard will eventually link jobs from its own monitoring page to the site monitoring.

8. Deployment

The monitoring system is in active development. New features get added and existing ones refined at a regular basis. The Job monitoring software is distributed as a tarball that must be installed and configured individually for each service. We briefly describe below the deployment procedure that we follow presently for LSF and PBS like batch systems :

```
> wget http://sarkar.web.cern.ch/sarkar/dist/jobmon_v1.1.tgz
> tar xzvf jobmon_v1.1.tgz -C /opt
> cd /opt/jobmon/install
```

A configuration file `/opt/jobmon/install/jobmnon.cfg` included in the distribution must be suitably edited and then a service configured as :

```
> ./configure_jobmon service --basedir /opt
```

where `service` is one of

- `db` : prepare MySQL DB tables and grant access suitably on the DB server node

- `webservice`: install the CGI based application server on the web server
- `ce`: prepare queue sensor on computing elements
- `wn`: prepare node sensor on worker nodes

For Condor based farms and the glideinWMS instances we may be able to combine the two kinds of sensors as a single service. The current status of the tool and all the relevant technical information are available in reference [7].

9. Browser Support

The client side uses the jQuery JavaScript framework extensively, for basic effects and event handling, User Interface (UI) as well as Ajax calls. jQuery has extended browser support significantly beyond Firefox and Seamonkey to include IE 7+, Safari 3+ and Opera 9.6+. Support for IE 6 is limited and there is no further plans while the Google Chrome has not been tried out yet.

10. Conclusion

The Grid job monitoring system has evolved as a useful tool for the site administrators and end-users alike. It is pretty robust due to its inherently distributed design. The major batch systems are supported while plug-ins can be easily developed for new batch system integration. The monitoring framework uses a basic secured web server; security issues have not yet been addressed in detail which we plan to study in future. The system scales easily even for the largest of Tier-2s. A number of scalability issues was successfully addressed by the implementation at the FNAL Tier-1. However, with the recent deployment at a large glideinWMS instance, which holds tens of thousands of jobs at a time, the system is up against a new challenge with scalability.

Acknowledgement

The authors are thankful to Giuseppe Bagliesi, Jon Bakken, Stefano Belforte, Giacinto Donvito, Igor Sfiligoi and Frank Würthwein for many helpful suggestions and support.

References

[1] S. Padhi et. al., Use of glide-ins in CMS for production and analysis, CHEP 09, Prague, Czech Republic

[2] Jesse James Garrett, Ajax: A New Approach to Web Applications,
http://adaptivepath.com/ideas/essays/archives/000385.php, 2005

[3] Huang, J. et. al., An overview of Quill: A passive operational data logging system for Condor
https://www.cs.wisc.edu/condordb/overview_07-18-2007.pdf

[4] CMS Dashboard main page, http://dashboard.cern.ch/cms

[5] jQuery Home Page, http://jquery.com

[6] J. Letts et. al., CMS Analysis Operations, CHEP 09, Prague, Czech Republic

[7] https://twiki.cern.ch/twiki/bin/view/CMS/ItalianT2ToolsJobMonitor
http://cmswiki.fnal.gov/twiki/bin/view/USCMS/Jobmon