WMSMonitor: a Monitoring Tool for Workload and Job Lifecycle in Grids

Daniele Cesini
INFN-CNAF Bologna, Italy
daniele.cesini@cnaf.infn.it

Danilo Dongiovanni
INFN-CNAF Bologna, Italy
danilo.dongiovanni@cnaf.infn.it

Enrico Fattibene
INFN-CNAF Bologna, Italy
enrico.fattibene@cnaf.infn.it

Tiziana Ferrari
INFN-CNAF Bologna, Italy
tiziana.ferrari@cnaf.infn.it

Abstract

Scheduling services are core Grid components of paramount importance to support the transparent distribution of tasks to remote shared resources in an efficient way. High availability of these core services is thus of great importance. Given the distributed nature of the system, monitoring the task lifecycle and the aggregate workflow patterns generated by users belonging to various communities is particularly challenging.

This paper deals with the problem of Grid workload monitoring by reviewing the related requirements, and illustrates the architecture and implementation of a tool, the WMSMonitor, which is designed to meet the needs of various users categories, such as administrators, developers, advanced Grid users and performance testers.

1. Introduction

Job scheduling is a fundamental issue on Grid environment. It must be efficient in order to minimize the waiting time before job execution and reliable providing job failure recovery mechanisms to increase the success/abort ratio. Moreover the resource discovery, brokering and failure recovery tasks should be as transparent as possible to the user. Difficulties in accomplishing these requirements increase proportionally to Grid size and to job submission rates. In the large scale worldwide Grid provided by the EGEE project [1], the scheduling task is performed by the *Workload Management System* (WMS) service together with its job status advancement tracking facility *Logging and Bookkeeping* (LB). They constitutes two of the key services composing the gLite middleware stack deployed on the EGEE infrastructure.

The WMS central role as a user gateway to Grid resources makes the high availability of the service a challenging issue, implying a systematic service status check and a prompt access to information required for problem debugging. Moreover, given the heterogeneous user communities it is important to extract from the WMS detailed information on the job flow and submission modalities. This also offers the chance to gain insight on Grid usage.

Even though several tools are available to monitor overall Grid performance and status [2], none of them provides detailed information on the WMS/LB status and job flow. In this paper we present WMSMonitor, a monitoring tool for the gLite WMS/LB. Through a usable Web interface, it provides a single access point to both WMS/LB service status variables and to the WMS job submission, processing and dispatching flow. Moreover, keeping an historical archive it offers data aggregation facilities and statistical reports.

Analyzing few use cases we show how WMSmonitor results to be an effective support tool for administrators and developers to visualize and interpret problematic service conditions and to spot performance bottlenecks. We also show how the continuous monitoring over months of intense submission activity turned to be helpful to user community for its capability to measure performance and collect aggregated statistics.

This manuscript is structured as follows: Section 2 is a description of the gLite WMS/LB; Section 3 describes the needs that motivated the tool development; Section 4 presents the WMSMonitor tool architecture, implemented sensors and Web presentation solution in details; Section 5 analyses use cases observed during EGEE infrastructure stress tests conducted by high energy physics communities.

2. The gLite WMS and LB services

WMS is a Grid service responsible for the distribution of user tasks to the remote computing resources. WMS supports this process by tracking events related to the tasks during their lifecycle, by adopting failure recovery procedures in case of problems and by performing resource discovery. This last step relies on the availability of information from a Grid Information Service publishing the set of available remote resources from which the list of candidates is selected in order to match the requirements specified by the users.

The LB service is tightly coupled to the WMS and tracks jobs managed by the WMS. It gathers events from WMS components providing the user with detailed information about the status of jobs during their whole lifecycle.

In the following, communities of users, in general belonging to the same scientific experiment or project, will be addressed as Virtual Organizations (VOs). The Job Description Language (JDL) [3] is the syntax used to define the profile of the workload submitted. It is based on Classified Advertisements also known as ClassAdds [4] that through key/value pairs allow the user to specify various job characteristics (e.g. input files, executables, arguments, output files, etc.) and to give directives for job distribution, such as requirements on the location of computing resources, hardware and software environment needed.

WMS can handle various task types such as: single batch jobs, MPI jobs, Direct Acyclic Graph jobs – for the submission of job sets with complex internal dependencies, job collections – for the handling of group of independent jobs, and finally parametric jobs, i.e. group of tasks depending on a user-defined parameter.

2.1 WMS/LB Architecture Overview

WMSMonitor supports the tracking of jobs over various phases of their processing cycle within the WMS. The job lifecycle is tied to the WMS and LB architectures [5] as shown in Figure1.

Workload Management Proxy (WMProxy) is the component responsible for accepting job submission requests from the user. It is implemented as a Web service accessible via either APIs or a command line interface. Firstly, WMProxy authenticates and authorizes users (credentials in the form of X509 proxy certificates are analysed).

In addition, a WMProxy subcomponent, the *limiter-script*, implements a mechanism to avoid machine overload. Some server status indicators such as disk usage, memory usage and load average, are periodically checked. If values exceed some configurable thresholds, additional job submissions are rejected. Moreover, WMProxy is in charge of handling the so-called user *SandBox* which is defined to be the set of files provided as input to the job for processing, or as output of the computation phase. The WMS provides retrieving tools to access such files. SandBoxes are transferred though a Grid File Transfer Protocol (GFTP) [6] server running on the WMS machine.

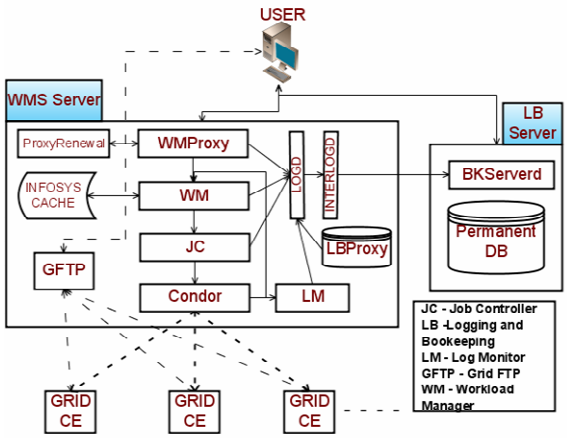

Figure 1. gLite WMS/LB architecture

WMProxy dispatches accepted submissions to the *Workload Management* (WM) component which, through a resource discovery process known as *Match Making* (MM), compares the job requirements against the available Grid resources, with the purpose of selecting the most suitable one.

The *Compute Element* (CE) is the Grid service that virtualizes computing resources to be used for job execution. In order to increase the internal scalability, WMS supports the submission of sets of jobs (job collections) through the invocation of a single atomic operation. This feature is complemented by the capability to perform a single MM computation for sets of jobs with similar requirements, the so-called *Bulk Match Making* mechanism [5]. If no candidates CEs are found, the job is kept pending on a task queue and periodically reprocessed by the WM until a configurable timeout expires. Resource discovery relies on detailed information about resources and services available in the infrastructure. Therefore, the WM periodically downloads data from various information providers and caches them internally. The number of CEs available for every group of Grid users (VO or subgroups) considered in the MM process is stored in the mentioned WM cache and indicated with the term *VOViews*. The WM also evaluates how to deal with failed jobs and decides, according to the submission directives of the user, if they must be resubmitted or aborted.

Then, the *Job Controller* (JC) prepares the job for the Condor [7][8] submission (i.e. creates the Condor submit file) and passes the job to Condor scheduler which is responsible for the actual submission to the remote resource. Finally, *Log Monitor* (LM) parses the Condor log files, intercepts and stores interesting job events such as changes in status ("running", "done", "abort", etc.). In addition, it triggers resubmission of failed jobs by queuing them to the WM.

The LB is a stand alone service which interacts with the WMS and stores any advancement step in the lifecycle of all jobs handled by the WMS. Users can retrieve detailed information about their own jobs by querying the LB server.

A local cache of the LB server (*LBProxy*) is maintained on the WMS for fast job information retrieval and to preserve the WMS from LB response time issues. The cache is refreshed as soon as jobs reach a final status.

WMS components are not meant to upload job events directly to the LB server, as this task is performed in two separate steps by the *Logd* and the *Interlogd* services. The first one receives information from the WMS components and stores them on physical files (the so-called *dg20log* files). They should be seen as a queue of events (one per file) that are waiting to be uploaded to the LB server. The Interlogd reads such files, extracts the relevant information, and sends it to the LB server. The dg20log files are removed once successfully uploaded.

As already stated, the decision to accept or refuse jobs is based on the user credentials (a X509 proxy certificate) sent with the job. This implies that the job lifetime can not exceed the validity time of the corresponding user proxy certificate. Since Grid proxy certificates usually last few hours there is a mechanism (it can be enabled by the user) to renew proxies that are about to expire. The service responsible for this task is called *ProxyRenewal*.

3. Requirements

Monitoring of Grid services, which are composed of several independent processes as shown in the previous sections, can be challenging due to the large number of interactions which take place. During its lifecycle, every WMS job is exposed to a number of possible sources of failure, within the WMS (in case of troubles with its inner components) and/or in the remote resources.

Given the intrinsic complexity and error-prone nature of distributed system, and to the large number of service instances deployed, in order to increase service availability, it is important to have tools that promptly detect failures and collect/analyze information about status of a pool service instances. In fact, in the lack of such monitoring systems, the failure detection process relies on manual checks on every instance.

For Grid services based on multiple internal components such as the WMS, manual debugging is made even more complex by the spread of information across several sources such as the LB database, the daemons log files and the tools for monitoring of the hardware which hosts the services (for example, *Lemon* [9]). This emphasizes the importance of providing a single access point to important debugging information and to status key indicators for multiple Grid service instances.

In addition to the above-mentioned needs, a further set of requirements applies to Grid services responsible for job scheduling. In order to control the full functionality and efficiency of the scheduling process it is important to gather information such as:

1. the number of jobs being submitted;
2. the job types;
3. the number of jobs flowing between the various service internal components (in order to spot bottlenecks);
4. the number of jobs successfully completed.

Grid infrastructures supporting several VOs typically deploy a large number of core service instances in order to ensure failover and load balancing. For this reason, aggregation of data gathered for service clusters is important in various respects. For example, data may be aggregated per VO, but monitoring tools should give administrators the freedom to specify arbitrary grouping rules (for example, according to the various roles: production servers, test servers, etc.).

The following Section details how these requirements have been taken into account designing the WMSMonitor application.

4. WMSMonitor presentation

The WMSMonitor has been developed to meet the needs of various user categories:

- Grid service developers and performance testers, who are interested in monitoring their services to improve their quality;
- advanced Grid users submitting a huge quantity of jobs, i.e. to test the service scalability for their VO and, possibly, the quality of the VO tools developed for automatic submission;
- resource center managers that need per-VO aggregated statistics on service load and the service availability offered;
- VO managers to obtain aggregated job statistics, i.e. to cross check their monitoring tools.

In this Section we provide a description of the considered metrics and of the software architecture; for this, each component of the WMSMonitor application is described in detail. Finally we overview some operational issues related to common usage of the tool.

4.1 Metrics

The set of metrics adopted by WMSMonitor can be broadly divided into three main categories: Grid service, system and job flow metrics (see Section 2.1).

Grid service metrics:

- daemon status for WMProxy, Proxy Reneval, WM, LM , LL, JC, LBProxy, FTPD
- number of file descriptors opened by WM, LM, JC, LL
- number of entries in the WM and JC queues and dg20log files
- number of GFTP open sessions
- number of VOViews
- Condor Job state statistics
- number of connections opened on ports used by LB service
- daemon status for BKServerd and LL

System metrics:

- CPU load average
- percentage of occupancy of disk partitions

Job flow metrics:

- number of jobs submitted to WMProxy from any user, also reporting the number of collections of jobs and the mean number of nodes per collection with relative standard deviation
- number of jobs queued to WM from WMProxy
- number of jobs queued to WM from LM, i.e. resubmitted after failure
- number of jobs queued to JC from WM
- number of jobs queued to Condor from JC
- number of jobs successfully completed
- number of jobs aborted

It should be noted that job flow metrics are derived from LB queries and refer to time intervals (e.g. number of jobs crossing the considered component within the past 15 minutes). On the other hand Grid service and system metrics report values referred to the exact time of measurement.

4.2 Architecture

WMSMonitor is composed by a server on a dedicated machine and agents hosted on the monitored Grid services (Figure 2).

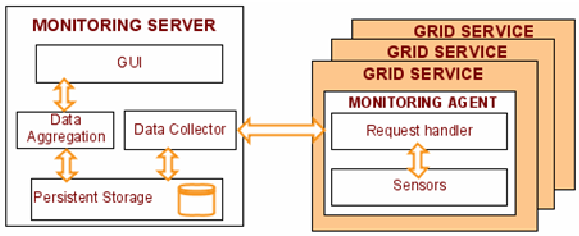

Figure 2. Overview of WMSMonitor architecture

The server architecture is organized in three layers: a presentation layer, an application layer implementing data collection and aggregation features and a persistent storage layer. The monitoring agent consists of a Web service exposing an interface for a set of sensors performing the measurements.

The data collector application periodically triggers the execution of sensors on the monitored Grid services through the Web service. The adopted *data pull* approach grants the on-demand data collection functionality which allows to: i) dynamically change the measurement frequency in case of troublesome conditions for specific instances; ii) recollect job flow information in order to remove historical data inconsistencies (see Section 4.4). Data collected from each agent are inserted in the persistent storage to provide an historical archive. Finally, a data aggregator extracts and elaborates information from the persistent storage making it available for a Web-based graphical user interface.

To implement the described functional modules we developed a PHP based application for data aggregation and presentation, over a MySQL DataBase as persistent storage. For the server/agent communication a SOAP Python module was used.

4.2 Data Collector and Monitor Agent

WMS	DATE	RUNNING JOBS	IDLE JOBS	WM QUEUE	JC QUEUE	VO VIEWS	LB EVENTS QUEUE	CPU LOAD	SANDBOX PARTITION	GENERAL STATUS	DAEMONS STATUS
wms006.cnaf.infn.it	2008-04-11 10:45:22	15	651	3	0	5361	69	0.68	81	⚠	✓
wms009.cnaf.infn.it	2008-04-11 10:45:27	3	19	0	0	5364	0	0.4	51	✓	✓
wms011.cnaf.infn.it	2008-04-11 10:45:29	61	671	0	0	5361	7	0.83	43	✓	✓
wms012.cnaf.infn.it	2008-04-11 10:45:48	56	969	7	0	5361	43	0.85	48	✓	✓
wms014.cnaf.infn.it	2008-04-11 10:45:53	1479	1940	3	0	5361	36	6.61	50	✓	✓
wms015.cnaf.infn.it	2008-04-11 10:46:11	125	2181	0	1	5361	6	2.56	2	✓	✓
wms017.cnaf.infn.it	2008-04-11 10:46:18	74	1442	0	0	5361	4	1.59	2	✓	✓
egee-rb-04.cnaf.infn.it	2008-04-11 10:46:21	0	19	0	0	5364	0	1.35	26	✓	✓

Figure 3. Web interface: main page

This services are responsible for the scheduled and un-scheduled collection of data from each monitored WMS/LB instance.

Data collector module runs on the WMSMonitor server at regular intervals or occasionally on demand. It implements two main operations (Figure 2): i) interacting with the agent request handlers running on monitored WMS/LB instances, it triggers the sensors execution; ii) it runs consistency checks on collected data and stores them in the Persistent Storage module. In the current implementation, every 15 minutes a cron job executes the data collector taking in input the list of WMSLB instances to monitor.

On the monitored instances side, when requests for the collection of a new series of data arrive the handler performs authentication operations, and calls a set of Python functions implementing the sensors for monitored variables. It also initializes the sensor supplying them with parameters coming from the data collector.

4.3 Web presentation service

The presentation service is a Web-based graphic user interface designed to offer to WMS administrators, developers and VO advanced users monitored information in a usable aggregated form. The graphical user interface is divided in a main section reporting an overview of principal status variables for all monitored WMS instances (Figure 3) and a details section for each specific instance (Figure 4). A section with statistics about WMS service usage by VOs is also provided (Figure 6). To access the Web pages a valid personal digital certificate and Adobe Flash Player installed on the browser are required.

4.3.1 Main page description. (Figure 3) Monitored WMS instances can be grouped according to predefined configurable sets, such as all the WMS dedicated to a VO or specific role services. In order to simplify the administration of WMS services we introduced two icons that spot at a glance the overall status of the instances services and running daemons respectively. These icons, normally in "OK" status, can switch to "Warning" or "Failure" status. The switch is triggered by a set of rules and thresholds defined to point out possible WMS/LB service failure conditions. From this main page one can access both the specific WMS instance details page and the VOs stats page.

4.3.2 Specific WMS instance details page description. (Figure 4 and Figure 5) This section of the Web interface is divided in peripheral boxes reporting latest series of acquired data in textual form and a central box reporting historic data series in graphical form implemented exploiting Open Flash Chart [10]. The column box on the left reports information about the WMS components described in Section 2.1 with the exception of Condor statistics. Data have been aggregated in order to provide the WMS administrator with a component focused view, that allows the prompt identification of the possibly failing service. On top of the page two boxes present HW status data for both WMS and associated LB instances, other than Condor statistics. In these boxes a configurable link is provided to reach other external tool Web pages monitoring the same instance, if available.

In the central box two sets of historic data series are reported. A tabbed menu is provided to switch between them. The first set of charts (Figure 5) plots Condor statistics, component queues and the job flow details between WMS components across a user specified time period. The time resolution of these plots is determined by the data acquisition period, 15 minutes by default. Since job flow sensors report the number of jobs passing trough each component between each two contiguous measurements and given the slightly variable duration of such time windows (Section 4.4), we decided to normalize data to time unit presenting the mean job flow rates in Hz (Figure 5, bottom charts). The second set of charts available in the tab menu of the central box reports historic data about component job flow as cumulative job number per day over a user specified time period (Figure 4, left side charts). In the same tab also a chart reporting the number of jobs per day ended in *done* or *aborted* state is presented (Figure 4, upper right chart).

In order to facilitate charts readability a box with numerical values pops up when pointing at the plot curves (Figure 5, bottom left chart).

4.3.3 VO statistics page. (Figure 6) This functionality provides the VO manager or advanced VO users with a summary view of all jobs processed by dedicated WMS/LB instances within a user specified time window. Two tabs are available aggregating data in graphical and textual form respectively.

In the *Charts* tab, on the left a daily summary cumulating data from all selected instances within the specified time period is reported. Bottom left chart reports the jobs submitted together with the jobs queued to Condor summarizing the number of jobs treated by the WMS point of view. The top left chart reports the final status reached by the treated jobs. On the right pie charts, the same information is aggregated per WMS instance over the whole time period.

In the *Table* tab a textual report presents cumulative data across the selected time window for each WMS instance: the number of jobs submitted, the number of collection, the number of jobs done/aborted and the number of jobs treated by Condor. Also a super-total reporting the cumulative data over all WMS instances and over whole time period considered is reported.

4.4 Operational issues

In this section we address some operational issues related to the WMSMonitor tool usage. We start mentioning the problem of dg20log files cumulating on the gLite WMS instances in case of system overload. These files temporarily store on WMS disk jobs events waiting to be archived into the LB database. Running a query to obtain job flow statistics on the LB instance with some dg20log files still on the relative WMS instance would potentially lead inconsistencies in the data. Therefore an auto-update functionality has been implemented to recollect correspondent data once the dg20log queue has been absorbed. Again, to assure consistency of the data archive on a daily basis we paid attention to dynamically and automatically partition the 24h in contiguous but not overlapping time interval for the job flow data extraction queries. In fact even though the data collection is triggered by a cron job on regular time intervals, we had to take into account different instances sensor response time, producing small unpredictable delays.

It is worth mentioning that several tests under controlled conditions were conduced to validate the WMSMonitor tool and assess reliability of collected data. Data consistency was cross-checked with data obtained from independent sources: no discrepancies were observed.

5. Use cases

In this section we first discuss the interpretation of monitored variables in real WMS/LB administration scenarios and then present use cases derived from monitoring a cluster of WMS/LB instances during EGEE infrastructure stress tests conducted by high energy physics communities.

Figure 4. Web interface: detailed information about specific WMS

With reference to Section 4.1 we discuss the interpretation and integration of information relative to WMS/LB instances from the site administrator point of view. The first category of measured variables are mainly meant to let the WMS/LB administrator promptly know for several instances of WMS/LB in parallel about the occurrence of potentially critical service failures. The quick detection of daemons error (error code is reported) or HW error conditions on WMS/LB instances, highlighted by graphical solutions in the Web interface, allows the administrator to promptly recover from service unavailability situations. As an example of such HW critical conditions consider that self protection mechanism implemented on WMS/LB prevents the server from accepting new submissions in case of high CPU loads (>20 by default), or high disk occupancy (>90%) (see Section 2.1). These critical conditions have been frequently observed during intense submission activity.

Other monitored variables are meant as a support for service unavailability prevention, informing the administrator about overloaded services. As an example we mention services using too many file descriptors or with anomalous queues (WM, JC, LB), an high number of connections opened by LB or high number of GFTP sessions, or no CEs resources available for the WMS MM operation (VOViews sensor) causing the WM queue to increase indefinitely till jobs' abortion, or disk space occupancy reaching the tolerance limit.

The category of variables (Section 4.1) focusing on job flow between WMS components gives useful information about internal WMS dynamic, and provides administrators, developers and performance testers (both from developers and VOs communities) with a powerful support tool for diagnosis of points of failures or bottlenecks of the service.

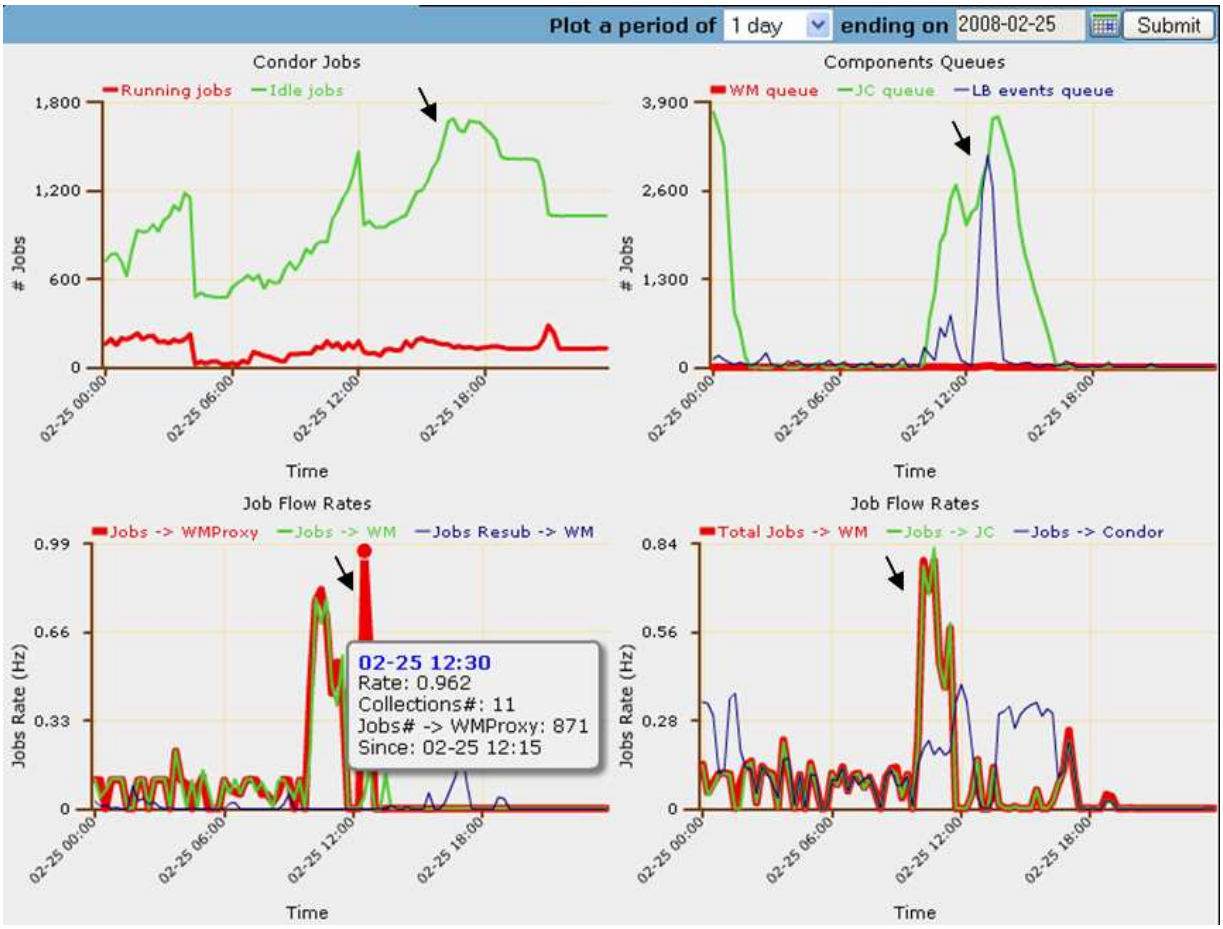

Figure 5. Internal WMS dynamics and job flow

Figure 5 illustrates an example of the mentioned WMS service internal dynamic exploration while monitoring a VO dedicated instance during a performance test. Starting from (Figure 5, bottom left) we see peaks of few thousands jobs submitted in about three hours to WMProxy which, grouped in collections (see Section 2), are quickly processed and queued to WM. Grouping jobs in collections with bulk MM enabled (see Section 2.1) reduces the WM time-through, so that jobs are quickly queued to JC which serially process and queues them to Condor at a slower rate (Figure 5, bottom right). In (Figure 5, top right) we observe how that results in a peak in the JC queue. Also the LB job events transfer is under stress in such conditions, resulting in a dg20log files queue (Figure 5, top right). Finally in (Figure 5, top left) we notice how in Condor, after submission peaks (Figure 5, bottom left), the number of idle jobs increases while the number of running jobs stays stable. This has been interpreted as an effect of full queues on CE requested for the test jobs, and confirmed by other information sources. The described scenario puts in evidence how the main contribution to gLite WMS time-through comes from JC component when users submit BulkMM enabled collections. This can be easily visualized when considering how the peaks of submissions to the WMProxy (Figure 5, bottom left) result in a large queue just for the JC component but not for the WM (Figure 5, top right). This example constitutes an important feedback for developers considering that this submission type (collection with BulkMM enabled) is becoming popular among VOs. Regarding the queue in job events transfers to the LB, it should be noted that these do not affect the WMS job processing time, but just causes a delay in users' perception of job advancement status. This delay affects the data collected by WMSMonitor in real time, but it is handled by the auto-updater functionality described in Section 4.4 once the queue is reabsorbed.

Figure 6. Aggregated statistics relative to a pool of WMS dedicated to a specific VO

As last use case we present the VO stats functionality implemented in the Web presentation service (Section 4.3). Figure 6 reports cumulative statistics relative to a continuative test performed over a month by an high energy physics VO on a set of dedicated WMS instances. Exploiting this functionality we learn that, during the whole test, roughly one million jobs were treated by the WMS pool considered. The distribution of jobs across the WMS instances is not homogeneous as can be seen in (Figure 6, bottom left pie chart), where one instance is clearly underused. This is useful feedback to the VO manager in order to better tune the VO submission tools. Jobs were

submitted at a mean rate of 30K jobs per day (Figure 6, bottom left chart). From the textual data in the *Table* tab (Section 4.3.3) of the same Web page section we derived that 76% of submitted jobs ended in successful state while 14% aborted and the remaining jobs ended in other final states (e.g. *Cancelled* by the user).

6. Conclusion

In this paper we presented WMSMonitor, a tool to monitor the WMS service, a Grid component which is responsible for the key role of user task scheduling in the gLite middleware stack deployed within the EGEE project infrastructure.

The analysis of the presented use cases showed how, through a usable Web interface, WMSMonitor provides a prompt overview of WMS service status and internal dynamics, supporting administrators in detecting and debugging problems, and developers in spotting service bottlenecks. The paper showed how the usage of its data aggregation facilities, over time and over multiple WMS instances, can give helpful support to several categories of users to monitor the submitted Grid workload during its entire lifecycle.

Encouraged by users feedback, future work will focus on supplying them with more metrics helpful to optimize Grid resources exploitation.

Acknowledgments

We wish to thank the WMS/LB developers team for their valuable help and support. Special thanks to Francesco Giacomini, Zdenek Salvet and Marco Cecchi.

References

[1] Enabling Grids for E-sciencE. Homepage: http://www.eu-egee.org/
[2] M. Gerndt, R. Wismueller, Z. Balaton, et alt.; Performance Tools for the Grid: State of the Art and Future, volume 30 of Research Report Series, Lehrstuhl fuer Rechnertechnik und Rechnerorganisation (LRR-TUM) Technische Universitaet Muenchen. Shaker Verlag, 2004. ISBN 3-8322-2413-0.
[3] The JDL Attributes Specification, EGEE JRA1 Technical Report (http://trinity.datamat.it/projects/EGEE/wiki/wiki.php?n=JDL.AboutJDL)
[4] http://www.cs.wisc.edu/condor/classad/
[5] P. Andreetto et alt.; The gLite Workload Management System; in Proc. Computing in High Energy and Nuclear Physics (CHEP) 2007, Victoria, British Columbia (CA), Sep 2007.
[6] W. Allcock; GridFTP: Protocol Extensions to FTP for the Grid; Open Grid Forum final document, GFD.20, Apr 2003
[7] D. Thain, T. Tannenbaum, and M. Livny, "Distributed Computing in Practice: The Condor Experience" *Concurrency and Computation: Practice and Experience*, Vol. 17, No. 2-4, February-April, 2005, pp 323-356.
[8] T. Tannenbaum, D. Wright, K. Miller, and M. Livny, "Condor - A Distributed Job Scheduler", in Thomas Sterling, editor, *Beowulf Cluster Computing with Linux*, The MIT Press, 2002.ISBN:0-262-69274-0
[9] http://lemon.Web.cern.ch/lemon/index.shtml
[10] The Open Flash Chart project home page (http://teethgrinder.co.uk/open-flash-chart/)