J Grid Computing (2010) 8:323–339
DOI 10.1007/s10723-010-9148-x

Experiment Dashboard for Monitoring Computing Activities of the LHC Virtual Organizations

Julia Andreeva · Max Boehm · Benjamin Gaidioz · Edward Karavakis · Lukasz Kokoszkiewicz · Elisa Lanciotti · Gerhild Maier · William Ollivier · Ricardo Rocha · Pablo Saiz · Irina Sidorova

Received: 24 August 2009 / Accepted: 19 February 2010 / Published online: 28 April 2010

Abstract The Large Hadron Collider (LHC) is preparing for data taking at the end of 2009. The Worldwide LHC Computing Grid (WLCG) provides data storage and computational resources for the high energy physics community. Operating the heterogeneous WLCG infrastructure, which integrates 140 computing centers in 33 countries all over the world, is a complicated task. Reliable monitoring is one of the crucial components of the WLCG for providing the functionality and performance that is required by the LHC experiments. The Experiment Dashboard system provides monitoring of the WLCG infrastructure from the perspective of the LHC experiments and covers the complete range of their computing activities. This work describes the architecture of the Experiment Dashboard system and its main monitoring applications and summarizes current experiences by the LHC experiments, in particular during service challenges performed on the WLCG over the last years.

J. Andreeva (✉) · B. Gaidioz · E. Karavakis · L. Kokoszkiewicz · E. Lanciotti · G. Maier · W. Ollivier · R. Rocha · P. Saiz · I. Sidorova
CERN, European Organization for Nuclear Research, 1211 Geneva 23, Switzerland
e-mail: julia.andreeva@cern.ch

M. Boehm
Hewlett-Packard, 65428 , Roselsheim, Germany

E. Karavakis
School of Engineering and Design, Brunel University, Uxbridge, UB8 3PH, UK

Keywords WLCG · Distributed infrastructure · LHC experiments · Monitoring

1 Introduction

Preparation is under way to restart the LHC [1]. The experiments at the LHC are all run by international collaborations, bringing together scientists from institutes all over the world. Each experiment is distinct, characterized by its unique particle detector. The two large experiments, ATLAS [2] and CMS [3], are based on general-purpose detectors designed to investigate the largest range of physics possible. Two medium-size experiments, ALICE [4] and LHCb [5], have specialized detectors for analyzing the LHC collisions in relation to specific phenomena. User communities of the LHC experiments are organized in the virtual organizations (VOs).

The LHC is estimated to produce about 15 petabytes of data per year. This data has to be distributed to computing centers all over the world with a primary copy being stored on tape at CERN [6]. Seamless access to the LHC data has to be provided to about 5,000 physicists from 500 scientific institutions. The scale and complexity of the task shortly described above requires complex computing solutions. A distributed, tiered computing model was chosen by the LHC experiments for the implementation of the LHC data processing task. The LHC experiments use the WLCG [7] distributed infrastructure for their computing activities. In order to monitor the computing activities of the LHC experiments, several specific monitoring systems were developed. Most of them are coupled with the data-management and the workload-management systems of the LHC VOs, for example PhEDEx [8], Dirac [9], PanDA [10] and AliEn [11]. In addition, a generic monitoring framework was developed for the LHC experiments—the Experiment Dashboard. If the source of the monitoring data is not VO-specific, the Experiment Dashboard monitoring applications can be shared by several VOs. Otherwise, the Experiment Dashboard offers experiment-specific monitoring solutions for the scope of a single experiment.

To demonstrate readiness for the LHC data taking, several computing challenges were run on the WLCG infrastructure over the last years. The latest one, Scale Testing for the Experiment Programme'09 (STEP09) [12], took place in June 2009. The goal of STEP09 was the demonstration of the full LHC workflow from data taking to user analysis. The analysis of the results of the STEP09 and of the earlier WLCG computing challenges proved the key role of the experiment-specific monitoring systems, including Experiment Dashboard, in operating the WLCG infrastructure and in monitoring the computing activities of the LHC experiments.

The Experiment Dashboard allows to estimate the quality of the infrastructure and to detect any problems or inefficiencies. Furthermore, it provides the necessary information to conclude whether the LHC computing tasks are accomplished.

The WLCG infrastructure is heterogeneous and combines several middleware flavors: gLite [13], OSG [14] and ARC [15]. The Experiment Dashboard project works transparently across all these different Grid flavors.

The main computing activities of the LHC VOs are data distribution, job processing, and site commissioning. The Experiment Dashboard provides monitoring covering the various computing activities mentioned above. In particular, the site commissioning aims to improve the quality of every individual site, therefore ameliorating the overall quality of the WLCG infrastructure.

The Experiment Dashboard is intensively used by the LHC community. Taking as an example the CMS Dashboard server, according to the Web statistics tool [16], during year 2009 monthly number of unique visitors increased three times, from 1,300 in the beginning of the year to 3,900 in autumn. In average 50,000 pages are viewed daily. The users of the system can be classified into various roles: managers and coordinators of the experiment computing projects, site administrators, and LHC physicists running their analysis tasks on the Grid.

Section 2 shortly describes the related work. The architecture of the Experiment Dashboard system and implementation details of the Experiment Dashboard framework are described in Section 3. Sections 4–6 overview the main monitoring applications of the Experiment Dashboard system. The applications are discussed in the following order: data-management monitoring applications, applications for the job monitoring, and applications for the site monitoring and commissioning activity.

Section 7 covers the recent developments, which focus on high level cross-VO views of the computing activities of the LHC experiments, both at the level of a single site and at a global level.

Close collaboration between the developers of the Experiment Dashboard and the user communities is described in Section 8.

In the conclusions we summarize the role of the Experiment Dashboard monitoring system and the current experiences of using it by the LHC community.

2 Related Work

One approach for monitoring Grid infrastructures, used by systems such as MonALISA [24] and GridICE [34] is to aggregate and display data from the local fabrics monitoring of the distributed clusters. They provide a system-level view of the Grid resources. Monitoring systems such as GridView [17] and Real Time Monitor [18] obtain information from the Grid services, for example from the Logging and Bookkeeping service [19]. Monitoring data provided by the systems mentioned above is useful for service providers and people operating distributed infrastructure, but does not necessarily cover the needs of the user communities, since it is missing application-level information.

One of the possible ways to evaluate Grid infrastructure from the perspective of the user communities is to execute functional tests, which emulate user activity at the Grid sites, and to collect and display the results of these tests. This approach is applied on the Grid infrastructures including WLCG and TeraGrid [35]. For WLCG this task is performed by the Service Availability Monitoring (SAM) [20], for TeraGrid by Inca [35] monitoring system. Though monitoring of functional tests is rather effective for detecting problems with the distributed sites and services it can not substitute monitoring of real user activities.

Most of the existing tools which provide monitoring of real user activities were, as a rule, developed as a part of the data management and workload management systems of the LHC experiments (Phedex, Dirac, PanDA). They provide data transfer and job processing monitoring including application-level information, namely, application exit code, name of the data collection, detailed failure reason, version of the experiment-specific software. The limitation of these systems is that they are working in the scope of a single experiment and do not provide common monitoring solutions which can be shared by several virtual organizations.

Experiment Dashboard aims to combine Grid-related monitoring information with the application-related data and to offer where possible common monitoring solutions which can work across several virtual organizations.

3 Experiment Dashboard Framework

The common structure of the Experiment Dashboard service consists of information collectors, data repositories, normally implemented in ORACLE databases, and user interfaces. The Experiment Dashboard uses multiple sources of information, like:

- Other monitoring systems, like the Imperial College Real Time Monitor (ICRTM) or the Service Availability Monitoring (SAM)
- gLite Grid services, such as the Logging and Bookkeeping service (LB) or CEMon [21]
- Experiment specific distributed services, for example the ATLAS Data Management services or distributed Production Agents for CMS
- Experiment central databases (like the PanDA database for ATLAS)
- Experiment client tools for job submission, like Ganga [22] and CRAB [23]
- Jobs instrumented to report directly to the Experiment Dashboard

This list is not exhaustive. Information can be transported from the data sources via various protocols. In most cases, the Experiment Dashboard uses asynchronous communication between the source and the data repository. For several years, in the absence of a messaging system as a standard component of the gLite middleware stack, the MonALISA monitoring system was successfully used as a messaging system for the Experiment Dashboard job monitoring applications. Currently, the Experiment Dashboard is being instrumented to use the Messaging System for the Grid (MSG) [25] for the communication with the information sources.

A common framework providing components for the most usual tasks was established to fulfill the needs of the dashboard applications being developed for all experiments. The schema of the Experiment Dashboard framework is presented in Fig. 1.

The Experiment Dashboard framework is implemented in Python. The tasks performed on regular basis are implemented by the Dashboard agents. The framework provides all the necessary tools to manage and monitor these "agents", each focusing on a specific subset of the required tasks, such as collection of the input data or computation of the daily statistics summaries.

To ensure a clear design and maintainability of the system, the definition of the actual monitoring

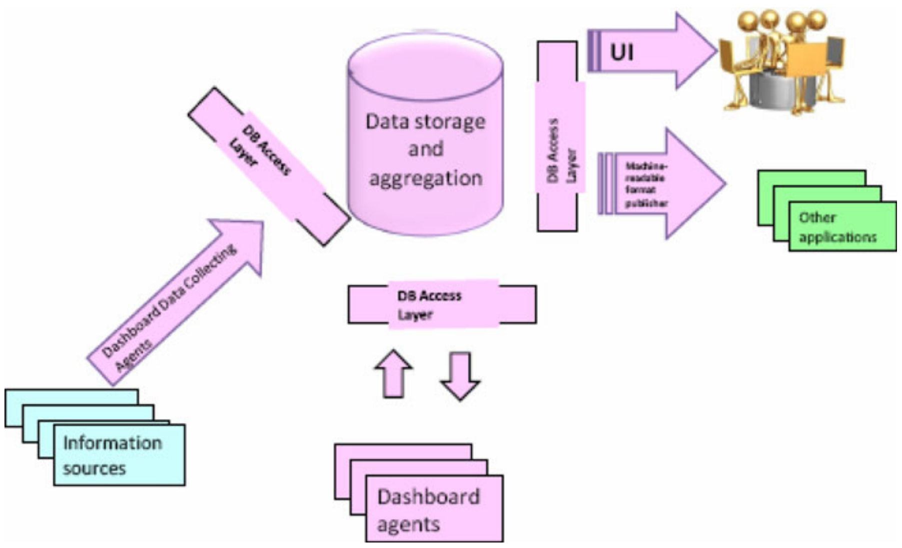


Fig. 1 The experiment dashboard framework schema

application queries is decoupled from the internal implementation of the data repository. Every monitoring application implemented within the Experiment Dashboard framework comes with the implementation of one or more data access objects (DAO), which represents the "data access interface": a public set of methods for the update and retrieval of information. Access to the database is done using a connection pool to reduce the overhead in creating new connections, therefore the load on the server is reduced and the performance increased.

The Experiment Dashboard requests are handled by a system following the Model-View-Controller (MVC) pattern. They are handled by the "controller" component, launched by the apache mod_python extension, which keeps the association between the requested URLs and the corresponding "actions", executing them and returning the data in the format requested by the client. All actions will process the request parameters and execute a set of operations, which may involve accessing the database via the DAO layer. When a response is expected, the action will store it in a python object, which is then transformed into the required format (HTML page, plain XML, CSV, image) by the "view" components. Applying the view to the data is performed automatically by the controller.

All the Experiment Dashboard information can be displayed using HTML, so that it can be viewed in any browser. Moreover, the Experiment Dashboard framework also provides the functionality to retrieve information in XML (extensible markup language), CSV (comma separated values), JSON (javascript object notation) or image formats. This flexibility allows the system to be used not only by users but also by other applications. A set of command line tools is also available.

The current Web page frontends are based on XSL style sheet transformations over the XML output of the HTTP requests. In some cases more dynamic Web interfaces are available by issuing asynchronous javascript requests from the client browser (AJAX), gradually building the Web page and offering improvements in navigation and performance.

Recently, support for the Google Web Toolkit (GWT) was added to the framework. Some of the applications have started to be migrated to this new client interface model, which gives great benefits both for users and developers—compiled code, easier support for all browsers, out of the box widgets.

All components are included in an automated build system based on the Python distutils, with additional or customized commands enforcing strict development and release procedures. In total, there are more than fifty modules in the framework, and fifteen of them being common modules offering the functionality shared by all applications.

The modular structure of the Dashboard framework enables flexible approach for implementing the needs of the customers. For example, for the CMS production system, Dashboard provides only implementation of the data repository. Data retrieved from the Dashboard database in XML format is presented to the users via user interface developed by the CMS production team in the CMS Web-tools framework [26].

Parts of the Dashboard framework are used by other projects, some unrelated to monitoring, as is the case with the ATLAS Distributed Data Management system described below.

4 Experiment Dashboard Monitoring of the Distributed Data Management System

The Experiment Dashboard provides monitoring applications for several data-management tasks, namely monitoring of data-transfer for the ALICE experiment, monitoring of the ATLAS Distributed Data Management (DDM) system and monitoring of the file transfer service (FTS) [27]. The last one is currently under development.

This section describes in detail the ATLAS DDM Dashboard as an example of the data-management monitoring applications.

4.1 Introduction to ATLAS DDM

The ATLAS DDM Dashboard plays a central role in ATLAS computing operations. It is widely used by the people taking computing shifts. ATLAS DDM monitoring serves about 1,000 unique visitors per month and up to 250,000 pages are viewed daily.

Matchmaking between computing tasks and available resources in ATLAS is done based on the location of the data and requires data proximity for resources to be considered eligible. The ATLAS DDM system is responsible for data placement on resources, and rapid export of data sets is important to ensure full utilization. This makes DDM a complex component with high availability requirements, and it therefore benefits from having a comprehensive monitoring system.

The ATLAS DDM system is composed of different components which interact with each other to perform the different bookkeeping and data movement tasks. The catalogs take care of the bookkeeping, while the site services try to answer the different user requests to spread existing data among the different sites.

With the expected amount of data being in the order of several petabytes per year, and the file sizes varying from the controlled production data (bigger and thus less files) to the more chaotic user analysis results (typically higher number of files), having collections of files greatly reduces the dimension of the bookkeeping task and improves performance as the dataset is also used as the unit of data movement. And being simply an abstraction over the physical data stored in individual files, datasets are also dynamic, as users can add or remove files from the collection, creating new versions of the same dataset.

4.2 Architecture of Dashboard DDM

The Dashboard DDM is composed of two main sets of components: the Web application and all the actions available through its HTTP based interface, and the agents, each taking care of a specific task. All of them use a single backend database, where all data concerning the monitoring of DDM and Grid fabric components is stored.

The application's most visible use is of course the access to the data using a Web browser.

XHTML is the default output format for any query resulting in response data, though other formats are available as mentioned above. The available functionality via the Web interface is depicted in Fig. 2.

The second part of the application is the information collection from the DDM system. It relies on a messaging system pattern, where different messages are delivered to pre-defined queues or topics. Currently HTTP is used in production for the delivery of the messages, but any other messaging backend supported by the Dashboard framework can be used, like MonALISA or Java Message Service (JMS).

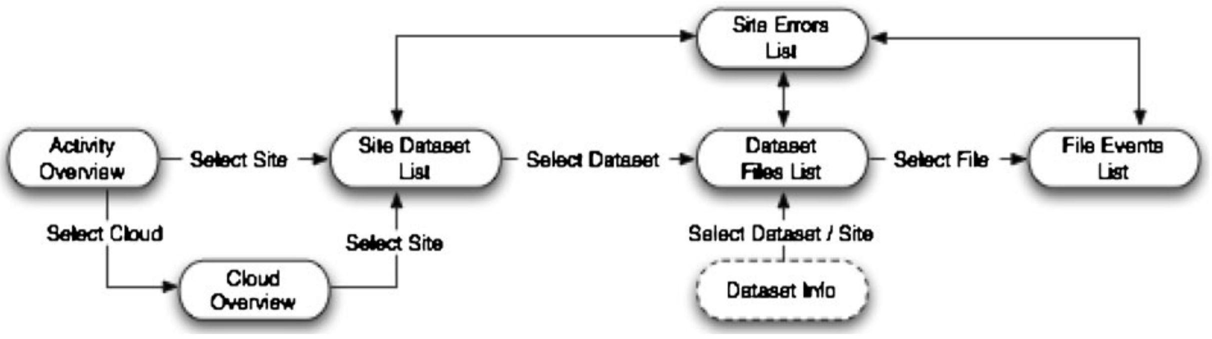


Fig. 2 Web application flow. Users navigate from Activity or Clouds (groups of sites) overview pages down to individual file event pages

The main reason of using HTTP for delivering these messages is the reduced number of external dependencies on the client. The only disadvantage of this option is the lack of reliability in the message delivery. This is a feature that does not exist natively in the HTTP protocol, so a client side layer with persistent storage had to be developed for cases where the endpoint is down or unreachable. Several modern messaging systems provide this functionality both at the client and broker levels.

Besides the collection of dataset and file transfer information from the site services, there are individual agents performing different tasks in the DDM Dashboard. They cover tasks like collecting service status and availability from the Site Availability Monitor (SAM) service, monitoring the available space in the storage elements, generating statistics regarding site and site data link performance—throughput, number of files and datasets moved, average dataset and file completion time, dataset and file time in queue, etc., and errors during both transfer and registration, generating and sending alarms to system operators. This information is gathered by doing periodic queries to the database.

4.3 Performance

Due to the amount of data being collected, a detailed design of the backend database was required. Moreover, the Web application endpoint was also taken into account, not really due to heavy data querying but due to it also being used to collect the different messages coming from the DDM system.

4.3.1 Backend Database

Rough figures used when designing the system indicated half a million datasets being created per year, each having 10 to 10,000 files. With around 80 storage areas at the time (both disk and tape) it was easy to calculate a worst case scenario and from these numbers it was decided to rely on the Oracle database service supported by the CERN IT (Information Technology) department—eliminating the need for the setup and support of a non-trivial database infrastructure.

The most impressive number comes not from the datasets or files, but from the individual file transfer state reports, which evolve over time. This is of course temporary data, useful for real time debugging but less useful once information becomes old and statistics have been gathered. Even so, the period of time where this information is kept should be made as big as possible. Oracle partitioning [37] was especially important for the file state history table, but is also used in both datasets and files tables.

4.3.2 Web Frontend

Choosing HTTP as the messaging protocol means that a server receiving messages from a different site service instances is hit very hard and should be highly optimized. Monitoring the system has shown peaks of up to 90 requests per second, with five requests per second in normal operation. Considering that most of these (90%) are bulk requests, the load on the service is very high. The choice was to use the Apache Web Server, which provides the required performance.

5 LHC Job Processing and the Experiment Dashboard Applications for Job Monitoring

The LHC job processing activity can be split in two categories: processing raw data and large-scale Monte-Carlo production, and user analysis. The main difference between the mentioned categories is that the first one is a large scale, well-organized activity, performed in a coordinated way by a group of experts, while the second one is chaotic data processing by members of the huge distributed physics community. Users running physics analysis do not necessarily have enough knowledge about the Grid and profound expertise in computing in general. With the restart of the LHC, a considerable increase of analysis users is expected. Clearly, for both categories of job processing, complete and reliable monitoring is a necessary condition for the success.

The organization of the workload management systems of the LHC experiments differs from one experiment to another. While in the case of ALICE and LHCb the job processing is organized via a central queue, in the case of ATLAS and CMS the job submission instances are distributed and there is no central point of control as in ALICE or LHCb. Therefore, the job monitoring for ATLAS and CMS is a more complicated task and it is not necessarily coupled to a specific workload management system. The Experiment Dashboard provides several job monitoring solutions for various use cases, namely the generic job monitoring applications, monitoring for ATLAS and CMS production systems, and applications focused on the needs of the analysis users. The generic job monitoring, which is provided for all LHC experiments, is described in more detail in Section 5.1. This application is generic and was used outside the LHC community by the VLEMED virtual organization [36]. Since the distributed analysis is currently one of the main challenges for the LHC computing, several new applications were built recently on top of the generic job monitoring, mainly for monitoring of the analysis jobs. Section 5.2 gives a closer look at the CMS Task Monitoring as an example of the job monitoring applications for user analysis.

5.1 Experiment Dashboard Generic Job Monitoring Application

The overall success of the job processing depends on the performance and stability of the Grid services involved in the jobs processing and on the services and software which are experiment-specific. Currently, the LHC experiments are using several different Grid middleware platforms and therefore a variety of Grid services. Regardless of the middleware platform, access from the running jobs to the input data as well as saving output files to the remote storage are currently the main reasons for job failures. Stability and performance of the Grid services, like the storage element (SE), storage resource management (SRM) and various transport protocols, are the most critical issues for the quality of the data processing. Furthermore, the success of the user application depends as well on the experiment-specific software distribution at the site, the data management system of the experiment and the access to the alignment and calibration data of the experiment known as “conditions data”. These components can have a different implementation for each experiment and they have a very strong impact on the overall success rate of the user jobs. The Dashboard Generic Job Monitoring Application tracks the Grid status of the jobs and the status of the jobs from the application point of view. For the Grid status of the jobs, the Experiment Dashboard used the Grid related systems as an information source. In the past, the Relational Grid Monitoring Architecture (RGMA) [28] and the Imperial College Real Time Monitor were used as information sources for Grid job status changes. Unfortunately, none of the mentioned systems provided complete and reliable data. The current development aimed to improve the situation of publishing the job status changes by the Grid services involved in the job processing, which is described in Section 5.1.2. To compensate the lack of information from the Grid-related sources, the job submission tools of the ATLAS and CMS experiments were instrumented to report job status changes to the Experiment Dashboard system. Every time when the job submission tools query the status of the jobs from the Grid services, the status is reported to the Experiment Dashboard. The jobs themselves are instrumented for the runtime reporting of their progress at the worker nodes. The information flow of the generic job monitoring application is described in the next section.

5.1.1 Information Flow of the Generic Job Monitoring Application

Similar to the common Dashboard structure, described in Section 3, the job monitoring system consists of the central repository for the monitoring data (Oracle database), collectors, and a Web server that renders the information in HTML, XML, CSV, or as images.

The main principles of the Dashboard job monitoring design are:

- to enable non-intrusive monitoring, which must not have any negative impact on the job processing itself,
- to avoid direct queries to the information sources and to establish asynchronous communication between the information sources and the data repository, whenever possible.

When the development of the job monitoring application started, the gLite middleware did not provide any messaging system, so the Experiment Dashboard uses the MonALISA monitoring as a messaging system. The job submission tools of the experiments and the jobs themselves are instrumented to report needed information to the MonALISA server via the apmon library, which uses the UDP protocol. Reporting is implemented in a lightweight way, not creating any monitoring overhead or any dependency between the reporting of job status information and job processing itself. The situation when a UDP packet cannot be sent does not prevent the job being executed. Statistics collected over 5 years show that the proportion of lost UDP packets has never exceeded 5%, and is normally at the level of 2–3%. This level of reliability is acceptable for the monitoring needs.

Every few minutes the Dashboard collectors query the MonALISA server and store job monitoring data in the Dashboard Oracle database. All information received by the MonALISA server is logged. If for any reason the database is not available, monitoring data is not lost. As soon as the database comes online, the Dashboard collector consumes information from the log file and stores it in the database.

The data related to the same job and coming from several sources is correlated via a unique Grid identifier of the job.

Following the outcome of the work of the WLCG monitoring working groups, the existing open source solutions of messaging systems were evaluated. As a result of this evaluation, Apache ActiveMQ [25] was proposed to be used for the Messaging System for the Grids (MSG). Currently, the Dashboard job monitoring application is instrumented to using MSG in addition to the MonALISA messaging.

The job status shown by the Experiment Dashboard is close to the real-time status. The maximum latency is 5 min, which corresponds to the interval between the sequential runs of the Dashboard collectors.

5.1.2 Instrumentation of the Grid Services for Publishing Job Status Information

As it was mentioned above, information about job status changes provided by Grid-related sources is currently not complete and covers only a subset of jobs. This has a bad impact on the trustworthiness of the Dashboard data. Though some job submission tools are instrumented to report job status changes at the point when they query Grid-related sources, this query is done on user request. For example, when a user never requests the status of his jobs and the jobs were aborted, there is no way for the Dashboard to be informed about the abortion of the jobs. As a result, they can stay in “running” or “pending” status, unless being turned into ‘terminated’ status with ‘unknown’ exit code by a so-called timeout Dashboard procedure. To overcome this limitation, the ongoing development aims to instrument the Grid services involved in the job processing to publish job status changes to the MSG. Dashboard collectors consume information from the MSG and store it in the central repository of the job monitoring data. The services which need to be instrumented and the concrete implementation depend of the way the jobs are submitted to the Grid.

In case the jobs are submitted via the gLite Workload Management System (WMS) the LB service keeps full track of the job processing. The LB provides the notification mechanism which allows to subscribe to the job status changes events and to be notified as soon as events matching the conditions specified by the user happen. A new component "LB Harvester" was developed in order to register at several LB servers and to maintain the active notification registration for each one. The output module of the harvester formats the job status message according to the MSG schema and publishes it into MSG.

Jobs submitted to Condor-G [29], do not use the WMS service and correspondingly do not leave a trace in the LB. The job status changes publisher component was developed in collaboration of Condor and the Dashboard teams. Condor developers have added a job logs parsing functionality to the Condor standard libraries. The publisher of the job status changes reads new events from standard Condor event logs, filters events in question, extracts essential attributes and publishes them to MSG. The publisher is run in the Condor scheduler as a Condor job. In this case, Condor itself takes care of publishing status changes.

5.1.3 Scalability and Performance

Taking as an example the CMS Dashboard Job Monitoring, the system keeps track of all jobs submitted by the CMS user community. One hundred thousand to 250,000 jobs are submitted daily and up to 30,000 jobs are running in parallel. A single MonALISA server which receives job status update messages can handle up to 3,000 updates per second. The Dashboard collector groups together messages related to the status of a particular job and performs updates of the database. Scaling up of the Dashboard job monitoring system can be achieved by increasing of the number of MonALISA servers and Dashboard collector instances. Currently, CMS is using 3 MonALISA servers. The actual update rate of the job status information in the database is 100–200 job status updates per second.

As was mentioned in the Section 3, current development aims to adapt the MSG system as a transport mechanism for the Dashboard job monitoring application. The MSG provides a reliable and scalable message bus for communication between information source and information consumer. The core component of the system is a message broker which routes information from sources to consumers. A network of interconnected message brokers ensures redundancy and scalability. Recent testing of the new job monitoring information flow using MSG confirmed that adapting this technology will allow for increased reliability and performance of data delivery from the information source to the Dashboard job monitoring repository.

The interactive user interface described in Section 5.1.4 shows the real-time situation regarding job processing in the scope of a single virtual organization. This interface uses raw information from the job monitoring database instance. Tuning of the database configuration and of the SQL queries and partitioning of the job monitoring schema provide sufficient performance of the Interactive User interface (2–4 s per request) under the condition that information is requested about recent jobs, submitted over last 48 h.

The job monitoring database instance for the CMS experiment contains information for more than 40 million jobs per year. In order to provide sufficient performance for accessing job monitoring information for time ranges longer than 48 h, raw monitoring data is aggregated in summary tables with hourly, daily and monthly granularity. This task is performed by Oracle scheduled jobs. Depending on the requested time range, the historical interface described in Section 5.1.4 queries information in the corresponding summary table.

5.1.4 Job Monitoring User Interfaces

The standard job monitoring application provides two types of user interfaces. First, the so called "interactive user interface", which enables very flexible access to recent monitoring data and shows the job processing for a given VO at runtime. The interactive UI contains the distribution of active jobs and jobs terminated during a selected time window by their status. Jobs can be sorted by various attributes, for example, the type of activity (production, analysis, test, etc.), site or computing element where they are being processed, job submission tool, input dataset, software version and many others. The information is presented in a bar plot and in a table. A user can navigate to a page with very detailed information about a particular job, for example, the exit code and exit reason, important time stamps of processing the job, number of processed events, etc.

Second, the "historical interface", which shows job statistics distributed over time. The historical view allows following the evolution of the numeric metrics such as the number of jobs running in parallel, CPU and wall clock consumption or success rate. The historical view is useful for understanding how the job efficiency behaves over time, how resources are shared between different activities, and how various job failures fluctuate as a function of time.

5.2 Job Monitoring for User Analysis

Distributed analysis on the WLCG infrastructure is currently one of the main challenges of the LHC computing. As it was already mentioned earlier, transparent access to the LHC data has to be provided for more than 5,000 scientists all over the world. Analysis users do not necessarily have expertise in Grid computing. The number of active LHC users is steadily growing. According to the Dashboard statistics since the beginning of 2008 more than 1,000 distinct users of the CMS VO were running analysis jobs on the WLCG infrastructure. 100–200 distinct CMS users submit their analysis jobs to the WLCG daily. The significant streamlining of operations and the simplification of end-users' interaction with the Grid are required for effective organization of the LHC user analysis. Simple, user-friendly, reliable monitoring of the analysis jobs is one of the key components of the operations of the distributed analysis. Such monitoring is required not only for the physicists running analysis jobs but also for the analysis support teams. In case of CMS, most of the analysis users interact with the Grid via the CMS Remote Analysis Builder (CRAB). User analysis jobs can be submitted either directly to the WLCG infrastructure or via the CRAB analysis server, which operates on behalf of the user. In the first case, the support team does not have access to the log files of the users' job or to the CRAB working directory which keeps track of the task generation. To understand the reason of the problem of a particular user's task, the support team needs a monitoring system capable of providing complete information about the task processing.

To serve the needs of the analysis community and of the analysis support team, the Dashboard Task Monitoring [30] application has been developed on top of the CMS job monitoring repository.

The application provides a comprehensive view of the task processing. It demonstrates the progress of the task and assists in detecting of the eventual problems and in providing debugging information.

The Task Monitoring information includes the job status of individual jobs in the task, their distribution by site and over time, the reason of failure, the number of processed events and the resubmission history. According to the feedback of the user community the tool provides intuitive navigation. Monitoring information is updated every 5 min.

The application offers a wide variety of graphical plots that visually assist the user to understand the status of the task.

The development was user-driven with physicists invited to test the prototype in order to assemble further requirements and identify any weaknesses of the application. The monitoring tool has become popular among the CMS users. According to our Web statistics, more than one hundred distinct analysis users are using it for their everyday work.

One of the important improvements foreseen for the Task Monitoring application is failure diagnostics for both Grid and application failures. The ideal situation would be to reach to a point where the user does not have to open the log file to search for what went wrong, but rather find all necessary information within the monitoring tool. A development effort is ongoing to improve the failure diagnostics reported to the Dashboard from the CRAB job wrapper.

Another attempt to improve the understanding of the underlying failure reasons of the user jobs is described in the next section.

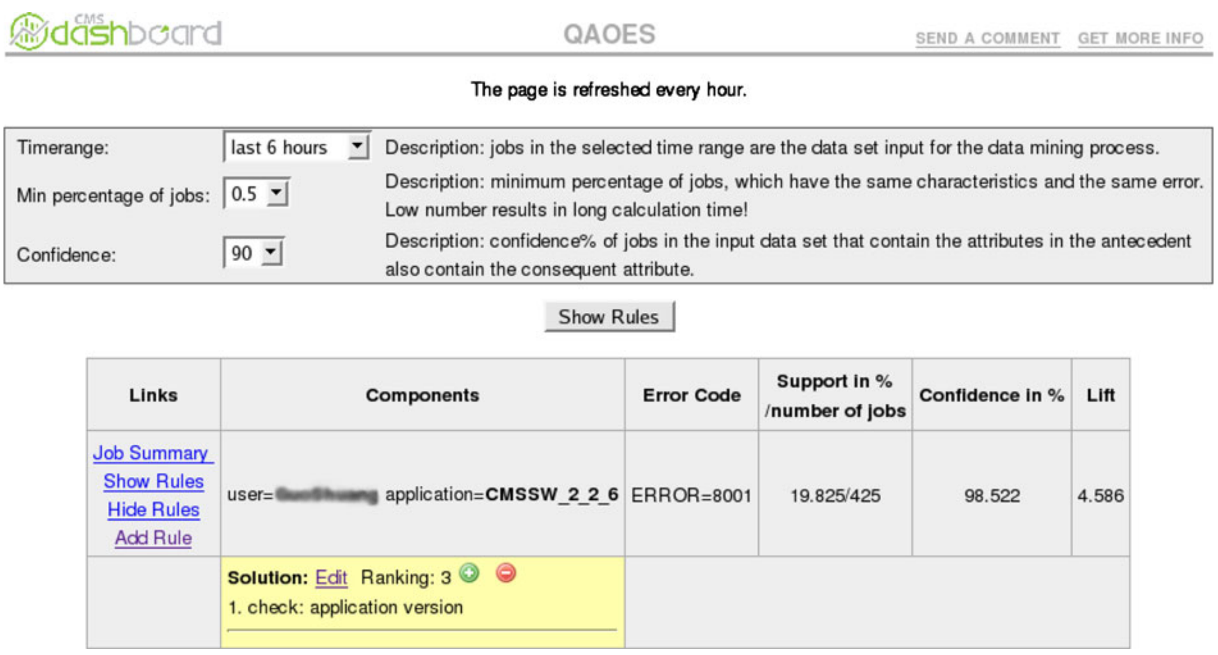
5.3 Quick Analysis of Error Sources

Some of the monitoring tools described in this section deliver information about failing Grid jobs, including exit codes, and provide a possible failure reason. However, the exit codes do not always indicate the Grid component which hosts the actual fault responsible for the erroneous behavior of Grid jobs. Instead, a more sophisticated methodology is required to locate problematic Grid components. In this context, we define a Grid component as an entity involved in the execution of a job, such as Grid service, used software, a software version, a dataset name and , and the user submitting the job. Within the Experiment Dashboard a tool called QAOES (Quick Analysis Of Error Sources) was developed. Logically the system consists of two steps: the first aims to detect potential problems on a Grid infrastructure automatically and the second provides previously collected human expertise about possible solutions to the detected problems. The information is merged and then presented in a Web interface, depicted in Fig. 3

5.3.1 Expert System and Association Rule Mining

Based on the list of possible problems, we collect human expert knowledge about the underlying problem diagnose and a possible solution. The expert system consists of a knowledge base keeping all the information in rule-based format and an inference engine, which consults the knowledge base to search for and retrieve rules matching the generated potential problems. This work is in progress. The rules implemented so far are covering relatively simple use cases.

We applied a data mining technique called association rule mining to previously collected job monitoring data. This approach reveals hidden information about Grid elements' behavior by taking dependencies between the characteristics of failing Grid jobs into account. To generate the Association Rules we use the a priori algorithm



The page is refreshed every hour.

Timerange:	last 6 hours	Description: jobs in the selected time range are the data set input for the data mining process.
Min percentage of jobs:	0.5	Description: minimum percentage of jobs, which have the same characteristics and the same error. Low number results in long calculation time!
Confidence:	90	Description: confidence% of jobs in the input data set that contain the attributes in the antecedent also contain the consequent attribute.

Links	Components	Error Code	Support in % /number of jobs	Confidence in %	Lift
Job Summary Show Rules Hide Rules Add Rule	user=[redacted] application=**CMSSW_2_2_6**	ERROR=8001	19.825/425	98.522	4.586
	Solution: Edit Ranking: 3 1. check: application version				

Fig. 3 The quick analysis of error sources Web interface shows one problem and one suggestion to solve the problem which is retrieved from the expert system

[31]. The output of this step is a list of associated Grid components which cause a certain exit code, together with quality measures (support, confidence). For example:

$$\{\text{SITE} = \text{A}, \text{SOFTWARE\_VERSION} = 5\}$$
$$\rightarrow \text{ERROR} = 7\,[\text{s} = 1.1\%, \text{c} = 80\%]$$

This process is executed every hour with job monitoring data of the past 6 h.

6 Site Monitoring

Currently the WLCG infrastructure includes more than 140 sites and the number of sites is still increasing. Operating such a large scale heterogeneous infrastructure requires stable and reliable behavior of the distributed services and the sites hosting these services. The LHC VOs need to extensively test all relevant aspects of the Grid site, such as the ability to efficiently use the network to transfer data, the functionality of all site services relevant to a given VO and the capability to sustain the VO workflows.

The applications described in this section aim to provide a straight forward way for monitoring of the Grid sites from the perspective of a VO.

6.1 Dashboard Site Availability Application Based on the Results of SAM Tests

The service availability monitoring (SAM) is a framework that allows the periodic execution of tests. These tests are setup in all the sites participating in the WLCG to verify the status of the Grid services. In addition to the tests defined by the WLCG operators, each VO can define a set of VO-specific tests that are performed at the sites used by a given VO.

The Dashboard Site Availability application based on the results of SAM tests was originally developed following a request from the CMS experiment. Later on, other experiments found it useful and the application was redesigned to be generic and usable by any experiment using the SAM framework for site evaluation.

The application enables a possibility for the VO to define their own site availability definitions, taking into account a given set of tests and services. In this context, VO-specific availability measures the capability of sites to provide certain functionalities. They can be related to a particular service, or set of services or a particular workflow. A single VO can introduce several availability definitions.

To introduce a new availability, the VO needs to define the set of service types to be taken into account for availability calculation (for example: computing element) and the set of tests which are critical for each given service type.

The results of the tests are stored in the SAM database. The stored Oracle procedures calculate the availability based on the test results. The Dashboard SAM Web portal displays the results of these tests and the calculated availability.

Two different views are provided: one to check the latest test for a given service; the second one is to evaluate how a site or a service evolves over time. Figure 4 presents the monthly site availability for all sites supporting CMS during three different time periods: January 2008, January 2009 and July 2009. There has been an enormous improvement both in the number (doubled from 28 to 56) and in the quality provided by the sites. As demonstrated in Fig. 4 in the beginning of 2008 18 best sites out of 28 showed monthly availability between 60% and 80%, while in summer of 2009 availability of 47 sites out of 56 was higher than 80%.

The Dashboard SAM Web portal is extensively used by the CMS VO for the site commissioning activity[32], which results in steady improvement of the state of the sites used by the CMS VO. Similar to other Dashboard applications, information from the Dashboard SAM Site availability application can also be retrieved in XML format. This format is used for importing data to the site local fabric monitoring systems.

6.2 Site Status Board

The site status board (SSB) provides a monitoring framework where the monitored metrics can be dynamically (re)defined by the user commu-

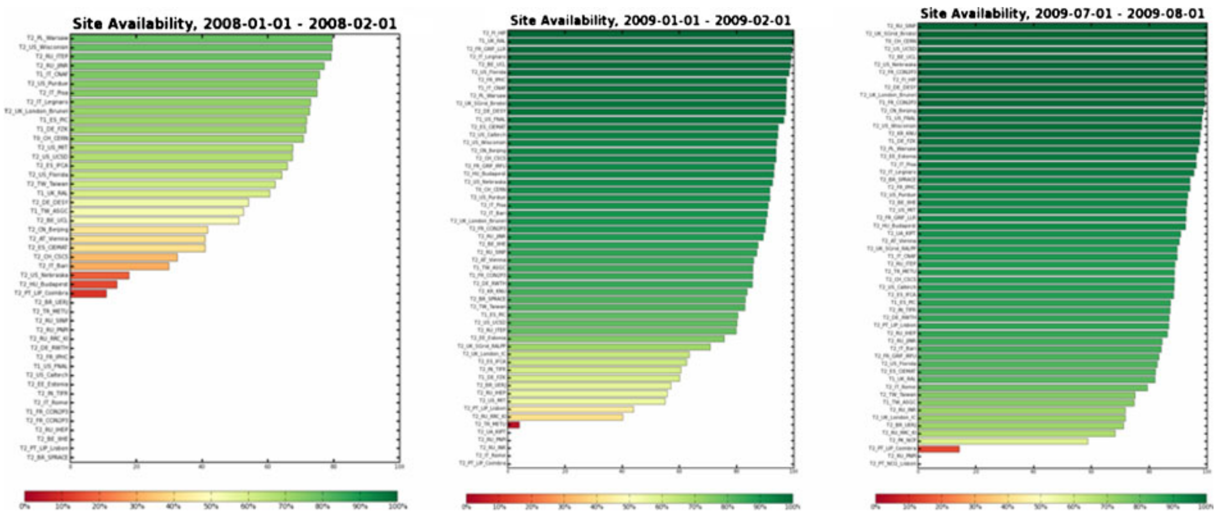


Fig. 4 shows monthly availability of the sites used by CMS. *Every bar* corresponds to a particular site, availability is in the range 0–100%. *Green color* corresponds to high availability, *red one* to the low availability

nity. SSB was designed to allow flexibility for the users in terms of defining the monitored metrics, their criticality or in configuring the users' views. Adding a new metric or an alternative view is straight forward and is being performed via a provided Web interface. The SSB is a generic application and can be used by any VO.

Initially, the SSB was developed for the CMS VO, for people who take computing shifts. The goal of the computing shifts is to follow up the CMS computing activities on the distributed infrastructure and to take actions in case of eventual problems. The display for the computing shifts provides the status of all the sites via single page with clear indication of any problem of any nature. Sometimes it might not be evident which information has to be taken into account to conclude whether a site is performing well from the perspective of the VO. Multiple iterations are required to define the set of important metrics. Therefore, the SSB was designed to be flexible and allow integration of new metrics.

At the moment, the application is used by all four LHC VOs.

Currently, CMS uses the SSB for several purposes:

- Computing shifts: For each site, the SSB displays the status of the services, efficiency of the job processing for production and analysis, current number of jobs, status of the data transfers, information about downtimes, and links to open issues of the site
- Site commissioning: This activity observes the efficiency of test jobs submitted to the sites and the status of the transfer links. CMS has a well-defined procedure that sites have to follow in order to be commissioned and be able to participate in the computing activities of the experiment
- Space monitoring: Displays the amount of occupied and free storage space at the different sites

Like many other Dashboard applications, the SSB has three components: collectors that gather information, a data repository, and a Web portal that displays the results. The collectors can fetch each metric from a different source. Moreover, each metric can define its own refresh rate.

VO administrators manage the metrics via a X509-authenticated Web interface. They can define which metrics are critical for each of the different activities. For visualization purposes the metrics can be grouped in a flexible way and various alternative views are enabled based on this grouping.

7 Integration of the VO-Specific Monitoring Systems

The VO-specific monitoring systems work in the scope of a single experiment. Very often different activities of a VO are monitored by different systems. There is no tool providing an overall view of the VO computing activities. On the other hand, a high-level cross-VO view is also missing.

This problem becomes more visible with the growing scale of activities of the LHC VOs on the WLCG infrastructure and with the intensive, simultaneous use of the WLCG resources by all LHC VOs. Moreover, the site administrators of sites serving several LHC VOs complained that it is not convenient to navigate to multiple VO-specific monitoring systems in order to understand whether their site is responding well to the needs of all LHC experiments.

The recent development aims to integrate the existing VO-specific monitoring systems and to provide a high-level view combining metrics from each of them. In this section we describe an example of such an application built on top of the VO-specific monitoring systems.

7.1 Siteview

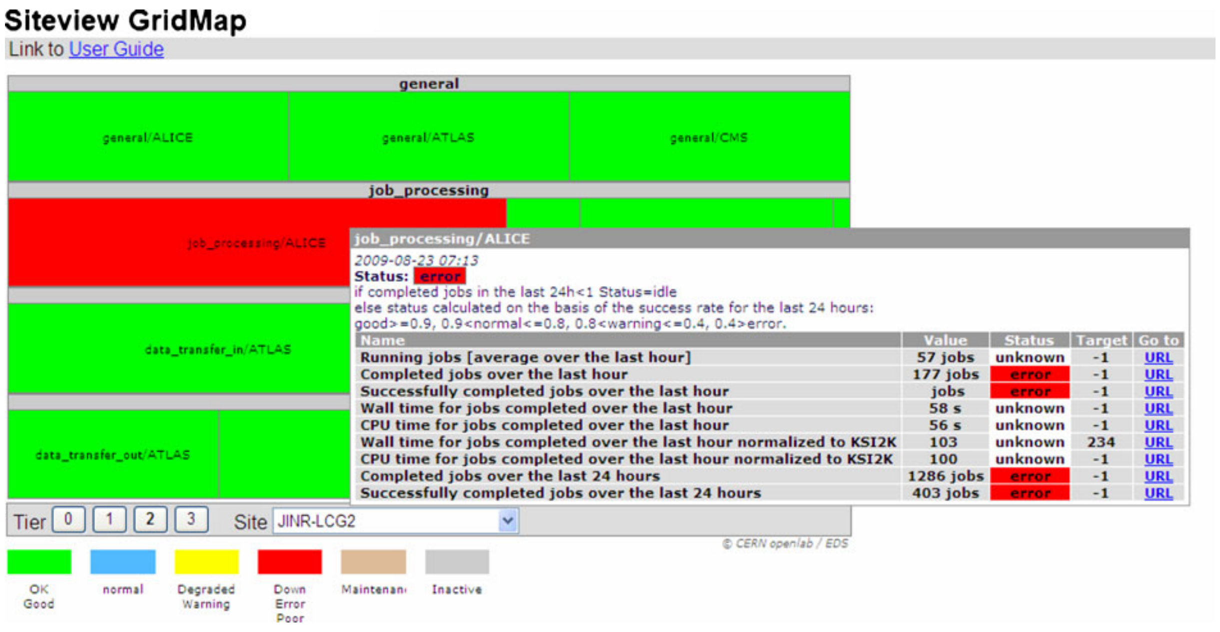
The Siteview application is based on the Experiment Dashboard Framework and the GridMap [33] visualization system.

The GridMap application was developed in collaboration of CERN and the HP Company. GridMap enables appropriate visualization for the distributed systems. The GridMap image represents a map consisting of groups of cells. Each cell can correspond to a single monitored instance or to a group of instances organized in a hierarchical structure. The color of the cell represents the status of a single instance or of the group of instances. The size of the cell can be defined by some important numeric monitoring metrics. With a mouse click on a cell, it turns into a sub-map showing all instances of the corresponding group. There is a good match between the capability of the GridMap to present the status of the complicated hierarchical structures and the requirements for the visualization of the distributed WLCG infrastructure.

The Siteview integrates data from multiple monitoring systems of the four LHC experiments, namely the MonALISA repository of ALICE, Dashboard for ATLAS DDM, ATLAS ProdSys Dashboard, PhEDEx, CMS Dashboard and Dirac. The repository of the common monitoring metrics is implemented in Oracle. The application provides not only a snapshot view but also the historical distribution of the common metrics. The purpose of the tool is to propagate information about the status of the most important VO activities at a site and to provide a single entry point for a transparent navigation of the VO-specific monitoring information to the site administrators. The Siteview map publishes the pre-cooked links to the relevant information, so that even a person not familiar with the monitoring system displayed the primary information source, can easily find corresponding data. The VO-specific monitoring systems publish monitoring data in CSV format via the http protocol. Data is consumed by the Dashboard collectors and stored in the Oracle database. The UI based on GridMap visualization runs on top of the metric repository. The Siteview uses the same database structure as the SSB application described in Section 6.2 The Siteview collectors were implemented following the example of the collectors of the SSB application. Thanks to this, all the features implemented for the SSB are immediately available for the Siteview.

An example of the Siteview screenshot is shown in Fig. 5.

The interface shows the map split in four parts. The first one is for the overall status of the site from the perspective of the VOs. The second one is for the job processing, the third and the fourth ones are for the incoming and outgoing transfers. Every cell corresponds to a particular VO. On moving the mouse over a particular cell the frame shows up. It contains the most important metrics and the links to the primary information sources. An example in Fig. 5 shows job processing metrics. On the click on a particular case the sub-map is displayed. It shows the distribution by job processing types for the job processing part or by channels for the transfer parts.



job_processing/ALICE

2009-08-23 07:13

Status: error

if completed jobs in the last 24h<1 Status=idle
else status calculated on the basis of the success rate for the last 24 hours:
good>=0.9, 0.9<normal<=0.8, 0.8<warning<=0.4, 0.4>error.

Name	Value	Status	Target	Go to
Running jobs [average over the last hour]	57 jobs	unknown	-1	URL
Completed jobs over the last hour	177 jobs	error	-1	URL
Successfully completed jobs over the last hour	jobs	error	-1	URL
Wall time for jobs completed over the last hour	58 s	unknown	-1	URL
CPU time for jobs completed over the last hour	56 s	unknown	-1	URL
Wall time for jobs completed over the last hour normalized to KSI2K	103	unknown	234	URL
CPU time for jobs completed over the last hour normalized to KSI2K	100	unknown	-1	URL
Completed jobs over the last 24 hours	1286 jobs	error	-1	URL
Successfully completed jobs over the last 24 hours	403 jobs	error	-1	URL

Fig. 5 Example of the site view

8 Collaboration with the User Community

As mentioned in the paper, close collaboration with the user community is essential for the success of the monitoring applications.

One of the principles followed during the development process of the Dashboard system is to involve potential users of a given application in designing the user interface and in validation of the early prototype. In order to understand the needs of the Dashboard users, members of the Dashboard team take part in the LHC computing shifts, attend computing meetings and conduct surveys asking for feedback about recent applications. Moreover, the usage of the Dashboard applications is monitored via Web statistics tools.

User input is taken into account for defining the content of the monitoring data collected in the data repository, the level of its aggregation and for designing the user interfaces. Depending on the role of a particular group of users the same monitoring information should be exposed in a different way.

Taking as an example information about data collection used by the analysis jobs, in the Task monitoring application described in Section 5.2, the only thing a physicist is interested in knowing is which of his actual tasks reads a particular data collection. Site administrators would like to monitor how many users/jobs are accessing a given data collection at their site, for how long, whether success rate of user jobs correlates with a particular data collection, etc... Computing teams responsible for data distribution are interested in understanding whether replication of data collections is done in an optimal way in order to ensure load balancing of the analysis jobs between various sites. All of these categories of users require a different presentation of the same monitoring data, different time ranges for monitoring the data evolution and various scopes.

User requirements regarding the flexibility which has to be provided by the monitoring application also strongly depend on the category of users. While the Task monitoring user interface is rather stable and very few feature requests were submitted after validation of the application by the user community, applications used for computing operations constantly require quick changes. Therefore the Site Status Board application which represents a sort of container which can keep various monitoring metrics and can expose these metrics in a customized form was developed for computing operations.

9 Conclusions

The Experiment Dashboard system became an essential component for the LHC computing operations. The variety of its applications covers the full range of the LHC computing activities. The system is being developed in a very close collaboration with the users. As a result, the Experiment Dashboard manages to respond well to the needs of the LHC experiments.

The Dashboard team has repeatedly found generic solutions during the development process which can be shared by all LHC VOs. Rather often, the application initially requested by a single experiment is being adapted to the rest of the community on the request of other VOs. This, certainly, allows not only to save on development effort for the implementation of the new requests, but facilitates the maintenance and support of the existing Dashboard applications. Some of the Dashboard applications , for example, generic job monitoring was used outside the LHC user community by VLEMED virtual organization.

The Dashboard framework provides all the necessary components for the development and management of the monitoring system and it is being used by other projects.

A big diversity of the VO-specific monitoring tools used by the LHC community creates certain problems. There is no global cross-VO picture of the LHC activities on the WLCG infrastructure. To overcome this limitation, the recent development effort was focused on integration of the monitoring data coming from the multiple VO-specific monitoring systems and on presenting it in a consistent way. The Dashboard framework and the GridMap visualization system are used for this purpose.

The future evolution of the Experiment Dashboard is driven by the needs of the LHC experiments, applying where possible common solutions.

Acknowledgements The authors would like to stress that Dashboard development is the result of collaboration with many colleagues in other projects and development teams like MonALISA, SAM, Ganga, Condor-g, LB and with the colleagues in the LHC experiments, in particular developers of the workload and data management systems and members of the VO computing projects and task-force teams. The authors would like to express their gratitude to all those people for the fruitful collaboration. They would like to thank LHC physicists providing feedback regarding user analysis monitoring tools. Special thank to Stefano Belforte and Andrea Sciaba for valuable contributions and useful feedback.

This work is co-funded by the European Commission through the EGEE-III project (www.eu-egee.org), contract number INFSO-RI-222667, and the results produced made use of the EGEE Grid infrastructure.

References

1. Evans, L., Bryant, P. (eds): LHC machine. J. Instr. 3:S08001. doi:10.1088/1748-0221/3/08/S08001 (2008)
2. The ATLAS Computing Group: ATLAS computing technical design report. ATLAS-TDR-017, CERN-LHCC-2005-022 (2005)
3. The CMS Computing Group: CMS computing technical design report. CMS-TDR-007, CERN-LHCC-2005-023 (2005)
4. The ALICE Computing Group: ALICE computing technical design report. CMS-TDR-012, CERN-LHCC-2005-020 (2005)
5. The LHCb Computing Group: LHCb computing technical design report. CMS-TDR-0011, CERN-LHCC-2005-019 (2005)
6. European Organization for Nuclear Research, CERN. http://public.web.cern.ch/public/. Accessed 21 April 2010
7. Shiers, J.: The Worldwide LHC Computing Grid (worldwide LCG), Computer Physics communications, vol. 177, issues 1–2, July 2007, pages 219–223, Proceedings of the Conference on Computational Physics 2006 - CCP (2006)
8. Rehn, J., et al.: PhEDEx high-throughput data transfer management system. In: CHEP06, Conference on Computing in High Energy and Nuclear Physics Proceedings, Mumbai, India, (2006)
9. Tsaregorodsev A., et al.: Dirac: A community Grid solution. In: CHEP07, Conference on Computing in High Energy and Nuclear Physics Proceedings, Victoria, BC, Canada (2007)
10. Nilsson, P.: PanDA system in ATLAS Experiment. ACAT'08, Italy (2008)
11. Saiz, P., et al.: AliEn—ALICE environment on the GRID. Nucl. Instrum. Methods **A502**, 437–440 (2003)
12. http://www.hpcwire.com/offthewire/STEP09-Demonstrates-LHC-Readiness-49631242.html. Accessed 21 April 2010

13. Laure, E., Fisher, S.M., Frohner, A., Grandi, C., Kunszt, P., et al.: Programming theGrid with gLite. Comput. Methods Sci. Technol. **12**(1), 33–45 (2006)
14. Pordes, R., et al.: The open science Grid. J. Phys. Conf. Ser. **78**, 012057 (2007)
15. Eerola, P., et al.: Roadmap for the ARC Grid middleware. PARA LNCS 4699 (2006)
16. CMS Dashboard stats. http://lxarda18.cern.ch/awstats/awstats.pl?config=lxarda18.cern.ch. Accessed 21 April 2010
17. Karmady, R., et al.: GridView: a monitoring and visualization tool. In: CHEP06, Conference on Computing in High Energy and Nuclear Physics Proceedings, Mumbai, India (2006)
18. Martyniak, J., et al.: Real time monitor of Grid job executions. In: CHEP09, Conference on Computing in High Energy and Nuclear Physics Proceedings, Prague, Chech Republic (2009)
19. Ruda, M.: A uniform job monitoring service in multiple job universes. High Performance Distributed Computing, Proceedings of the 2007 workshop on Grid monitoring Monterey, California USA (2007)
20. Collados, D.: Evolution of SAM in an enhanced model for monitoring WLCG services. In: CHEP09, Conference on Computing in High Energy and Nuclear Physics Proceedings, Pargue, Chech Republic (2009)
21. Aiftimiei, C., P, et al.: Using CREAM and CEMON for job submission and management in the gLite middleware. In: CHEP09 Conference Proceedings, Pargue, Chech Republic (2009)
22. Moscicki, J., et al.: Ganga: a tool for computational-task management and easy access to Grid resources. Comput. Phys. Commun. arXiv:0902.2685v1
23. Spiga, D., et al.: The CMS remote analysis builder (CRAB). Lect. Notes Comput. Sci. **4873**, 580–586 (2007)
24. Legrand, I., Newman, H., Cirstoiu, C., Grigoras, C., Toarta, M., Dobre, C.: MonALISA: an agent based, dynamic service system to monitor, controland optimize Grid based applications. In: Proceedings of Computing for High Energy Physics, Interlaken, Switzerland (2004)
25. Casey, J., Rodrigues, D., Schwickerath, U., Silva, R.: Monitoring the efficiency of user jobs. In: CHEP'09: 17th International Conference on Computing in High Energy and Nuclear Physics, Prague, Czech Republic (2009)
26. Metson, S., et al.: CMS offline webtools. In: CHEP'07: Conference on Computing in High Energy and Nuclear Physics Procedings, Victoria, BC, Canada (2007)
27. Schulz, M., et al.: Building the WLCG file transfer service. In: CHEP07: Conference on Computing in High Energy and Nuclear Physics Proceedings, Victoria, BC, Canada (2007)
28. Cooke, W., et al.: The relational Grid monitoring architecture: mediating information about Grid. J. Grid Comput. **2**(4), 1–17 (2004)
29. Thain, D., Tannenbaum, T., Livny, M.: Distributed computing in practice: the condor experience. Concurrency Comput. Pract. Ex. **17**(2–4), 323–356 (2005)
30. Karavkis, E., et al.: CMS dashboard for monitoring of the user analysis activities. In: CHEP'09: 17th International Conference on Computing in High Energy and Nuclear Physics, Prague, Czech Republic (2009)
31. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: Proceedings of the 20th International Conference on Very Large Data Bases, VLDB, Santiago, Chile, 487–499 (1994)
32. Belforte, S., et al.: The commissioning of CMS sites: improving the site reliability. In: 17th International Conference on Computing in High Energy and Nuclear Physics Proceedings, Prague, Czech Republic (2009)
33. GridMap visualization, http://www.isgtw.org/?pid=1000728. Accessed 21 April 2010
34. Andreozzi, S., et al.: GridICE: a monitoring service for Grid systems. Future Gener. Comput. Syst. **21**(4), 559–571 (2005)
35. Smallen, S., et al.: User-level Grid monitoring with Inca 2. In: High Performance Distributed Computing, Proceedings of the 2007 workshop on Grid monitoring Monterey, California, USA (2007)
36. Andreeva, J., et al.: The experiment dsahboard for medical applications. In: 3rd EGEE User Forum, Clermont-Ferrand, France (2008)
37. Oracle Partitioning. http://www.oracle.com/us/products/database/options/partitioning/index.htm. Accessed 21 April 2010