iopscience.iop.org

Home Search Collections Journals About Contact us My IOPscience

CMS Dashboard Task Monitoring: A user-centric monitoring view

This content has been downloaded from IOPscience. Please scroll down to see the full text.

2010 J. Phys.: Conf. Ser. 219 072038

(http://iopscience.iop.org/1742-6596/219/7/072038)

View the table of contents for this issue, or go to the journal homepage for more

Download details:

IP Address: 134.83.92.127
This content was downloaded on 11/05/2015 at 14:17

Please note that terms and conditions apply.

CMS Dashboard Task Monitoring: A user-centric monitoring view

Edward Karavakis[1,2], Julia Andreeva[2], Akram Khan[1], Gerhild Maier[2], Benjamin Gaidioz[2]

[1] School of Engineering and Design, Brunel University, Uxbridge, UB8 3PH, UK

[2] CERN, European Organisation for Nuclear Research, CH-1211 Geneva 23, Switzerland

E-mail: Edward.Karavakis@cern.ch

Abstract. We are now in a phase change of the CMS experiment where people are turning more intensely to physics analysis and away from construction. This brings a lot of challenging issues with respect to monitoring of the user analysis. The physicists must be able to monitor the execution status, application and grid-level messages of their tasks that may run at any site within the CMS Virtual Organisation. The CMS Dashboard Task Monitoring project provides this information towards individual analysis users by collecting and exposing a user-centric set of information regarding submitted tasks including reason of failure, distribution by site and over time, consumed time and efficiency. The development was user-driven with physicists invited to test the prototype in order to assemble further requirements and identify weaknesses with the application.

1. Introduction

The Experiment Dashboard [1] is a monitoring system developed for the LHC experiments in order to provide the view of the Grid infrastructure from the perspective of the Virtual Organisation. The CMS Dashboard provides a reliable monitoring system that enables the transparent view of the experiment activities across different middleware implementations and combines the Grid monitoring data with information that is specific to the experiment.

The scientists must be able to monitor the execution status, application and grid-level messages of their tasks that may run at any site on the distributed WLCG infrastructure. The existing CMS monitoring systems provide this type of information but they are not focused on the user's perspective.

The CMS Dashboard Task Monitoring project addresses this gap by collecting and exposing a user-centric set of information to the user regarding submitted tasks. It provides a clear and precise view of the status of the task including job distribution by sites and over time, reason of failure and advanced graphical plots giving a more usable and attractive interface to the analysis and production user. The development was user-driven with physicists invited to test the prototype in order to assemble further requirements and identify weaknesses with the application.

In the first section of this paper, the concept of the Experiment Dashboard monitoring system will be described in detail. The next sections provide an overview of the Task Monitoring application and its features. The two final sections focus on the known issues, the future work and draw some conclusions.

2. Architecture

The CMS Task Monitoring application is part of the Experiment Dashboard system [2] which is widely used by the four LHC experiments. The *Controller* is the main piece of the web application (Figure 1). It receives all client requests and decides what to do with them. For each client request there should be a corresponding *Action*, which will normally involve some interaction with the model of the application (some business logic that might involve accessing or updating persistent data).

A client request might involve producing some output. This output is identified by its mime/type and will have a *View* associated with it. The *Action* will put any data that it collected/produced in a shared area (the *ActionContext*) so that it can later be taken by the *View* to produce the output to the client. All the relationship between client requests, *Actions*, *Views* and its associated mime/types is defined in a single configuration file, the *ActionMapping* file. A widely used format for data retrieval is HTML but information can also be retrieved in XML, CSV or image formats allowing any third party application to use the system.

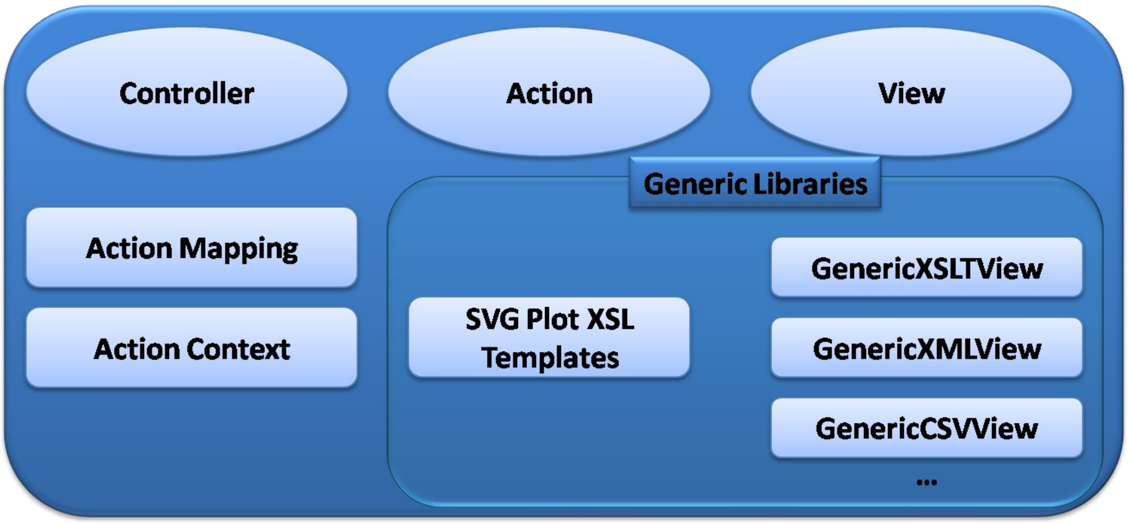


Figure 1. Web Application Architecture

The Dashboard Task Monitoring application is built on top of the Dashboard Job Monitoring system which uses multiple sources of information [3]. There are two main architectural principals of the Dashboard Job Monitoring system:

- Monitoring should not be intrusive to the information source. Thus, it does not pool information from the primary monitoring sources on a regular basis to avoid adding additional load on the services responsible for the job processing.
- The Dashboard uses a message-oriented architecture. There is no synchronous connection to the primary information producer. The job submission tools as well as the jobs themselves are instrumented to report in real time important events to the MonALISA [6] servers. The Dashboard Collectors regularly consume information published by the MonALISA servers. At the time when the development of the Dashboard started in the summer of 2005, no messaging system was provided as a standard component of the Grid Middleware stack. The MonALISA

system was selected to be used as a messaging system for the Dashboard. Currently, the Dashboard development team is integrating the Dashboard with the Messaging System for the Grid (MSG) [4].

The data collectors gather both Grid-related information as well as information specific to the application which is run by the users (Figure 2). The Grid-related information is obtained in the XML format from the Logging and Bookkeeping Database using the Imperial College Real Time Monitoring publisher (ICRTM) [5]. The application-specific information is gathered throughout a job's lifetime via the MonALISA monitoring system.

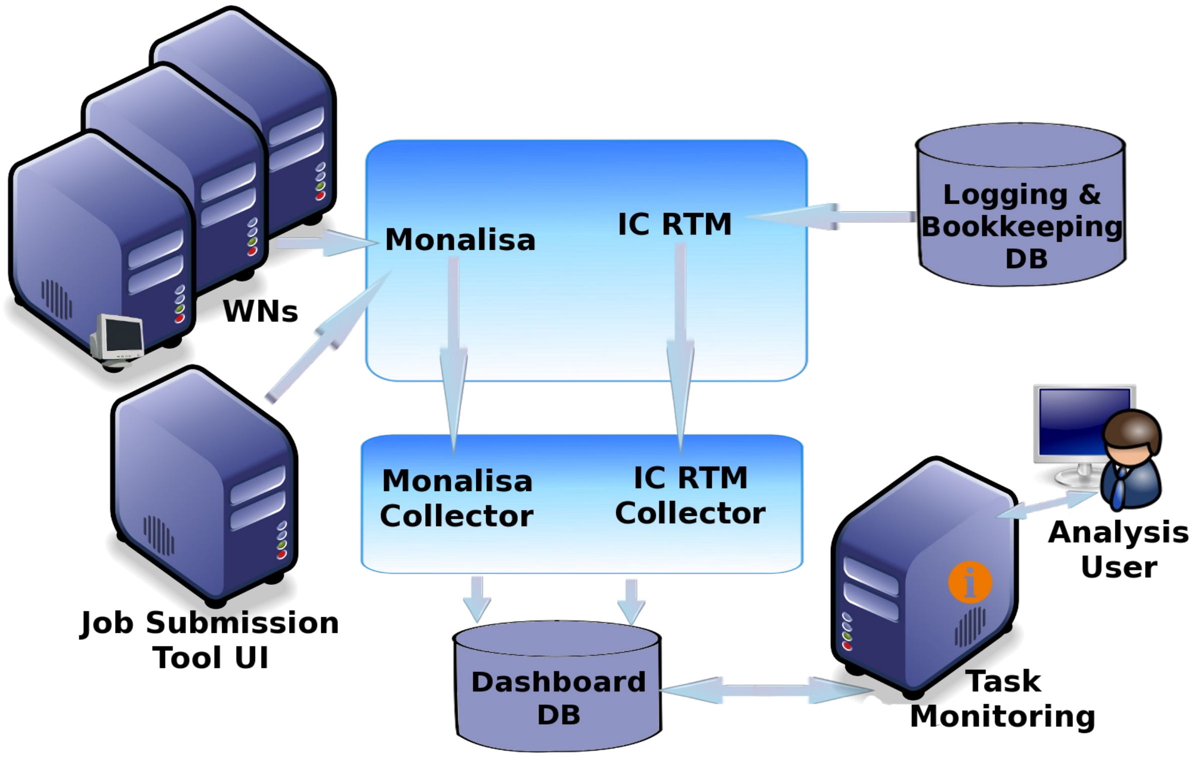


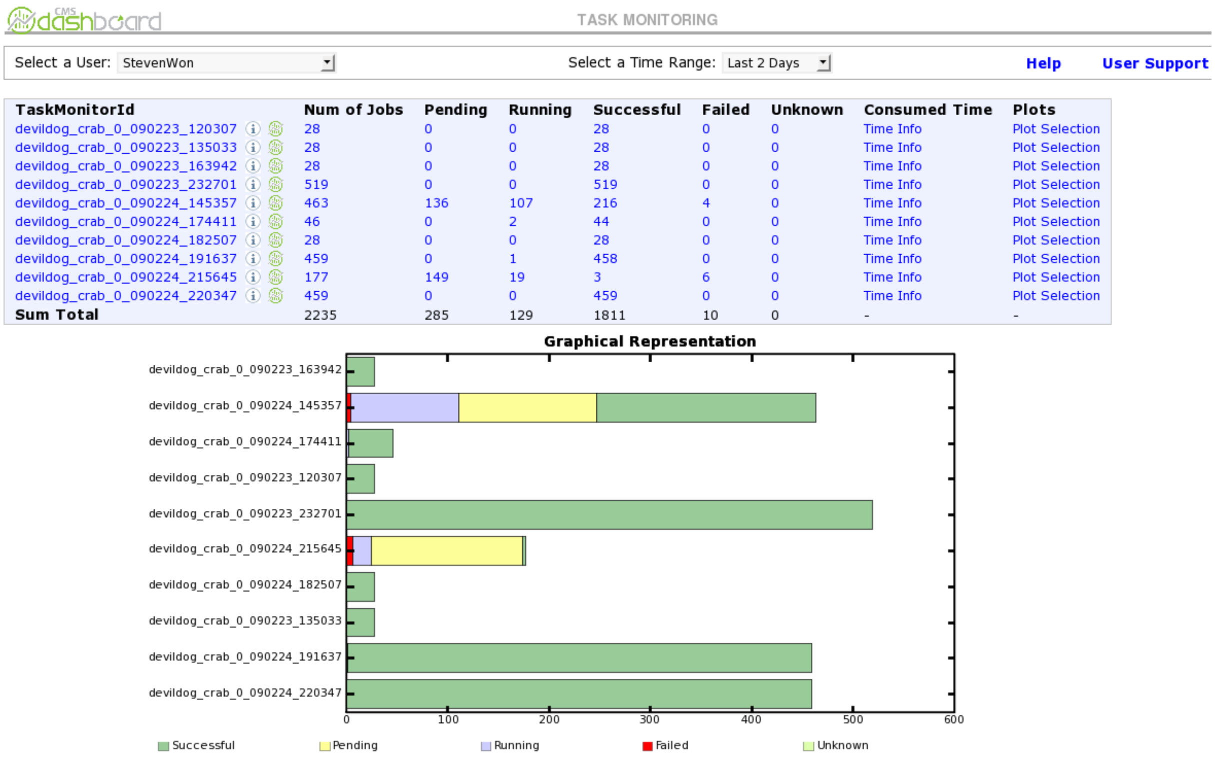
Figure 2. Job Information Gathering

The job submission tools of the CMS experiment and the job wrappers generated by these tools are instrumented to report meta-information about a user's tasks and the progress of a user's job to the MonALISA server. The Dashboard then presents all this information in a coherent way, as if all of it came from one source [7].

3. Monitoring Features

Task Monitoring provides monitoring functionality regardless of the job submission method or the middleware flavour and it works transparently across various Grid infrastructures which is the reason why it is so heavily used by many analysis users [8][9]. It is easy to understand how it works and how to navigate throughout the tool. It is clean and intuitive in layout and it contains no unnecessary information (Figure 3).

The tool has very low latency and it updates in 'real time' from the worker nodes where the jobs are running. A snapshot of the user interface can be seen in Figure 3.



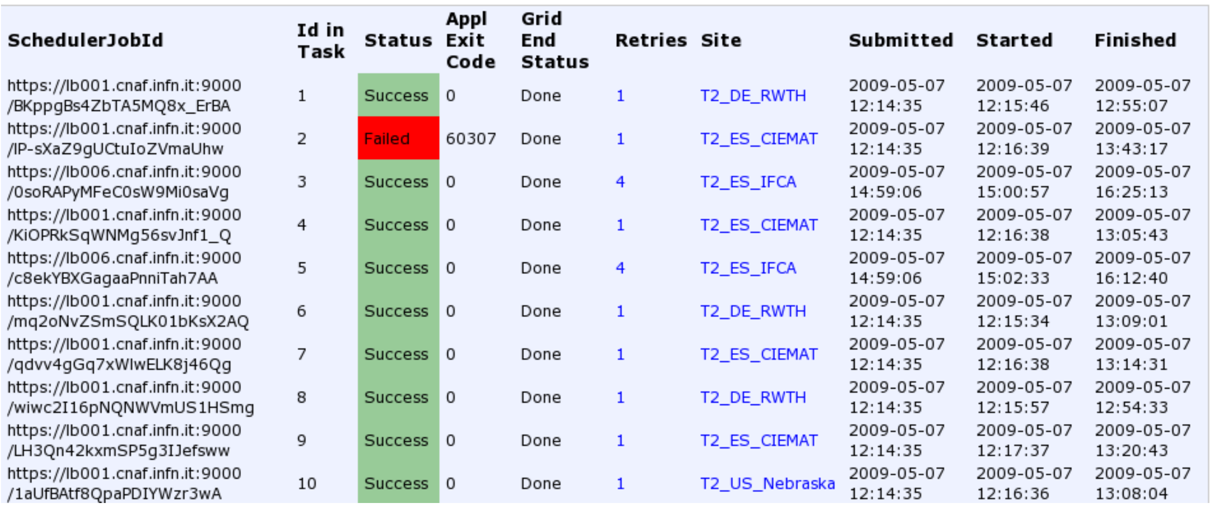
Select a User: StevenWon

Select a Time Range: Last 2 Days

TaskMonitorId	Num of Jobs	Pending	Running	Successful	Failed	Unknown	Consumed Time	Plots
devildog_crab_0_090223_120307	28	0	0	28	0	0	Time Info	Plot Selection
devildog_crab_0_090223_135033	28	0	0	28	0	0	Time Info	Plot Selection
devildog_crab_0_090223_163942	28	0	0	28	0	0	Time Info	Plot Selection
devildog_crab_0_090223_232701	519	0	0	519	0	0	Time Info	Plot Selection
devildog_crab_0_090224_145357	463	136	107	216	4	0	Time Info	Plot Selection
devildog_crab_0_090224_174411	46	0	2	44	0	0	Time Info	Plot Selection
devildog_crab_0_090224_182507	28	0	0	28	0	0	Time Info	Plot Selection
devildog_crab_0_090224_191637	459	0	1	458	0	0	Time Info	Plot Selection
devildog_crab_0_090224_215645	177	149	19	3	6	0	Time Info	Plot Selection
devildog_crab_0_090224_220347	459	0	0	459	0	0	Time Info	Plot Selection
Sum Total	2235	285	129	1811	10	0	-	-

Figure 3. User Interface

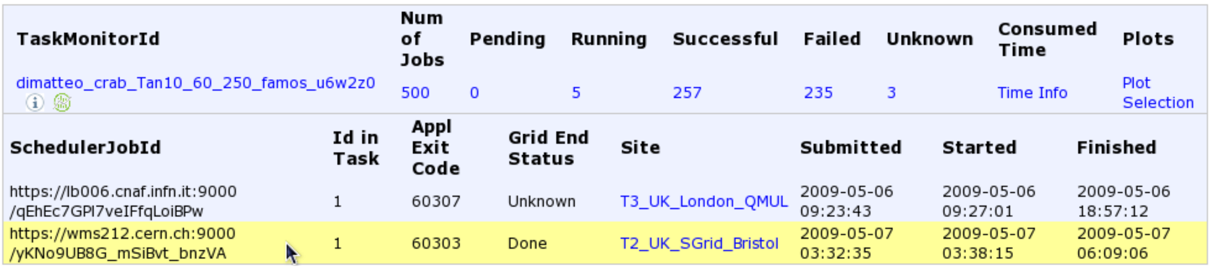
Clicking on the information link next to the name of the task provides meta-information such as input dataset, version of the software used by the task and of the submission tool and the task creation time. Clicking on the number of jobs corresponding to a given status provides a detailed information of all the jobs of a selected category (Figure 4).



SchedulerJobId	Id in Task	Status	Appl Exit Code	Grid End Status	Retries	Site	Submitted	Started	Finished
https://lb001.cnaf.infn.it:9000 /BKppgBs4ZbTA5MQ8x_ErBA	1	Success	0	Done	1	T2_DE_RWTH	2009-05-07 12:14:35	2009-05-07 12:15:46	2009-05-07 12:55:07
https://lb001.cnaf.infn.it:9000 /lP-sXaZ9gUCtuIoZVmaUhw	2	Failed	60307	Done	1	T2_ES_CIEMAT	2009-05-07 12:14:35	2009-05-07 12:16:39	2009-05-07 13:43:17
https://lb006.cnaf.infn.it:9000 /0soRAPyMFeC0sW9Mi0saVg	3	Success	0	Done	4	T2_ES_IFCA	2009-05-07 14:59:06	2009-05-07 15:00:57	2009-05-07 16:25:13
https://lb001.cnaf.infn.it:9000 /KiOPRkSqWNMg56svJnf1_Q	4	Success	0	Done	1	T2_ES_CIEMAT	2009-05-07 12:14:35	2009-05-07 12:16:38	2009-05-07 13:05:43
https://lb006.cnaf.infn.it:9000 /c8ekYBXGagaaPnniTah7AA	5	Success	0	Done	4	T2_ES_IFCA	2009-05-07 14:59:06	2009-05-07 15:02:33	2009-05-07 16:12:40
https://lb001.cnaf.infn.it:9000 /mq2oNvZSmSQLK01bKsX2AQ	6	Success	0	Done	1	T2_DE_RWTH	2009-05-07 12:14:35	2009-05-07 12:15:34	2009-05-07 13:09:01
https://lb001.cnaf.infn.it:9000 /qdvv4gGq7xWlwELK8j46Qg	7	Success	0	Done	1	T2_ES_CIEMAT	2009-05-07 12:14:35	2009-05-07 12:16:38	2009-05-07 13:14:31
https://lb001.cnaf.infn.it:9000 /wiwc2I16pNQNWVmUS1HSmg	8	Success	0	Done	1	T2_DE_RWTH	2009-05-07 12:14:35	2009-05-07 12:15:57	2009-05-07 12:54:33
https://lb001.cnaf.infn.it:9000 /LH3Qn42kxmSP5g3IJefsww	9	Success	0	Done	1	T2_ES_CIEMAT	2009-05-07 12:14:35	2009-05-07 12:17:37	2009-05-07 13:20:43
https://lb001.cnaf.infn.it:9000 /1aUfBAtf8QpaPDIYWzr3wA	10	Success	0	Done	1	T2_US_Nebraska	2009-05-07 12:14:35	2009-05-07 12:16:36	2009-05-07 13:08:04

Figure 4. Detailed Job Information

Clicking on any name on the 'Site' column opens the Site Status Board for the CMS Sites [10], providing a 24-hour status availability of the selected site allowing to identify any problematic site and blacklist it from resubmissions. Also, clicking on the *'Retries'* column provides a detailed re-submission history of every single job which can be very useful for debugging purposes. An example can be seen in Figure 5; the job produced an output to the Storage Element (SE) but the staging out finished with an error (exit code: 60307), thus, all following resubmissions had no chance to succeed, since the file was already created on the SE (exit code: 60303). Before any further resubmission, the output file generated by the previous attempt should be removed from the SE.



TaskMonitorId	Num of Jobs	Pending	Running	Successful	Failed	Unknown	Consumed Time	Plots
dimatteo_crab_Tan10_60_250_famos_u6w2z0	500	0	5	257	235	3	Time Info	Plot Selection

SchedulerJobId	Id in Task	Appl Exit Code	Grid End Status	Site	Submitted	Started	Finished
https://lb006.cnaf.infn.it:9000 /qEhEc7GPl7veIFfqLoiBPw	1	60307	Unknown	T3_UK_London_QMUL	2009-05-06 09:23:43	2009-05-06 09:27:01	2009-05-06 18:57:12
https://wms212.cern.ch:9000 /yKNo9UB8G_mSiBvt_bnzVA	1	60303	Done	T2_UK_SGrid_Bristol	2009-05-07 03:32:35	2009-05-07 03:38:15	2009-05-07 06:09:06

Figure 5. Detailed Resubmission Information

The application offers a wide variety of graphical plots that will visually assist the user to understand the status of the task. These plots show the distribution by site of successful, failed, running and pending jobs as well as for the processed events (Figure 6a) and they can help identify any problematic site and blacklist it from further resubmissions (Figure 6b). They also demonstrate the terminated jobs in terms of success or failure and over the time range that the task has been running (Figure 6c). In the case of failure, the distribution by reason is demonstrated, whether it be Grid-Aborted or Application-Failed jobs (Figure 6d).

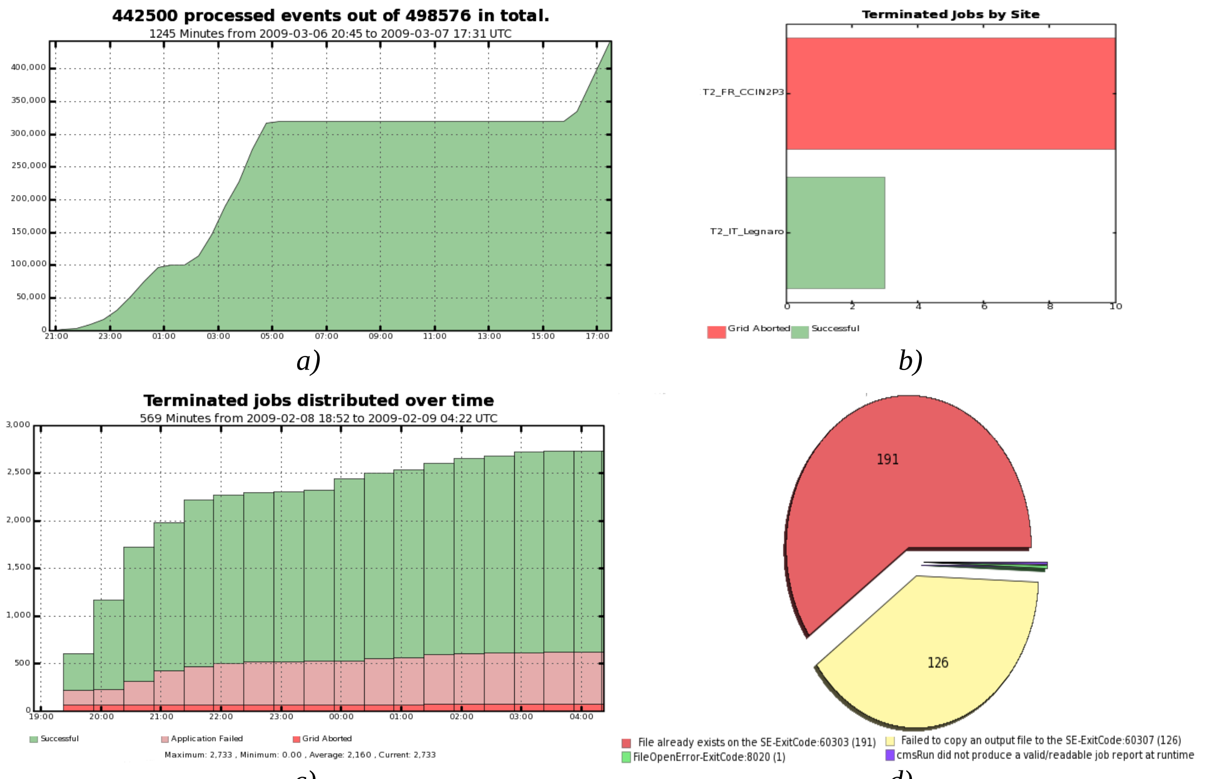


Figure 6. Graphical Plots: a) Processed Events over Time, b) Terminated Jobs by Site, c) Terminated Jobs over Time, d) Reason of Failure

Various kinds of consumed time plots are available such as the distribution of CPU and Wall Clock time spent for successful and failed jobs and the average efficiency distributed by site (Figure 7).

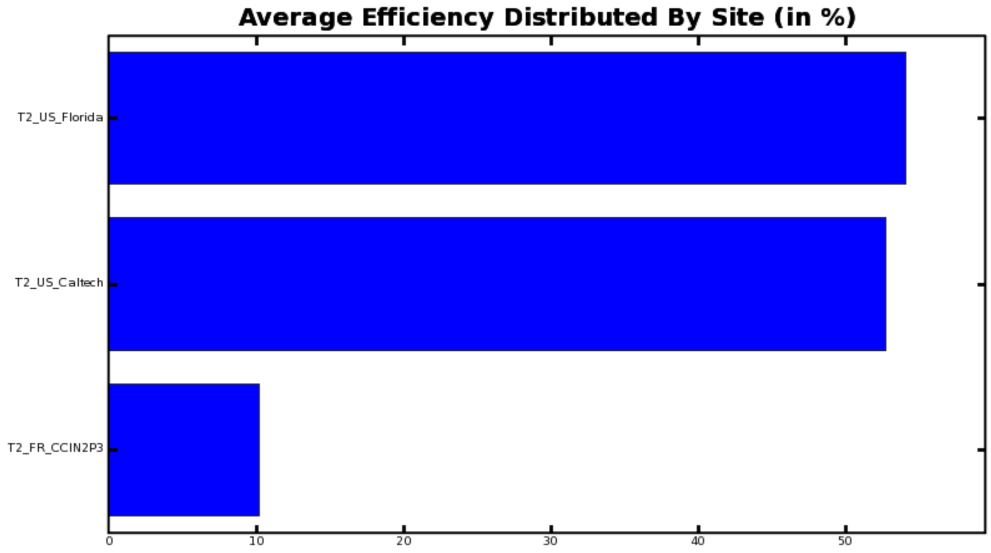
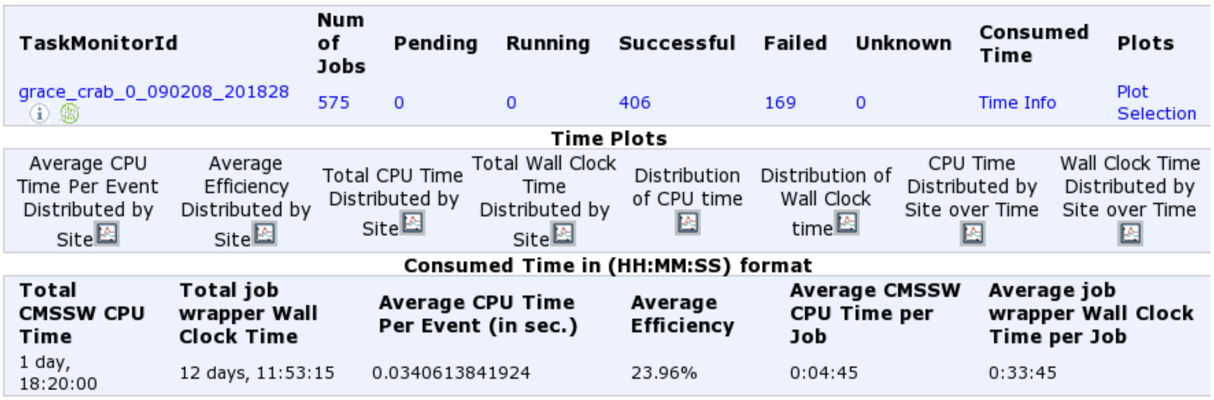


Figure 7. Efficiency Distributed by Site

These plots will help the user to see how the CPU time per event and efficiency can vary depending on the site that the jobs are running on. The user gets information regarding the time that has been consumed for a specific task or a given job:

- For any given task (Figure 8), the following information is available: the average efficiency of the task, the total and the average CMSSW CPU and job wrapper Wall Clock time usage and the average CPU time spent per event.
- At the job-level the user gets information about the efficiency of every single job separately.



TaskMonitorId	Num of Jobs	Pending	Running	Successful	Failed	Unknown	Consumed Time	Plots
grace_crab_0_090208_201828	575	0	0	406	169	0	Time Info	Plot Selection

Time Plots

Average CPU Time Per Event Distributed by Site	Average Efficiency Distributed by Site	Total CPU Time Distributed by Site	Total Wall Clock Time Distributed by Site	Distribution of CPU time	Distribution of Wall Clock time	CPU Time Distributed by Site over Time	Wall Clock Time Distributed by Site over Time

Consumed Time in (HH:MM:SS) format

Total CMSSW CPU Time	Total job wrapper Wall Clock Time	Average CPU Time Per Event (in sec.)	Average Efficiency	Average CMSSW CPU Time per Job	Average job wrapper Wall Clock Time per Job
1 day, 18:20:00	12 days, 11:53:15	0.0340613841924	23.96%	0:04:45	0:33:45

Figure 8. Consumed Time information for a selected task

4. Experience of the CMS User Community with Task Monitoring

In the CMS Community, the CMS Remote Analysis Builder (CRAB) [12] is used for the job submission. CRAB is a Python programme simplifying the process of creation and submission of CMS analysis jobs to the Grid environment. CRAB can be used in two ways: i) as a standalone application and ii) with a server. The standalone mode is suited for small tasks and it submits the jobs directly to the scheduler and these jobs are under the user's responsibility. In the server mode, suited for larger tasks, the jobs are prepared locally and then passed on to a dedicated CRAB Server which then interacts with the scheduler on behalf of the user and performs additional services such as automatic resubmissions and output retrieval.

Rather often, Task Monitoring discovers previously undetected problems with the CRAB Server or the Workload Management Systems (WMS). The Dashboard reports a job as 'finished' when the job finishes on the worker node but the job status updates by the Grid services can introduce some latency and they are quite often delayed due to a component of the CRAB Server or due to problems of the WMS or of the Logging and Bookkeeping system (LB) [13]. Thus, when the users see a big delay in status updates in CRAB compared to the status shown in Task Monitoring, they report the problem and after investigation either the CRAB Server is fixed or the faulty WMS is blacklisted.

We have performed a publicity campaign to bring awareness to the CMS User Community for the Task Monitoring application, collect feedback, assemble further requirements and identify weaknesses with the application. According to our web statistics [8][9], more than one hundred distinct analysis users are using Task Monitoring for their everyday work.

5. Known Issues and Future Work

The overall improvement of the Task Monitoring application strongly depends on the completeness of the job monitoring information in the Dashboard data repository. One of the known issues is the incomplete information regarding the Grid status of the jobs. Currently, only information from the Imperial College Real Time Monitoring (ICRTM) is used for this purpose. Unfortunately, only a fraction of the CMS jobs are monitored by the ICRTM [4]. Jobs submitted via Condor_G [11] or Condor-glideins escape it and not all Workload Management Systems (WMSs) are monitored by the ICRTM. There is a lot of development effort, driven by the Dashboard team, to improve this situation by instrumenting the Logging and Bookkeeping system (LB) for publishing job status changes information to the Messaging System for the Grid (MSG). Then, this information will be available to all possible interested clients including the Dashboard. Condor_G is also being instrumented to report job status changes to the MSG.

Currently, a weak point of the application is the failure diagnostics for both Grid and application failures due to the incomplete information we receive regarding the reason of the failure. The ideal situation would be to reach to a point where the user shouldn’t have to open the log file to search for what went wrong. The user should get everything from the monitoring tool. A development effort is ongoing to improve the failure diagnostics reported to the Dashboard from the job wrapper.

We plan to develop a search functionality that will allow the user to search for a specific task or job using a search pattern. Finally, we are working on developing a script to automatically generate a set of commands for processing a set of jobs, such as resubmissions, killings, getting logging info and retrieving the output. These commands will be both in CRAB format and in various underlying middleware format. The user could select a subset or all of the failed jobs for a given task and be able to download and run a single command file that will do the resubmission automatically. This feature is needed when CRAB's task directory is not available for an unknown reason and the user can not use the CRAB UI in order to manipulate the jobs of the task.

6. Conclusion

While the existing monitoring tools are coupled to a specific middleware, Task Monitoring provides monitoring functionality regardless of the job submission method or the middleware platform offering a complete and detailed view of the user's tasks including failure diagnostics, processing efficiency and resubmission history.

The monitoring tool has become very popular among the CMS users. According to our web statistics, more than one hundred distinct analysis users are using it for their everyday work. Close collaboration with several CMS users resulted in the tool being focused on their exact monitoring needs.

Acknowledgments
The authors would like to thank all the participating users of the CMS Community for their feedback and support. This work was funded by EGEE. EGEE is a project funded by the European Union under contract INFSO-RI-031688.

References

[1] Experiment Dashboard, http://dashboard.cern.ch
[2] ARDA Dashboard Developer's Guide, http://dashb-build.cern.ch/build/nightly/doc/guides/common/html/dev/index.html
[3] J. Andreeva et al. 'Experiment Dashboard: the monitoring system for the LHC experiments'. In GMW'07: Proceedings of the 2007 workshop on Grid monitoring, ACM
[4] J. Andreeva et al. 'New Job Monitoring Strategy on the WLCG Scope'. In CHEP'09: Proceedings of the 2009 International Conference on Computing in High Energy and Nuclear Physics, IOP
[5] Imperial College Real Time Monitoring (ICRTM), http://gridportal.hep.ph.ic.ac.uk/rtm/
[6] Monitoring Agents Using a Large Integrated Services Architecture (MonALISA), http://monalisa.cern.ch/monalisa.html
[7] P. Saiz et al. 'Grid Reliability'. In CHEP'07: Proceedings of the 2007 International Conference on Computing in High Energy and Nuclear Physics, IOP
[8] Dashboard Production Server Statistics, http://lxarda18.cern.ch/awstats/awstats.pl?config=lxarda18.cern.ch
[9] Dashboard Application Usage Statistics, http://lxarda18.cern.ch/usage.html
[10] Dashboard Site Status for the CMS Sites, http://dashb-ssb.cern.ch/ssb.html
[11] Condor-G, http://www.cs.wisc.edu/condor/condorg/
[12] CMS Remote Analysis Builder (CRAB), https://twiki.cern.ch/twiki/bin/view/CMS/SWGuideCrab
[13] Logging and Bookkeeping System, http://egee.cesnet.cz/cs/JRA1/LB/