Experience in Grid Site Testing for ATLAS, CMS and LHCb with HammerCloud

Johannes Elmsheuser[α], Ramón Medrano Llamas[β], Federica Legger[α], Andrea Sciabà[β], Gianfranco Sciacca[γ], Mario Úbeda García[β], Daniel van der Ster[β]

α: Ludwig-Maximilians-Universität München, Munich, Germany
β: CERN, Geneva, Switzerland
γ: Universität Bern, Switzerland

E-mail: daniel.vanderster@cern.ch

Abstract. Frequent validation and stress testing of the network, storage and CPU resources of a grid site is essential to achieve high performance and reliability. HammerCloud was previously introduced with the goals of enabling VO- and site-administrators to run such tests in an automated or on-demand manner. The ATLAS, CMS and LHCb experiments have all developed VO plugins for the service and have successfully integrated it into their grid operations infrastructures.

This work will present the experience in running HammerCloud at full scale for more than 3 years and present solutions to the scalability issues faced by the service. First, we will show the particular challenges faced when integrating with CMS and LHCb offline computing, including customized dashboards to show site validation reports for the VOs and a new API to tightly integrate with the LHCbDIRAC Resource Status System. Next, a study of the automatic site exclusion component used by ATLAS will be presented along with results for tuning the exclusion policies. A study of the historical test results for ATLAS, CMS and LHCb will be presented, including comparisons between the experiments' grid availabilities and a search for site-based or temporal failure correlations. Finally, we will look to future plans that will allow users to gain new insights into the test results; these include developments to allow increased testing concurrency, increased scale in the number of metrics recorded per test job (up to hundreds), and increased scale in the historical job information (up to many millions of jobs per VO).

1. Introduction

HammerCloud was previously introduced [1] as a solution to stress test the Worldwide LHC Computing Grid sites. The goal of the service is to enable system administrators and grid operators to easily measure the sites' performances, and thereby to find and eliminate bottlenecks in the application, storage or network infrastructures. In the years just before and after data taking started at the LHC, HammerCloud was heavily used for such stress tests, and was also centrally responsible for a series of global tests to measure the worldwide throughput of the grid for physics data analysis.

In 2011 and 2012, however, the main usage of HammerCloud changed. During the day-to-day and continuous operations of a grid site there is often little room or time available to carry out large scale stress tests. However, while the number of stress tests has decreased, the usage

of HammerCloud as a continuous validation service has increased dramatically. Continuous validation is an important concept which can contribute significantly to the efficiency and reliability of the WLCG. In its basic form, HammerCloud sends a stream of repeated short full-workload test jobs to all of the grid sites. The outcomes of the jobs are recorded and, depending on successes or failures, HammerCloud can trigger actions including updating status visualizations, emailing operators, and whitelisting/blacklisting sites.

Today, three LHC VOs include HammerCloud in their grid operations as a site validation service:

The ATLAS experiment runs a set of tests known as the Analysis Functional Test and Production Functional Test. These tests run continually at all ATLAS grid sites and when a sites fails the jobs the site can be automatically excluded from the workload management system.

The CMS experiment runs HammerCloud as a Job Robot service and results are consulted by grid operators to identify faulty sites.

The LHCb experiment has integrated HammerCloud with its DIRAC Resource Status System; this system can trigger tests in HammerCloud when a site needs to be validated.

In this way, HammerCloud has filled a niche in the grid testing market. The requirements for a high-rate of tests, with each test having a high-complexity, while allowing high-customizability of the tests unfortunately exclude popular services such as Nagios from being able to perform this service effectively.

In the remainder of this paper we will describe the design and operations of this multi-VO service (in section 2) and highlight the recent experiences of the ATLAS, CMS, and LHCb experiments (in section 3). Next, we perform an error analysis to find common problems between the VOs in section 4, and finally, we conclude in section 5.

2. The HammerCloud Service

HammerCloud is a multi-VO service hosted centrally by CERN IT [2] and operated in collaboration with the ATLAS, CMS, and LHCb grid operations teams. With its current roles (as detailed in section 3) it has a relatively high criticality. Though it is not a core service like the data or workload management services, it is nonetheless important since a prolonged outage of HammerCloud would allow faults in grid sites to go undetected and thereby the job error rates would increase. As such, the operations of the HammerCloud service have been tuned to deliver a high quality of service without requiring intensive operations manpower.

2.1. Service Architecture and Requirements

Architecturally, the core HammerCloud software stack is depicted in figure 1. HammerCloud is a python application written using Ganga [3] to submit and monitor the test jobs, and Django as a database access layer and to present results to the users. Ganga is used as the backend job management tool since it has a convenient set of modules to access the experiment middleware stacks (namely PanDA for ATLAS, CRAB for CMS, and DIRAC for LHCb), and also since it provides helpful functionality to submit, clone, and resubmit a series of identical jobs while monitoring their state in a background thread.

The testing demands of the HammerCloud users are increasingly large: the ATLAS and CMS functional tests each execute tens of thousands of jobs per day. The construction, submission, tracking, and occasional canceling of these jobs is a resource intensive activity. The core HammerCloud submission algorithm is as follows:

(i) Construct a set of N test jobs matching the test template requirements. Normally this involves first querying the experiment's data management service for input data files

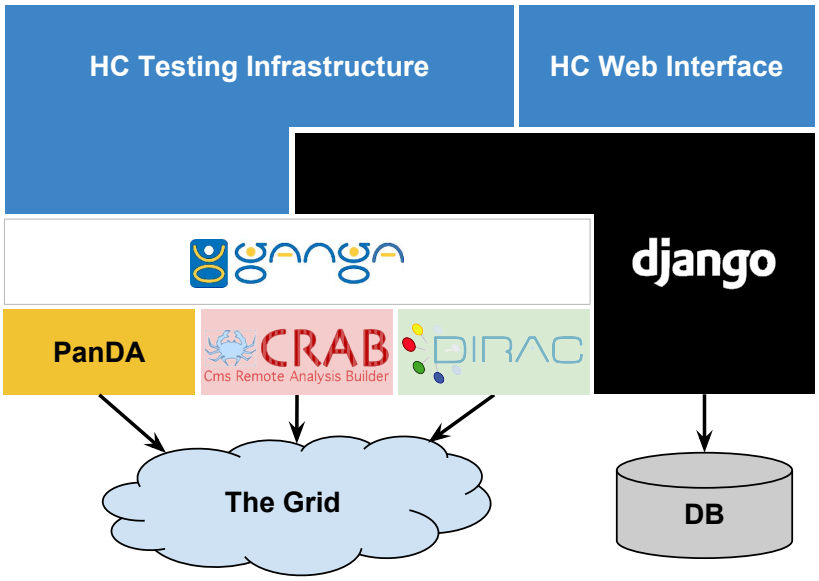

Figure 1. High level picture of the HammerCloud architecture.

available at each grid site, and second building a set of Ganga job objects (normally one for each tested site).

(ii) Submit each of the N test jobs to the experiment's workload management system.

(iii) Using the Ganga background monitoring thread, poll the status of each of the submitted and running jobs in the workload management system.

(iv) When a job has completed (whether as a success or failure), HammerCloud clones the job object, optionally randomly selects a new input data file, and resubmits it anew to the workload management system.

(v) HammerCloud loops around steps (iii) and (iv) until the test has expired (normally after 24 to 48 hours).

At any given time, HammerCloud is monitoring the status of around 1000-2000 jobs for the functional testing. When the service is used for on-demand stress testing at just a few sites, this number can easily double or triple.

2.2. Service and Operational Design

The heavy usage of HammerCloud has led to a service design which can scale to these increasing demands. At the core of HammerCloud is a set of MySQL databases which record the test configurations and results for each VO. Previously managed by the HammerCloud team, the service has recently been changed to take advantage of the new Database-on-Demand service [4] provided by CERN IT. DB-on-Demand provides a simple web interface to request and manage database instances (currently only MySQL) running on virtualized hardware. This has led to a decrease in operations effort (needed to backup and maintain the databases) while also providing room for scalability (DB-on-Demand allows growth through increased VM sizes and numbers).

Submission and management of the test jobs themselves is handled by a series of submission nodes. Each VO typically runs many tests in parallel (up to 10-20 each), and these tests can be load balanced across many machines. Different VO workload management systems required a

varied level of client-side resources to submit and manage the test jobs. For example, ATLAS and CMS tests that require jobs be submitted to the Glite WMS seem to generate a high I/O load on the HammerCloud submission nodes. By contrast, the workload management systems featuring a simple Python or RESTful interface (such as the CRAB server and PanDA) require fewer resources to submit jobs. Since the load generated by a test cannot be known before it runs, HammerCloud continually measures the Linux CPU load average of each of the submission nodes and balances the tests accordingly.

The web frontend of HammerCloud uses a shared web-server across all of the VOs. Designed as a Django web application, the interface is coded with generic models and views (for example for HammerCloud administration, to view tests and browse results), but also allows the possibility of customized views for specific VO applications. The optimization of Django views of the large HammerCloud database is non-trivial; the abstract model layers in Django make it relatively simple to naively generate huge loads on the database. In one case, while optimizing the main test index page (e.g. `http://hammercloud.cern.ch/hc/app/atlas/`) we found that to generate a list of around 20 tests Django was making more than 1000 queries to the database. Solving this problem required a careful restructuring of the code to provide query hints and allow reuse of cached data objects.

To improve the operations procedures, we have developed a number of sensors to evaluate the health of the service and notify the operators. The HammerCloud web service is monitored by Service Level Status [5] which sends alarms when the server (or database) is down. More importantly, the status of the individual tests is monitored pro-actively. The tests rely on several components which are occasionally unreliable – querying the data and workload management systems can be glitchy from time to time. To detect the cases where the job submission components fail, we have developed health scripts which continually confirm the state of the tests; namely, that the number of running tests is correct, that the required Ganga processes are running, and that all of the test log files are growing. In the cases were a test has been found to have failed, it is automatically restarted. Generally, the operations of HammerCloud is hands-off; when any of the database or submission nodes crash the affected tests are restarted automatically by the health sensors.

Finally, while HammerCloud can be thought of as a typical IT service such as web, database, or grid services, in practice the overall HammerCloud service provides a significant *human* component which cannot go unstated. HammerCloud is operated both by a small core team at CERN IT (responsible for core operations and scalability) and also a team of VO grid operators. In particular, each VO dedicates expert resources to maintain up-to-date test templates, define meaningful testing strategies, and provide support to the sites to craft benchmarking stress-tests and help troubleshoot errors. The success of HammerCloud relies on these VO experts as much as the core operations support.

3. Experiences

Though it was initiated as an ATLAS-specific testing framework for distributed analysis, HammerCloud has evolved into a multi-VO service with multiple novel applications. We detail the varied VO use-cases here.

3.1. ATLAS

The ATLAS experiment originally conceived of HammerCloud as a stress testing service to measure the grid performance before the LHC started data taking. Since then, the experiment's usage of the service has grown to include:

- On-demand stress testing to measure application performance under different site and application configurations.

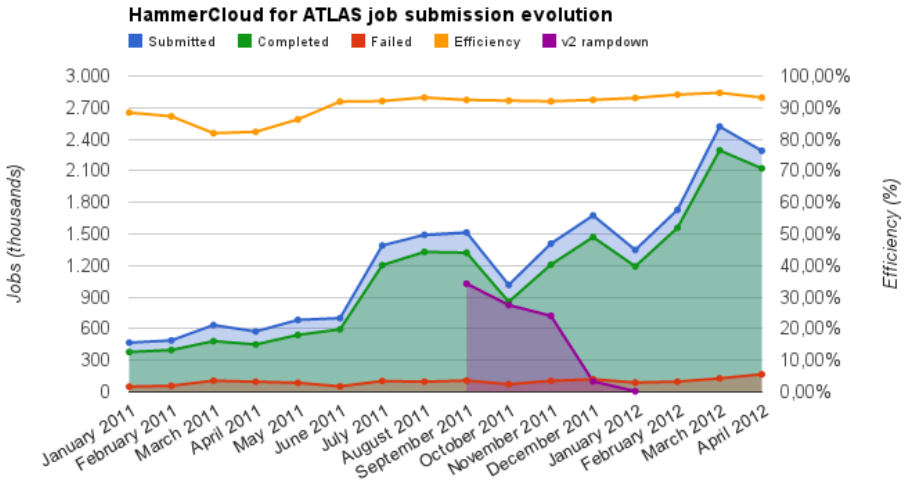

Figure 2. Test job activity for ATLAS.

- Continuous validation of the PanDA analysis queues in the so-called *Analysis Functional Test* (AFT) and PanDA production queues in the *Production Functional Test* (PFT).
- Automatic exclusion (i.e. blacklisting) of sites which fail the AFT and PFT and their subsequent re-inclusion after they have succeeded the test jobs. More details about the AFT and PFT testing can be found in [6].
- Global testing of the PanDA Pilot [7] release candidates.
- I/O performance tests of the ROOT analysis framework. More details are included below.
- Cloud computing tests. More details are available in [8].
- Global nightly build testing of the *Athena* analysis framework.

All tolled, these use-cases combine for an increasing number of test jobs sent per month. The time evolution of the service is depicted in figure 2; in recent months the ATLAS instance has sent more that 2 million test jobs per month. While this sounds like a large resource commitment, it is notable that each of these test jobs is very short (typically less than 5 minutes) so they accumulate large volumes without consuming a large number of CPUs at the sites.

One unfortunate side-effect of these large number of jobs has been the large load placed on the ATLAS distributed data management system (DDM). Since each analysis job necessarily produces a unique output dataset, HammerCloud was able to generate huge numbers of user output datasets; in some months the HammerCloud AFT and PFT created around one third of all user output datasets. Following this discovery, PanDA and HammerCloud were modified to exceptionally re-use the output datasets. Presently the rate of output dataset creation is less than 100 times less than before this change.

3.1.1. ROOT I/O Testing ATLAS analysis relies heavily on the performance of its ROOT-based data formats. In [9] the experiment has built a task-specific data mining tool around HammerCloud which is being used to measure and optimize the performance of ROOT D3PD analysis. In these tests, various parameters are being evaluated, with an emphasis on understanding the behavior of enabling input data caching in ROOT and on measuring the performance of WAN vs. LAN input data access. HammerCloud is configured to run a set of ROOT-based functional tests at grid sites world wide including those with varied storage

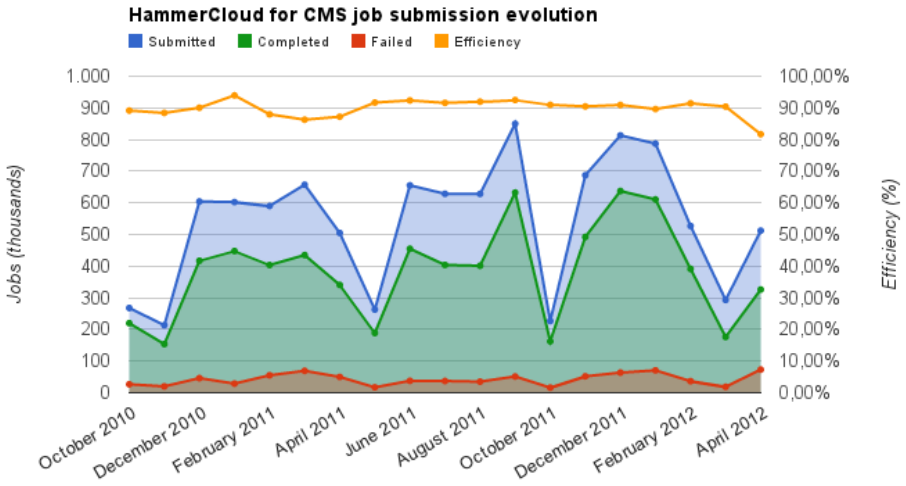

Figure 3. Test job activity for CMS.

element softwares (e.g. dCache, DPM, Xrootd, EOS). More details about the results of the tests are available in these proceedings [9].

3.2. CMS

Until April 2012, the CMS collaboration was using a functional testing engine very similar to GangaRobot and HammerCloud, called Job Robot. The success rate of the Job Robot jobs was one of the quality metrics used to calculate the site readiness, a single-valued estimator used to classify a site as "ready" or "not ready" to accept CMS jobs [10]. The fraction of days in ready status over the latest two weeks is used inside the CMS collaboration to rank sites.

In order to fully replace the Job Robot with HammerCloud functional testing, more views had to be added to the CMS HammerCloud application, such as the Job Errors table which presents the number of grid and application errors at each site. This allows a more detailed classification of the site readiness and helps the debugging of site errors using the logs obtained from the jobs. This view shows CRAB error codes that are parsed using a custom tool that can check the *LoggingInfo* and the *stdout* files when the job finishes. It detects grid errors and application exit codes, helping site administrators to debug the problems that arise.

On the operational side, and as shown in figure 3, the operation of HammerCloud for CMS has been uniform since 2010, although there were two exceptional problems in May 2011 and October 2011. The first problem happened during development, and the second one was a HammerCloud problem related to a Django race condition following an incorrect transaction isolation configuration in the database.

For CMS, about 700.000 jobs are submitted each month, resulting from 9 functional tests that are configured regionally. These functional tests are storing 85 performance metrics per job, regarding for example the I/O performance, number of events processed and CPU efficiency. Globally, the CMS site efficiency is very good, having a success rate of 90% in average for all the 84 sites tested.

One difference between the CMS and ATLAS instances is relative job priorities in the experiments' workload management systems. In the case of ATLAS, all jobs submitted through PanDA are submitted with the maximum priority to ensure that at least one job is running in a site at a given time; this allows HammerCloud to use less intensive resubmission parameters. In the case of CMS, jobs are submitted with the default priority, therefore the behaviour of the

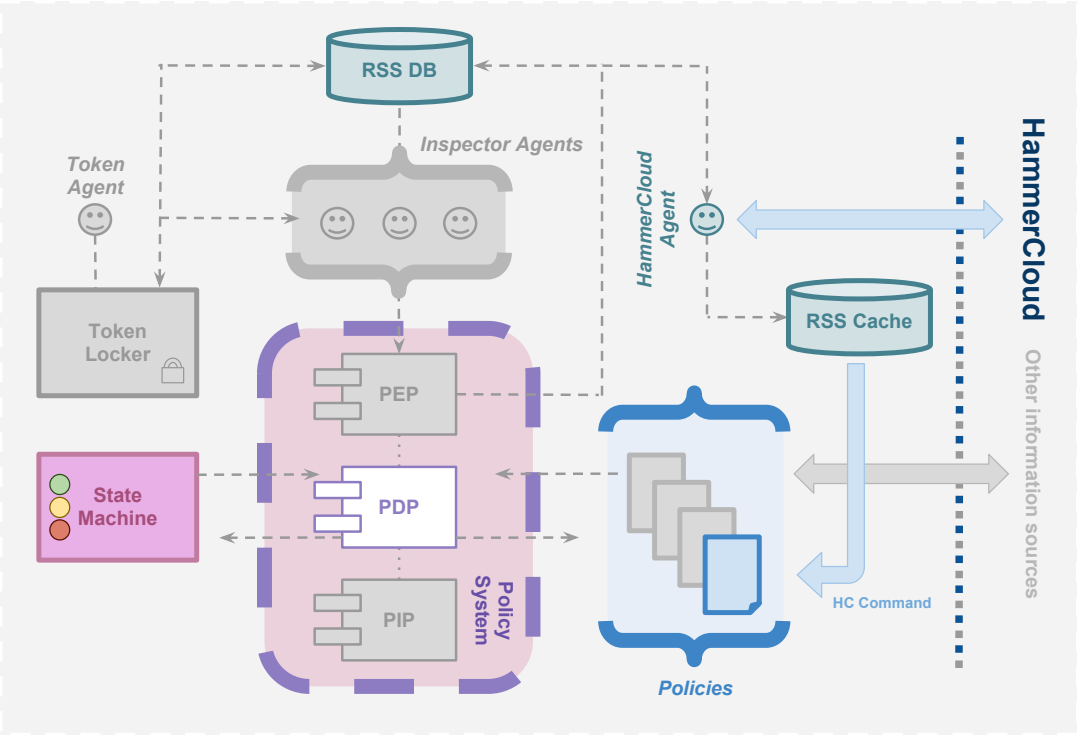

Figure 4. Integration of HammerCloud with the LHCbDIRAC Resource Status System.

site is less predictable and depends on the site occupancy at the submission time. To limit this effect a bulk of around 30 jobs is submitted and the test length for regions with highly used sites has been increased to two days, reducing also the leftover jobs that are killed at the end of a test.

3.3. LHCb

The LHCb instance of HammerCloud runs on a dedicated VO-box and is fully integrated with the Resource Status System (RSS) of DIRAC through its extension, LHCbDIRAC. The RSS is the central point for resource status information of the LHCb grid resources. It also acts as a monitoring tool with automated procedures to reshape the grid ontology used by LHCbDIRAC when needed (for example, during a downtime or related to a throttled site).

The LHCb instance runs without the supervision of any human operator; it is fully managed by an agent in LHCbDIRAC named HammerCloudAgent (note that a DIRAC agent is a stateless light-weight component, comparable to a cron-job). This agent has the knowledge of the LHCbDIRAC needs, and sets the HammerCloud service accordingly.

Figure 4 describes the HammerCloud integration with the new RSS; this work is currently in development. Details about the Resource Status System are available elsewhere in these proceedings [11] but here we highlight the components relevant for HammerCloud (highlighted in color in the figure).

First, the HammerCloud agent reads the resources database to discover site elements in status *probing* (see figure 5 for the complete element state machine). *Probing* resources are those which have been previously banned but the reason why they were *banned* has disappeared; thus, they move one step up to *probing*. If the probing checks are enabled in the system, no resource will

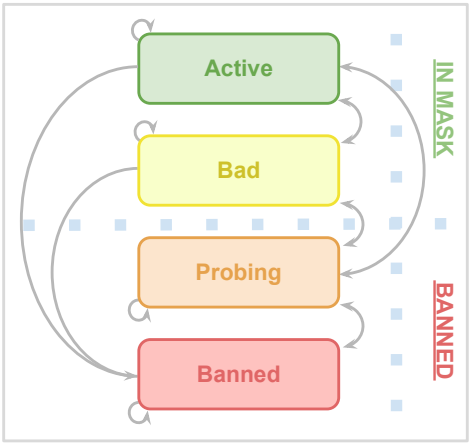

Figure 5. State machine of the LHCbDIRAC resource elements.

be able to leave this state until all of the checks are satisfactory. One of these probing checks is a HammerCloud stress-test.

When triggered, the HammerCloud agent launches a test of the *probing* resource; when done, it updates the RSS Cache with its results. Once the results are in the cache, the policy system evaluates the output of the test making use of the HammerCloud policy (designed to analyze the test output) and propose a new status. If the test is not satisfactory, it will propose *probing*; otherwise, it will propose *bad* or *active*.

3.3.1. Future Plans The usage of HammerCloud for continuous functional testing (like ATLAS and CMS) is not foreseen in the mid-term plan, as its functionality is already covered by the usage of the RSS combined with stress testing on demand. There are certainly use cases that could further exploit HammerCloud functionalities, but these need to be reviewed first, especially in relation to the metrics reported by HammerCloud that could be used to perform a statistical analysis of the failures.

4. Error Analysis and Search for Redundant Tests

4.1. Motivation

The HammerCloud job result databases present a unique opportunity to look for cross-VO correlations in the grid site faults. There are examples of sites which share services between VOs; in cases where these shared services are faulty, it is conceivable that both VOs would lose the ability to use that site. At CERN all VOs share, for example, the network and some parts of the AFS home directory service, while other services are not shared, such as the grid computing elements and storage servers. In the case when CERN's AFS is globally broken, then it is highly likely that no VO would be able to run jobs there.

If, by examining the HammerCloud error records, we can observe a large correlation between VOs (meaning that when one VO observes problems then the other VO is also likely to see it), then it could be argued that the test results should be shared between VOs. Such sharing of these results could then lead to a decreased number of tests required, thereby improving the testing efficiency. Take for example the ATLAS AFT and PFT tests; historically, ATLAS ran the AFT tests with automatic exclusion of the faulty sites but relied on grid operations shifters to look for errors in the production queues[1] and manually blacklist these sites. After some considerable

[1] PanDA splits each grid site into an *analysis* and *production* queue to separate the chaotic user jobs from the

effort, simultaneous testing of production queues was added to HammerCloud in the PFT tests. Now that the AFT and PFT have run in parallel for multiple months, we have observed that in many cases, production and analysis queues are auto-excluded at the same moment. This indicates that perhaps some parts of the AFT or PFT are redundant, and there should be some effort made to reduce the number of tests run to decrease the resource consumption.

4.2. Method

In this section we look for similar behaviors between VOs. We began by extracting records for February to April 2012 from the ATLAS and CMS results databases. We then counted the daily failure rates (number of failed jobs / number of total jobs) for each site, and recorded the results in a table. Finally, we computed the correlation between all sites, where the correlation value of two variables X and Y is computed as:

$$\frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{(n-1)s_X s_Y} \tag{1}$$

where $\bar{x}$ is the mean value of x, s_X is the standard deviation of x and n is the number of measurements. For each site we have highlighted the related sites having the largest positive correlation in daily failure rates.

4.3. Results for ATLAS Correlations

First, we present the results of correlations between ATLAS sites to confirm the aforementioned observation that AFT and PFT auto-exclusion events seem to happen at the same time. In table 1 we highlight several sites that have a very high correlation between the error rates seen in their production and analysis queues. At this collection of sites (having a correlation coefficient close to 1.0), it is possible that either the AFT or the PFT is redundant; if ATLAS was concerned about resources dedicated to testing it may be feasible to stop one stream of tests and treat the production and analysis queues as a single entity for auto-exclusion purposes.

Some sets of PanDA queues are interfaces to the same computing clusters; we expect these queues to be highly correlated. Indeed, the French queues ANALY_IN2P3-CC, ANALY_IN2P3-CC-T2, and ANALY_IN2P3-CC-T2_RD are all inter-correlated between 0.99 and 1.0. Once again, independent testing of these queues is shown to be redundant.

We also observed some highly correlated queues hosted at separate sites. For instance, the two Canadian queues ANALY_ALBERTA-WG1 and ANALY_SFU_bugaboo (located in Edmonton and Vancouver) are correlated with a coefficient equal to 0.97. The Spanish queue ANALY_IFAE is correlated to the state of its Tier 1 centre ANALY_PIC with 0.95.

4.4. Results for ATLAS-CMS correlations

Next, we looked at the correlations between the daily error rates seen by ATLAS and CMS queues hosted at the same physical location. These are listed in table 2. In general, the results show that there is very little correlation in the error rates between ATLAS and CMS. Some sites, such as PIC and IN2P3-CC show a correlation compatible with zero. Others, such as CERN, RAL and CSCS demonstrate some small correlations between the different VO tests, but further investigations would be required to understand the nature of those relationships.

5. Future Plans and Conclusion

This paper has presented the successful development, operations, and evolution of HammerCloud during 2011 and 2012. More than 8.300.000 ATLAS jobs and more than 8.000.000 CMS jobs have

centrally controlled data processing, respectively. For example, PanDA jobs at CERN can run under either of two queues, ANALY_CERN or CERN-PROD.

Table 1. Example correlations between ATLAS analysis and production queues

Production Queue	Analysis Queue	Correlation Coeff.
AGLT2	ANALY_AGLT2	0.93
CPPM	ANALY_CPPM	0.96
CYFRONET-LCG2	ANALY_CYF	0.94
TUDresden-ZIH	ANALY_DRESDEN	0.95
UNI-FREIBURG	ANALY_FREIBURG	0.93
praguelcg2	ANALY_FZU	0.98
IFIC	ANALY_IFIC	0.96
INFN-LECCE	ANALY_INFN-LECCE	0.99
INFN-NAPOLI-ATLAS	ANALY_INFN-NAPOLI	0.97
UKI-NORTHGRID-LANCS-HEP	ANALY_LANCS	0.98
LIP-LISBON	ANALY_LIP-Lisbon	0.97
UKI-NORTHGRID-LIV-HEP	ANALY_LIV	0.997
NIKHEF-ELPROD	ANALY_NIKHEF-ELPROD	0.96
UKI-SOUTHGRID-OX-HEP	ANALY_OX	0.96
RAL-LCG2	ANALY_RAL	0.95
UKI-LT2-RHUL	ANALY_RHUL	0.94
RRC-KI	ANALY_RRC-KI	0.98
CA-SCINET-T2	ANALY_SCINET	0.99
SLACXRD	ANALY_SLAC	0.98
SWT2_CPB	ANALY_SWT2_CPB	0.91
TECHNION-HEP	ANALY_TECHNION-HEP	0.99
UKI-LT2-UCL-HEP	ANALY_UCL	0.99

Table 2. Example correlations between co-located ATLAS and CMS queues

ATLAS Queue	CMS Queue	Correlation Coeff.
ANALY_RAL	T1_UK_RAL	0.13
ANALY_CERN_XROOTD	T1_CH_CERN	0.21
ANALY_FZK	T1_DE_KIT	0.20
ANALY_PIC	T1_ES_PIC	0.0014
ANALY_IN2P3-CC	T2_FR_CCIN2P3	0.0075
ru-PNPI	T2_RU_PNPI	0.012
ANALY_CSCS	T2_CH_CSCS	0.52

been submitted only in 2012. These jobs are the result of more than 5400 ATLAS tests and 5700 CMS tests, and are used to validate around 130 ATLAS and 84 CMS grid sites concurrently. In 2012, all these tests have generated more than 2 billion raw metrics. While these figures provide a wealth of data to learn and analyze, they are the major scalability concern in the development and operations side.

Along with the general optimizations described in section 2.2, there are many improvements planned to more efficiently handle the load generated by the job submission and the data management, improve the quality and stability of the service, and fulfill the site requests:

(i) Reduce the memory used by the processes: each test is handled by a UNIX process that is driven by the GangaRobot framework for the job submission and monitoring. All the

data storage and database querying is handled by the Django O/RM, which compiles all the queries to specific SQL, handles data and query caching and manages sessions and connections to the database. Since these processes manage a lot of database objects, the memory usage is very high (about 1 GiB of proportional set per process), therefore the length of the test is limited (currently 24 hours in general) and the system memory is the limiting factor to run more test in a submit note. A reduction on the memory required by the Django O/RM would make possible the increase of the test length, reducing the overhead of the initial submission and test generation which would alleviate the load generated to the DDM mechanisms, like DQ2 in the case of ATLAS.

(ii) Simplification of the statistics engine: the current statistical engine provided by HammerCloud is very powerful, since it allows to combine almost every metric collected by the testing engine. The major drawback found by the users is that the interface and procedures to get statistics is very inefficient and difficult to understand, and also the quality of the plots generated can be improved. A simpler statistics engine will be implemented, which also would be used as a monitoring tool for sites and clouds.

(iii) An API to access raw data: another highly requested feature, that is very related with the previous is access to the raw metrics collected by HammerCloud to allow other tools, like Nagios probes, take advantage of the data being collected. The current JSON API will be extended and optimized to allow queries in such a way.

(iv) Isolation of architectural components: currently, the deployment of HammerCloud involves a key machine that runs a small database with Django authentication data, the Apache web server, the web cache and some of the ATLAS tests. Further deployment improvements will make these three elements run in separated machines, also the database will be moved to a high replication service ran by the CERN IT department.

(v) Improvement of deployment procedures: in relation with the previous item, the HammerCloud team is rather small in comparison with the volume of the service and the number of machines that manages. Going further in a well defined set of deployment procedures and mechanisms, like the distribution of HammerCloud in RPM packages or prebuilt virtual appliances mixed with an agile perspective of the software development will increase the quality of service provided.

In closing, HammerCloud has become the *de facto* tool for grid functional testing both in ATLAS and CMS and has a big impact on the monitoring operations in both VOs. LHCb has a novel approach to integrate HammerCloud with its workload management system; the practical application of this integration are highly anticipated. Further work is needed to improve the efficiency of the testing, following the correlation study presented in section 4. Indeed, the ATLAS automatic site exclusion is an example of the possible automations that can be achieved with a continuous functional testing and a machine learning engine with highly optimized policies.

References

[1] D. van der Ster et al. HammerCloud: A Stress Testing System for Distributed Analysis. 2011 J. Phys.: Conf. Ser. 331 072036 doi:10.1088/1742-6596/331/7/072036

[2] M. Girone et al. The "Common Solutions" Strategy of the Experiment Support group at CERN for the LHC Experiments. In these proceedings.

[3] M. Kenyon et al. Key developments of the Ganga task-management framework. In these proceedings.

[4] R. Gaspar Aparicio et al. The Database on Demand service. In these proceedings.

[5] F. Barreiro Megino et al. Service monitoring in the LHC experiments. In these proceedings.

[6] F. Legger et al. Improving ATLAS grid site reliability with functional tests using HammerCloud. In these proceedings.

[7] P. Nilsson et al. Recent Improvements in the ATLAS PanDA Pilot. In these proceedings.

[8] D. van der Ster et al. Exploiting Virtualization and Cloud Computing in ATLAS. In these proceedings.
[9] W. Bhimji et al. The ATLAS ROOT-based data formats: recent improvements and performance measurements. In these proceedings.
[10] J. Flix, J. Hernández and A. Sciabà. Monitoring the Readiness and Utilization of the Distributed CMS Computing Facilities. J. Phys.: Conf. Ser. 331. 072020.
[11] F. Stagni et al. Grid administration: towards an autonomic approach. In these proceedings.