HammerCloud: A Stress Testing System for Distributed Analysis

Daniel C. van der Ster¹, Johannes Elmsheuser², Mario Úbeda García¹, Massimo Paladin¹
1: CERN, Geneva, Switzerland
2: Ludwig-Maximilians-Universität München, Munich, Germany
E-mail: daniel.colin.vanderster@cern.ch

Abstract. Distributed analysis of LHC data is an I/O-intensive activity which places large demands on the internal network, storage, and local disks at remote computing facilities. Commissioning and maintaining a site to provide an efficient distributed analysis service is therefore a challenge which can be aided by tools to help evaluate a variety of infrastructure designs and configurations. HammerCloud is one such tool; it is a stress testing service which is used by central operations teams, regional coordinators, and local site admins to (a) submit arbitrary number of analysis jobs to a number of sites, (b) maintain at a steady-state a predefined number of jobs running at the sites under test, (c) produce web-based reports summarizing the efficiency and performance of the sites under test, and (d) present a web-interface for historical test results to both evaluate progress and compare sites. HammerCloud was built around the distributed analysis framework Ganga, exploiting its API for grid job management. HammerCloud has been employed by the ATLAS experiment for continuous testing of many sites worldwide, and also during large scale computing challenges such as STEP’09 and UAT’09, where the scale of the tests exceeded 10,000 concurrently running and 1,000,000 total jobs over multi-day periods. In addition, HammerCloud is being adopted by the CMS experiment; the plugin structure of HammerCloud allows the execution of CMS jobs using their official tool (CRAB).

1. Introduction
Present day high energy physics research relies on the availability of powerful computing resources which allow physicists to process vast quantities of collision data in a timely manner. The resources of the World-wide LHC Computing Grid (WLCG) have been deployed to meet this demand. Nevertheless, the optimization of existing WLCG computing sites and configuration of newly commissioned resources motivates tools which aid in the design and configuration of a grid site to ensure that its capabilities meet or exceed the requirements of the foreseen user applications.

HammerCloud is an automated site testing service with two main goals. First, it is a stress-testing tool which can deliver large numbers of real jobs to objectively aid in the commissioning of new sites and to evaluate changes to site configurations. Second, it is an end-to-end functional testing tool which periodically sends short jobs to continually validate the site and related grid services. In addition to delivering the test jobs to grid sites, HammerCloud provides a user-friendly web interface to test results. The service presents plots of job efficiency and many
performance metrics to enable quick comparisons to past configurations and between other sites.

This paper presents the current status of HammerCloud as it has been designed and deployed for three LHC experiments, namely ATLAS, CMS, and LHCb.

2. HammerCloud Design

The HammerCloud service (general architecture in Figure 1) is implemented as a python application using Ganga [1, 2] to submit jobs to relevant grid execution backends and features a Django-based web interface for both public testing results browsing and administrative operations such as creating and maintaining tests.

The basic concepts and terminology of HammerCloud are enumerated here:

Test: A test is the basic object describing the actions to be carried out by the HammerCloud backend. Each test is described by options including the analysis code to run, the input datasets to process, a list of sites to test, parameters describing how many jobs to send to each site, and a start time and end time.

Template: A test template is composed of a set of test options which can be reused from test to test. Templates do not have an associated start time or end time.

User: A HammerCloud user has the ability to configure and request a test at a particular start time. The test request can be approved or rejected by an HammerCloud operator or automatic approval logic.

Stress Templates and Tests: A template or test of category stress is used for on-demand testing of a set of grid sites with the same analysis code and list of input dataset. Generally, a user selects a stress template, adds start/end times and the sites to test, and then submits the request for approval.

Functional Test: A template of category functional, when marked active by an operator, is automatically-scheduled as functional tests in order to periodically validate a large set (or
grid sites with a short analysis job. Tests of this sort are useful for not only site validation but also testing new grid middleware (e.g. client or pilot software).

2.1. HammerCloud Backend Logic

The HammerCloud Backend Logic was designed to serve a few purposes:

- Execute tests at the scheduled times: this includes generating test jobs, submitting the jobs to the sites under test, monitoring the jobs when they run and complete, resubmitting further jobs to keep each site at the target workload during the test, and to kill leftover running jobs at the end of a test.

- Automatically schedule activated functional templates as needed.

- Compute summary statistics for tests: this includes plotting metric summaries and overall HammerCloud activity.

When a test is running at a given site, HammerCloud monitors the jobs in active states (submitted, running) to determine if more jobs should be submitted to the site. Each site \( i \) in a test has associated resubmission parameters \( \text{resubmit\_enabled} \) (here, \( E_i \)), \( \text{num\_datasets\_per\_bulk} \) (\( D_i \)), \( \text{min\_queue\_depth} \) (\( S_i \)), and \( \text{max\_running\_jobs} \) (\( R_i \)). When a test first starts, HammerCloud generates a job for each site with a configuration to process \( D_i \) input datasets; since each job can process partial datasets, \( D_i \) datasets may map to \( J_i \) jobs where \( J_i > D_i \). After the first set of jobs has been submitted to all sites, HammerCloud begins monitoring the states of the jobs with the following resubmission algorithm (written in python-based pseudocode). Note that experiment HammerCloud plugins may make minor changes to this resubmission algorithm.

```python
for site in sites:
    (s,r) = getNumSubmittedAndRunningJobs(site)
    if E[i] and s < S[i] and r < R[i]:
        submitNewBulkJob(site)
```

The effects of this algorithm on the number of running jobs at a site is depicted in Figure 2.

2.2. Django-based Frontend

A web interface has been implemented in using Django; this framework separates data models from business logic and interface views, thereby enabling a clean and efficient design of the web application. The HammerCloud web interface serves two use cases:

- An administrative interface for HammerCloud operators and users. HammerCloud operators use this to maintain test templates and other data types including sites, analysis files, and dataset lists. HammerCloud users make test requests via the administrative interface.

- A public results browser lists all scheduled, running, and completed tests; allows users to get summary statistics for each test, including metric plots and tables; and presents summary statistics for all HammerCloud tests, such as a historical plot of number of running jobs across all HammerCloud tests.

The home page for HammerCloud is presented in Figure 3. Here, the ATLAS instance of HammerCloud is shown. The home page presents the list of currently scheduled or running tests, separated into the stress and functional test categories.

When a user clicks a test, they are shown a test summary (example of a CMS summary shown in Figure 4). The test summary shows the overall job successes and some performance metric histograms, such as the event processing rate. Below, the sites involved in a test are listed along with their respective job statuses; users can easily click to view the individual test jobs directly in order to debug problems.
Some functional tests might be deemed critical by the experiment grid operators. The HammerCloud GangaRobot view (example in Figure 5) shows a summary chart displaying the success of each site when running these critical tests. More information about the GangaRobot view in HammerCloud can be found in [3].

Users can also view the results of the HammerCloud test jobs directly in the experiment monitoring system. For ATLAS, functional test jobs are labelled with...
Figure 5. The GangaRobot view of test results [3].

Figure 6. View in the Panda Monitor of HammerCloud FUNCTIONAL test jobs for ATLAS.

Figure 7. Number of running test jobs over the past year.

`processingType=gangarobot`: this way, the Panda [4] monitor displays the results of these specific jobs. An example of this view is in Figure 6.

3. Usage in LHC Computing

HammerCloud was initially designed and implemented in cooperation with the ATLAS experiment. The work began as manually submitted stress tests by multiple ATLAS users, and the results of the early tests motivated the development of an automated stress testing system that became HammerCloud. Automated job robots such as HammerCloud have been
demonstrated previously, such as the CMS Job Robot and the ATLAS GangaRobot predecessors.

Since fall 2008, HammerCloud has been used by ATLAS for around 300,000 CPU-days of
site testing. In almost 4000 tests, 10,000,000 jobs have been executed with an 87% overall
success rate at more than 100 ATLAS sites or queues. The service was heavily used during the
year before data taking in order to prepare sites for the high loads expected from real users.
This preparation was successful to help enable the high reliability of the ATLAS grid sites
that we observe today. Presently, ATLAS sites continue to use HammerCloud to evaluate new
configurations, and the functional tests are used to automatically exclude malfunctioning sites
from the Panda [4] job brokerage system. Many recent examples of HammerCloud usage can be
found in these proceedings [5, 6, 7, 8, 9, 10, 11, 12, 13].

Following the ATLAS success with HammerCloud, the CMS and LHCb experiments have
worked to develop HammerCloud plugins to test their grid sites. CMS and LHCb prototypes have
been delivered in mid-2010. The CMS HammerCloud service has been working in production
since fall 2010 [14, 15], and the service is currently being further integrated into the daily CMS
grid operations procedures. The LHCb plugin has been used in production, for example to help
commission a new storage service at Rutherford Appleton Laboratory; finalizing the rollout of
this service for LHCb in planned for early 2011.

4. Conclusions and Future Work
HammerCloud has been developed to enable grid site administrators and operators to easily and
repeatedly test their facilities. Via on-demand stress tests and frequent short functional tests,
HammerCloud provides end-to-end testing of an experiment’s distributed analysis facilities.
Being involved since the conception of HammerCloud, ATLAS has made significant use of the
service. CMS and LHCb plugins for HammerCloud have been developed in 2010 and the service
is presently being incorporated into the daily grid operations of these experiments.

Acknowledgements
Partial support for this work was provided by EGI-InSPIRE INFSO-RI-261323.

References
[1] J.T. Moscicki et al. Ganga: A tool for computational-task management and easy access to Grid resources.
2009. Comp. Phys. Comm. 180 11
[2] J. Elmsheuser et al. Reinforcing User Data Analysis with Ganga in the LHC Era: Scalability, Monitoring and
User-support. In these proceedings.
[3] F. Legger et al. Distributed analysis functional testing using GangaRobot in the ATLAS experiment. In these
proceedings.
[4] T. Maeno et al. Overview of ATLAS PanDA Workload Management. In these proceedings.
[5] R. Jones. One Small Step: The Experiment Offline Systems after One Year. In these proceedings.
[6] G. Duckeck et al. Atlas operation in the GridKa Tier1/Tier2 Cloud. In these proceedings.
[7] D. van der Ster et al. Commissioning of a CERN Production and Analysis Facility based on xrootd. In these
proceedings.
[8] Y. Kemp et al. LHC Data Analysis Using NFSv4.1 (pNFS): A Detailed Evaluation. In these proceedings.
[9] W. Bhimji et al. Optimising Grid Storage Resources For LHC Data Analysis. In these proceedings.
[10] W. Ehrenfeld. Using TAGs to Speed up the ATLAS Analysis Process. In these proceedings.
[11] S. Skipsey et al. Establishing Applicability of SSDs to LHC Tier-2 Hardware Configuration. In these
proceedings.
[12] D. Duellman et al. Storage Service Developments at CERN. In these proceedings.
[13] A. Peters et al. Exabyte Scale Storage at CERN. In these proceedings.
[14] C. Grandi et al. CMS Distributed Computing Integration in the LHC Sustained Operations Era. In these
proceedings.
[15] J. Hernandez et al. Monitoring the Readiness and Utilization of the Distributed CMS Computing Facilities
during the First Year LHC Running. In these proceedings.