A Dynamic Web User Interface for HammerCloud

Openlab Summer Student Programme Report

Massimo Paladin

August 7, 2009

Contents

List of Figures

Chapter 1

Introduction

1.1 Overview

This report is to explain my work at CERN during the Openlab Summer Student Programme. During my staying at CERN I've been working in IT/GS/DMA. I have been working on *HammerCloud*, a stress-testing system to commission grid sites for distributed analysis activities.

I developed a new web interface for better visualization of the stress test results and an administration interface for HammerCloud operators. This interface has generated positive feedback from the users and it is now in production for the ATLAS experiment.

1.2 Problem: HammerCloud web user interface

HammerCloud is a system that performs stress-testing of distributed analysis facilities which do LHC physics. It has been in use since Fall 2008, and was notably quite successful during the STEP'09 tests to run more than 1 million jobs worldwide. However, the HammerCloud web interface was the weakest point of the system. It was a static interface that couldn't scale over large tests.

The weak points include:

- The pages were generated a priori by a scheduled script, requiring the storage of all the pages. They were generated every 10 minutes for the running tests, not providing live data.
- The test result page could be unusable for large tests. In table 1.1 you can see 4 screenshots, all from the same page. Those 4 screenshots represent only a small part of the page; as you can see for a large test this page was completely unreadable.

- All the plots were generated a priori from a scheduled script with the requirement to store all of the files. The consequence was wasted storage on the server.

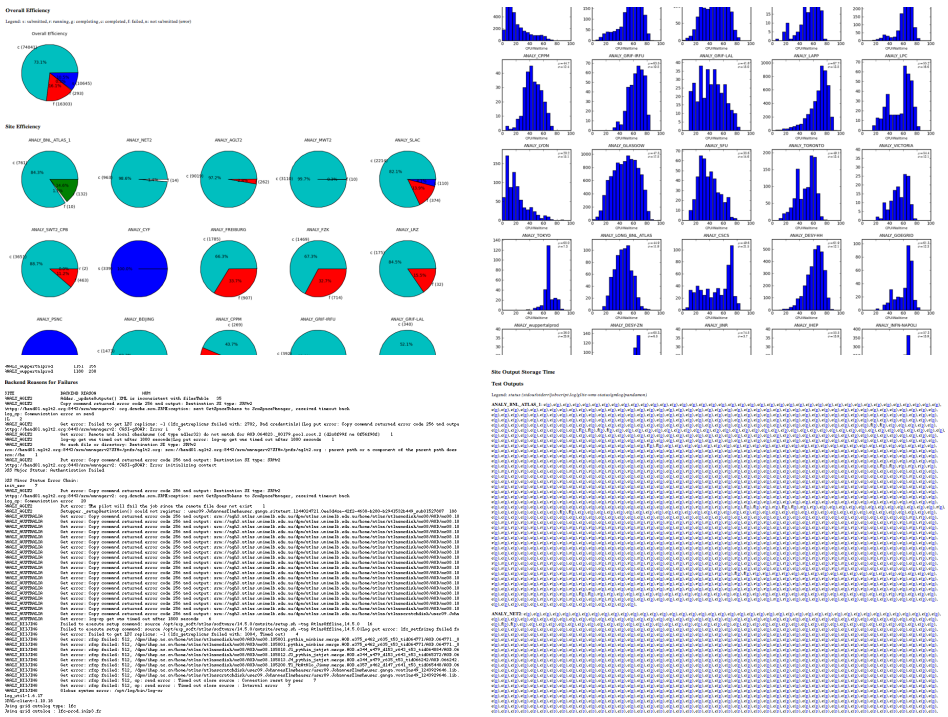


Table 1.1: Old test result page

1.3 Outline

A brief explanation about the chapters content:

- Chapter 2: in this chapter we explain the requirements and the technologies used for the development of the new web interface
- Chapter 3: in this chapter you can find three specific guides for the different users of the system: normal users, operators and developers
- Chapter 4: in the last chapter are the conclusions and possible improvements for the system

Chapter 2

A Dynamic Interface for HammerCloud

In the previous chapter I explained the problems of the old HammerCloud web interface. The target of my staying at CERN was the development of the new interface. In this chapter I am going to explain the requirements of the new interface and the technologies used for its development.

2.1 Design

Looking at the major shortcomings of the old interface we defined some goals and requirements for the new interface:

- dynamic interface
- better visualization for test result page
- easier error diagnosis
- better plotting system
- better usage history of the clouds
- administration interface for the operators

The new interface should be dynamic and scale over a large amount of tests and big tests, it should generate the pages dynamically to avoid the static pre-generation performed by the old one.

The test page requires a new design, it should be lighter and more usable, the contents have to be reorganized in a better way to prevent huge pages. It would be good to have the possibility to compare the metrics over the sites.

One of the causes of huge pages in the old interface was the printing of all the error codes generated from the tests. These have to be grouped and reorganized by their type selecting only the useful informations.

The plotting system of the old interface required to store all the plots in the server, wasting a huge amount of storage. The dynamic generation of the plots is not computionally possible; we have to generate them with a scheduled script. Thanks to *Google Charts API*[1] we don't need to store all the charts but only the informations to generate them.

The usage of the clouds chart should be improved keeping long term history and offering an increased granularity like a per site history with an adequate way to browse it.

The major requirements is an administration interface to able the HammerCloud operators to create and manage tests without the administrator support. This feature should be designed carefully looking at the security threats.

2.2 Technologies used

The reference programming language for the new interface is *Python* but I used a framework to speed up the development. The technologies used are:

- Django[2]: web application framework written in *Python*
- Google Charts API: free server-side plotting system offered by *Google*
- Memcached[3]: object caching system supported by *Django*
- Jquery[4] & Jquery UI[5]: Javascript libraries to simplify javascript code
- MySQL[6]: relational database management system

2.2.1 Django

Django is a web application framework written in *Python*, it permits rapid development thanks to helpful features:

- Object-relational mapper: possibility to code the database model with *Python* classes. *Django* provides facilities to insert, update, delete and read data from the database thanks to the code of the model
- URL pattern system: bind view functions to specific urls using regular expressions
- Template system: permits to template the html code in files separate from the source code. The view functions can render this template files with some data to build the page

- Automatic administration interface: thanks to the coded database model, *Django* provide an automatic and customizable administration interface for the modification of the coded tables
- Cache System: support different caching systems like *memcached* to permit caching at different granularity

2.2.2 Google Charts API

Google Charts API is a plotting system, completely server-side offered by *Google*. The idea of this system is that to create a new plot you need to build an ad hoc url looking the API, specifying all the parameters for the chart. After that you can request the url and *Google* will return you the wanted chart in the *PNG* format.

For instance, if you request the url in figure 2.1 you get the chart in figure 2.2.

```
http://chart.apis.google.com/chart?chxt=x,y,x,x&chds=0,2860.0&
chd=t:170.0,1989.0,2593.0,2088.0,1332.0,918.0,652.0,325.0,115.0,
57.0,18.0,1.0,0.0,0.0,0.0,0.0&chxp=2,50.0|3,80.0&chxr=0,0,30,0.0|
1,0,2860.0,286.0&chco=4d89f9&chbh=a,1,1&chs=300x300&cht=bvg&
chtt=Overall+Events/s&chxl=2:|Hz|3:|μ=7.1+σ=3.6
```

Figure 2.1: Google Charts URL example

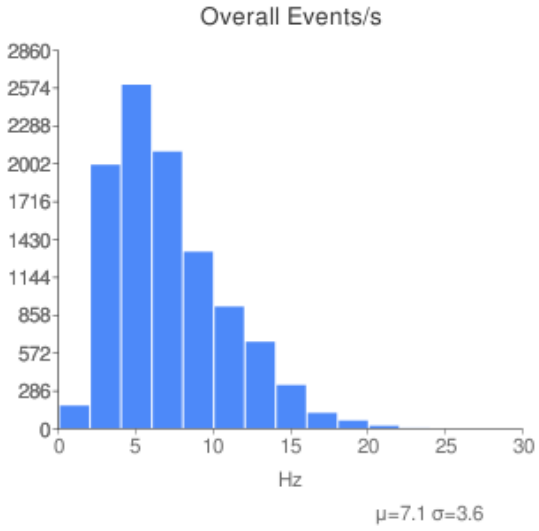


Figure 2.2: Google Charts example

2.2.3 Memcached

Memcached is an object caching system. It is a daemon that provides high level caching. *Django* has a backend for it and permits to use it very easily improving the performance of the interface.

2.2.4 Jquery & Jquery UI

Jquery is a javascript library that simplifies HTML document traversing, event handling, animating, and Ajax interactions for rapid web development. *Jquery UI* is a library that uses *Jquery* and provide easy animations and fancy widgets with a short amount of code.

Chapter 3

Usage Guides

In this chapter guidelines are provided for three types of users: the normal users, the operators and the developers. These have the aim to give an overview of the system for the respective points of view.

3.1 User Guide

This section is for the normal user that wants to browse through the test results. Starting from the home page (figure 3.1) at the address
http://gangarobot.cern.ch/hc
you can access to all the available contents. In the top of the page you can find a menu with the following entries:

- Clouds: point to a page that shows the list of the Clouds.
- Tests: is a drop down menu that permits to reach the Test list page. Each entry of this submenu points to a different state of the tests, you can access to the list of the scheduled tests, of the running tests and so on for all the test states.
- Last Tests: is a drop down menu with quick links to the last 10 tests
- Time: is a link that points to a page with statistics over the history
- Usage: is a link that points to a page showing the usage history charts of the running and completed jobs. It is possible to look at the overall charts or you can pick a specific cloud or site.
- Administration: a link that points to the administration interface, reserved for the HammerCloud operators.

In the center of the page you can see a chart that shows the usage of the clouds. Just below that chart there is a table that shows the running tests and the next 10 scheduled tests, you can click over a row to access to the result page.

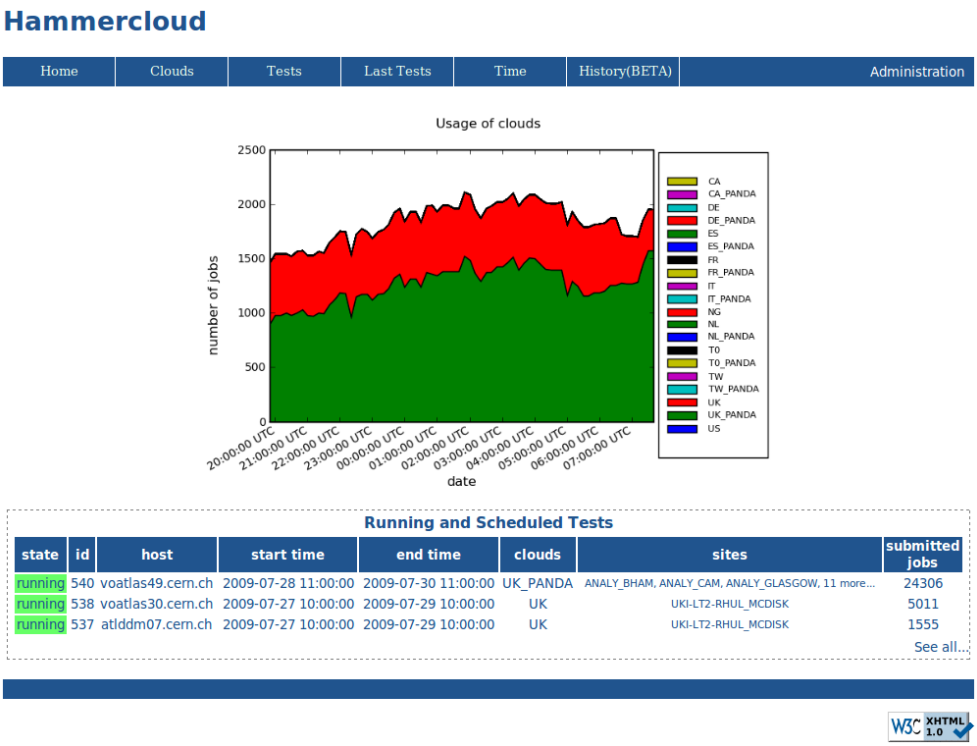


Figure 3.1: HammerCloud home page

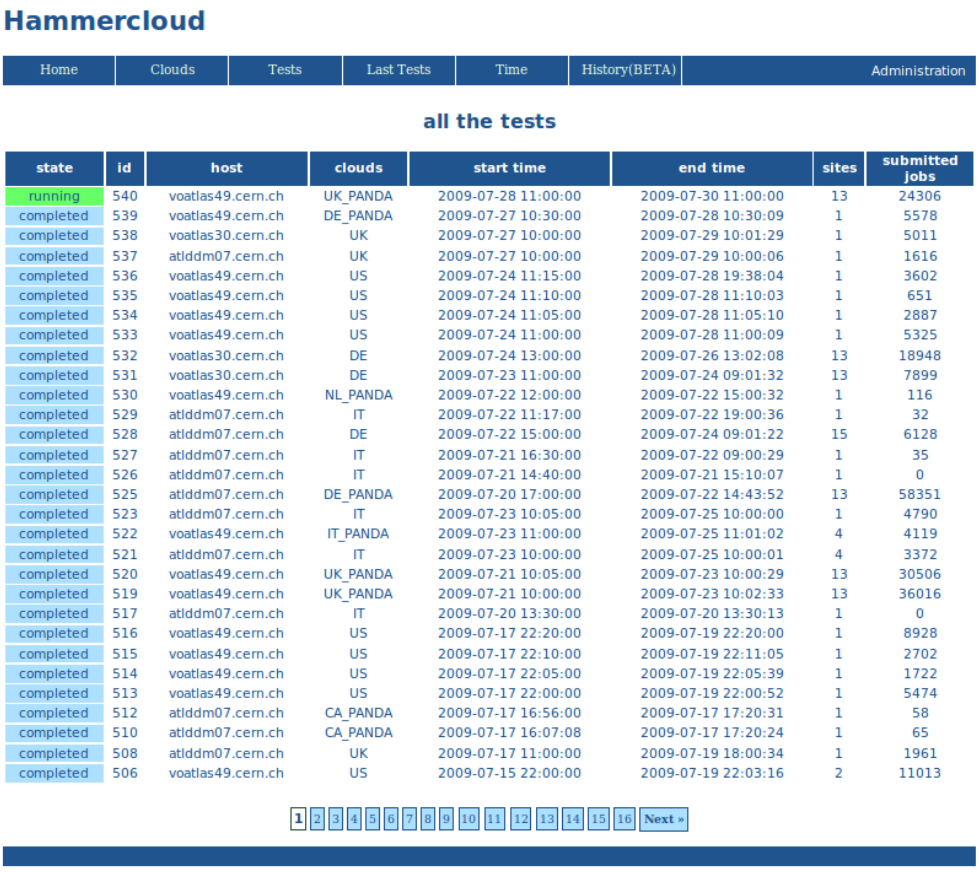
3.1.1 Tests list page

In the tests list page 3.2 you can find the list of the tests with some informations, you can click over a test to open the result test page.

3.1.2 Test page

In the single test page 3.3 all the information about a test are showed. The page is divided into 4 tabs:

- Overall: in this tab, figure 3.3, there is an overview of the test, there is some general information, the output logs of the test and some overall charts. Below there is an accordion that shows technical information and error diagnosis about the test.
- Sites: in this tab there is a *per site* view, you can see the metrics plot and all the log informations about the test grouped by site, this is useful to see all the informations relative to one single site.
- Metrics: in this tab there is a *per metric* view, you can find all the metric plots grouped by metric type, this is very useful when you want



state	id	host	clouds	start time	end time	sites	submitted jobs
running	540	voatlas49.cern.ch	UK_PANDA	2009-07-28 11:00:00	2009-07-30 11:00:00	13	24306
completed	539	voatlas49.cern.ch	DE_PANDA	2009-07-27 10:30:00	2009-07-28 10:30:09	1	5578
completed	538	voatlas30.cern.ch	UK	2009-07-27 10:00:00	2009-07-29 10:01:29	1	5011
completed	537	atlddm07.cern.ch	UK	2009-07-27 10:00:00	2009-07-29 10:00:06	1	1616
completed	536	voatlas49.cern.ch	US	2009-07-24 11:15:00	2009-07-28 19:38:04	1	3602
completed	535	voatlas49.cern.ch	US	2009-07-24 11:10:00	2009-07-28 11:10:03	1	651
completed	534	voatlas49.cern.ch	US	2009-07-24 11:05:00	2009-07-28 11:05:10	1	2887
completed	533	voatlas49.cern.ch	US	2009-07-24 11:00:00	2009-07-28 11:00:09	1	5325
completed	532	voatlas30.cern.ch	DE	2009-07-24 13:00:00	2009-07-26 13:02:08	13	18948
completed	531	voatlas30.cern.ch	DE	2009-07-23 11:00:00	2009-07-24 09:01:32	13	7899
completed	530	voatlas49.cern.ch	NL_PANDA	2009-07-22 12:00:00	2009-07-22 15:00:32	1	116
completed	529	atlddm07.cern.ch	IT	2009-07-22 11:17:00	2009-07-22 19:00:36	1	32
completed	528	atlddm07.cern.ch	DE	2009-07-22 15:00:00	2009-07-24 09:01:22	15	6128
completed	527	atlddm07.cern.ch	IT	2009-07-21 16:30:00	2009-07-22 09:00:29	1	35
completed	526	atlddm07.cern.ch	IT	2009-07-21 14:40:00	2009-07-21 15:10:07	1	0
completed	525	atlddm07.cern.ch	DE_PANDA	2009-07-20 17:00:00	2009-07-22 14:43:52	13	58351
completed	523	atlddm07.cern.ch	IT	2009-07-23 10:05:00	2009-07-25 10:00:00	1	4790
completed	522	voatlas49.cern.ch	IT_PANDA	2009-07-23 11:00:00	2009-07-25 11:01:02	4	4119
completed	521	atlddm07.cern.ch	IT	2009-07-23 10:00:00	2009-07-25 10:00:01	4	3372
completed	520	voatlas49.cern.ch	UK_PANDA	2009-07-21 10:05:00	2009-07-23 10:00:29	13	30506
completed	519	voatlas49.cern.ch	UK_PANDA	2009-07-21 10:00:00	2009-07-23 10:02:33	13	36016
completed	517	atlddm07.cern.ch	IT	2009-07-20 13:30:00	2009-07-20 13:30:13	1	0
completed	516	voatlas49.cern.ch	US	2009-07-17 22:20:00	2009-07-19 22:20:00	1	8928
completed	515	voatlas49.cern.ch	US	2009-07-17 22:10:00	2009-07-19 22:11:05	1	2702
completed	514	voatlas49.cern.ch	US	2009-07-17 22:05:00	2009-07-19 22:05:39	1	1722
completed	513	voatlas49.cern.ch	US	2009-07-17 22:00:00	2009-07-19 22:00:52	1	5474
completed	512	atlddm07.cern.ch	CA_PANDA	2009-07-17 16:56:00	2009-07-17 17:20:31	1	58
completed	510	atlddm07.cern.ch	CA_PANDA	2009-07-17 16:07:08	2009-07-17 17:20:24	1	65
completed	508	atlddm07.cern.ch	UK	2009-07-17 11:00:00	2009-07-19 18:00:34	1	1961
completed	506	voatlas49.cern.ch	US	2009-07-15 22:00:00	2009-07-19 22:03:16	2	11013

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 Next »

Figure 3.2: Tests List page

to compare the sites over a single metric.

- Other: other links

In the right corner at the top of the tabs there are two links that permit operators to clone and modify a test. The clone operation can be performed only by operators. For the modification of a test and his parameters the test has to be in scheduled or running state and the user has to be enabled from an operator.

3.1.3 Usage history page

A few words about the Usage History page (figure 3.4). This page shows two charts showing the number of the running and completed jobs over the history; these charts are generated with *Google Visualization API*. By default the overall charts are displayed but with the two drop down lists you can pick a single cloud or a single site for specific charts.

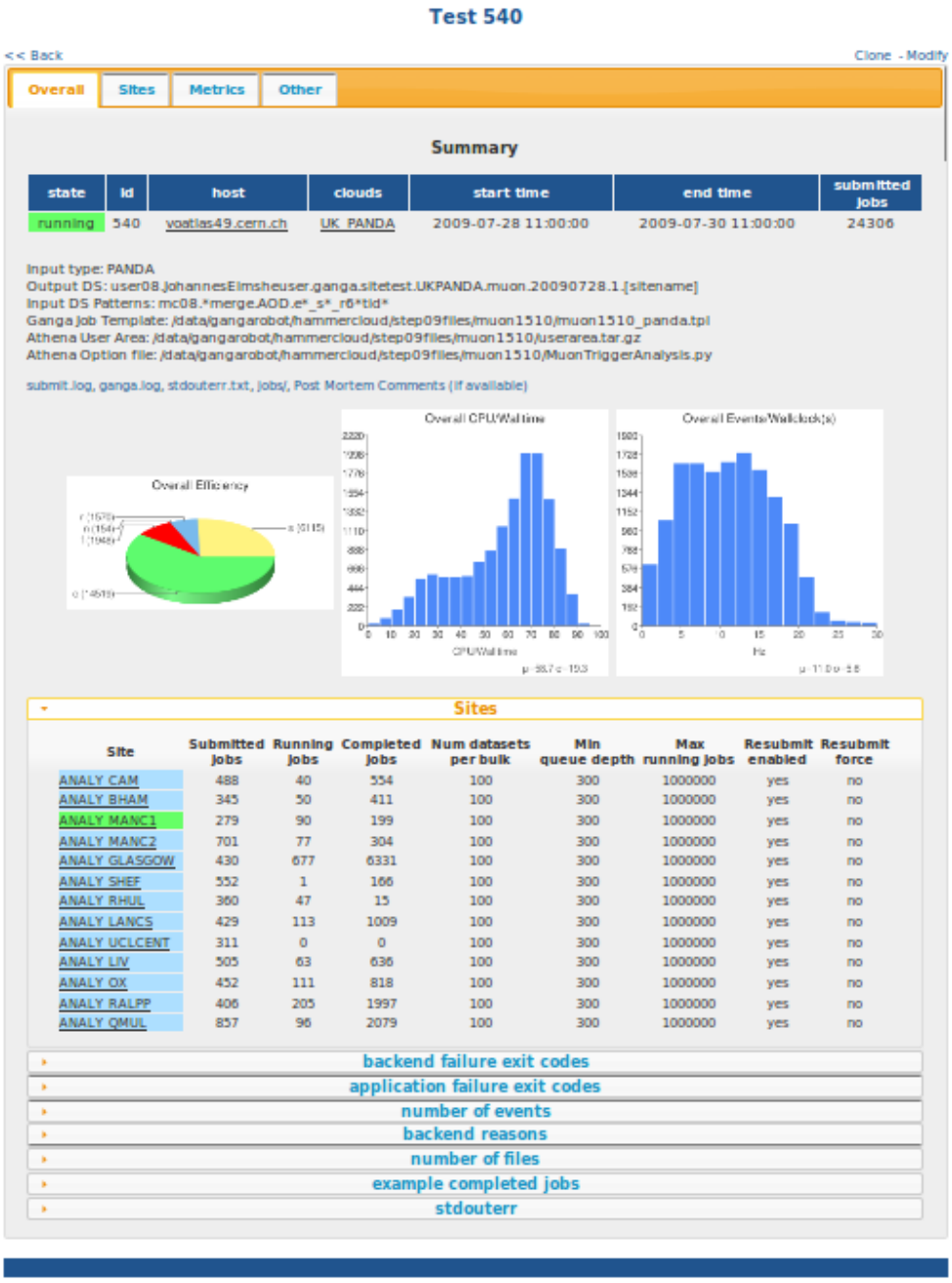


Figure 3.3: Tests page

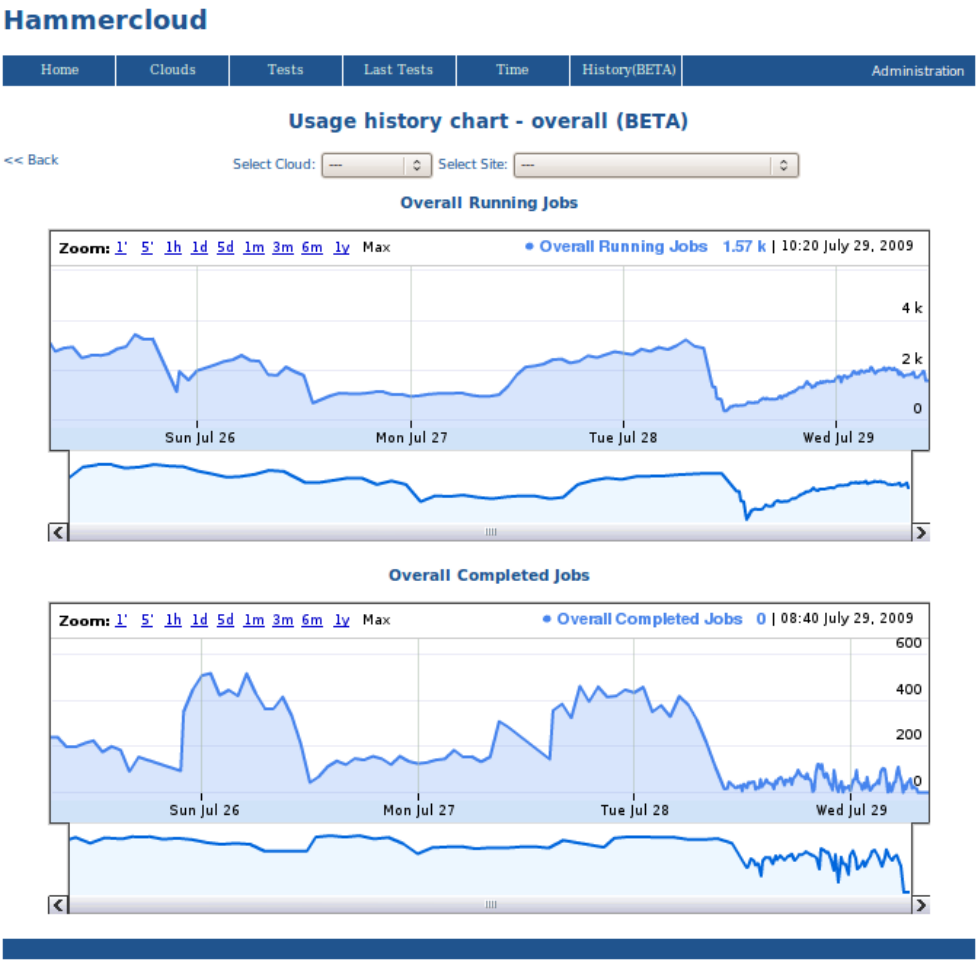


Figure 3.4: Usage History page

3.1.4 Cloud and Site page

In Cloud and Site pages is possible to see the usage history charts relative to the running and completed tests.

3.2 Operator Guide

This section is for the operators that we distinguished between two types: the staff and the normal operators. Staff operators are allowed to create, modify and delete tests and their related associations with the sites, the dspatterns and the users. Normal Operators have site-oriented permissions, they can modify the sites properties related to a test.

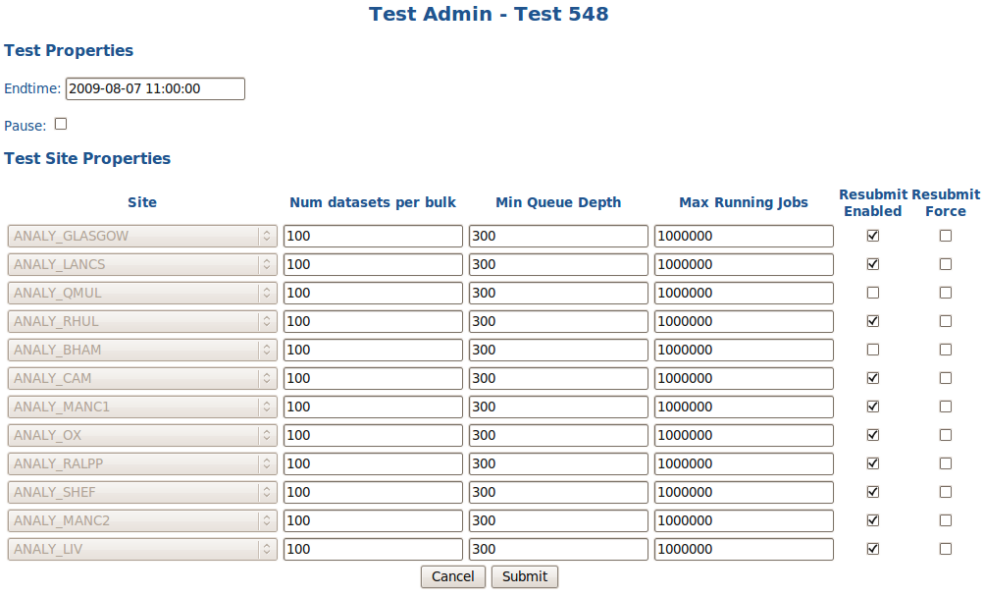


Figure 3.5: Test modification page

3.2.1 Normal Operators

The normal operators can modify the sites properties of the tests, to do that they need an account that can be created from the staff operators and have to be enabled to modify a test from a staff operator.
The link to access to the test modification page can be found in each single test page (section 3.1.2, figure 3.3), in the right corner at the top of the page.

The modification page (figure 3.5) is divided in two parts, in the first part the operator can change the *endtime* of the test and can pause it. In the second part the operator can change all the site properties related to that test. After the editing click *Submit* to save the new values.

3.2.2 Staff Operators

The administration interface for the staff operators is accessible to this address:
http://gangarobot.cern.ch/hc/admin
To enter in the administration interface the user need a staff account that can be created by one of the administrator of the system. The home page of the administration (figure 3.6) give you access to read, modify and create tests and its related objects like the associations with the sites, the dspatterns and the users.

Create a new test

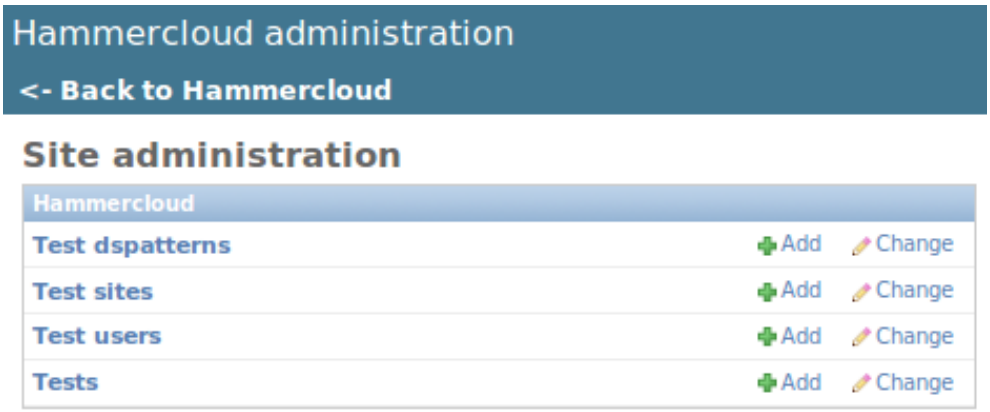


Figure 3.6: Administration page

To add a new test just click to the *add* link of the test object, this link will bring you in a page where you can set all the required parameters for the test.

In the first part of that page 3.7 you can specify all the general parameters for the test.

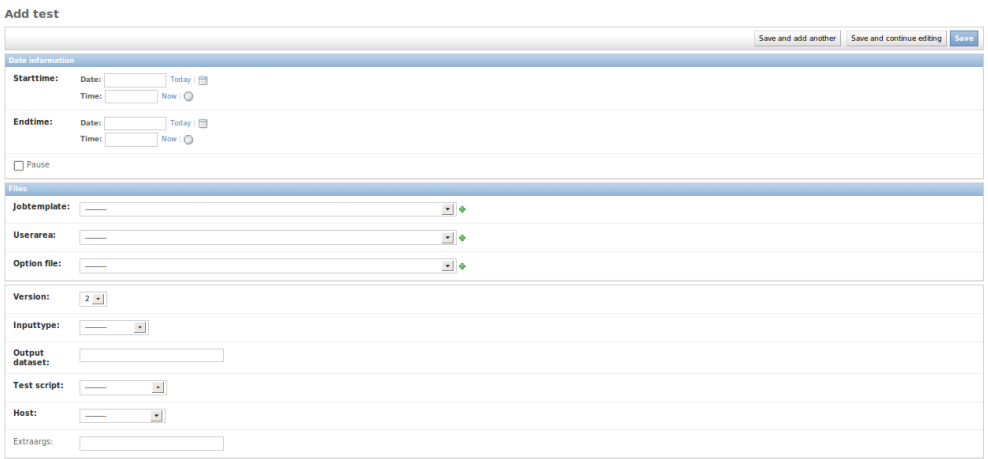


Figure 3.7: Add a new test - part 1

In the second part (figure 3.8) you can pick the cloud and the site that you want to add to the test specifying some per site parameters. When you pick a cloud the system will add one entry to Test Site association for each site of the selected cloud.

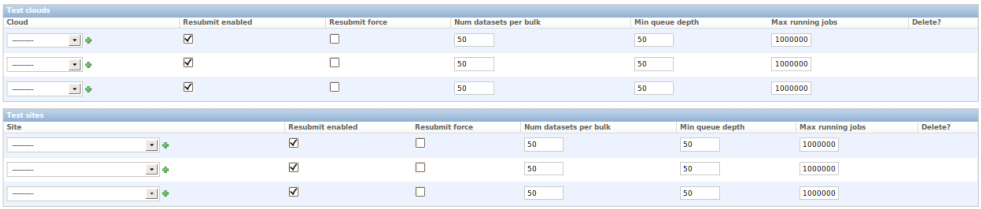


Figure 3.8: Add a new test - part 2

In the third part (figure 3.9) you can:

- add new dspatterns to the test
- add new user associations, meaning to allow a non staff user to modify some parameters of the test from the user interface
- give a recurrence to the test; the system will create multiple instance of the same test changing the *starttime* and the *endtime*

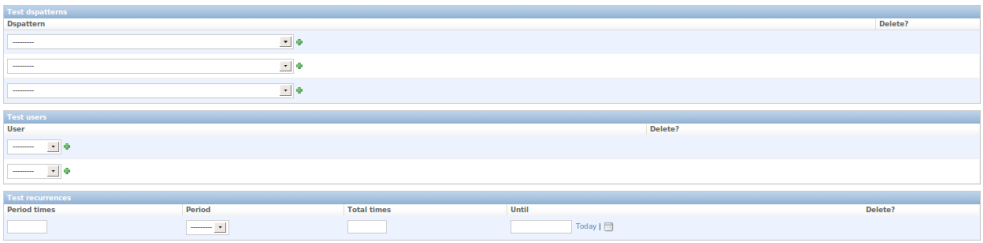


Figure 3.9: Add a new test - part 3

When you have finished with the test editing the test will be in a draft state, to schedule it you have to go in the tests list that you can reach from the administration interface. From the tests list you can check the checkbox of the test that you want to schedule and select the action *Send selected tests for approval* and press the *Go* button. The system will send an email to the administrator informing that there is a new test to approve and schedule.

Clone a test

If you want to create a test similar to an old test you can just go in the tests list of the administration interface, check the tests that you want to clone, select the action *Clone multiple tests* from the drop down list and press the *Go* button. That action clones the test. Afterwards you have to change some parameters like the *starttime* and the *endtime* or other parameters that you want to change and finally you can send it for approval.

3.3 Developer Guide

3.3.1 HammerCloud Web User Interface

This section is for the developers that want to maintain and add functionalities to the system. I am going to explain the source code structure of the interface, with some helpful informations to add contents to the system. You can see the structure of the source code in figures 3.10 , 3.11 and 3.12). The first figure contains all the *Python* files that build the system. In the second figure there is the media directory; this is the directory where you should put all the static contents of the interface that are going to be served from the static web server. In the third image there is the templates directory, here you can find all the *html* templates of the interface.

Keep in mind the *Django* features explained in 2.2.1 that will help you to understand the meaning of the different files.

Source Code

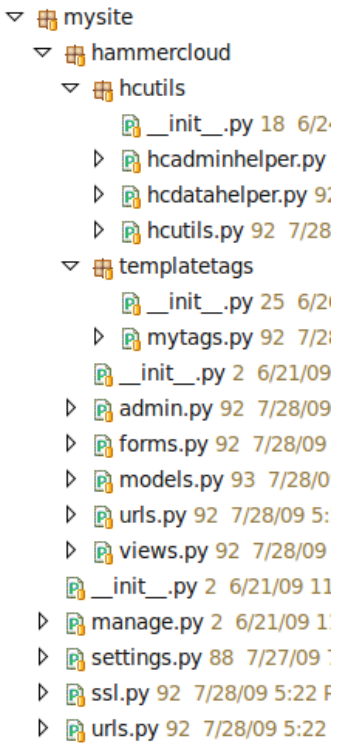


Figure 3.10: HammerCloud WUI source code

In figure 3.10 you can find an explanation of the source code files.

- settings.py : this file contains the Django settings for the application

- urls.py : url patterns for the entire website
- ssl.py : middleware for *Django* to force *SSL* in selected pages
- manage.py : *Django* file that permits to use the application locally with a small server offered from *django*
- hammercloud/ : this directory contains all the *HammerCloud* web interface application files
- hammercloud/urls.py : url patterns for the application
- hammercloud/models.py : this file is one of the most important, it codes every table of the database into *Django* objects with methods that provides all the data that have to be showed
- hammercloud/views.py : this file contains all the view functions, every page of the interface is a view, and all the views are managed here
- hammercloud/forms.py : this file contains the code of some forms used in the interface
- hammercloud/admin.py : contains the configurations and the customizations for the automatic administration interface provided from *Django*
- hammercloud/hcutils/hcadminhelper.py : this file contains some utilities used on the test modification page of the user interface
- hammercloud/hcutils/hcdatahelper.py : this file contains some data configurator and aggregator used from *views.py* to gather the data to send to the templates system
- hammercloud/hcutils/hcutils.py : some useful classes of general behaviour used in the application
- hammercloud/templatetags/mytags.py : personal filters for the template tags system used in *Django*

Media Files

In this section an explanation about the media files listed in figure 3.11

- admin/ : this directory just point to the admin media directory of *Django* directory
- css/ : directory that contains all the stylesheets of the application
- css/smothness/ : theme for the *Jquery UI* facilities

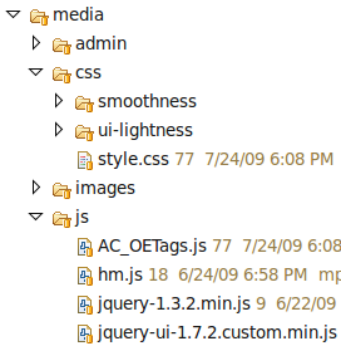


Figure 3.11: HammerCloud WUI media

- css/ui-lightness/ : theme for the *Jquery UI* facilities
- css/style.css : main stylesheet for the application
- images/ : directory for the images
- js/ : directory containing all the javascript used in the application
- js/AC_OETags.py : javascript from *Adobe* to check *Flash* requirements on the client browser
- js/hm.js : custom javascript for the application
- js/jquery-1.3.2.min.js : javascript for *Jquery*
- js/jquery-ui-1.7.2.custom.min.js : javascript for *Jquery UI*

Template Files

In this section an overview over the templates files listed in figure 3.12

- 404.html : template for the 404 http error
- 500.html : template for the 505 http error
- admin/base_site.html : customization of the administration template
- hammercloud/ajaxtestmetrics.html : template for the *Metrics* tab of the single test page
- hammercloud/ajaxtestsites.html : template for the *Sites* tab of the single test page
- hammercloud/base.html : base template, the one to be extended if you want to create new pages

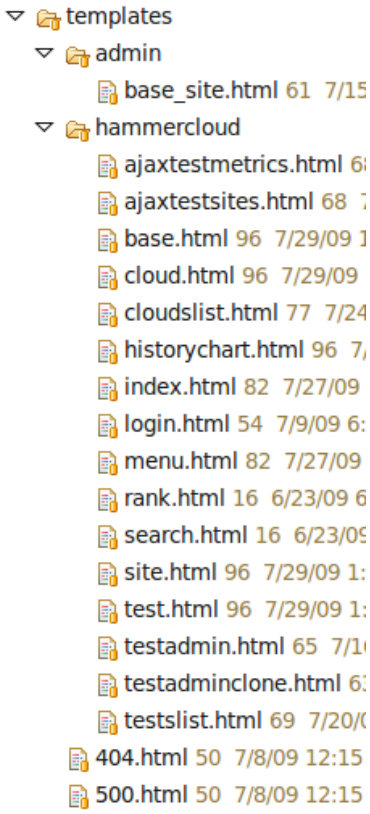


Figure 3.12: HammerCloud WUI templates

- hammercloud/cloud.html : single cloud page template
- hammercloud/cloudslist.html : clouds list page template
- hammercloud/historychart.html : template for the usage history page
- hammercloud/index.html : template for the homepage
- hammercloud/login.html : template for the login page
- hammercloud/menu.html : template for the top menu, included in the *base* and *index* templates
- hammercloud/rank.html : draft of the template for the future rank page
- hammercloud/search.html : draft of the template for the future search page
- hammercloud/site.html : template for the single site page

- hammercloud/test.html : template for the single test page
- hammercloud/testadmin.html : template for the tests modification page of the users interface, not the one in the automatic administration interface
- hammercloud/testadminclone.html : template for the test clone confirmation page
- hammercloud/testslist.html : template for the page that shows the tests lists

Add a new page to the interface

To add a new page to the web interface you should follow these step:

- add a new url pattern to *hammercloud/urls.py*
- create a template file inside the template directory
- add a new function to *hammercloud/views.py* that manage the new page. If you need data related to the model and they are not provided, add a new method to the right class in *hammercloud/models.py* that generate the data that you need in the function

3.3.2 Server Scripts

There are 2 server scripts that collect data for the web interface, the first is related to the plots generation and the second one is the one in charge to collect the usage of the clouds.

The dynamic plots generation was too expensive computanionally in terms of database queries to be computed at each page request. The only option was to generate the plots with a server script. The second script records the usage of the single sites over the history providing usage history charts over all the clouds, for a single cloud and for a single site.

Plots generator

The file *hcplotgenerator.py* is responsible for creating the metric plots for the tests. This script run every 10 minutes and generate the metric plots for all the running tests and the test completed from not more than 2 hours. This script uses the python module *hctestplot.py* that contains the metric functions.

Add a new plot

To add a new plot to the tests you should follow these steps:

- add a new tuple in the table *metric_type*
- add the method that generates the new plot into *hctestplot.py* module according to the schema of the existing plots
- add the name of the plot to generate into the module *hcplotgenerator.py* in one or both the lists that appear in the top of the file. These lists contain the plots to generate for the normal and the panda tests.

The web interface will automatically print the new plots after the first run of the scripts.

Usage generator

The module *hcusagegenerator.py* is the one responsible for logging the number of the running and complete jobs over the history, this script runs every 10 minutes on the *gangarobot* server. The script logs the following informations:

- every ten minutes records the number of running jobs for each site flagging them as *daily*
- every ten minutes counts the number of the completed jobs completed 100 minutes before. Because of delays on the data acquirement the number of completed jobs is updated til 200 minutes later the stop time of the jobs.
- every hour aggregates the number of the running and completed jobs of the 25th hour in the past keeping only one point for each hour and flagging these points as *weekly*
- every 7 days aggregates the number of the running and completed jobs of the 8th day in the past keeping only one point for each day and flagging these points as *monthly*
- every 90 days aggregates the number of the running and completed jobs of the 11th week in the past keeping only one point for each week and flagging these as *yearly*

Chapter 4

Conclusions and Improvements

4.1 Conclusions

Personally, I feel that my experience at CERN was very rewarding. I valued the importance of being part of a team and learned many new things thanks to the kind people in my group. I was particularly interested in being exposed to real problems that I have not previously seen during my studies.

4.2 Possible Improvements

A list of possible improvements that would be useful:

- Sites Rank: study and develop a metric to measure the sites quality in order to build a rank
- Search Page: build a search page where it is possible to pick multiple sites and tests and show aggregated plots and reports
- NICE authentication: add *NICE* authentication to the system to permits the user to login with their account or their certificate
- Usage History: improve usage history page to allow comparison between sites and clouds

Bibliography

[1] Google Chars API - free server-side plotting system (http://code.google.com/apis/chart/)

[2] Django - Python web application framework (http://www.djangoproject.com/)

[3] Memcached - object caching system (http://www.danga.com/memcached/)

[4] Jquery - javascript framework (http://jquery.com/)

[5] Jquery UI - javascript framework for user interface widgets (http://jqueryui.com/)

[6] MySQL - database management system (http://www.mysql.com/)