Integrated experiment activity monitoring for wLCG sites based on GWT

Alejandro Guinó Feijóo and Xavier Espinal
Port d’Informacio Científica (PIC), Universitat Autònoma de Barcelona (UAB), Edifici D - E-08193 Bellaterra, (Cerdanyola del Valles, Barcelona), Spain
E-mail: aguin@ifae.es, espinal@pic.es

Abstract.
The goal of this work is to develop a High Level Monitoring (HLM) where to merge the distributed computing activities of an LHC experiment (ATLAS). ATLAS distributed computing is organized in clouds, where the Tier-1s (primary centers) provide services to the associated Tier-2s centers (secondaries) so they are all seen as a cloud by the experiment. Computing activities and sites stability monitoring services are numerous and delocalized. It would be very useful for a cloud manager to have a single place where to aggregate available monitoring information. The idea presented in this paper is to develop a set of collectors to gather information regarding site status and performance on data distribution, data processing and Worldwide LHC Computing Grid (WLCG) tests (Service Availability Monitoring), store them in specific databases, process the results and show it in a single HLM page. Once having it, one can investigate further by interacting with the front-end, which is fed by the stats stored on databases.

1. Introduction
LHC is well on the way since early November 2009 and experiments are receiving, computing and delivering data almost uninterruptedly. The WLCG Tier-1s started the operations in 2006 but they are now in full production since the LHC start. One of the peculiarities about the Tier-1s is that big fraction of them give support to several LHC experiments: ATLAS, CMS, LHCb or ALICE; each one of these having it’s intrinsic workflows for the different fundamental experiment activities: data simulation, real data reprocessing, user analysis and data distribution.

On top of that, WLCG project is constantly monitoring the performance of the sites by sending job probes either from a general point of view (experiment transversal probes) or experiment-oriented probes. The consequence is that sites end up looking at a plethora of monitoring pages without having an overall integrated view, which would make life easier to site administrators and experiment experts.

1.1. Project Proposal
Information fragmentation and the nature of ATLAS Computing Model, based on computing clouds formed by a Tier-1 and several Tier-2s, led us to develop the monitoring for an entire ATLAS cloud, the PIC cloud, composed by PIC-Tier-1 and six associated Tier-2s: IFAE, IFIC and UAM in Spain, and LIP-LISBON, LIP-COIMBRA and NCG-INGRID in Portugal.
First, we had to find the way to gather all the information to satisfy all monitoring daily needs (data distribution, data processing and WLCG tests), and this is done by the collectors. Those applications collect all the data needed in a regular basis and store them in specific SQL databases.

Having all the data, the next step was to process the results and show it in a single High Level Monitoring (HLM) page. Once having it, one can investigate further by interacting with the front-end which is fed by the stats stored at the databases.

The system behind this front-end, was developed using a browser-based development kit (Google Web Toolkit) which produces a high-performance AJAX client-server environment (Web Interface), written in JAVA, that runs across all major browsers.

2. The Collectors
We have seen that these applications are essential to gather the data and populate the databases and, finally, feed the web interface. This is their main goal and they have to do the work efficiently, using few resources and running as fast as possible. Hence these programs, in fact scripts, have been developed using many languages (prototypes) in order to test their behavior. To take the best decision, we studied the environment (OS), the needs and specially the operations they should do to accomplish its mission.

Data comes in XML format from different sources across the Grid, including WLCG tests (Service Availability Monitoring) or ATLAS Dashboard data (data distribution, data processing). It has to be treated, selecting what it’s useful, adapting, cleaning and storing the values that will be sent to a SQL database, made to the measure of the Web environment. At the same time, they have to log any warning or error (database connection fails, HTTP connection fails, memory fails, etc) and provide a database connection to finally store the values collected in the process.

We should have in mind all the reasons stated till now to carefully choose the language to code these scripts. At the end, we chose Python (an interpreted, interactive, object-oriented, extensible programming language [1]) but, why Python? Python provides all the tools needed to process XML and deal with HTTP and SQL connections with a more than acceptable performance/usability ratio, compared to all the major (or more extended) languages today, as well as the following features:

- Python offers a great integration on Linux environments (its part of Linux Standard Base) which means the collectors can be easily designed, coded and deployed on any Linux-like OS whether they are Debian, or RPM based.
- Dynamic typing. Assignment statements in Python bind names to objects of any type.
- Python code is more readable than the majority of pure scripting languages like PERL, which means improved code maintenance and ease of use.
- Python offers more productivity than pure object-oriented languages (5-10 times more than Java [2])

3. The Web Interface
At this point data is stored and ready to describe what’s happening in our cloud. Now, it’s time to build the main part of the project, what is going to finally determine the easiness of the cloud administration: the web interface. Applying the same policy as before, we built several prototypes with three ideas in mind:

- It has to contain a clear view of the data using any sort of graphical representation.
- It has to offer a global view of the state of the cloud when displayed, at first sight.
- It has to allow the administrator to go deeper into results to help him wonder what happens.
Besides these, we still wanted efficiency, easiness of use and low use of computing resources so we chose Google Web Toolkit (GWT), over the other available options. Again, why GWT over the other options? These are the main reasons we considered to use it:

- GWT is a development toolkit for building and optimizing complex browser-based applications [3].
- GWT let you build Asynchronous JavaScript and XML (AJAX) browser-based application avoiding all the problems that these technologies had in the past (JavaScript coding in large and complex applications, weak typing, etc)
- GWT let you write an application in Java, with all the benefits of object-oriented programming (OOP) over non OOP languages like PHP or ASP.
- GWT presents a transparent, ordered client-server structure without having to be an expert in browser quirks.
- It’s open source and all the modules and add-ons are completely free, including the most advanced graphical tools.
- It offers the benefits of a highly interactive web site (like Silverlight or Flash) without requiring browser plugins. It just needs JavaScript enabled.

Now that we have selected the tool, we are going to briefly explain how we coded the client-server web interface structure.

3.1. The Client-side
As told before, GWT provides a Java API that lets you build component based GUIs while avoiding JavaScript, and abstracting the HTTP protocol and underlying browser DOM model. A developer can build its front-end without knowing anything about AJAX or any of its components. [4]

Here, at the client-side, we designed and implemented the structure of the project and everything that can be seen and used by any user of the system, coded in JAVA. Then, the GWT compiler, has converted the Java classes into browser compliant JavaScript and HTML.

![Figure 1](image.png)

**Figure 1.** One of the representations built entirely in JAVA describing one of the data flows over the time at the PIC cloud

3.2. The Server-side
Our project needs the data collected to populate the structure created in the client-side, so here is where the server-side appears. At the same time JavaScript and HTML are deployed to the client-side, the server side classes are deployed on Tomcat or any other Servlet container.
We should have in mind that the heart of AJAX is making data read/write calls to a server from the JavaScript application running in the browser [5], so GWT provides many methods to communicate with a web server (RPC communications, HTTP requests, etc). We chose RPC (Remote Procedure Calls) because GWT provides a library of classes that makes the RPC communications with a J2EE server extremely easy.

![Figure 2](image)

**Figure 2.** Data flows from the server to the client to build the entire environment

4. Related work

The idea of applications with the ability to gather data/information to represent it, is not new. There are many applications, programs or scripts related to different computing fields, like personal information or topographic applications that do that. This project has the same purpose of those applications with the only difference that is specifically designed to monitor the data flows related to a system, an ATLAS cloud. This difference defines the requirements, the design and even the implementation of the project, that can be summarized in one main idea: have all the information needed to control the cloud data flows in a easy-to-use, self-explaining and unique environment, with the ability to let the administrator to go deeper into the first view [6]. Once finished, the entire project can be easily exported to any of the other ATLAS clouds as shares the same data sources. All the technologies used, the structure of the project and the data sources make it possible.

5. Conclusions and future work

We have seen the needs of a cloud administrator, and have presented a solution designed to help him having the cloud in control. The best notice is that the project was finished successfully and the tests over the ground have proven that it works well when deployed. All the Tier-2s and the the Tier-1 itself, have considered that this new tool will be very useful and hope that the work needed to control the cloud will diminish considerably.

However, there are a lot of work to do. The efficiency of the client when it builds the different representations, the structure of the project, the data treatment and the amount of RPC calls and a large list of other issues could be improved. We should have in mind that this is just a prototype.
Figure 3. The system up and running

6. Acknowledgments

This work was partially supported and makes use of results produced by the project "Implantación del Sistema de Computación Tier1 Español para el Large Hadron Collider Fase II" funded by the Ministry of Science and Innovation of Spain under reference FPA2007-66152-C02-00. Port d’Informació Científica (PIC) was founded on June 2003 as a scientific-technological center jointly maintained through the Departament d’Educació i Universitat, Centro de Investigaciones Energéticas, Medioambientales y Tecnológicas (CIEMAT), Universitat Autònoma de Barcelona (UAB) and Institut de Física d’Altes Energies (IFAE). The purpose was the establishment of a scientific research support center with a focus on experiments which had a large data volume and a big collaborative environment.
References
[1] Python 2011 Url - http://www.python.org/
[2] F S 2009 Python and java: A side-by-side comparison Url - http://pythonconquerstheuniverse.wordpress.com/2009/10/03/python-java-a-side-by-side-comparison/
[3] GWT 2011 Url - http://code.google.com/intl/ca/webtoolkit/
[4] Izabel 2007 Introduction to gwt Url - http://developerlife.com/tutorials/?p=80
[5] Google 2011 Communicating with a server Url - http://code.google.com/intl/ca/webtoolkit/doc/1.6/FAQ_Server.html
[6] Feijóo A G 2010 Monitorització d’alt nivell per el computing de l’experiment atlas (lhc)
[7] CERN 2010 Worldwide lhc computing grid Url - http://lhc.web.cern.ch/LCG/public/
[8] ATLAS 2005 Atlas computing tdr (technical design report) Url - http://atlas-proj-computing-tdr.web.cern.ch/atlas-proj-computing-tdr/PDF/Computing-TDR-final-July04.pdf (Preprint ATLASTDR--017, CERN-LHCC-2005-022)
[9] Jones R and Barberis D 2008 Atlas computing model Url - http://iopscience.iop.org/1742-6596/119/7/072020
[10] Gaidioz B 2007 Arda dashboard production system monitoring Url - http://dashb-build.cern.ch/build/nightly/doc/guides/production/html/user/index.html