Expanding the user base beyond HEP for the Ganga distributed analysis user interface

IOP Conf. Series: Journal of Physics: Conf. Series **898** (2017) 052032 doi:10.1088/1742-6596/898/5/052032

Expanding the user base beyond HEP for the Ganga distributed analysis user interface.

R Currie, U Egede, A Richards, M Slater and M Williams

Blackett Laboratory, Imperial College London, Prince Consort Road, London SW7 2BW UK, Particle Physics Group, School of Physics and Astronomy, University of Birmingham, Edgbaston, Birmingham, B15 2TT, UK

E-mail: rcurrie@cern.ch

Abstract. This document presents the result of recent developments within Ganga[1] project to support users from new communities outside of HEP. In particular I will examine the case of users from the Large Scale Survey Telescope (LSST) group looking to use resources provided by the UK based GridPP[2][3] DIRAC[4][5] instance. An example use case is work performed with users from the LSST Virtual Organisation (VO) to distribute the workflow used for galaxy shape identification analyses. This work highlighted some LSST specific challenges which could be well solved by common tools within the HEP community.

As a result of this work the LSST community was able to take advantage of GridPP[2][3] resources to perform large computing tasks within the UK.

1. Introduction

Ganga [1] is an easy to use Job submission and management tool developed in Python [6]. The project has a large user-base with approximately 300 active daily users and is responsible for submitting and managing over 200,000 jobs per day. The Ganga project is used extensively by large LHC based experiments with a proven track record on LHCb and ATLAS. Recently Ganga has been adopted as the default entry point for new users looking to make use of GridPP resources within the UK.

Job submission in Ganga is performed using either a custom written python scripts or from the existing interactive Ganga environment making use of an IPython [7] console interface.

Some usage statistics collected from Ganga users throughout 2015 to 2016 are shown in Figures 1 and 2. Figure 1 shows the breakdown of the number of users throughout this time period and the relative sizes of different user communities. Figure 2 shows the total number of jobs which were submitted during this time period.

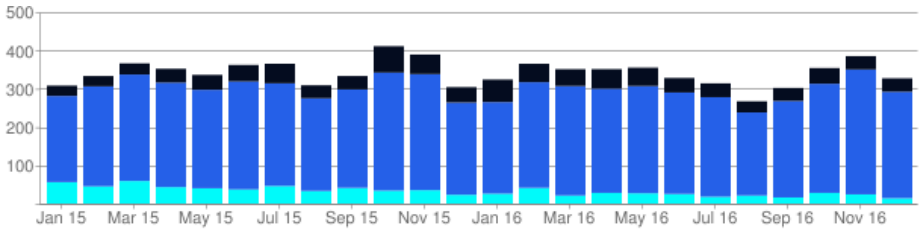

Figure 1. This figure (in colour online) shows the number of unique Ganga users throughout the time period of 01/01/2015 to 31/12/2016. Dark blue indicates LHCb users, light blue indicates Atlas users and black indicates other users.

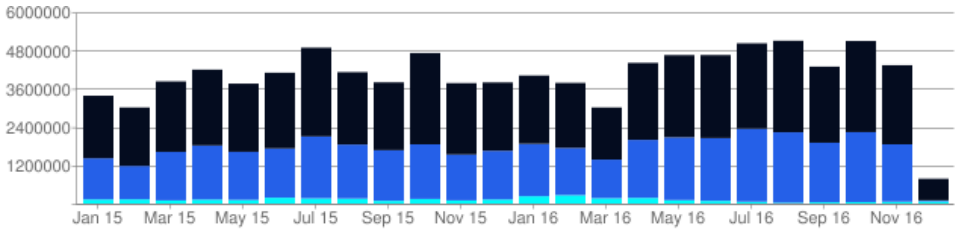

Figure 2. This figure (in colour online) shows the number submitted Ganga jobs submitted throughout the time period of 01/01/2015 to 31/12/2016. Black indicates the number of split jobs, dark blue the number of jobs which have been split and light blue the number of jobs which haven't been split.

2. Development to Support smaller VOs

In order to support users from many smaller VOs the Ganga project is uniquely positioned and to be able to support many disparate workflows and external technologies. With this in mind, the Ganga project has recently undergone significant developments to better support a more modular design. These changes have primarily focused on ensuring that various components of the Ganga core framework have well defined behaviour as well as being thread safe and robust.

Ganga 6.1 was the first version of the Ganga project to be released after migration to use the git source code management system. This had the advantage of allowing Ganga to migrate to the github.com website allowing the project to become less CERN centric and be accesible for all users. Ganga 6.3 was later released to support users requiring advanced credential management.

Perhaps the main advantage of this has been allowing Ganga to migrate to make use of a new continuous integration testsuite. This testsuite has been integrated with the Ganga project on github (`github.com/ganga-devs/ganga`) to increase the quality of the released tools to the end-users.

3. Supporting multiple VOs

One of the main efforts in Ganga 6.3 has been the development to support users who require multiple credentials to be used for different tasks. These users are typically members of multiple VOs or have multiple roles within the same HEP experiment. In order to support a growing number of multi-VO users Ganga 6.3 has developed support for managing the different credentials within its core framework.

Ganga prior to version 6.3 supported the ability to be configured to manage a single credential for a user specified VO. These credentials are managed through the use of standard VOMS tools with user proxies constructed upon demand.

The main development within Ganga 6.3 has been to develop a service within the infrastructure which allows for providing different credentials for different tasks. As with proir releases this new credentials service has been designed to use well tested VOMS tools to manage multiple proxies on behalf of the user.

Future developments are also orientated to allow for communication with multiple instances of proxy based job management systems such as DIRAC.

4. Ganga Job Submission

Ganga makes use of a "Job" object which has several key attributes highlighted in Figure 3. Each of these attributes has an interface describing the expected behaviour of a plugin which is assigned to each "Job" instance. These plugins are python classes built on top of the Ganga core framework.

Ganga makes use of the plugins assigned to a "Job" object to build a job-script used to submit a job to different backends. A job-script contains all of the instructions for automating common user related activities. These plugins are all Python based classes derriving from the "GangaObject" within the Ganga framework. This object provides features such as persistency, object inheritance and a user-focussed configuration framework.

The configuration of these "Job" objects is saved to disk using an XML based repository storing a jobs configuration and its current state.

The plugins which account for most of the content of a given job-script are the "backend" and "application" plugins. Many of the tasks performed within these plugins are common between experiments and often deal with similar tasks.

Examples of some of the tasks automated by these plugins are:

- Managment of job input files
- Description of experimental data
- Configuring job output
- Description of what a job should execute

Recent developments have also focussed on reducing the complexity of generated job-script files. This development has made writing new application plugins much simpler due to the reduced amount of duplicated behaviour between objects and experiments. In the case of the LHCb community this has had the result of a three fold reduction in the amount of auto-generated code for job submission and management. This reduces the amount of work required to debug and maintain these job-scripts and to support each community.

These scripts and shared components are used daily by hundreds of users which means that all user groups benefit from testing of many different workflows.

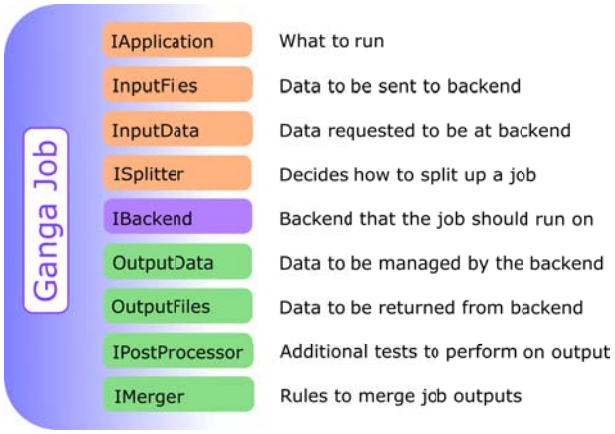

Figure 3. This figure (in colour online) shows the components within a Ganga Job. The orange top 4 plugins are used to configure what is run and what input data is needed. The purple backend indicates where the job is to be run. The bottom 4 green plugins configure what is to be done once the job has completed.

5. LSST Workflow

The LSST workflow supported within Ganga was the submission of computing jobs to perform galaxy shape identification using the Im3Shape[8] application.

This workflow is a highly CPU limited workflow which processes a relatively small amount of data and as such is similar to many of the distributed workflows within HEP.

Early usage of Ganga by LSST users involved the submission of large complex jobs built around a bash script which had evolved from the use of a local batch computing system. The main disadvantage to this was the poor management of files which were required by and produced by the application. Working with users of the Im3Shape application it was established the best way to distribute this software is to use CVMFS[9]. Additionally the use of Ganga objects to handle data required for input and output of each individual subjobs was found to be well suited to the job requirements.

Data management and the matching of jobs to the correct sites across the UK was managed by the GridPP DIRAC instance. Using Ganga a test processing of data was launched in Feburary 2016 which used to Im3Shape to analyse a test dataset. Figure 4 shows the total amount of CPU hours which were used throughout this processing.

The LSST workflow requires a large amount of configurable options to be passed to the Im3Shape application when it launches on a worker node. In order to simpligy the management of this an Im3Shape application plugin within Ganga was developed to store all of the configurable options in an easily accessible way. This plugin was designed to be minimal which puts new focus on the development of Ganga's core infrastrucutre to meet the needs of new users.

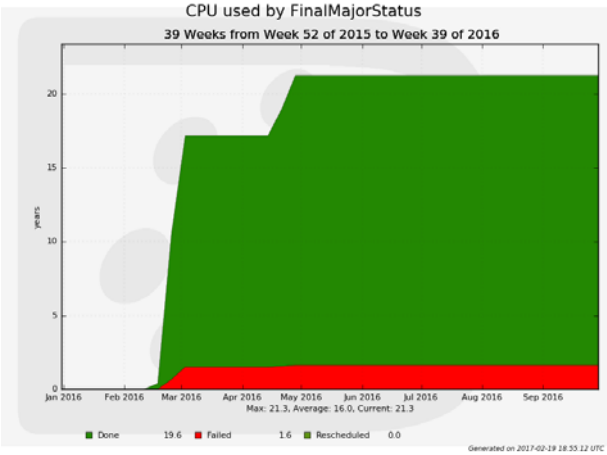

Figure 4. This figure shows the amount of CPU used by the LSST VO within the GridPP[2][3] instance of DIRAC[4][5]. The rise in CPU time in Feburary of 2016 was associated with the processing of data using the Im3Shape[8] application.

6. Conclusions

Ganga has shown to be a highly configurable and modular system to allow for complex analyses to be submitted and managed across many different computing resources.

Working with smaller VOs and the LSST community has been rewarding as it has allowed Ganga to increase the modularity of it's existing infrastructure and to introduce new features.

Migrating the Ganga project to use git and making use of resources from github has improved the quality and stability of Ganga releases. Making use of industry standard tools and services with a new continuous integration framework reduces the complexity and overhead in maintaining the Ganga project.

Ganga 6.3 has recently been released offering more support for users requiring more complex credential management. This release is also widely used and trusted by many experiments for job submission and management.

References

[1] Moscicki J T et al, 2009, Ganga: a tool for computational-task management and easy access to grid resources, *Comp. Phys. Comm.* Vol. **180** Issue 11
[2] Faulkner P J W et al, 2006, The GridPP collaboration, GridPP: development of the UK computing grid for particle physics, *J. Phys. G: Nucl. Part. Phys.* **32** N1–20
[3] Britton D et al, 2009, GridPP: the UK grid for particle physics, *Phil. Trans. R. Soc.* A **367** 2447–57
[4] Bauer D et al, 2015, The gridpp DIRAC project - DIRAC for non-LHC communities, *J. Phys.: Conf. Ser.* **664**
[5] Bauer D et al, 2015, The gridPP DIRAC project: implementation of a multi-VO DIRAC service, *J. Phys. Conf. Ser.* **664**
[6] Van Rossum G and Drake Jr. F L, 2006, *The Python language reference manual: revised and updated for version 2.5* (Network Theory Limited, Bristol)
[7] Perez F and Granger B E, 2007, IPython: a system for interactive scientic computing, *Comp. in Sci. and Eng.* **21**
[8] Zuntz J et al, 2013, IM3SHAPE: A maximum-likelihood galaxy shear measurement code for cosmic gravitational lensing, *Mon. Not. R. Astron. Soc.*
[9] Buncic P et al, 2010, CernVM – a virtual software appliance for LHC applications, *J. Phys.: Conf. Ser.* **219**