UNIVERSITY OF BIRMINGHAM

First Steps with Ganga

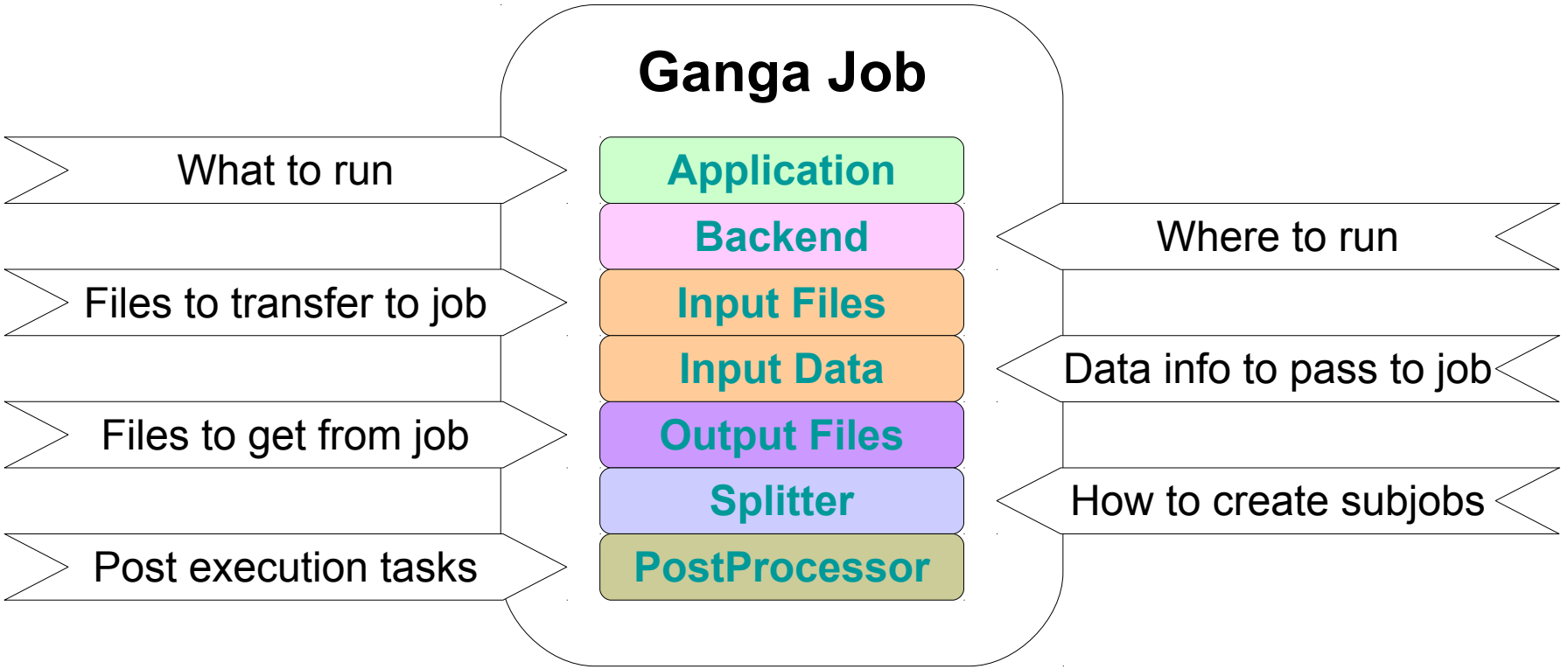
HEPSYSMAN Workshop, 14th January, 2016

Mark Slater, Birmingham University

Ganga Overview

For those who don't knoew, Ganga is a general job management tool used by many HEP experiments and beyond to simplify the submission and monitoring of both local and grid based tasks

It is built on the idea of *independent modules* that perform the various functions required by a typical job



Benefits of Ganga

There are several reasons why Ganga can help with job submission and management:

- It provides a common 'API' for many different backends
- Multiple ways of submission (command line, IPython, Service, etc)
- Hides much of the complexity for monitoring, etc.
- Easily customisable/expandable to suit the need and situation
- Written in plain python so will work with almost everything
- Active developers on hand to help
- Significantly lowers the barrier to entry for the grid
- Has many advanced features for performing complex tasks

Typical Job Submission

Ganga provides you with interfaces to the application your running (if available) and the backend you're running on. This allows you to completely configure the job and how it's run

```
In [2]:j = Job()

In [3]:j.application = Executable()

In [4]:j.application.exe = File("test.sh")

In [6]:j.inputfiles = [ LocalFile("calib.root") ]

In [7]:j.outputfiles = [ LocalFile("out.txt") ]

In [8]:j.backend = LCG()

In [9]:j.backend.requirements.cputime = 3600

In [10]:j.submit()
```

Start by creating a basic Job object

Set what you want the job to do (run the test.sh script)

What extra fies are needed to run the job?

What output is this script going to poduce?

Where do you want it to run (the LCG grid in this case) and configure this as you want

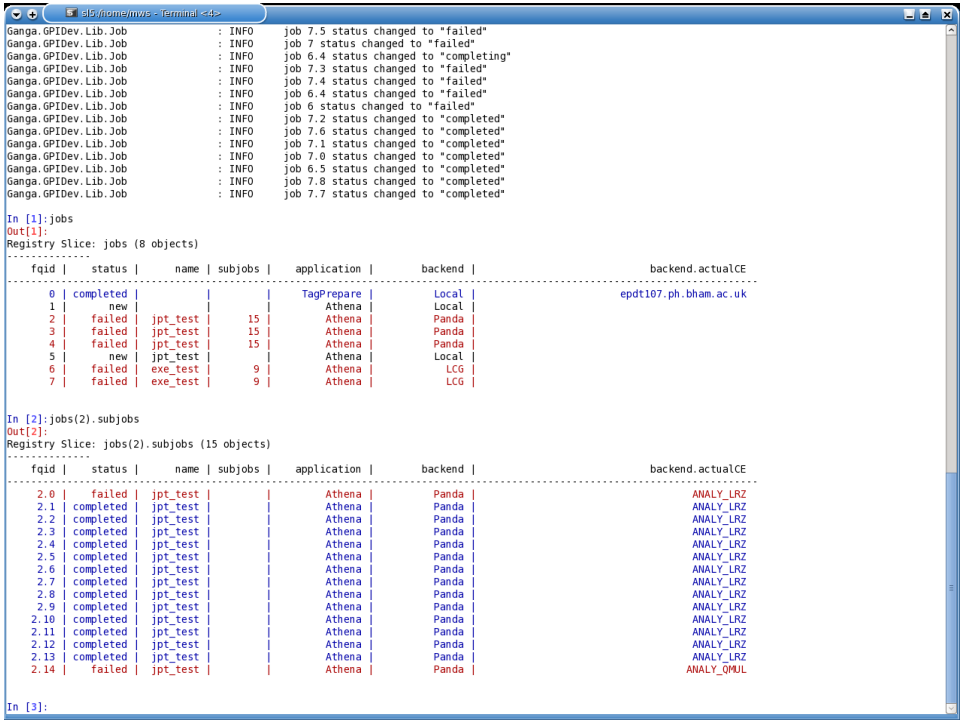
And finally, submit the job!

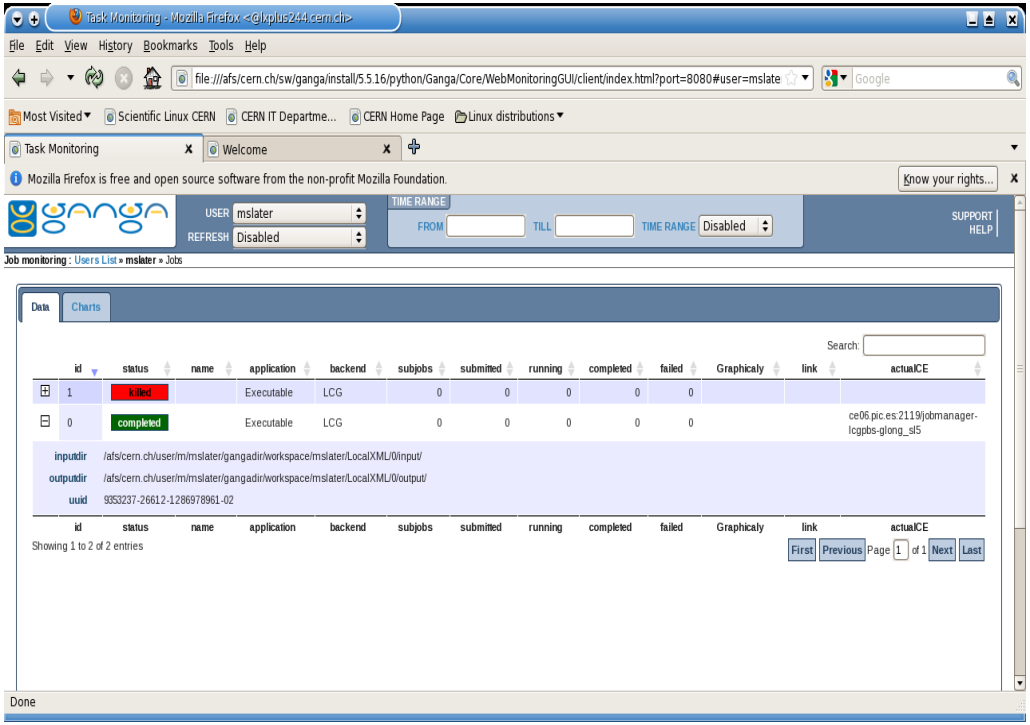
Monitoring Jobs

After submission, you can use Ganga to monitor and manage your jobs using Ganga's IPython interface by just typing 'ganga'

Ganga keeps track of all your jobs over all backends and gives you access to all the information using a local job repository

There is also a web gui available by starting Ganga with the –-webgui option





Configuring Ganga

Ganga is incredibly configurable – the behaviour of all Ganga objects, including default values, can be set in three places:

- .gangarc file which lists all avaiable variables and their description
- From the command line by specifying the options when running Ganga
- At runtime with Ganga using the 'config' dictionary style variable

In addition to this, you can also get help on all the Ganga Objects using the help system and (type 'help(<object>)')

A typical IPython tab-complete service is also available for specifying arguments as well as viewing methods/variables of a class

Scripting in Ganga

As Ganga is written in Python and runs through the IPython interface, you have complete flexibility about how you submit and manipulate your jobs

```
# only tar up once - use this for all jobs
a = Athena()
a.prepare()

for dsln in open("ds_input.txt").readlines():

     # assume you have <dsname>  <jobname>
     toks = dsln.strip().split()

     # submit the job over this dataset
     j = Job()
     j.name = toks[1]
     j.application = a
     j.inputdata = DQ2Dataset()
     j.inputdata.dataset = toks[0]
     j.splitter = DQ2JobSplitter()
     j.splitter.numsubjobs = 20
     j.backend = Jedi()
     j.submit()
```

The script to the left loops over an input file containing a dataset names and job names and submits a job for each. Note that the preparation of the Athena area only happens once!

Also, note that Ganga can store 'job templates' as well so reducing the code to 2-3 lines!

The script to the right loops over any completed jobs and retrieves the datasets produced by each

```
# loop over the set of completed jobs and
# grab the datasets
# (could also use jobs.select(status='completed'))

for j in jobs:

     if j.status == 'completed':
          for sj in j.subjobs():
               sj.outputdata.retrieve()

          # could also do:
          # os.system('dq2-get %s' % \
               j.outputdata.datasetname)
```

To run these, do either:

ganga <scriptname> **OR** ***execfile('<scriptname>') at the python prompt***

Workshop Organisation

This workshop is rather 'freeform' but is based on the Ganga Tutorial that is now on the github page:

https://github.com/ganga-devs/ganga/wiki/Full-Tutorial

This covers all the 'Core' aspects of Ganga, i.e. those that aren't associated with a specific experiment

It is expected that people will work through this, with the rough timings given in the agenda. However, if you want to skip ahead or cover something different then do go ahead!

Disclaimer!!

Though I have tried to make sure everything is working, Ganga is quite a large project and you may encounter problems/bugs/etc. Either with Ganga or with the Tutorial that I didn't come across. Let me know and I'll post a bug report and get it fixed ASAP!

Also – I have very little idea how long this will take!

First Steps With Ganga

There are several ways to get/use Ganga but for today, it is recommended to use the installation on CVMFS through a copy of the GridPP CERN VM:

https://www.gridpp.ac.uk/wiki/GridPP_and_the_CERN_VM_service

This will ensure we don't have to worry about different configurations, etc. etc.

However, if you want to specifically try Ganga with your Tier3 resources, then please do – the version on CVMFS should work without issue!

You should now start working through the tutorial sections below that cover the very basics of Ganga and will give you an idea of how it works:

- Installation and Basic Usage
- Configuration
- Job Manipulation
- Running Executables

Useful Links

The main Ganga web page is still hosted at CERN:

https://ganga.web.cern.ch/ganga/

The Ganga github page hosts the project including the Tutorial Wiki and Issue tracker:

https://github.com/ganga-devs/ganga

As well as directly posting new issues here you can get help at:

project-ganga<AT>cern.ch

OR

project-ganga-developers<AT>cern.ch

Or emailing me: mws@hep.ph.bham.ac.uk