UNIVERSITY OF BIRMINGHAM

Ganga/DIRAC Overview Part 2: Introduction to Ganga

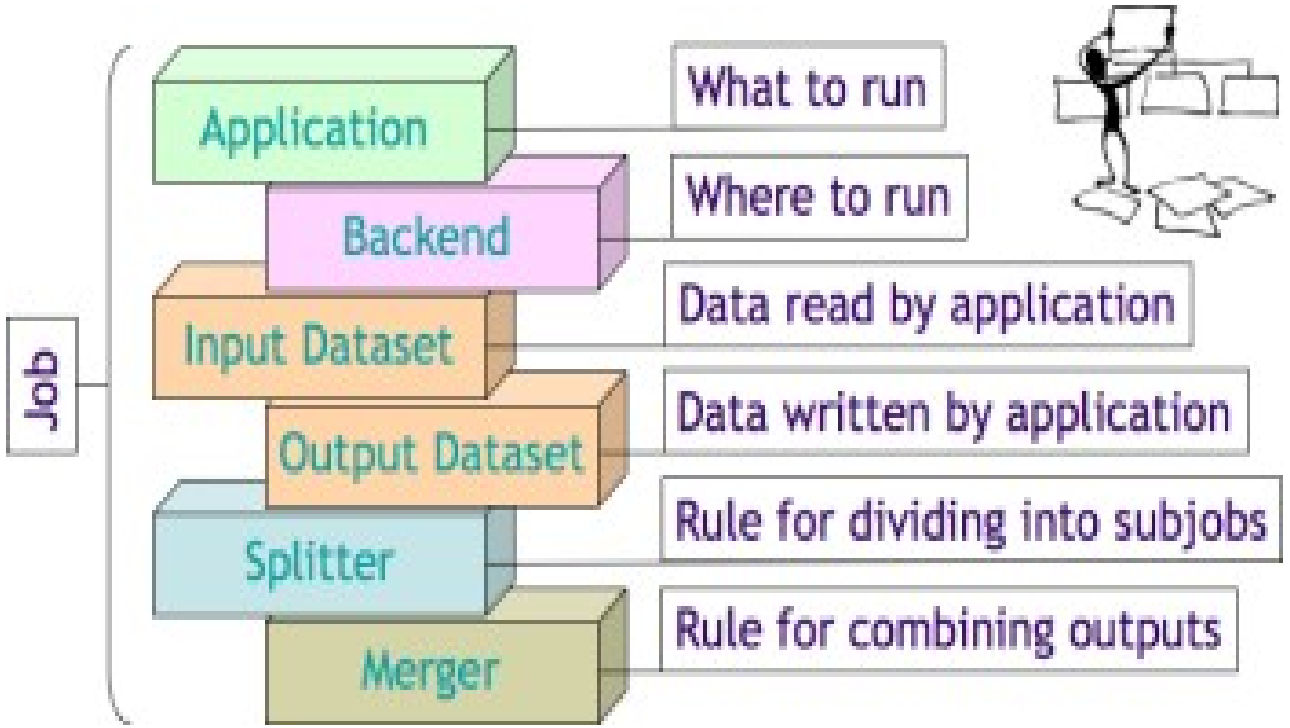
HEPSYSMAN, 13th January, 2014

Mark Slater, Birmingham University

Ganga Overview

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

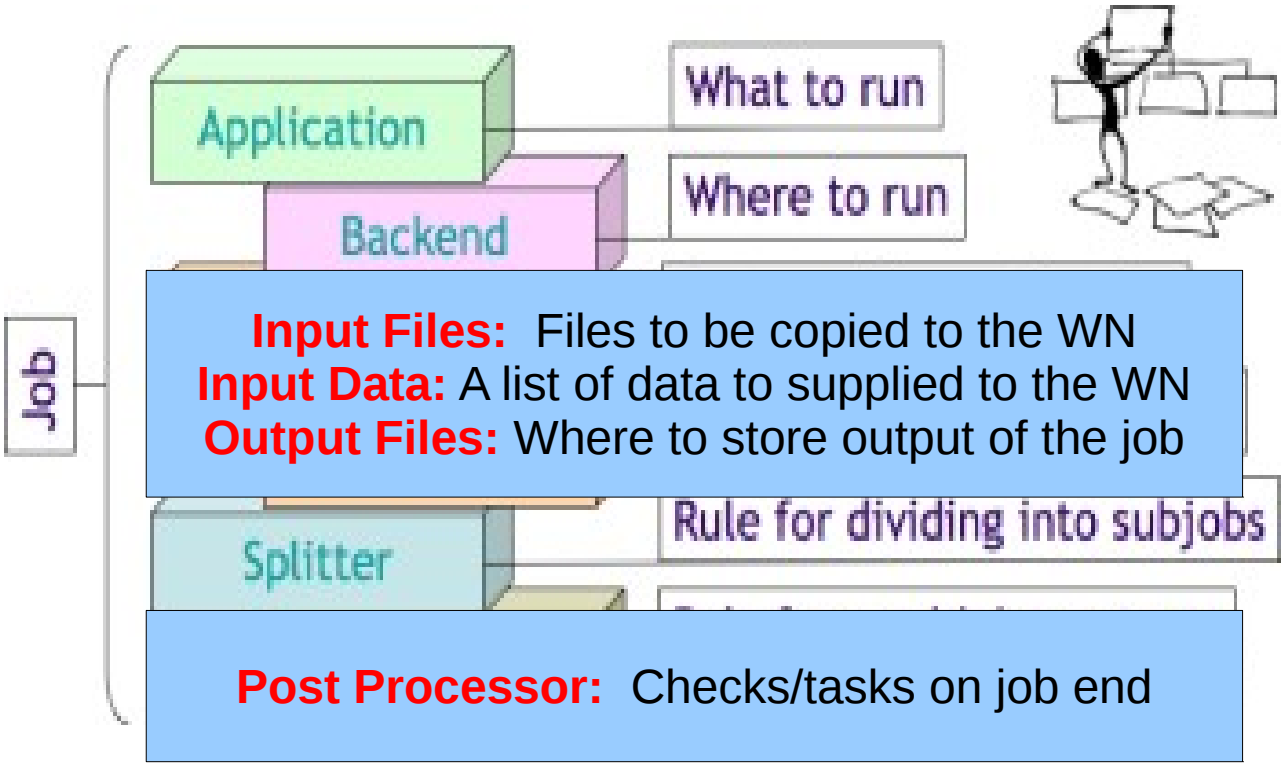
It is built on the idea of *independent modules* that perform the various functions required by a typical job



Ganga Overview

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



Most users deal with mulitple computing backends – one of the principles of Ganga is to simplify changing from e.g. LSF → Dirac

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 allow 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 [5]:j.outputsandbox = ["out.txt"]

In [6]:j.backend = LCG()

In [7]:j.backend.requirements.cputime = 3600

In [8]:j.submit()
```

Start by creating a basic Job object

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

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!

UNIVERSITY OF BIRMINGHAM

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 = Panda()
     j.submit()
```

Moving on from the basic example, here's a typical Atlas submission script that loops over an input file containing a dataset names and job names and submits a job for each
Note the re-use of the Athena application object!

The script to the right loops over any completed jobs and retrieves the DQ2 dataset 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

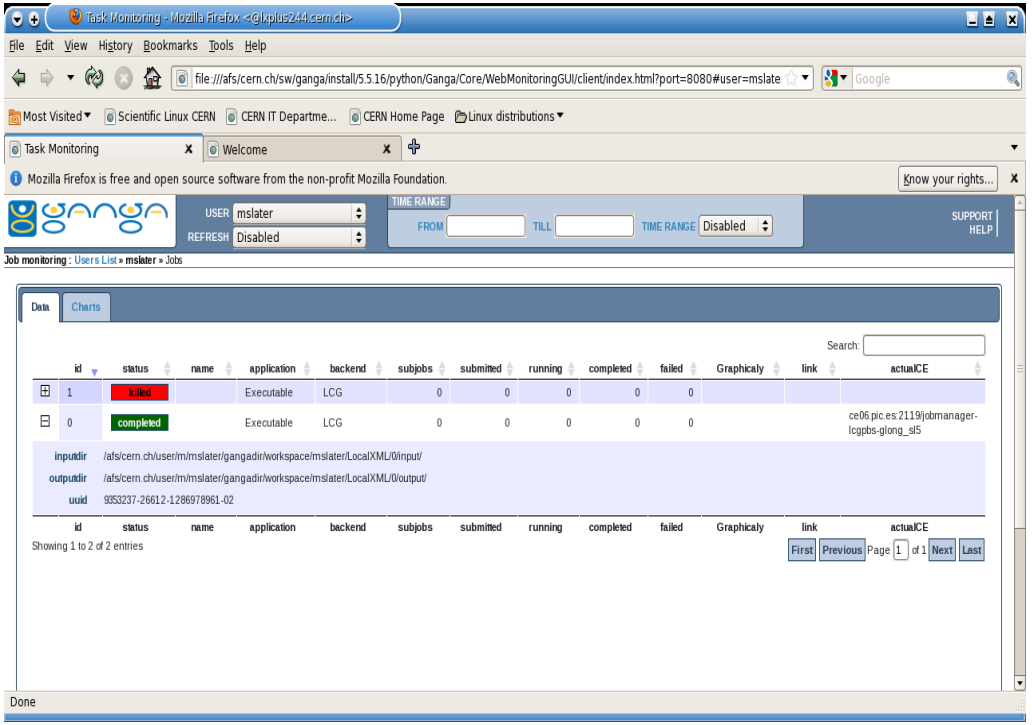
UNIVERSITY OF BIRMINGHAM

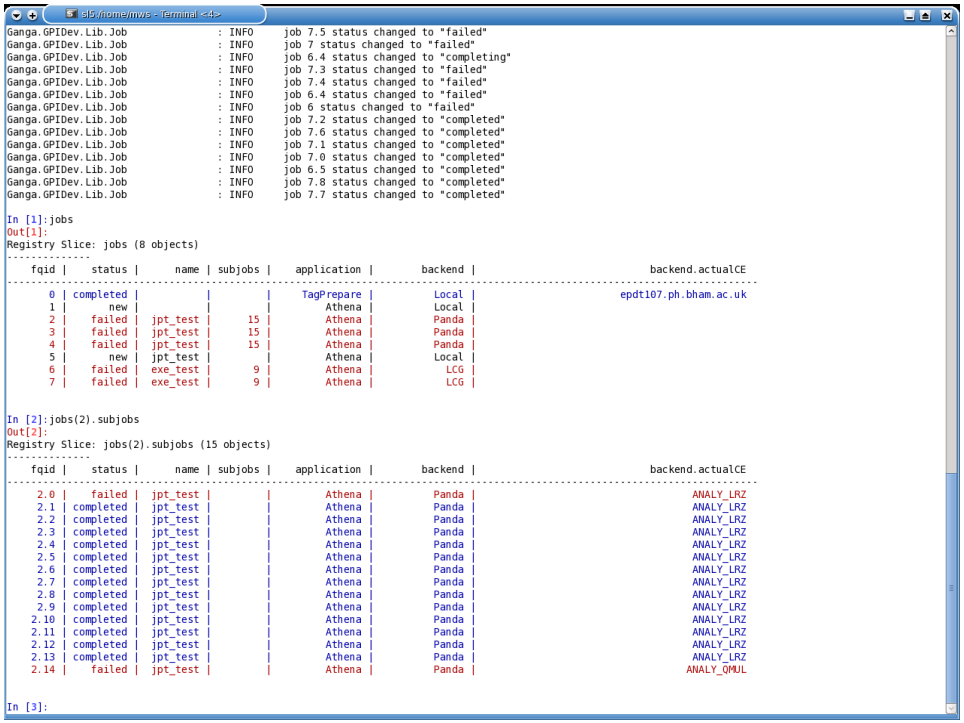
Monitoring Jobs

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

Basic Ganga Demo

Useful Links

The Main Ganga Website

http://ganga.web.cern.ch/ganga/

The General Ganga Manual

http://ganga.web.cern.ch/ganga/user/html/GangaIntroduction6/

Source Code

http://svnweb.cern.ch/world/wsvn/ganga/?

Savannah

https://savannah.cern.ch/bugs/index.php?group_id=195