

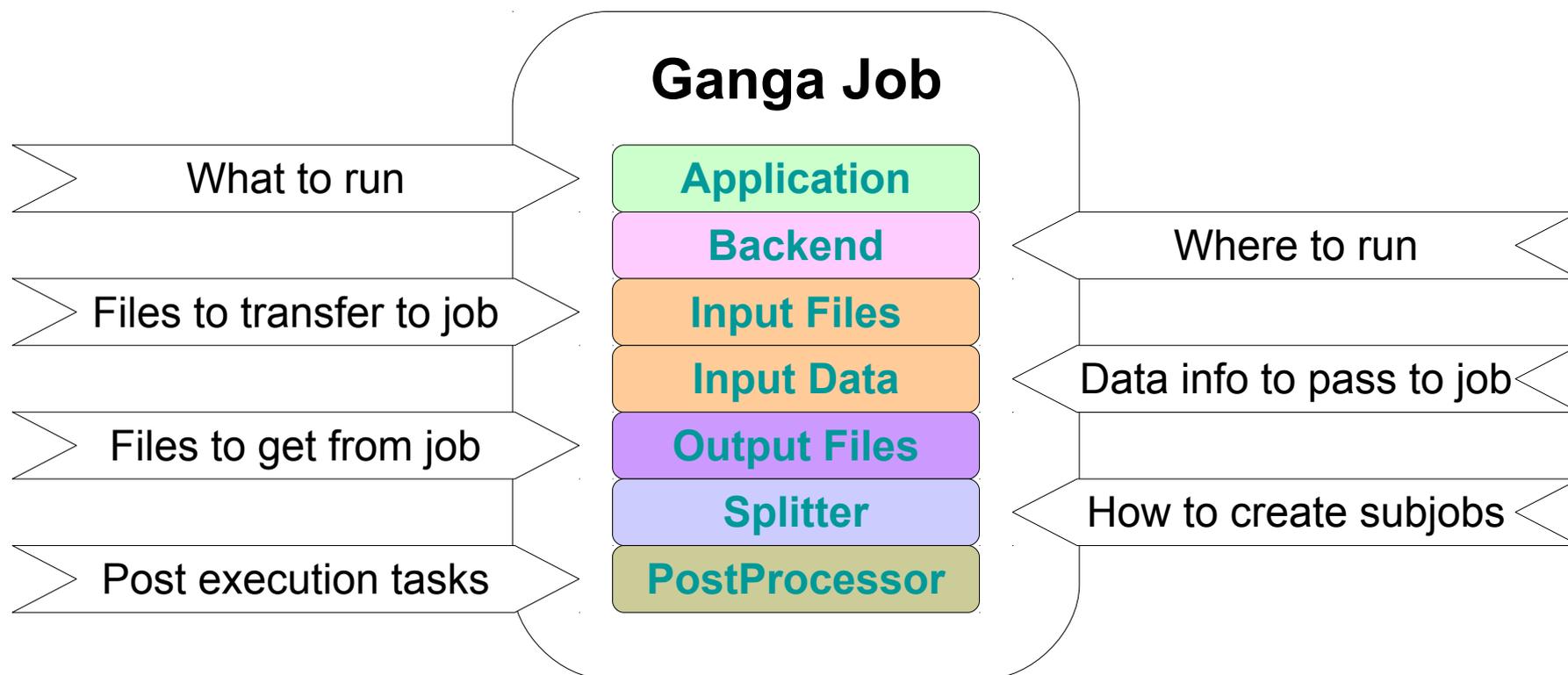


First Steps with Ganga

HEPSYSMAN Workshop, 14th January, 2016
Mark Slater, Birmingham University

For those who don't know, 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



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

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 files are needed to run the job?

What output is this script going to produce?

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

And finally, submit the job!

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

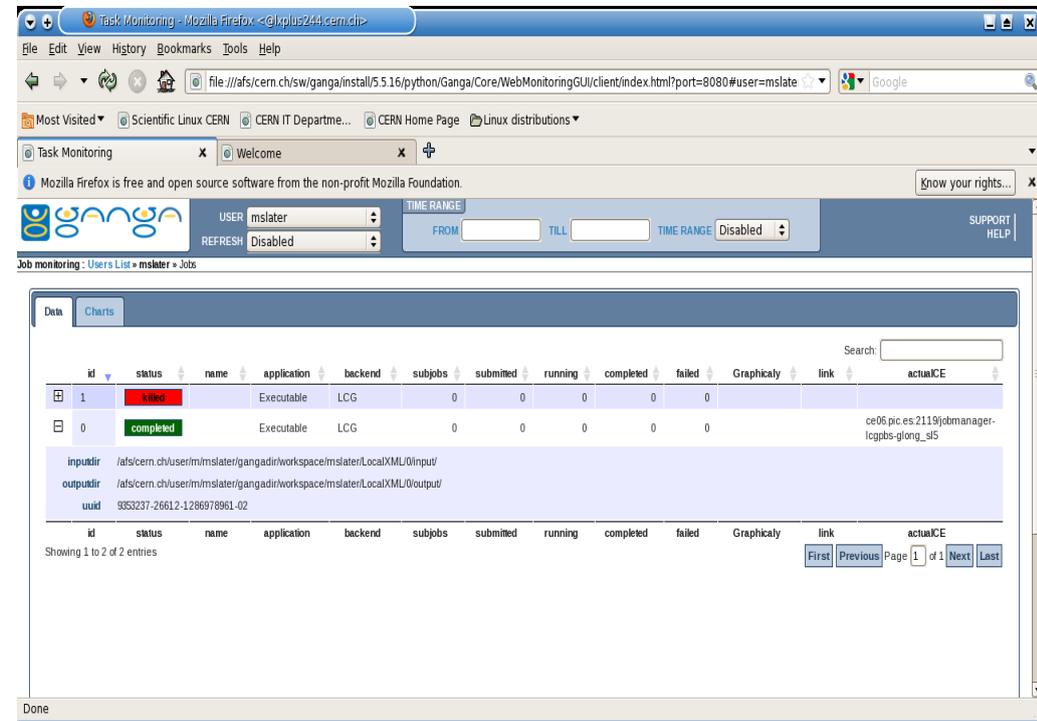
There is also a web gui available by starting Ganga with the --webgui option

```

Ganga.GPDev.Lib.Job : INFO job 7.5 status changed to "failed"
Ganga.GPDev.Lib.Job : INFO job 7 status changed to "failed"
Ganga.GPDev.Lib.Job : INFO job 6.4 status changed to "completing"
Ganga.GPDev.Lib.Job : INFO job 7.3 status changed to "failed"
Ganga.GPDev.Lib.Job : INFO job 7.4 status changed to "failed"
Ganga.GPDev.Lib.Job : INFO job 6.4 status changed to "failed"
Ganga.GPDev.Lib.Job : INFO job 6 status changed to "failed"
Ganga.GPDev.Lib.Job : INFO job 7.2 status changed to "completed"
Ganga.GPDev.Lib.Job : INFO job 7.6 status changed to "completed"
Ganga.GPDev.Lib.Job : INFO job 7.1 status changed to "completed"
Ganga.GPDev.Lib.Job : INFO job 7.0 status changed to "completed"
Ganga.GPDev.Lib.Job : INFO job 6.5 status changed to "completed"
Ganga.GPDev.Lib.Job : INFO job 7.8 status changed to "completed"
Ganga.GPDev.Lib.Job : INFO job 7.7 status changed to "completed"

In [1]: jobs
Out[1]:
Registry Slice: jobs (8 objects)
-----
fqid | status | name | subjobs | application | backend | backend.actualCE
-----
0 | completed | | | TagPrepare | Local | epdt107.ph.bham.ac.uk
1 | new | | | | | 
2 | failed | jpt_test | 15 | Athena | Panda | 
3 | failed | jpt_test | 15 | Athena | Panda | 
4 | failed | jpt_test | 15 | Athena | Panda | 
5 | new | | | | | 
6 | failed | exe_test | 9 | Athena | LCG | 
7 | failed | | | | | 

In [2]: jobs(2).subjobs
Out[2]:
Registry Slice: jobs(2).subjobs (15 objects)
-----
fqid | status | name | subjobs | application | backend | backend.actualCE
-----
2.0 | failed | jpt_test | | Athena | Panda | ANALY_LRZ
2.1 | completed | jpt_test | | Athena | Panda | ANALY_LRZ
2.2 | completed | jpt_test | | Athena | Panda | ANALY_LRZ
2.3 | completed | jpt_test | | Athena | Panda | ANALY_LRZ
2.4 | completed | jpt_test | | Athena | Panda | ANALY_LRZ
2.5 | completed | jpt_test | | Athena | Panda | ANALY_LRZ
2.6 | completed | jpt_test | | Athena | Panda | ANALY_LRZ
2.7 | completed | jpt_test | | Athena | Panda | ANALY_LRZ
2.8 | completed | jpt_test | | Athena | Panda | ANALY_LRZ
2.9 | completed | jpt_test | | Athena | Panda | ANALY_LRZ
2.10 | completed | jpt_test | | Athena | Panda | ANALY_LRZ
2.11 | completed | jpt_test | | Athena | Panda | ANALY_LRZ
2.12 | completed | jpt_test | | Athena | Panda | ANALY_LRZ
2.13 | completed | jpt_test | | Athena | Panda | ANALY_LRZ
2.14 | failed | jpt_test | | Athena | Panda | ANALY_QMUL
  
```



The screenshot shows the Ganga web GUI in a Mozilla Firefox browser. The URL is `file:///afs/cern.ch/sw/ganga/install/5.16/python/Ganga/Core/WebMonitoringGUI/client/index.html?port=8080#user=mslater`. The page displays job monitoring information for user 'mslater'. A table shows job details:

id	status	name	application	backend	subjobs	submitted	running	completed	failed	Graphically	link	actualCE
1	failed	Executable	LCG		0	0	0	0	0			
0	completed	Executable	LCG		0	0	0	0	0			ce06.pic.es2119jrbmanager-lcgbs-glong_sl5

Additional details for job 0 are shown below the table:

```

inputdir /afs/cern.ch/user/m/mslater/gangadir/workspace/mslater/LocalXML0/input/
outputdir /afs/cern.ch/user/m/mslater/gangadir/workspace/mslater/LocalXML0/output/
uuid 9353237-26612-1286978961-02
  
```

The page also includes a search bar, navigation buttons (First, Previous, Page 1 of 1, Next, Last), and a status bar at the bottom showing 'Showing 1 to 2 of 2 entries'.

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

- .gangarc file which lists all available 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

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

To run these, do either:

ganga <scriptname>

OR

execfile('<scriptname>') at the python prompt

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!

```
# 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)
```

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!

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 ●

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](mailto:project-ganga@cern.ch)

OR

[project-ganga-developers<AT>cern.ch](mailto:project-ganga-developers@cern.ch)

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