Ganga: flexible use of of virtualisation for user based large scale computations

Ulrik Egede, Monash University
PyHEP

13 July 2020

@UlrikEgede

What is Ganga?

Many scientific computations are too large to take care of by simply running a script from the command line and waiting for it to execute. To get around this many different systems has been used over the years

- Getting your work done like this often means that it gets broken into multiple pieces - tedious and error prone

Ganga will aim to solve this:

- Debug and develop locally, run globally;
- Pack up ancillary files;
- Split your task up, submit each of the smaller pieces, keep track of which fail, resubmit them;
- Merge all the pieces together.

How is it implemented?

Ganga is an mainly interactive framework implemented in Python

Presents user with an interactive iPython prompt

A well defined and fully documented API shields user from internal implementation

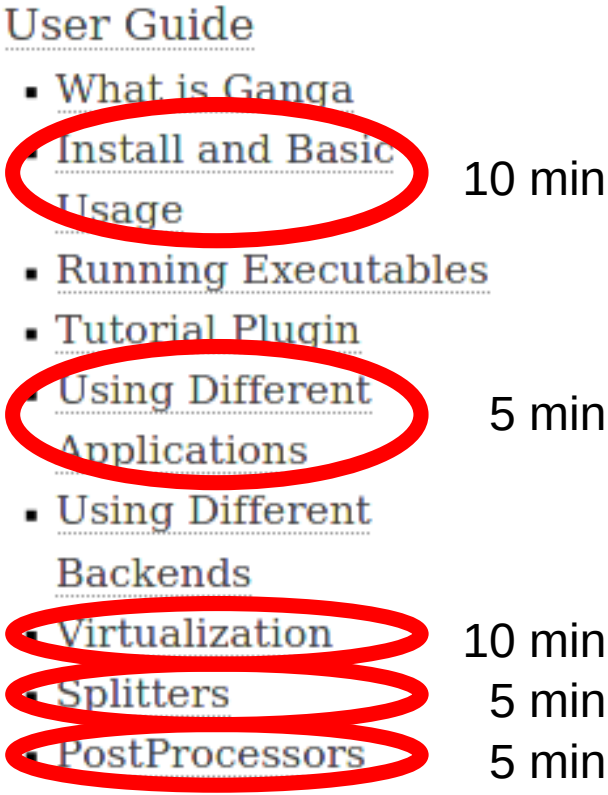
Central to API is the *Job* that defines a given large computational task

- What to execute: application
- Where to run: backend
- How to divide task: splitter
- Which data to use: inputdata
- Which ancillary files: inputfiles
- Which output: outputdata
- Which environment: virtualization
- Put pieces together: merger

Today's tutorial

We will go through part of the User Guide

- Installation – today via Binder
- Basic usage of defining a job and running it
- Different applications
- Virtualization
- Splitters
- Postprocessors



https://ganga.readthedocs.io/en/tutorialbinder/UserGuide/index.html

Questions on Slido: https://app.sli.do/event/lnhlklul

Install and basic usage

Need to know how to create a job, how to submit it and how to look at the output

```
j = Job()
j.submit()
```

```
Ganga In [1]: jobs
Ganga Out [1]:

Registry Slice: jobs (1 objects)
--------------
    fqid |    status |      name | subjobs |    application |        backend |
-------------------------------------------------------------------------------
       0 | completed |           |         |     Executable |          Local |
```

```
j = jobs(0)
j.peek('stdout')
```

https://ganga.readthedocs.io/en/tutorialbinder/UserGuide/InstallAndBasicUsage.html

Using different applications

Will as an example look at a tutorial application that is dedicated to factorizing prime numbers

- Illustrates how applications can make the use of dedicated applications easier

```
j = Job(application = PrimeFactorizer(number=1527), inputdata = PrimeTableDataset())
```

- Applications are defined for several experimental collaborations
  - Take care of compilation, configuration, collection of shared libraries, ...

https://ganga.readthedocs.io/en/tutorialbinder/UserGuide/UsingDifferentApplications.html

Virtualization

If you have a container defined that has your full environment in terms of software, you can now use this in Ganga.

Makes running on remote heterogeneous resources easier

- A container that provides the weather forecast for any airport in the world

```
j1 = Job(name='Weather', \
         virtualization=Docker(image='uegede/weather:1.2'), \
         application=Executable(exe='weather', args=['MEL']))
```

- An example illustrating that we can run a Fedora container in the Debian based Binder

```
j2 = Job(name='Fedora', \
         virtualization=Docker(image='fedora:latest'), \
         application=Executable(exe='cat', args=['/etc/redhat-release']))
```

https://ganga.readthedocs.io/en/tutorialbinder/UserGuide/Virtualization.html

Splitters and postprocessors

The *splitters* define how your job can be broken up in pieces for individual execution

- Can be based on the data, arguments, random number seeds

The *postprocessors* define what actions should be taken when monitoring discovers that job is done

- Merging output from subjobs crated by splitter
- Send notification
- Run short local job on output

https://ganga.readthedocs.io/en/tutorialbinder/UserGuide/Splitters.html
https://ganga.readthedocs.io/en/tutorialbinder/UserGuide/PostProcessors.html