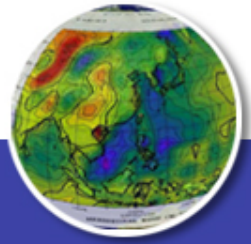
Ganga
The Job Submission Tool

WeiLong Ueng

wlueng@twgrid.org



Objectives

- **This tutorial gives users to understand**
  - Why require Ganga in Grid environment
  - What advantages of Ganga
  - The Architecture of Ganga
  - The Implementation of Ganga
  - How to use Ganga for job management

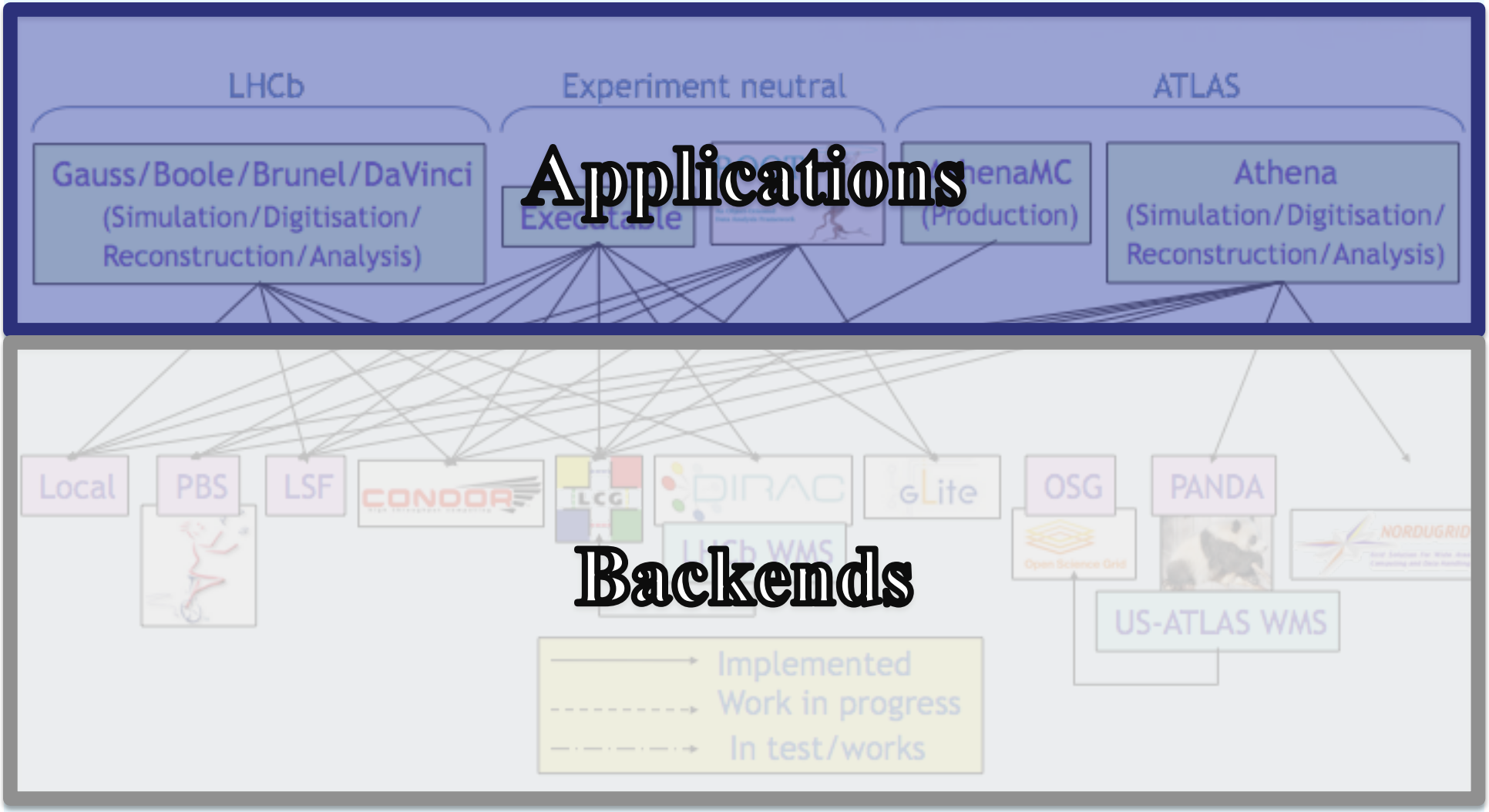
Outline

- Overview
- Ganga Job Components
- Ganga Architecture
- Ganga Interface
- Ganga & DIANE
- Ganga - Use Case

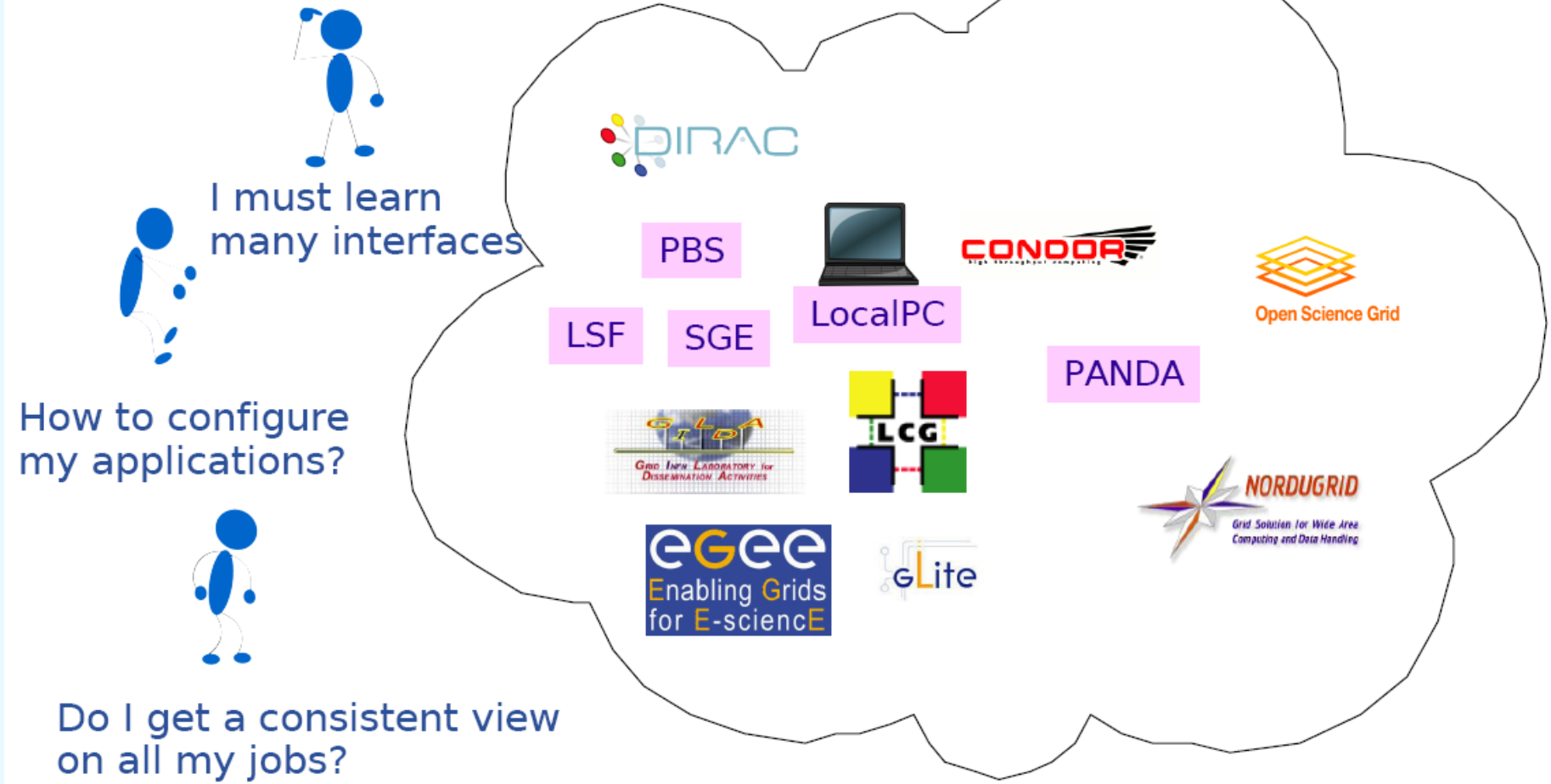
Background

- **Growing numbers of distributed system**
  - Scaling of distributed systems, from local batch systems and community-specific services to generic, global Grid infrastructures
  - Debug on PC, then perform small-scale testing in local resources and finally run at full-scale in globally distributed Grids
- **Supports Different Backends**
  - Backends: PBS, Condor, LCG, EDS… etc
  - How to deploy on various backends in an efficient way

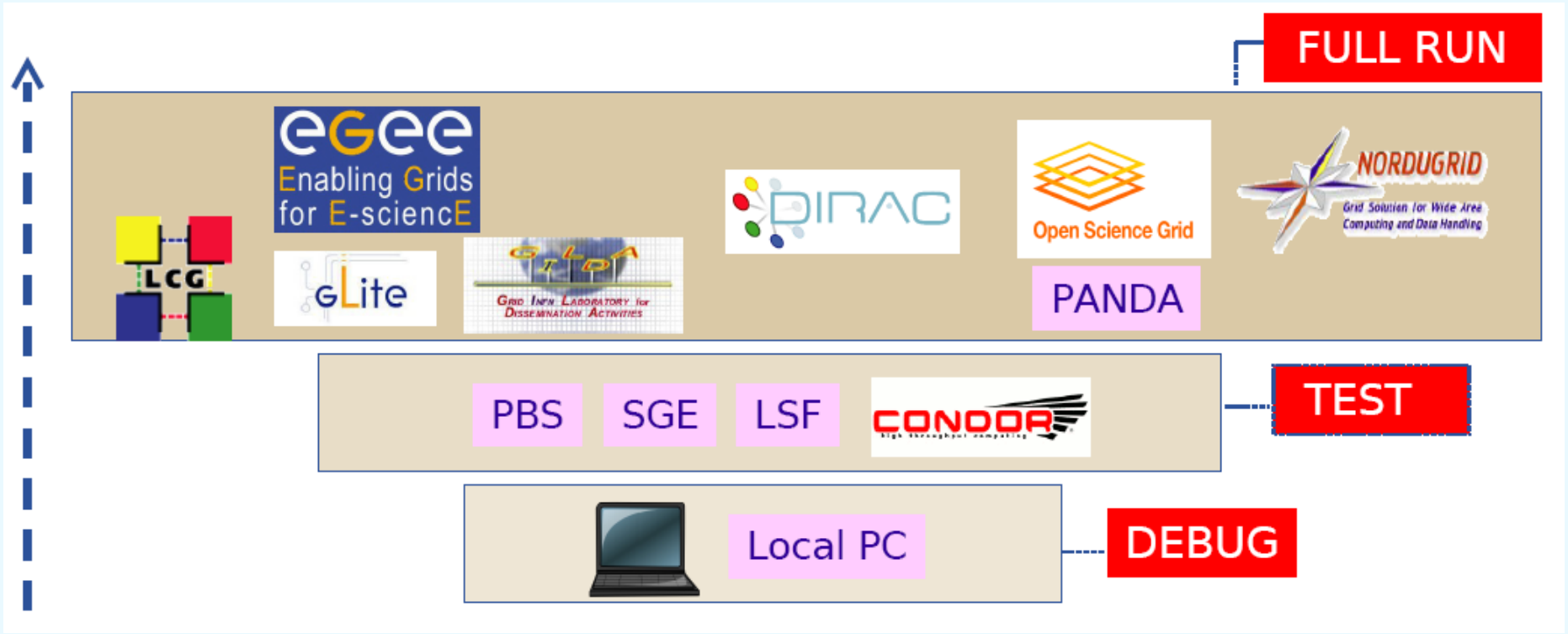
Motivation


LHCb
Experiment neutral
ATLAS
Gauss/Boole/Brunel/DaVinci
(Simulation/Digitisation/
Reconstruction/Analysis)
Executable
AthenaMC
(Production)
Athena
(Simulation/Digitisation/
Reconstruction/Analysis)
Applications
Local
PBS
LSF
CONDOR
LCG
DIRAC
gLite
OSG
PANDA
NORDUGRID
LHCb WMS
US-ATLAS WMS
Backends
Implemented
Work in progress
In test/works

Motivation

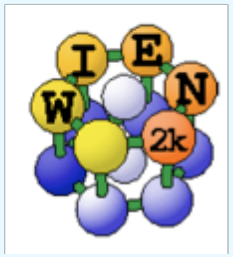

I must learn
many interfaces
How to configure
my applications?
Do I get a consistent view
on all my jobs?
DIRAC
PBS
LocalPC
CONDOR
Open Science Grid
LSF
SGE
PANDA
LCG
GILDA
Grid Infn. Laboratory for Dissemination Activities
NORDUGRID
Grid Solution for Wide Area Computing and Data Handling
eGee
Enabling Grids for E-sciencE
gLite

Motivation


FULL RUN
eGee
Enabling Grids for E-sciencE
LCG
gLite
GILDA
Grid Infn Laboratory for Dissemination Activities
DIRAC
Open Science Grid
PANDA
NORDUGRID
Grid Solution for Wide Area Computing and Data Handling
PBS
SGE
LSF
CONDOR
TEST
Local PC
DEBUG

Applications

- Originally design to meet the needs of the ATLAS and LHCb for a Grid user interface
- configuring and running applications based on the Gaudi / Athena framework common to the two experiments



Why We Need Ganga

without Ganga

```
Executalbe = "echo";
StdOut = "stdout";
StdErr = "stderr";
Arguments = "hello world";
... ...
```

write JDL

start grid proxy

```
voms-proxy-init -voms atlas
```

submit job

```
glite-wms-job-submit ... hello.jdl
```

check job status

```
glite-wms-job-status ... xxxx:yyyy:zzzz
```

get job outputs

```
glite-wms-job-outputs ... xxxx:yyyy:zzzz
```

with Ganga

```
j = Job(backend=LCG(middleware='glite'))
j.application = Executable()
j.application.exec = 'echo'
j.application.args = ['hello world']
j.submit()

j.status
```

- Ganga knows about the application setup/preparation in terms of backend jobs
- Ganga does bookkeeping for you
- Ganga offers programming interface natively in python

What is Ganga

- **Ganga is an easy-to-use frontend for job definition and management**
- **Simplified use of Grid**
- **allows trivial switching between testing on a local batch system and large-scale processing on Grid resources**
- **implemented in Python**
- **The latest version: GANGA 5.4.0**
  - released 28 Oct, 2009

The Purpose of Ganga

- **Easy for users to create, submit and monitor the progress of jobs**
- **Move transparently from different resources**
- **Keeps track of all jobs and their status through a repository that archives all information between independent Ganga sessions**
- **Simplify the progression from**
  - Rapid prototyping on a local computer
  - Small-scale tests on a local batch system
  - The analysis of a large dataset using Grid resources

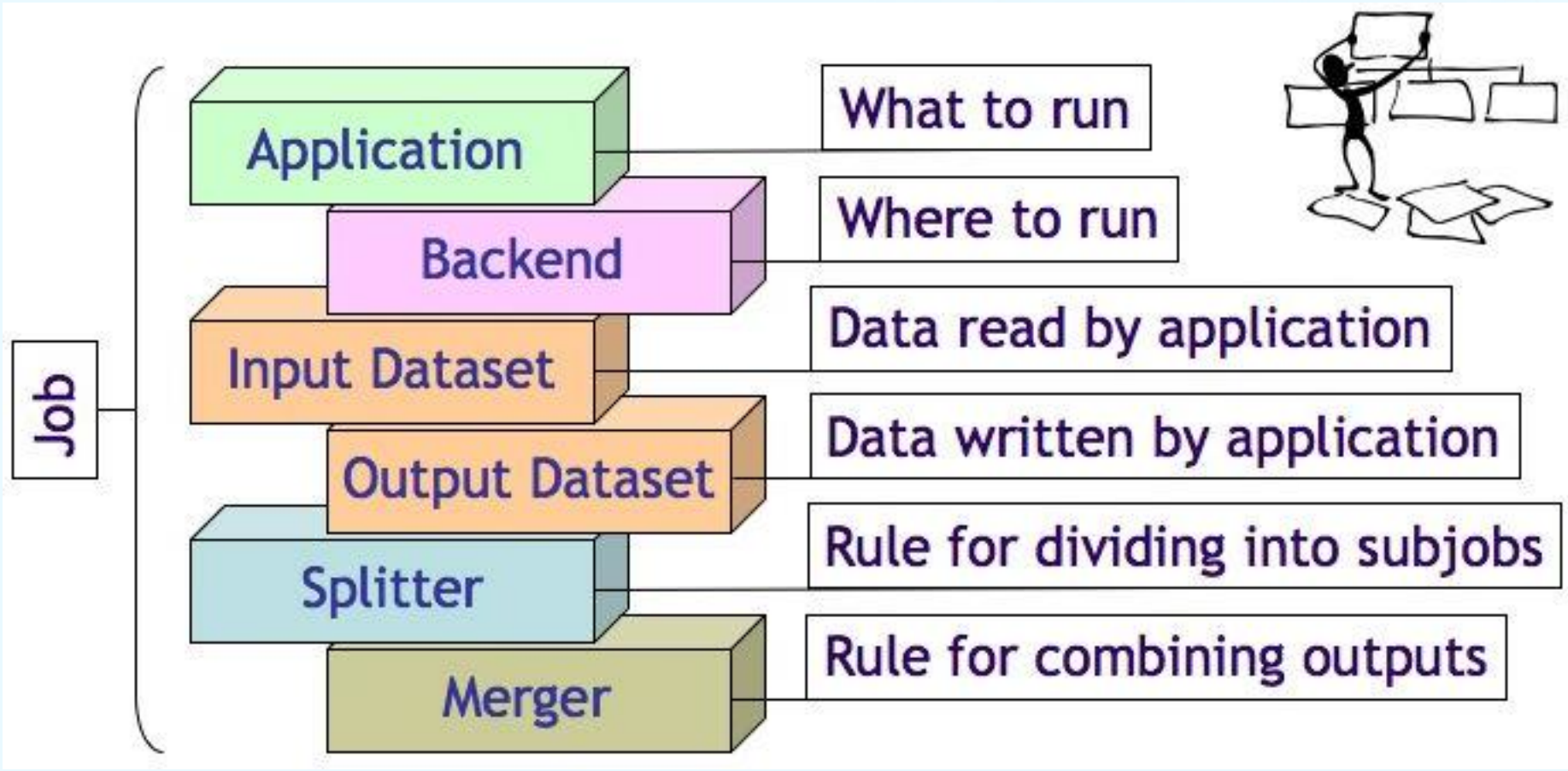
Features of Ganga

- **A user- and application-oriented layer above existing job submission and management technologies**
- **Interchangable and Interoperability**
    - Ex. Globus, Condor, UNICORE or gLite
- **Encapsulate all low-level setup of the application**
- **Hidden the heterogeneity of backends and data access**

Job Component

- The code to execute
- Input data for processing
- Data produced by the application
- The specification of the required processing environment
- Post-processing tasks
- Metadata for bookkeeping

Ganga Job Components



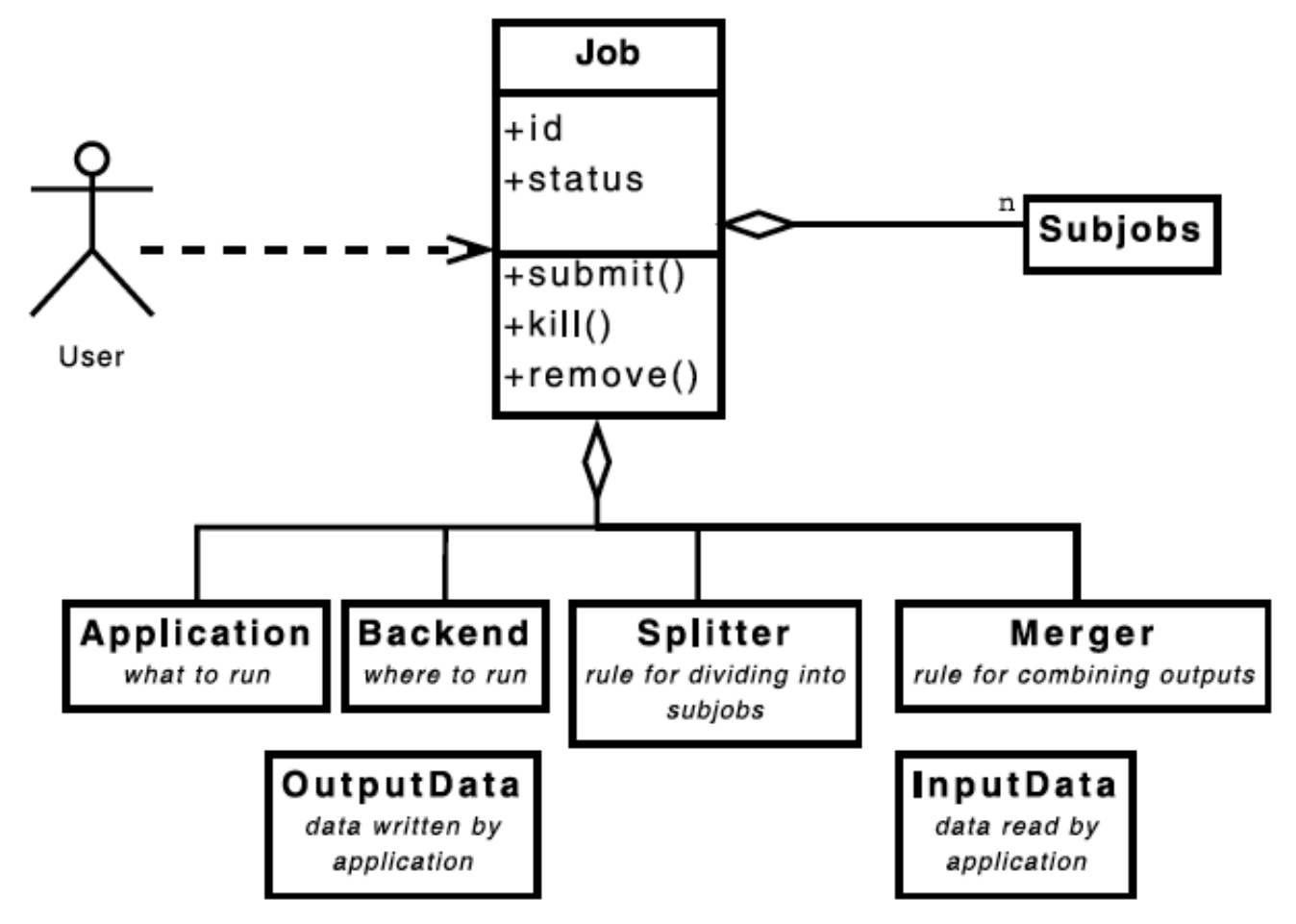
Ganga Job Components

- **Application**
  - The type of computational task
  - Define the software to be run
- **Backend**
  - Define the processing system to be used
  - Including Localhost, batch systems(PBS), Load Sharing Facility(LSF), Sun Grid Engine, Condor, Grid Systems

Ganga Job Components

- **Dataset**
  - Uniquely identify a particular collection of data
  - Provide methods for obtaining information about it, such as its location and size.
  - The Descriptions of Data Collections
    - Vary from different problem domains
- **Splitter**
  - Specify the number of subjobs to be created
- **Merger**
  - Allows for the aggregation of subjob outputs

The Component Structure of A Job



To Form a Complete Job

- **Application**
  - exe: the path to an executable binary or script
  - args: a list of arguments to be passed to the executable
  - env: a dictionary of environment variables and the values they should be assigned before the executable is run

```
j = Job()
j.application = Executable(exe=File('/opt/anotherscript'), args=['-d',File('/
etc/x')])
j.submit()
```

To Form a Complete Job

- **Backend**
  - Parameters describing the behaviour of a processing system
  - Differs among different backend systems
  - Methods for job executions, retrieval of jobs status

```
In [86]:plugins("backends")
Out[86]: ['LSF', 'LCG', 'Dirac', 'gLite', 'PBS', 'Condor', 'Local']
```

To Form a Complete Job

- **Dataset**
  - Datasets
    - the files or databases stored externally
  - Sandbox
    - consists of files transferred from the user's file system together with the job
  - Sandbox mechanism handles small files (typically up to 10MB)

j.inputsandbox = [File(extra_file')]
j.inputsandbox = ['extra_file']

j.outputsandbox = ['b.dat','a*.txt']

To Form a Complete Job

- **Splitter**
  - ArgSplitter:
    - Deals with executing the same task many times over with changing arguments each time
  - args: list of sets of arguments to be passed to an application

```
In [21]:s = ArgSplitter(args=[['A'],['B'],['C']])
In [22]:j = Job(splitter=s)
In [23]:j.submit()
Ganga.GPIDev.Lib.Job          : INFO    submitting job 164
Ganga.GPIDev.Adapters         : INFO    submitting subjob 16400001
Ganga.Lib.Localhost           : INFO    job 16400001 submitted
Ganga.GPIDev.Adapters         : INFO    submitting subjob 16400002
Ganga.Lib.Localhost           : INFO    job 16400002 submitted
Ganga.GPIDev.Adapters         : INFO    submitting subjob 16400003
Ganga.Lib.Localhost           : INFO    job 16400003 submitted
Out[23]: 1
```

To Form a Complete Job

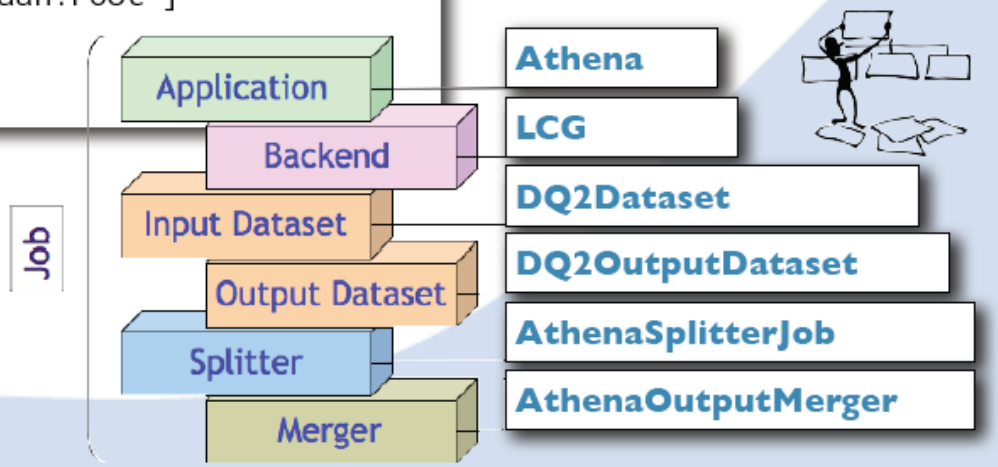
- **Merger**
  - Combing data in a particular format
  - (ex. Text strings or data representing histograms)
  - TextMerger:
    - concate the files of standard output and error returned by a set of subjobs
  - RootMerger:
    - sums the histograms produced in ROOT format

```
In [4]:j.merger=TextMerger()
In [5]:j.merger.files=['stdout']
In [6]:j.merger.ignorefailed = True
In [7]:j.submit()
```

```
In [17]:j.peek()
total 3.0K
-rw-r--r--  1 moscicki sf  903 Jun 11
13:08 stdout.merge_summary
-rw-r--r--  1 moscicki sf 1.2K Jun 11
13:08 stdout
```

Job Representation - Example

```
j = Job()
j.application=Athena()
j.application.prepare(athena_compile=False)
j.application.option_file='$HOME/athena/12.0.5/InstallArea/jobOptio
j.splitter=AthenaSplitterJob()
j.splitter.numsubjobs = 3
j.merger=AthenaOutputMerger()
j.inputdata=DQ2Dataset()
j.inputdata.dataset='csc11.005145.PythiaZmumu.recon.AOD.v11004103'
j.inputdata.match_ce=True
j.outputdata=DQ2OutputDataset()
j.outputdata.outputdata=['AnalysisSkeleton.aan.root']
j.backend=LCG()
j.submit()
```



Job Representation - Example

```
j = Job()
j.application=Athena()
j.application.prepare(athena_compile=False)
j.application.option_file=['$HOME/atlas/testarea/14.2.10/PhysicsAnalysis/
AnalysisCommon/UserAnalysis/run/AnalysisSkeleton_topOptions.py']
j.application.max_events='100'
j.inputdata=DQ2Dataset()
j.inputdata.dataset='trig1_misal1_mc12.005232.PythiaVBFH170wwll.recon.AOD.
v13003003_tid017852'
j.outputdata=DQ2OutputDataset()
j.outputdata.outputdata=['AnalysisSkeleton.aan.root' ]
j.splitter=DQ2JobSplitter()
j.splitter.numsubjobs=3
j.merger=AthenaOutputMerger()
j.backend=LCG()
j.backend.requirements.cloud='DE'
j.submit()
```

Application

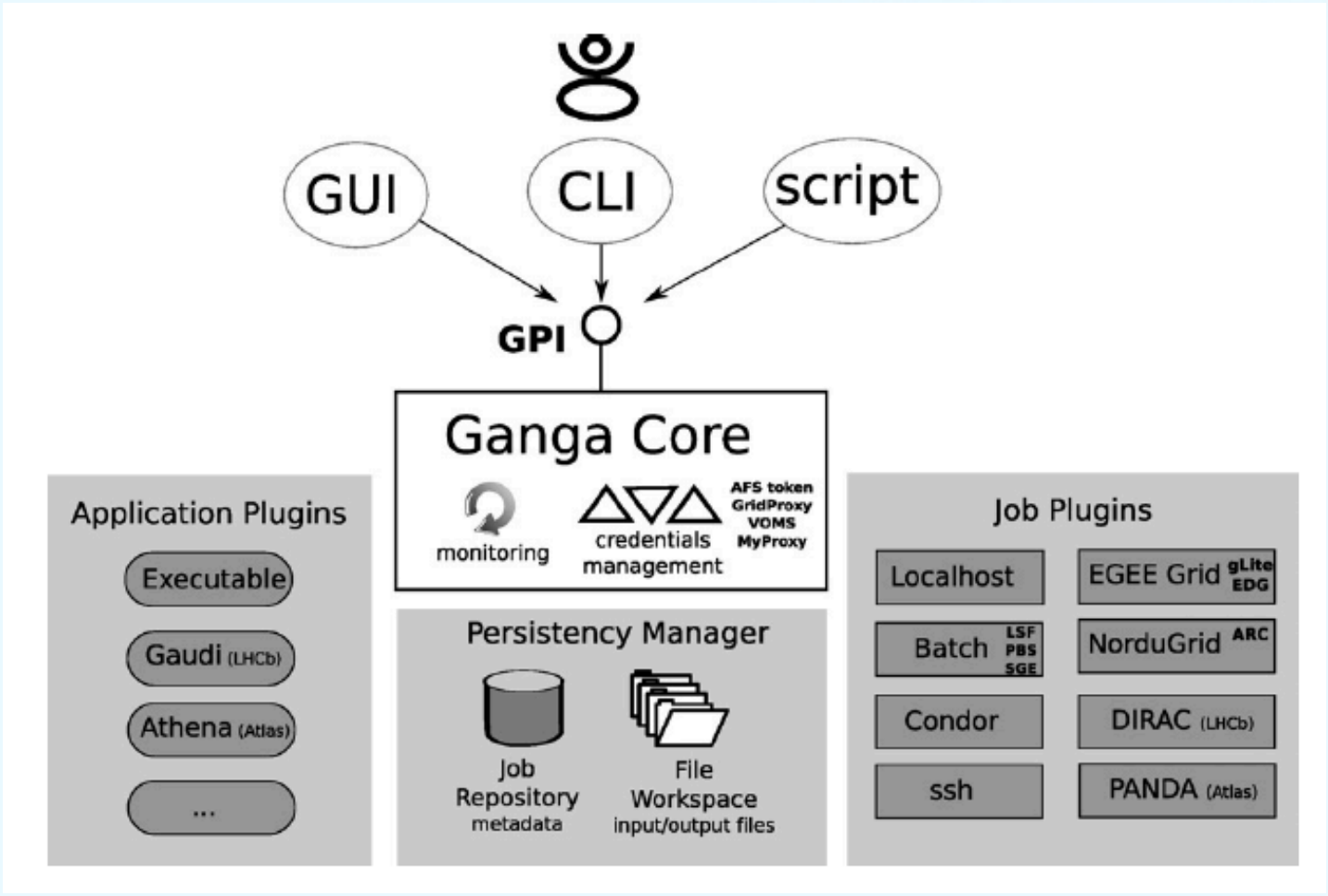
Dataset

Dataset

Splitter & Merger

Backend

Ganga Architecture



Ganga Interfaces

- Built on top of the Ganga Public Interface (GPI)
- A Text-based Command Line in Python reference
- A File-based Scripting Interface
- Graphical User Interface (GUI)

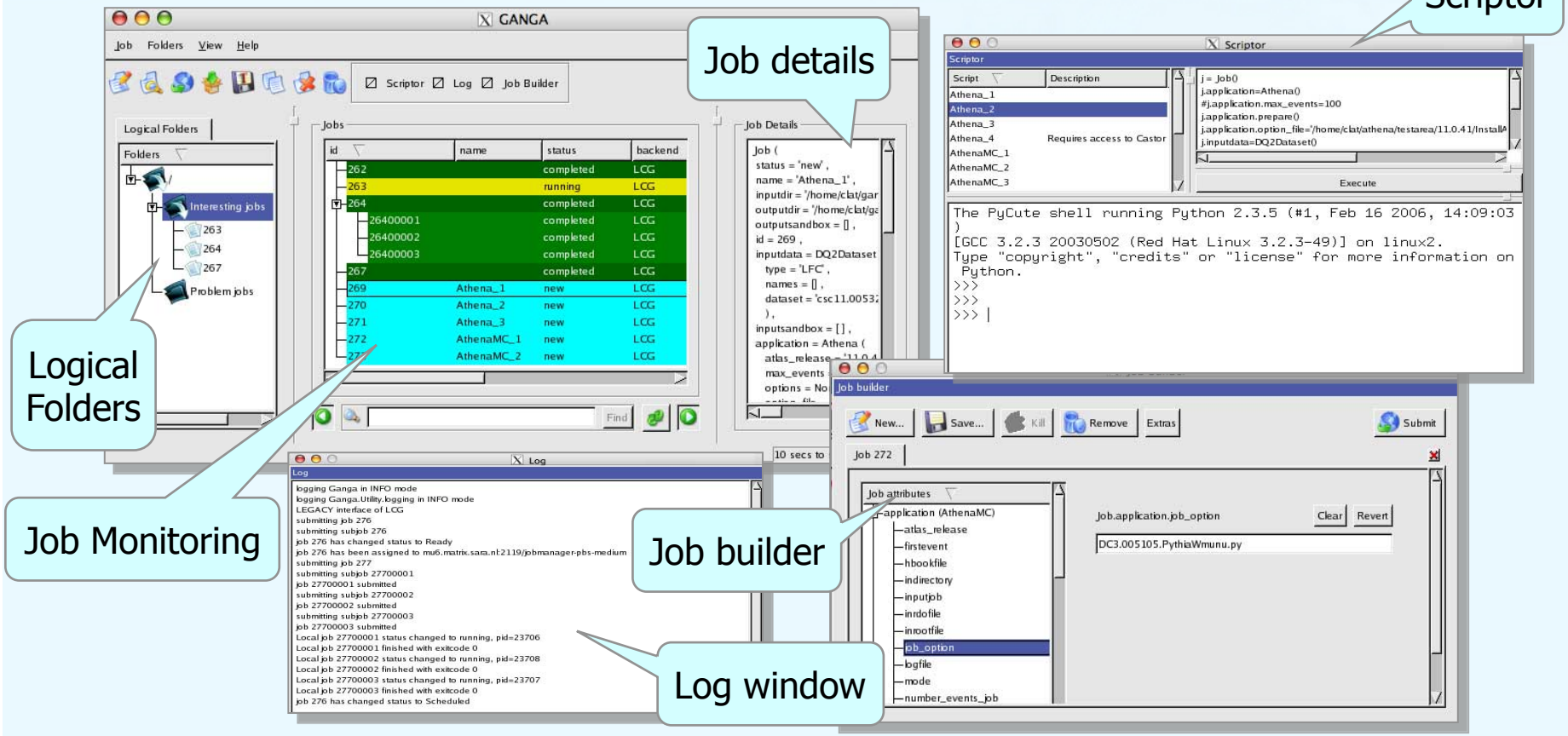
A File-based Scripting Interface

- **From the command line, a script myScript.py can be executed in the Ganga environment using:**
  - Shell> ganga myScript.py
- **myScript.py**

```
import os
dir=os.environ['HOME']+'/mydir'
file=os.listdir(dir)
for line in file:
    print line
    j = Job(Executable(exe=File(dir+'/'+line))
    j.submit
```

An example of a ganga script

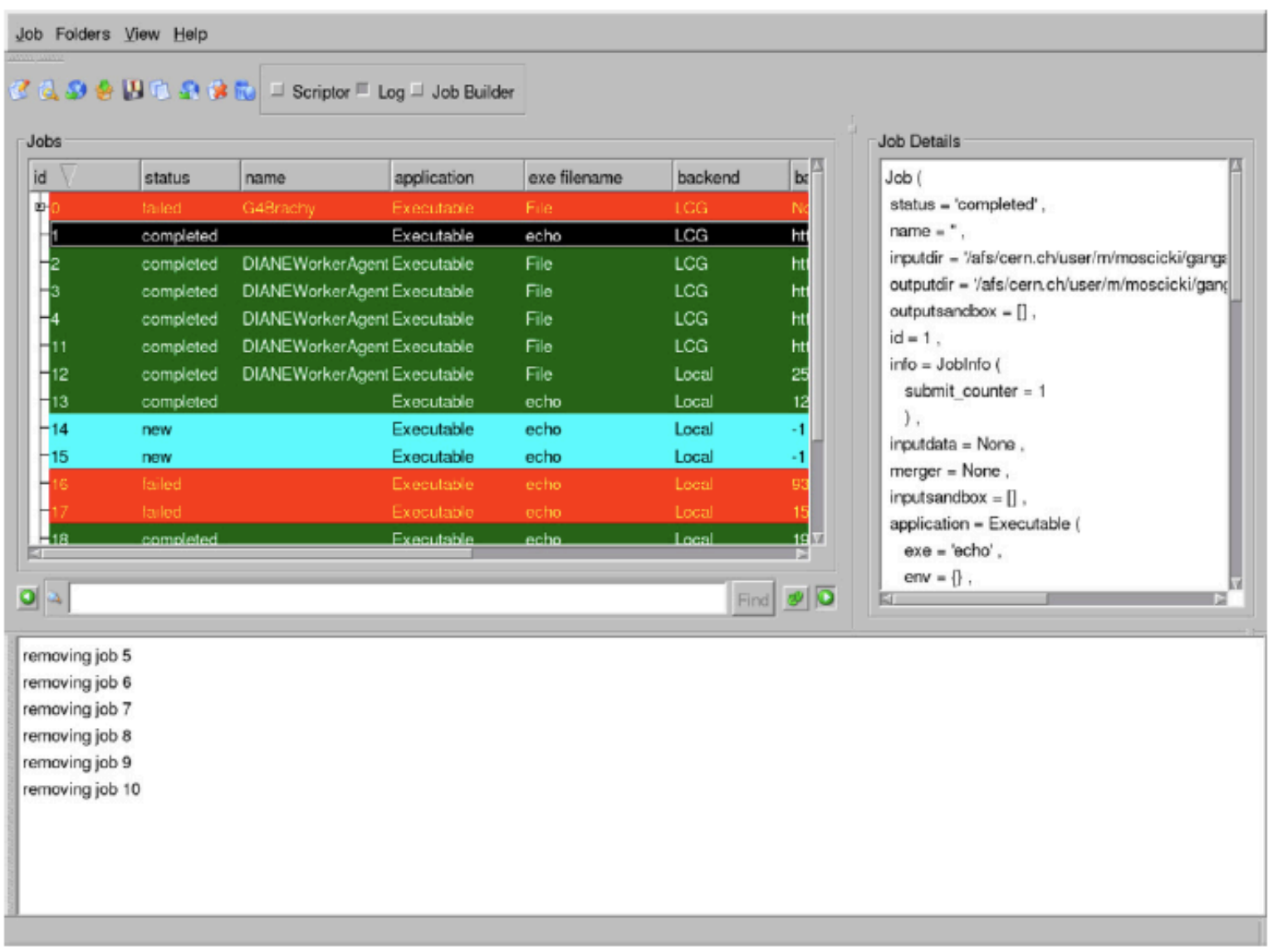
Ganga GUI



There is also a scripting interface (like pAthena)

I will use the line mode……

Ganga GUI


Job Folders View Help
Scriptor Log Job Builder
Jobs
id status name application exe filename backend
0 failed G4Brachy Executable File LCG
1 completed Executable echo LCG
2 completed DIANEWorkerAgent Executable File LCG
3 completed DIANEWorkerAgent Executable File LCG
4 completed DIANEWorkerAgent Executable File LCG
11 completed DIANEWorkerAgent Executable File LCG
12 completed DIANEWorkerAgent Executable File Local
13 completed Executable echo Local
14 new Executable echo Local
15 new Executable echo Local
16 failed Executable echo Local
17 failed Executable echo Local
Find
Job Details
Job (
status = 'completed' ,
name = '' ,
inputdir = '/afs/cern.ch/user/m/moscicki/ganga
outputdir = '/afs/cern.ch/user/m/moscicki/gang
outputsandbox = [] ,
id = 1 ,
info = JobInfo (
submit_counter = 1
) ,
inputdata = None ,
merger = None ,
inputsandbox = [] ,
application = Executable (
exe = 'echo' ,
env = {} ,
removing job 5
removing job 6
removing job 7
removing job 8
removing job 9
removing job 10

Ganga Public Interface (GPI)

- **User-level Interface separated from low-level, internal API**
- **Templates of Job Configurations**
  - Frequently used job configurations
- **Hierarchical JobTree**
  - Jobs can be labelled and organised in a hierarchical jobtree

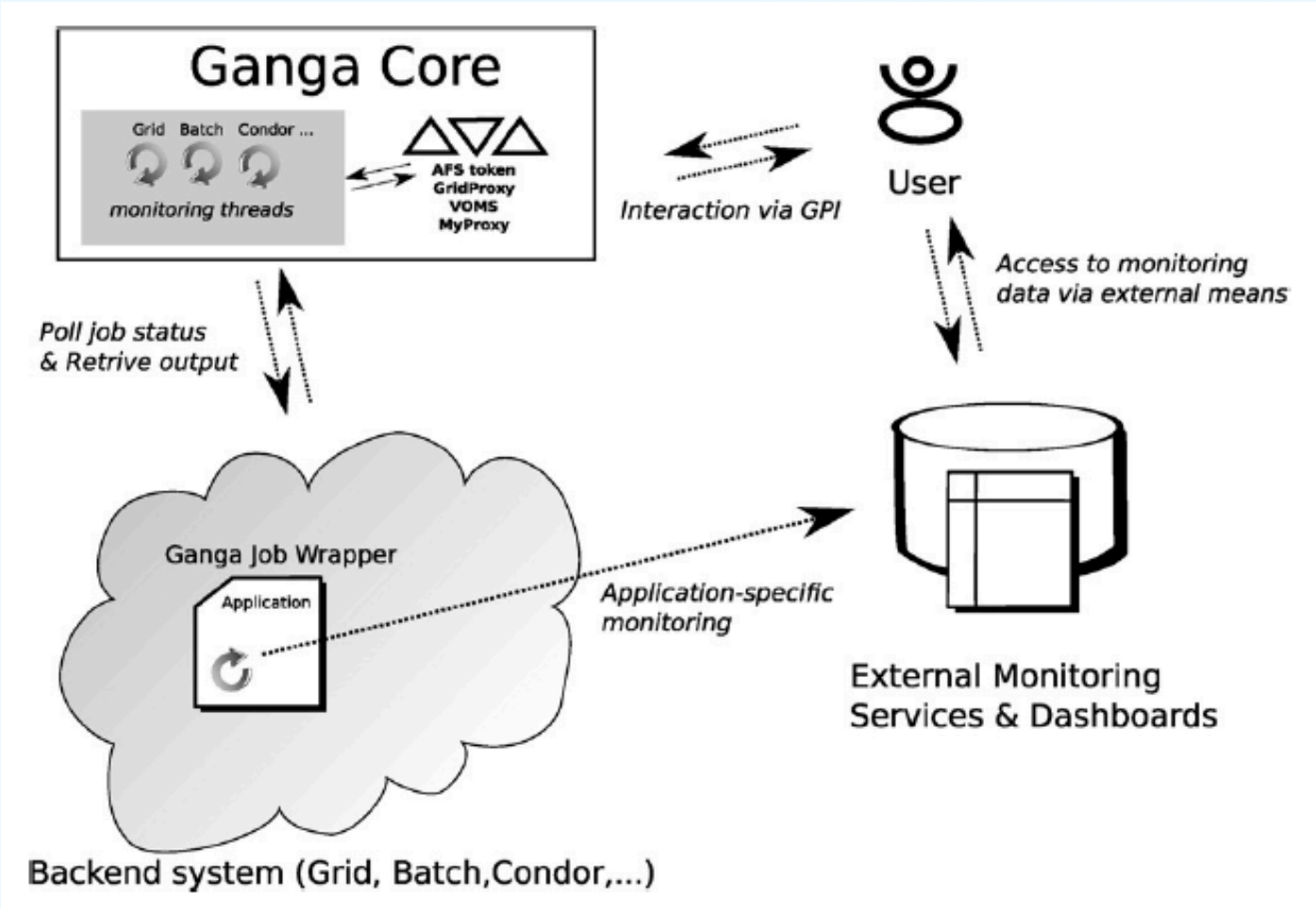
Ganga Core

- **Credentials Management**
  - User credentials, including classic Grid proxies with extensions for VOMS
  - Renew and destroy the credentials using GPI
  - Multiple security models
- **Monitoring**
  - Internal Monitoring
  - External Monitoring

Ganga Core – Job Monitoring

- **Internal Monitoring**
  - Polling job status for varying backends
  - The remaining validity of authentication credentials
- **External Monitoring**
  - Dynamically adding third-party monitoring sensors
  - Monitoring sensor
    - Client side
    - Remote environment
  - Allow collection of both
    - generic execution information
    - Application-specific data

Core - Job Monitoring



Persistency Manager

- **Job Repository Database**
  - Job Persistence and Objects
  - Job Bookkeeping and Metadata
  - Local and Remote Repository
  - Schema Migration
- **File Workspace**
  - The Input and Output files associated with the Jobs

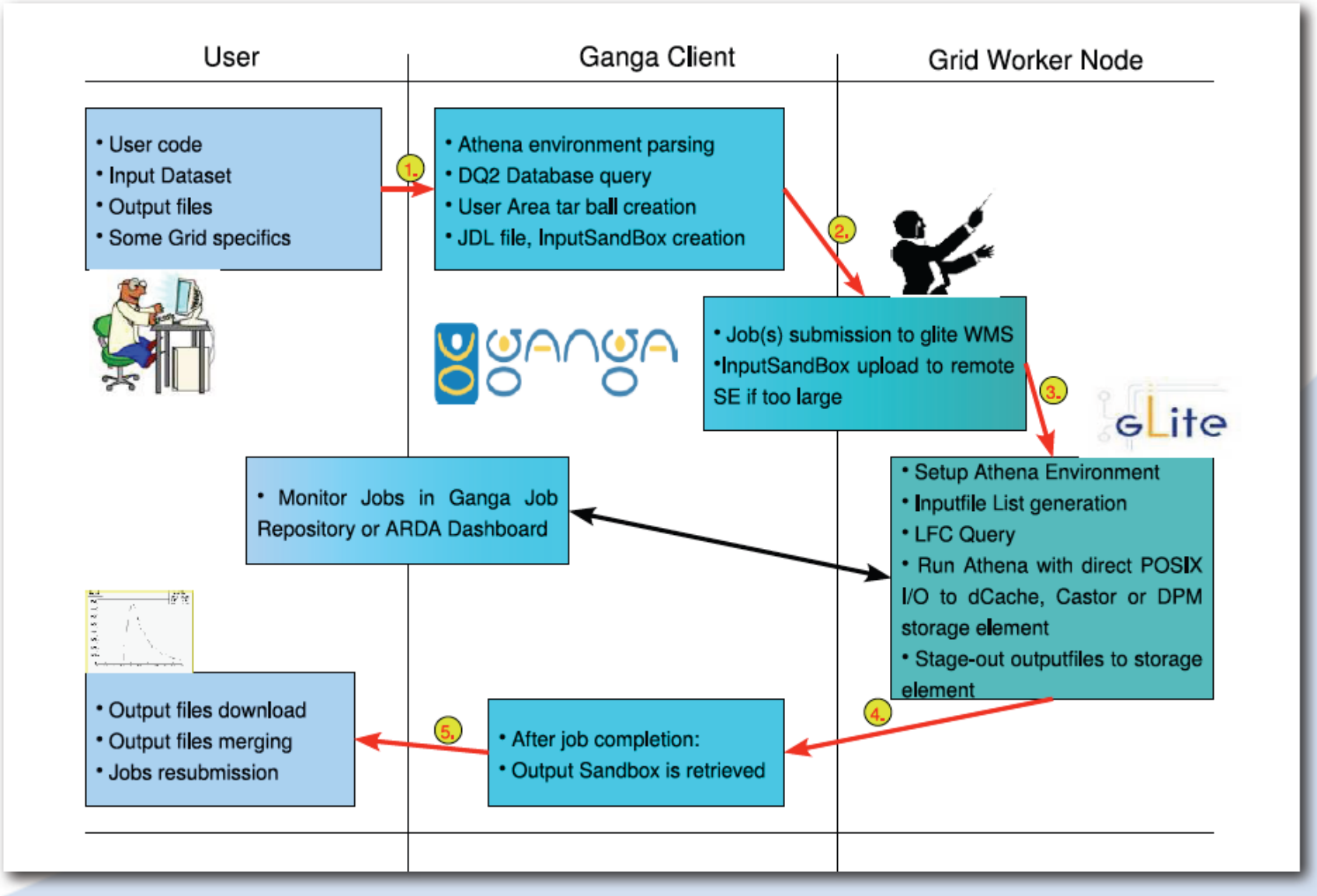
Application Plugins

- **Provides a configuration object which contains backend- independent information:**
  - The application to be run
  - The user code to be executed
  - The values to be assigned to any configurable parameters
  - The data to be processed

Job Plugins

- **Accepts an input configuration object and calls the application runtime handler responsible for the backend-specific part of the application configuration and management:**
  - Job splitting
  - Packaging of user code
  - Submitted jobs to the backend
  - Monitoring job progress
  - Retrieving output files when jobs complete

Ganga Workflow



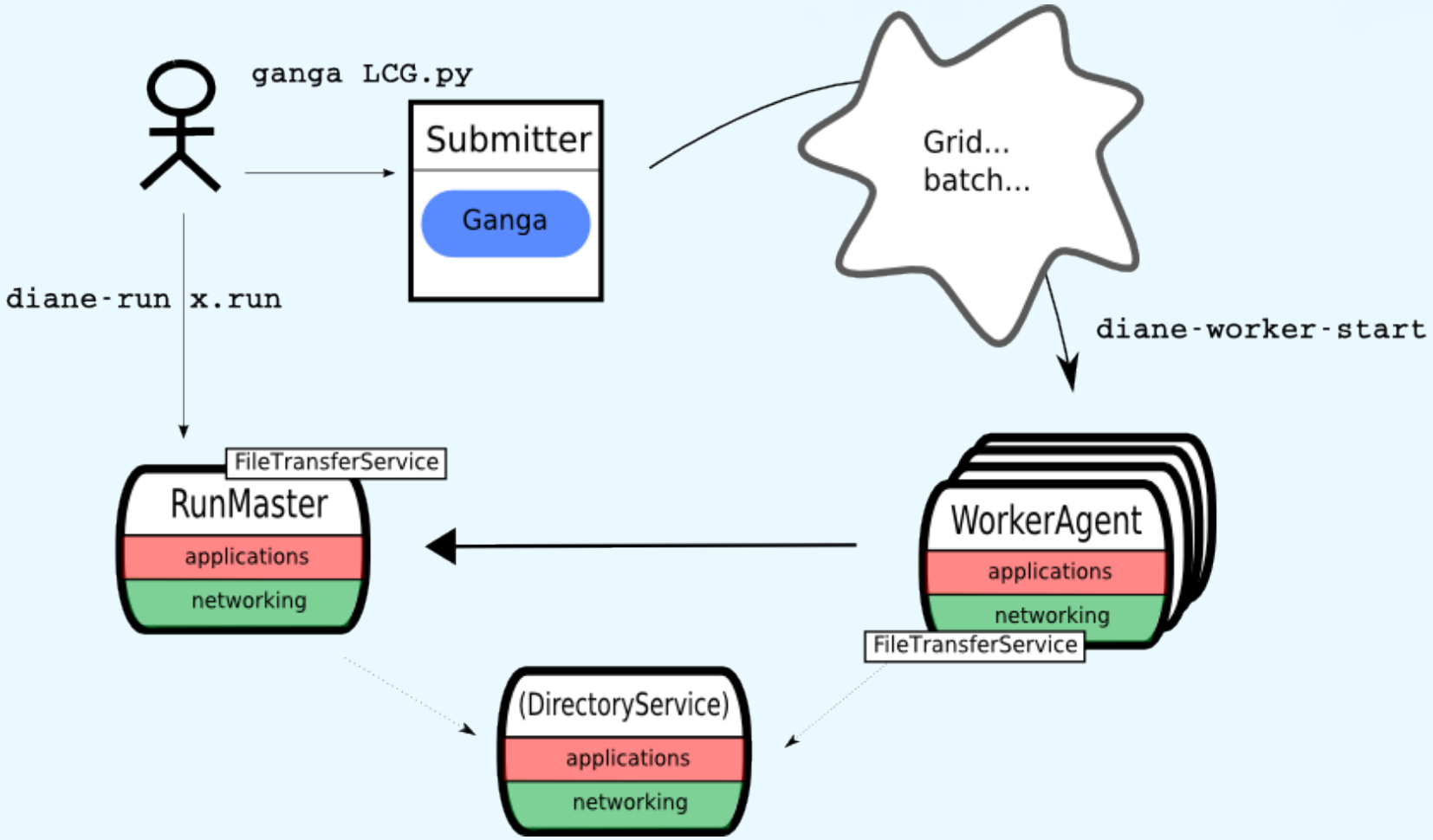
Python script – Running Athena

```
j = Job()
j.application=Athena()
j.application.exclude_from_user_area=["*.o","*.root*","*.exe"]
j.application.prepare(athena_compile=True)
j.application.atlas_release="14.2.10"
j.application.option_file=['my_jobOption'] <= jobOption filename (absolute path) //
j.application.max_events='100'
j.inputdata=DQ2Dataset()
j.inputdata.dataset="mc08.007081.singlepart_gamma_E10.recon.AOD.e339_s439_r462_tid023328"
j.inputdata.number_of_files=1
<= if you want to analyze a specified number of files
j.inputdata.type='DQ2_LOCAL'
file directly on SE // 'DQ2_DOWNLOAD' to force copy of file to WN )
j.outputdata=DQ2OutputDataset()
j.outputdata.outputdata=['recon.CBNT.pool.root']
j.outputdata.location="IN2P3-LAPP_LOCALGROUPDISK"
j.splitter=DQ2JobSplitter() <= if you want to analyze complete dataset
j.splitter.numsubjobs=6 <= (specifiy the number of subjobs)
j.merger=AthenaOutputMerger()
j.backend=LCG() j.backend.requirements.cloud='FR' //
j.backend.requirements.sites= ['LYON_MCDISK'] (dq2-ls -r command) j.submit()
```

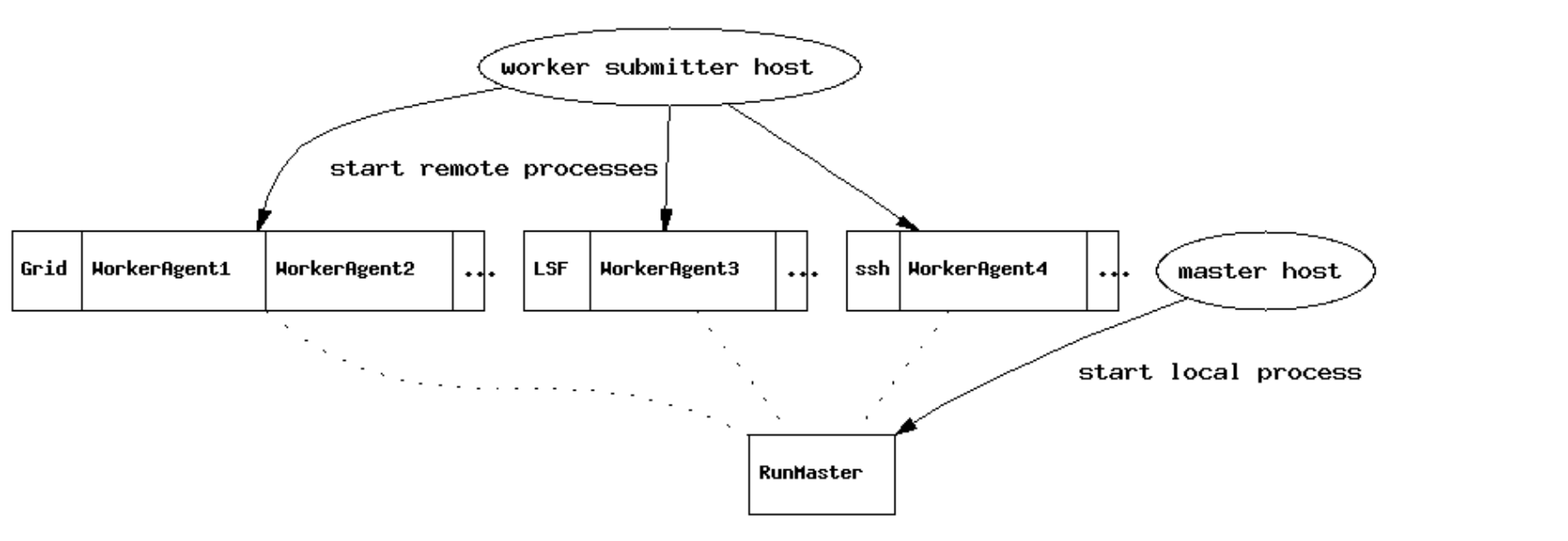
Ganga & DIANE

- **DIANE**
  - A lightweight agent-based scheduling layer on top of the Grid
  - Master & Worker Model
- **Ganga used in combination with DIANE**
  - DIANE works as an overlay scheduling system
  - Ganga works as a well-structured job management tool
- **DIANE worker agents are executed as Ganga jobs**

Ganga Submitter in DIANE Architecture



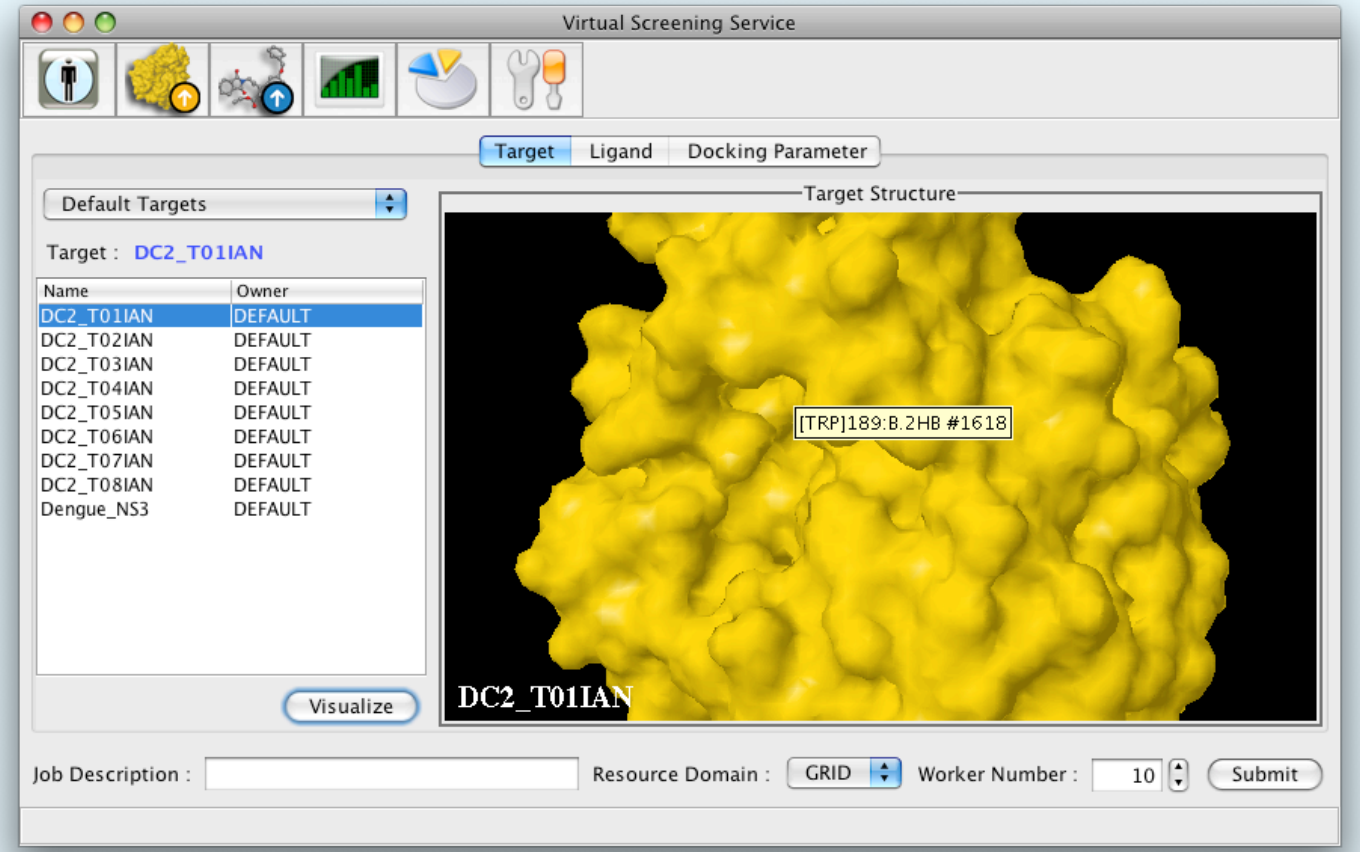
Ganga Job Submission tool in DIANE



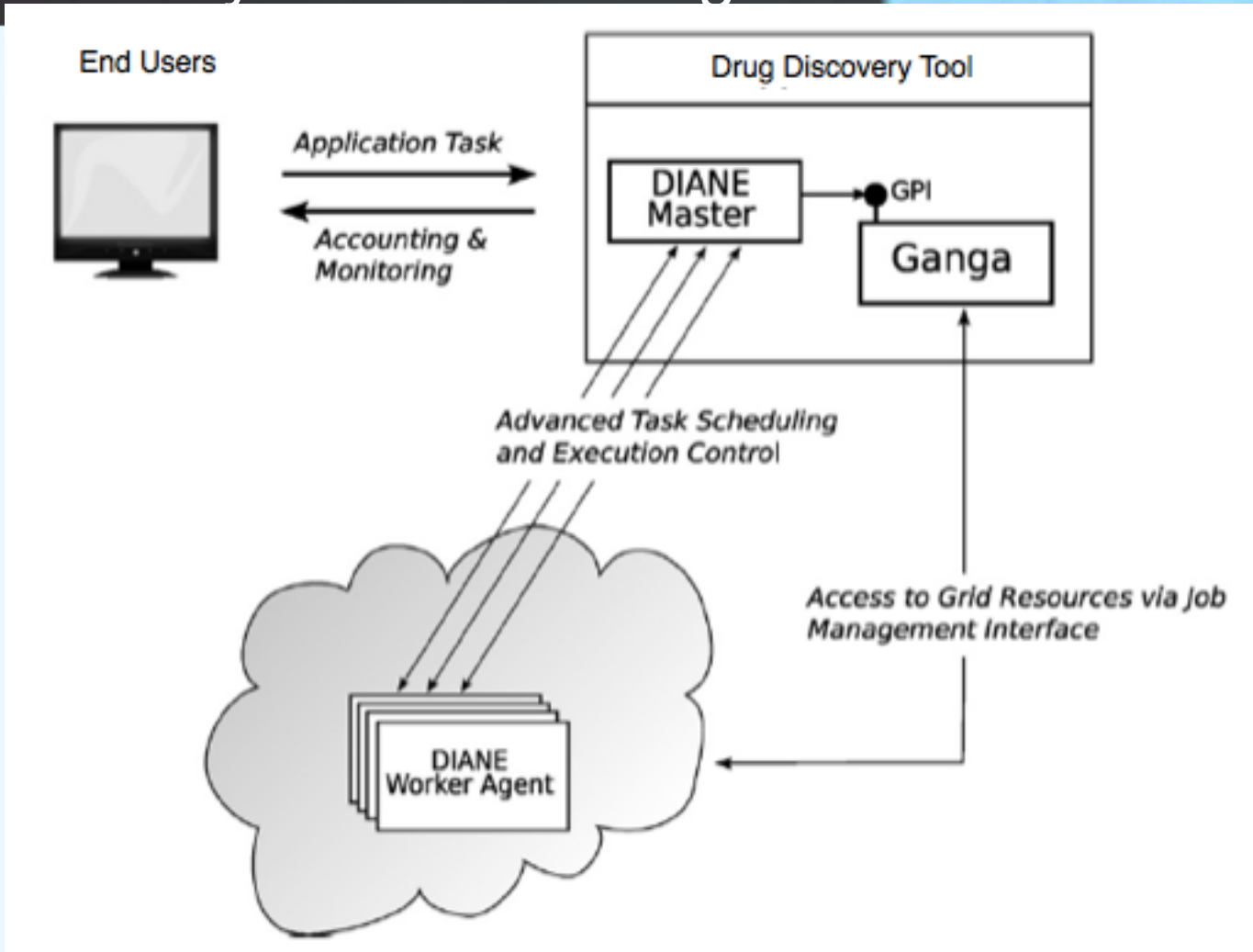
Use Case
Ganga Interfaced to DIANE Framework

- **Drug Discovery, Auto-docking Tool developed by ASGC, TW**
  - Ganga embedded in web-based services
  - This tool is for analysis of candidate drugs against avian flu
- **Ganga as a job management component embedded in DIANE, with the application**

Drug Discovery Tool with Ganga



Drug Discovery Tool with Ganga



Ganga as a job management component embedded in DIANE, with the application, Drug Discovery Tool

Referencs

- **Ganga Home**
  - http://ganga.web.cern.ch/ganga/index.php
- **Ganga User Guide**
  - http://ganga.web.cern.ch/ganga/user/index.php
- **Ganga GPI Reference**
  - http://ganga.web.cern.ch/ganga/release/4.3.2/reports/html/Manuals/GangaTutorialManual.html

~ The End ~