A High Energy Physics Data Analysis Tool

Hurng-Chun Lee (ASGC, CERN IT-GS)
ISGC 2008, Taipei, Taiwan
Outline

- Ganga at-a-glance
- Ganga and ATLAS distributed analysis
What’s Ganga

An easy-to-use enduser tool for preparing, submitting and managing jobs

jointly developed by ATLAS and LHCb

generic framework also for non-HEP users
Issue No.1

I must learn many interfaces

How to configure my applications?

Do I get a consistent view on all my jobs?

User: I want to concentrate on my studies
Issue No. 2

User: I want to come across different platforms seamlessly
Ganga: the goals

- Simplify use of computing resources
- Make it easy and integrated with application environment
- Allow quick transition between local PC, cluster, Grid...
- Organize work, keep history of jobs,...
- A generic framework
object oriented approach in defining jobs
- well-defined job building blocks
- programmable and extensible
Grid job management (client view)
Grid job management (client view)

without Ganga

```plaintext
Executable = "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`
Grid job management (client view)

without Ganga

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

write JDL

start grid proxy

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

submit job

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

check job status

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

get job outputs

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

with Ganga

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

```bash
write JDL
```

```bash
start grid proxy
```

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

```bash
submit job
```

```bash
glite-wms-job-outputs ... xxxx:yyyy:zzzz
```
Grid job management (client view)

**without Ganga**

```python
Executable = "echo"
StdOut = "stdout"
StdErr = "stderr"
Arguments = "hello world"

voms-proxy-init -voms atlas

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

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

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

**with Ganga**

```python
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
User has access to functionality of GANGA components through GUI, CLI or batch scripts.

Data Management:
- Job submission: selection between Grid and local system
- Input data selection, output location specification
- Storage and recovery of job information in a local or remote DB
- Storage of sandbox files

Application configuration:
- Executable
  - Gaudi
  - DaVinci
  - Athena
  - Other
- Job/Application plugins
- Application Manager
- Monitoring
  - Localhost
  - LSF
  - PBS
  - Condor
  - Other
- Backend plugins
- Ganga Core
- Job manager
- Archivist
- File Workspace, Input/Output Sandbox
- Job Repository
ATLAS data analysis

- 1 application: Athena
- 3 grids: WLCG, OSG, NorduGrid
- Distributed data management system: DQ2

- The computing model
- Data is distributed in Tier1/2s
- Analysis jobs are sent to data
Analysis workflow
What user does is ...

```
ganga athena
   --inDS csc11.005320.PythiaH170wwll.recon.AOD.v11004107
   --outputdata AnalysisSkeleton.aan.root
   --split 3
   --lcg
AnalysisSkeleton_topOptions.py
```

Single “Athena”-like command
Or ... for Object-Oriented Programming Gurus

```python
import Job
j = Job()
j.application = Athena()
j.application.prepare(athena_compile=False)
j.application.option_file=f'${HOME}/athena/12.0.5/InstallArea/jobOptions'
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()
```
Or ... for Object-Oriented Programming Gurus

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

- **User**
  - User code
  - Input Dataset
  - Output files
  - Some Grid specifics

- **Ganga Client**
  - Athena environment parsing
  - DQ2 Database query
  - User Area tar ball creation
  - JDL file, InputSandbox creation

- **Grid Worker Node**
  - Job(s) submission to glite WMS
  - InputSandbox upload to remote SE if too large

- **Monitor Jobs in Ganga Job Repository or ARDA Dashboard**

- **Output files download**
  - Output files merging
  - Jobs resubmission

- **After job completion:**
  - Output Sandbox is retrieved

- **Setup Athena Environment**
  - Inputfile List generation
  - LFC Query
  - Run Athena with direct POSIX I/O to dCache, Castor or DPM storage element
  - Stage-out outputfiles to storage element
Job monitoring and bookkeeping

MonALISA
MONitoring Agents using a Large Integrated Services Architecture

User

GANGA

Octopus

Accounting

Monitoring

ATLAS dashboard

Ganga Job

Monitoring loop
Real executable

Submit

Job logging & bookkeeping

gLite LB Service
Job monitoring and bookkeeping

- MonALISA: MONitoring Agents using a Large Integrated Services Architecture
- ATLAS dashboard
- GANGA
- Octopus
- Accounting
- Monitoring
- Job inspection
- Monitoring loop
- Real executable
- plug-able monitoring components
- supporting multiple monitoring plugins

User

GANGA

Submit

Ganga Job

Octopus

Job logging & bookkeeping

gLite LB Service
Getting your output

- by the time the Ganga job is finished
- output dataset is created on a SE closes to the CE with registration in ATLAS DDM
- one could switch to use an alternative SE
- fetching output dataset
  - using DDM enduser tool - dq2_get
  - j.outputdata.retrieve()
- merging outputs
  - j.merger.merge()
Switch to OSG and NorduGrid ...

```bash
# OSG

ganga athena
    --inDS csc11.005320.PythiaH170wwll.recon.AOD.v11004107
    --outputdata AnalysisSkeleton.aan.root
    --split 3
    --lcg
    AnalysisSkeleton_topOptions.py

# NorduGrid

ganga athena
    --inDS csc11.005320.PythiaH170wwll.recon.AOD.v11004107
    --outputdata AnalysisSkeleton.aan.root
    --split 3
    --ng
    AnalysisSkeleton_topOptions.py

# WLCG/EGEE

ganga athena
    --inDS csc11.005320.PythiaH170wwll.recon.AOD.v11004107
    --outputdata AnalysisSkeleton.aan.root
    --split 3
    --panda
    AnalysisSkeleton_topOptions.py
```
Switch to OSG and NorduGrid ...

```
ganga athena
   --inDS csc11.005320.PythiaH170wwll.recon.AOD.v11004107
   --outputdata AnalysisSkeleton.aan.root
   --split 3
   --lcg
AnalysisSkeleton_topOptions.py
```

```
ganga athena
   --inDS csc11.005320.PythiaH170wwll.recon.AOD.v11004107
   --outputdata AnalysisSkeleton.aan.root
   --split 3
   --panda
AnalysisSkeleton_topOptions.py
```

```
ganga athena
   --inDS csc11.005320.PythiaH170wwll.recon.AOD.v11004107
   --outputdata AnalysisSkeleton.aan.root
   --split 3
   --ng
AnalysisSkeleton_topOptions.py
```
You are not alone ...

- users: >1370 unique users world-wide
- very proactive development team
- user support:
  - hyper-news channels for ATLAS/LHCb support
- bug report
- To startup:
  - http://cern.ch/ganga
Ganga user communities

ATLAS
LHCB
HARP
Geant 4
COMPASS
Garfield
Academia Sinica
Genomics Research Center
New features in Ganga5

- more friendly interface
- new configuration framework for plugins
- job slicing framework
- more performant and less memory consumption
- many bug fixes in particular in ATLAS and LHCb application plugins

Coming soon ...
Conclusion

- There is a complex computing model behind the HEP data analysis
- End user should not be frustrated by the complexity
- Ganga has been properly designed and implemented for supporting HEP data analysis
- If you are not HEP user, you are also encouraged to try it out
- Enjoy your new Grid experience from [http://cern.ch/ganga](http://cern.ch/ganga)