Ganga
User-friendly Grid job submission and management tool for LHC and beyond

Dan van der Ster for the Ganga Team
CHEP 2009 – Prague, Czech Republic – 23-27 March 2009
Outline

• Motivation for a modular end-user tool for The Grid.

• Introduction to Ganga
  – Review of functionality, highlighting new features in Ganga 5
  – Stable release system and extensive testing framework

• Who is using Ganga?
  – Ganga for LHCb, ATLAS, and others...

• Novel usage in the community
What Users Want

- **Users want:**
  - Development on the laptop; full analysis on “The Grid™”.
  - To get results quickly, utilizing all of the resources available, wherever they are.
  - A familiar and consistent user interface to all of the resources.
- **Users don’t want:**
  - To know the details of the grids or the resources.
  - To learn yet another tool in order to access some resources.
  - To have to reconfigure their application to run on different resources.

“configure once, run anywhere”
Introduction to Ganga

• **Ganga is a user-friendly job management tool.**
  – Jobs can run locally or on a number of batch systems and grids.
  – Easily monitor the status of jobs running everywhere.
  – To change where the jobs run, change one option and resubmit.

• **Ganga is the main distributed analysis tool for LHCb and ATLAS.**
  – Experiment-specific plugins are included.

• **Ganga is an open source community-driven project:**
  – Core development is joint between LHCb and ATLAS
  – Modular architecture makes it extensible by anyone
  – Mature and stable, with an organized development process
Enabling Grids for E-sciencE

• Choose your own interface: CLI, GUI, or Scripting.

*** Welcome to Ganga ***
Version: Ganga-4-2-8
Documentation and support: http://cern.ch/ganga
Type help() or help('index') for online help.

• In [1]: jobs
• Out[1]: Statistics: 1 jobs
--------------
• # id status name subjobs application
  # backend.actualCE
  # 1 completed Executable
• compute.hpc.unimelb.edu.au:2119/jobmanage

To start Ganga:
  > ganga

#!/usr/bin/env ganga
#-*-python-*-# import time
j = Job()
j.backend = LCG()
j.submit()
while not j.status in ['completed', 'failed']:
  print('job still running')
  time.sleep(30)

To start the GUI:
  > ganga --gui

./myjob.exec

ganga ./myjob.exec

• GPI & Scripting

Dan van der Ster – Ganga: User-friendly Grid job submission and management tool for LHC and beyond
Submitting a Job with Ganga

Enabling Grids for E-sciencE

What is a Ganga Job?

**Run the default job locally:**

```
Job().submit()
```

**Default job on the EGEE grid:**

```
Job(backend=LCG()).submit()
```

**Listing of the existing jobs:**

```
jobs
```

**Get help (e.g. on a job):**

```
help(jobs)
```

**Display the nth job:**

```
jobs(n)
```

**Copy and resubmit the nth job:**

```
jobs(n).copy().submit()
```

**Copy and submit to another grid:**

```
j=jobs(n).copy()
j.backend=DIRAC()
j.submit()
```

**Kill and remove the nth job:**

```
job(n).kill()
job(n).remove()
```
One new recent feature is the **Remote backend**

**Remote** allows users to submit jobs from anywhere, even if you don’t have the batch/grid client tools installed:

- Local ganga packages the input sandbox, connects to a remote ganga instance via ssh
- The remote instance actually submits and monitors via the real backend (e.g. LCG/Dirac/Panda)
- Useful for example to work around quota issues etc…
• One of the strengths of Ganga is the stable release procedure and extensive testing
• Release Procedure:
  – Ganga developers rotate through 6 week terms as release manager
  – Release manager’s job is quite easy, most of the process is automated.
  – When enough tags have been collected, a pre-release is created
• Testing Framework:
  – Each pre-release is validated with nearly 500 test cases
  – Tests of new features and to prevent regressions
  – Ideally, each bugfix gets a test case.

<table>
<thead>
<tr>
<th>Package</th>
<th>xml</th>
<th>localxml</th>
<th>localxml</th>
<th>localxml</th>
<th>localxml</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ganga</td>
<td>255</td>
<td>36</td>
<td>254</td>
<td>37</td>
<td></td>
</tr>
<tr>
<td>GangaAtlas</td>
<td>1</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>GangaLHCb</td>
<td>141</td>
<td>18</td>
<td>146</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>GangaNG</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>GangaRobot</td>
<td>31</td>
<td>3</td>
<td>33</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>ALL</td>
<td>428</td>
<td>52</td>
<td>437</td>
<td>50</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Category</th>
<th>xml</th>
<th>localxml</th>
<th>localxml</th>
<th>localxml</th>
<th>localxml</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bugs</td>
<td>50</td>
<td>9</td>
<td>48</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>GPI</td>
<td>250</td>
<td>48</td>
<td>259</td>
<td>36</td>
<td></td>
</tr>
<tr>
<td>ALL</td>
<td>300</td>
<td>57</td>
<td>307</td>
<td>47</td>
<td></td>
</tr>
</tbody>
</table>

Failures usually due to timeouts
Who is using Ganga?

- Many projects using Ganga + DIANE
  - DIANE: tool to efficiently use resources
• Over 1000 unique users in the past 6 months:
  – Generally 50% ATLAS (blue), 25% LHCb (green), 25% other
• Monthly ~500 unique ~2000 unique since January 2007
• **Which interface is most popular?**
  - Normally scripting (blue) and command line interface (green) see 50%/50% usage
  - GUI is not used often… good for tutorials and learning, but CLI and scripts are more efficient.

Peak is from one user running many scripting instances
More than 137 unique domains using Ganga in past 6 months
Ganga is the Grid UI for LHCb
Main use is for running Gaudi jobs, including:

- Configuring Gaudi jobs
- Much easier to work with multiple configurations
- Specify the datasets
- Run the jobs locally, on batch systems and on the Grid via Dirac
- Managing the output data, Ntuples and histogram files.

Also support ROOT jobs

See Andrew Maier’s talk:

- “User analysis of LHCb data with Ganga” 26-Mar-2009 @ 15:00
• Ganga is one of two front-ends to the ATLAS grids (OSG, EGEE, NG)
  • Pathena and Ganga derive from a common “ATLAS Grid” library
• Plugins provided for all the analysis workflows:
  • Athena & AthenaROOTAccess processing AODs, DPDs, ESDs
  • AthenaMC app for private small MC production
  • DQ2 data management plugins
  • Flexible data access modes: local posix I/O, copy and process, FileStager

• See Johannes Elmsheuser’s talk:
  – “Distributed Analysis in ATLAS using GANGA” 24-Mar-2009 @ 17:50
• As ATLAS moves toward the PanDA pilot-based workload management system for analysis, GangaPanda is getting development attention.

• One novel feature is the “Personal Pilot” application.
  – Ganga sends a job to the Panda system
  – If the site selected for the job does not actively receive pilots (e.g. glexec not available) then Ganga can send pilots directly to the site using the LCG backend.

• See poster “A PanDA Backend for the Ganga Analysis Interface”
- Ganga is not only useful for end users:
  - The Python API allows development of Grid applications

- Two examples come from the ATLAS Distributed Analysis Testing tools.
  - GangaRobot is used to run many short functional tests daily to continuously validate the DA workflows.
    - Results are fed into Ganga so that broken sites can be avoided.

- HammerCloud is used to run large stress tests:
  - Measuring the behaviour of the storage, networks, databases, etc… under load.

- See talk “Functional and Large-Scale Testing of the ATLAS Distributed Analysis Facilities with Ganga” 26-Mar-2009 @ 15:20
There are novel contributions coming from outside the core Ganga team:

- KISTI (Korea Institute of Science and Technology Information) and the WISDOM project
  - Continued effort to use grids for Avian flu, Malaria drug searches
- Some of their resources could already be reached by the existing LCG/gLite backend.
- But, other resources are Globus sites, managed by GridWay.
  - No GridWay backend, so they developed one.
- Further, they wanted to hide the backend details from the users:
  - InterGrid backend: selects between LCG and GridWay backends using load information.

Next Steps

• What is next for Ganga Core development?

• Developers meeting in January 2009 highlights:
  – Further repository improvements:
    ▪ Testing an XML repository for improved speed and error resilience
  – Unified output management:
    ▪ Remove the distinction between “sandboxes” and “data”, allowing arbitrary treatment of all job I/O
  – Multi-stage or multi-application jobs:
    ▪ Prevent unnecessary storage of intermediate data by sending all stages of a job as one
  – Timekeeping:
    ▪ Incorporate timestamps of status transitions in the Job structure.
Conclusions

• Ganga is a user-friendly job management tool for Grid, Batch and Local systems
  – “configure once, run anywhere”

• A stable development model:
  – Well organized release procedure with extensive testing
  – Plugin architecture allows new functionality to come from non-core developers
  – Not just a UI – provides a Grid API on which many applications are built
  – Strong development support from LHCb and ATLAS, and 25% usage in other VOs

• For more information visit http://cern.ch/ganga
• BACKUP SLIDES
• A new major version (Ganga 5) was released in summer 2008

• Improved configuration interface
  – Instance settings on command line:
    ▪ `ganga -o[Logging]Ganga=DEBUG`
    or within `ganga`:
    ▪ `config.Logging # list the Logging section`
    ▪ `config.Logging.Ganga='DEBUG'`
  – User settings in `$HOME/.gangarc`
    ▪ `ganga -g generates an empty .gangarc with all options listed.`
      
        [Logging]
        Ganga = DEBUG
    – System settings in `$GANGA_CONFIG_PATH`
      ▪ E.g. system-wide `GangaAtlas-v5.ini` on the CERN AFS.

• Job slices and collective operations
  – `jobs[0:2] # list the 2 oldest jobs`
  – `jobs[-3:].submit() # submit the 3 most recent jobs`

• Job selection and collective operations
  – `jobs.select(status='completed') # list completed jobs`
  – `jobs.select(status='failed').resubmit() # retry all failed jobs`
  – `jobs.select(xrange(1,1000,2)).kill() # kill the odd jobs`
  – `jobs.select(backend='Panda').select(status='failed').copy().submit()`

• Also better jobs display, type checking of parameter and config options, repository speed and memory improvements