An Approach to Grid Interoperability using Ganga

Hangi Kim, KISTI
with Soonwook Hwang, YoonKee Lee, EunSung Kim, HeonYoung Yeom
ISGC2009, April 16th-23rd 2009, Academia Sinica, Taipei
Why are we interested in Grid Interoperability?

- **KISTI**
  - has been an official partner of EGEE project since EGEE-II, 2006
  - one of the founding members of PRAGMA

- **EGEE**
  - has Grid infrastructure based on gLite middleware services

- **PRAGMA**
  - has been operating its own Grid testbed based on Globus middleware services
INTRODUCTION

• Multiple Grid Infrastructures available
  – EGEE (gLite), PRAGMA (Globus), OSG, TeraGrid (Globus), DEISA(Unicore), etc.
  – Users have to learn commands or tools for each grid infrastructure
    • Not easy to use resources across multiple Grids

• Our Goal
  – High-level Tools that enable the use of as many as resources possible from multiple grid infrastructures without having to know the details of each grid middleware

• Our Approach
  – Chose to use the Ganga as a high-level tool for job execution management exploiting multiple Grid infrastructures
  – Chose to use the WISDOM as a target application
WISDOM (Wide In Silico Docking On Malaria)

- International initiative to deploy large-scale in-silico docking on a public grid infrastructure
- An attempt to find potential drugs against neglected or emerging diseases
  - e.g., Malaria, Avian Flu
WISDOM Production Environment

- Millions of independent docking jobs
  - Millions of chemical compounds
  - One or more targets
Ganga

- **easy-to-use user interface for job submission and management**
  - Implemented in Python: Interactive shell or Script files
  - Use **python syntax** to submit a job
- provide a plug-in mechanism for job submission on multiple computing backends (e.g., Local PC, Cluster and Grid) and applications (e.g., ROOT, GAUDI and Athena)
  - easily extended and customized to meet the needs of different user communities
- Currently, support only the LCG/gLite backend for Grid, not Globus backend
New Plug-ins Development in Ganga

- Have integrated new plug-ins into Ganga:
  - AutoDock applications
  - GridWay backend
  - InterGrid backend
New Plug-ins Development in Ganga (cont’d)

- **AutoDock Application**
  - Customize configuration for running autodock applications
  - *A LigandSplitter module* has been developed to spread ligand-protein docking jobs on the grid

- **GridWay Backend**
  - Provides access to Globus resources

- **InterGrid Backend**
  - Provides access to both LCG/gLite and Globus resources via the LCG and Gridway backend, respectively
  - Intelligent resource selection
    - Based on load on Grid
Interoperability Issues

• Authentication
  – Most grids use X509 credentials
  – Use VOMS proxy
  – `voms-proxy-init` command (gLite) has compatibility with `grid-proxy-init` command (globus)
Interoperability Issues (cont’d)

• The original shell script developed in the WISDOM project **fails to run** on Globus resources
  – The autodock program takes a file describing a 3D structure of target protein and a ligand file as its input files
  – The WISDOM script relies on the globus-url-copy command for copying the two files into the grid node
    • LCG: globus-url-copy – OK
    • Globus: globus-url-copy – Failed (No credential delegation supported)
Interoperability Issues (cont’d)

• Our approach
  – We had to modify the original WISDOM shell script not to rely on the globus-url-copy for file copying
  – The new shell script relies on the file staging feature of the Grid.
  • Put all the required files needed by the autodock code into
    – inputSandbox in case of LCG backend
    – inputFile in case of Gridway backend

It Worked !!!
EXAMPLE

- One AutoDock Job

```python
In [1]: !cat autodock_lcg.py
#!/usr/bin/python
j=Job()
j.application=Autodock()
j.application.exe="/bin/sh"
j.application.script="/home/horn/ganga_autodock/autodock.sh"
j.application.binary="/home/horn/ganga_autodock/autodock.tar.gz"
j.application.protein="/home/horn/ganga_autodock/1u2y.tar.gz"
j.application.ligand="/home/horn/ganga_autodock/ligands/9004736_1.pdbq"
j.application.parameter="/home/horn/ganga_autodock/dpf3gen.awk"
j.backend=LCG()
j.submit()
```
EXAMPLE
• Multiple AutoDock Jobs (*for* statement)

```python
In [1]:!cat autodock_gridway_10.py
#/usr/bin/python
for x in range(10):
    j=Job()
    j.application=Autodock()
    j.application.exe="/bin/sh"
    j.application.script="/home/horn/ganga_autodock/autodock.sh"
    j.application.binary="/home/horn/ganga_autodock/autodock.tar.gz"
    j.application.protein="/home/horn/ganga_autodock/1u2y.tar.gz"
    j.application.ligand="/home/horn/ganga_autodock/ligands/9004736_"+str(x)+".pdbq"
    j.application.parameter="/home/horn/ganga_autodock/dpf3gen.awk"
    j.backend=Gridway()
    j.submit()
```
• Multiple AutoDock Jobs (*for* statement)
• Multiple AutoDock Jobs (for statement)
• Intergrid Job
Thank you for your attention!

謝謝