Charon Extension Layer

Modular environment for Grid jobs and applications management

Jan Kmuniček

CESNET NA4 effort
Contents

• Introduction
• Charon Infrastructure
• Applications Management
• Jobs Management
• Conclusions
• **What is Charon Extension Layer?**
  
  – uniform and modular approach for (complex) computational jobs submission and management
  
  – generic system for use of application programs in the Grid environment (LCG/gLite middleware, …)
• Why Charon Extension Layer?
  - many various batch systems & scheduling components used in grid environment
  - each batch system has unique tools and different philosophy of its utilization
  - LCG/gLite provided tools are quite raw and simple
  - many additional tasks to use computer resources properly
• **Application management**  
  – single/parallel execution without job script modification

• **Job management**  
  – easy job submission, monitoring, and result retrieving

• **Command Line Interface (CLI) approach**
• Requirements

  – easy application initialization
  – version conflict handling
  – inter-application conflicts and/or dependencies handling
  – same usage in single/parallel execution
  – different levels of parallelizations

Module System
– similar approach as in Environment Modules Project*
  ▪ applications are activated by the modifications of shell environment (e.g. PATH, LD_LIBRARY_PATH, etc.)

– particular build of application is described by realization (e.g. by instructions, which describe shell environment modifications)

– realization is identified by name consisting from four parts:

  \textbf{name[::version[::architecture[::parallelmode]]]}

– user can specify only part of realization, in that case, module system completes full name of realization in such a way that the application will best fit available computational resources

*) http://modules.sourceforge.net/
• Commands of Module System

module [action] [module1 [module2] …]

- main command of Module System

  actions:
  - add (load), remove (unload)
  - avail, list*, active, exported, versions, realizations
  - disp, isactive

* list is default action

modconfig

- menu driven configuration of Module System
  (vizualizations, autorestored modules, etc.)
• Example of Module Activation

```
$ module add povray

Module specification: povray (add action)
=================================================================
WARNING: Nonoptimal architecture is used for module 'povray'
Cache type          : system cache
Architecture        : i786
Number of CPUs      : 1
Max CPUs per node   : 1
Exported module     : povray:3.6
Complete module     : povray:3.6:i386:single
```

• Module Name Completion

```
povray → povray:3.6:auto:auto → povray:3.6:i386:single
user     → default values          → resolved final name
```
• **Module Name Completion**

  name - specified by user (it is mandatory)

  version - specified by user / default

  architecture - specified by user / default / automatically determined
  - Module System tries to find such realization, which is the closest to system architecture

  parallelmode - specified by user / default / automatically determined
  - para - always
  - p4 - NCPU > MaxCPUs/node
  - shmem - 1 < NCPU <= MaxCPUs/node
  - node - NCPU <= MaxCPUs/node
  - single - NCPU=1
• Current enhancements of Module System
  – easy-to-use **configuration of user preferences** using configuration by a single command which is menu driven
    ▪ user can setup the behavior of module name completion, visualization of listed modules
    ▪ user is allowed to select modules to be automatically loaded during site activation
    ▪ user can change priority between system and user module caches
  – user modules
    ▪ user is permitted to extend available application portfolio by own module specifications
• **Model I - METACentrum (Czech national GRID)**

- Applications are on shared volume available to all grid elements

- Applications cannot be shared with all grid elements
- Their “sharing” is provided by their deployment to SE (once time)
- Only required applications are then installed on CE during every job execution

**Legend:**
- UI - user interface
- CE - computing element
- SE - storage element
- WN - worker node
- app - application

---

**Model II – EGEE GRID**

---
• Interactive browser of the module database

- extension of Module system containing real-time list of available software realizations
- this service can show the list of available applications with or without the accessible application versions
- this information is integrated with the detailed description of applications (documentation of particular compilation and installation in the MediaWiki)

http://troll.chemi.muni.cz/whitezone/development/charon/isoftrepo/
<table>
<thead>
<tr>
<th>Categories</th>
<th>Categories (versions)</th>
<th>List of realizations</th>
<th>Tree of realizations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Molecular Mechanics and Dynamics</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>auto dock</td>
<td>gromacs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Quantum Mechanics and Dynamics</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>abinit</td>
<td>dallion</td>
<td>pegamess</td>
<td>uspp</td>
</tr>
<tr>
<td>Conversion and Analysis</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>babel</td>
<td>cats</td>
<td>opendevidebe</td>
<td>bbplus</td>
</tr>
<tr>
<td>ball</td>
<td>octave</td>
<td>openslab</td>
<td>pub2pgsql</td>
</tr>
<tr>
<td>gquill</td>
<td>retinal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Visualization</td>
<td>grace</td>
<td>iggplot</td>
<td>molscript</td>
</tr>
<tr>
<td>gnpplot</td>
<td>master3d</td>
<td></td>
<td></td>
</tr>
<tr>
<td>povray</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nuclear Magnetic Resonance</td>
<td>dasha</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Physics, Astrophysics, Technical and Material Simulations</td>
<td>gnpb</td>
<td>mb</td>
<td>octave</td>
</tr>
<tr>
<td>Compilers and Supporting Environment</td>
<td>gnpb</td>
<td></td>
<td></td>
</tr>
<tr>
<td>System</td>
<td>gnpb</td>
<td>general</td>
<td>glibc</td>
</tr>
<tr>
<td>charon</td>
<td>certificates</td>
<td></td>
<td></td>
</tr>
<tr>
<td>hvtolem</td>
<td>ui-voce</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(c) 2006 Martin Petrik, Petr Kulhanek, National Centre for Biomolecular Research, Faculty of Science, Masaryk University</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Attendance is monitored by</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Jobs management

• **Requirements**
  
  – easy job submission
  
  – user should focus only on desired task not to all things related to submissions
  
  – easy parallel executions of applications
  
  – often repeated things should be processed automatically
  
  – keep information about job during execution and/or after execution

Charon System
• **Overview**
  – it is application in the context of Module System
  – it separates resources settings from job submission

• **Job Submission and Management**
  – `psubmit <resources> <job_script> [NCPU] [syncmode]`
  – `pinfo`
  – `psync`
  – `pkill`
  – `pgo` (does not work in EGEE GRID environment)

• **Charon Setup**
  – `pconfigure`
• Typical job flow

```
[myjob]$ psubmit gilda test1
```

submit job

```
[myjob]$ pinfo
```

monitor job

```
[myjob]$ psync
```

get results

No additional arguments are required – all information about job is stored in control files in job directory.
• **Job Restrictions**

  – job is described by script (specific input files can be autodetected)

  – **each job has to be in separate directory** – control files need to be unique

  – job directories must not overlap – because job directory is copied to WN and then back

  – only relative paths to job directory contents have to be used in job script – only data from job directory will be present on WN

  – software should be activated by Module System – only then best utilization of resources can be reached
**Configuration**

- **Sync Mode** – option for data transfer between UI and WN
  - gridcopy
    - *all data within job directory as input and as as result*
  - stdout
    - *all data within job directory as input*
    - *only standard output as result (other data are discarded)*

- **Resources** – identification of particular CE

- **Properties** – fine grained selection of computational resources (through Requirements item in JDL)

- **Alias** – uniform combination of above setup items in a singleword

`pconfigure` menu driven command for configuration
Future considerations

• Currently implemented features
  – multi-grid (sites) approach
    ▪ virtual encapsulation of computational resources
  – support for Job Provenance service

• Planned features
  – advanced job statistics
  – workflow support (DAG jobs)
  – access through graphical user interface
Conclusions

• **Single job management**
  
  – **encapsulation of a single computational job**
  
  – minimization of overhead resulting from direct middleware usage (JDL file preparation, etc.)
  
  – easy submission and navigation during job lifetime

• **Application management**
  
  – **powerful software management** and administration
  
  – comfortable enlargement of available application portfolio
• Uncover all details!
  – visit available web pages at