Charon Extension Layer

Modular environment for Grid jobs and applications management

Jan Kmuníček

CESNET NA4 effort

Contents

- Introduction
- Charon Infrastructure
- Applications Management
- Jobs Management
- Conclusions

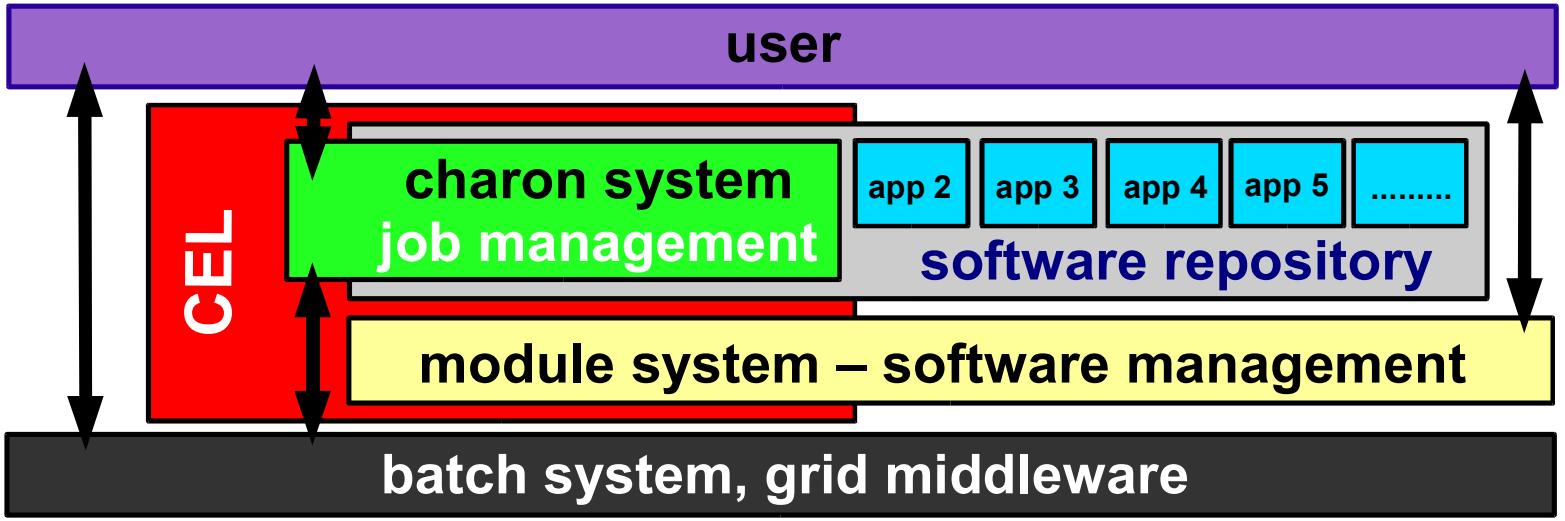
Introduction

- **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, …)

Introduction

- **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

CHARON Infrastructure



- **Application management**
  - single/parallel execution without job script modification
- **Job management**
  - easy job submission, monitoring, and result retrieving
- **Command Line Interface (CLI) approach**

Applications management

- **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

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:

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/

Module System *continued*

- **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.)

Module System *continued*

- **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 System *continued*

- **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

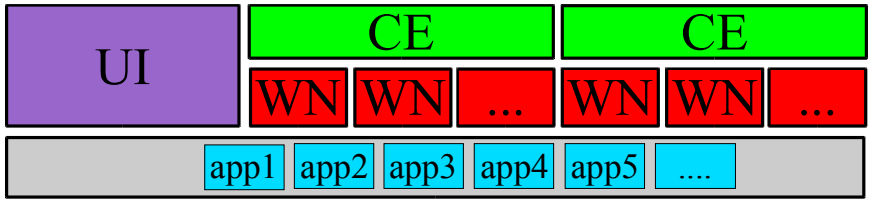
  - para - always
  - p4 - NCPU > MaxCPUs/node
  - shmem - 1 < NCPU <= MaxCPUs/node
  - node - NCPU <= MaxCPUs/node
  - single - NCPU=1

Module System *continued*

- **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

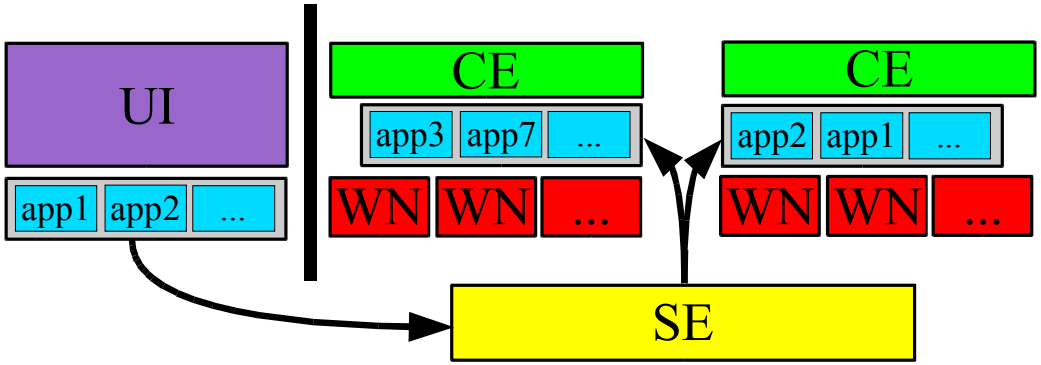
Applications in GRID

- **Model I - METACentrum (Czech national GRID)**



Applications are on shared volume available to all grid elements

- **Model II – EGEE GRID**



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

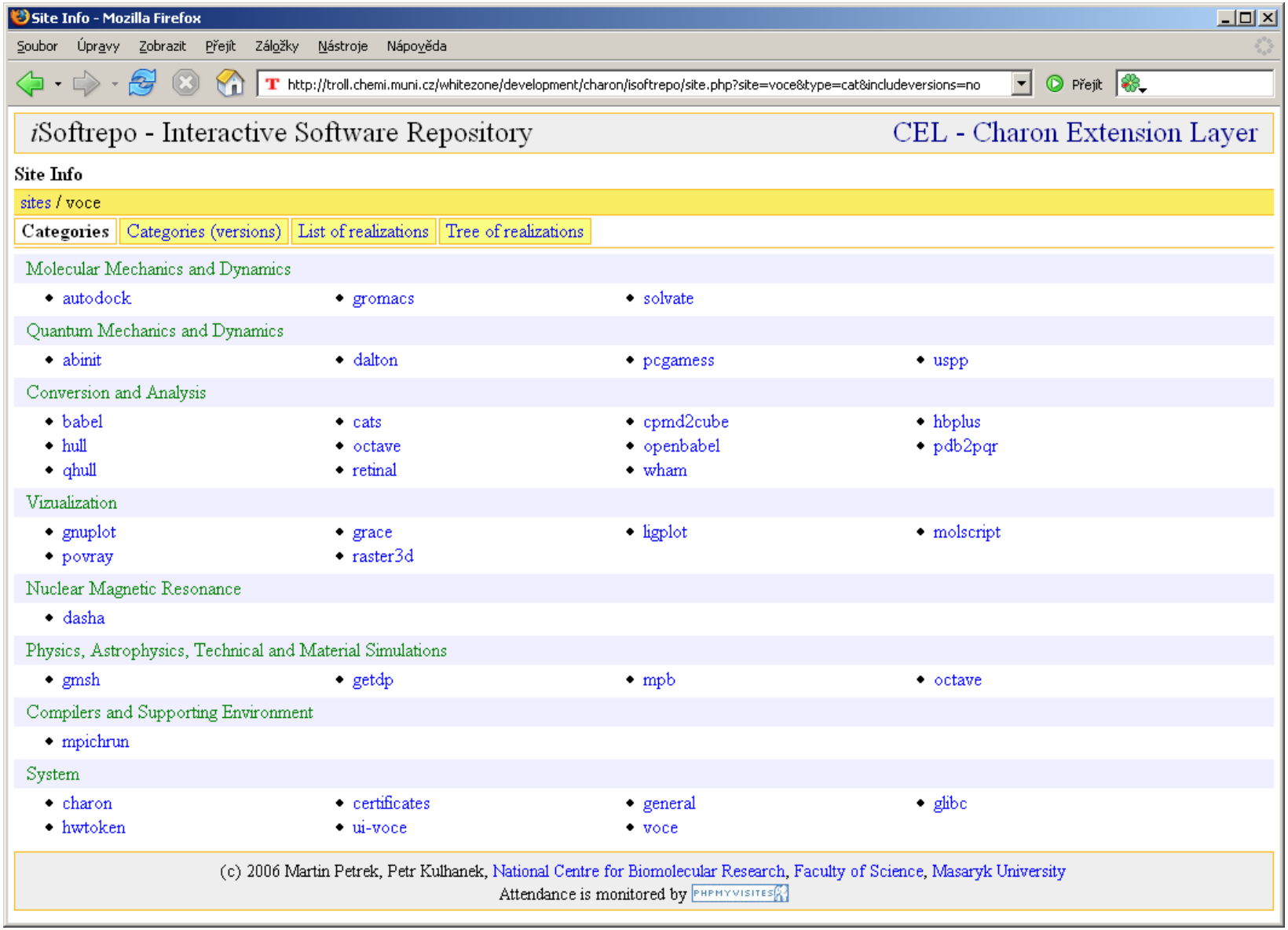
- applications cannot be shared with all grid elements
- their “sharing” is provided by their deployment to SE (once time)
- only required applications are then installed on CE during ever job execution

Applications in 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/

Applications in GRID



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 process automatically
  - keep information about job during execution and/or after execution

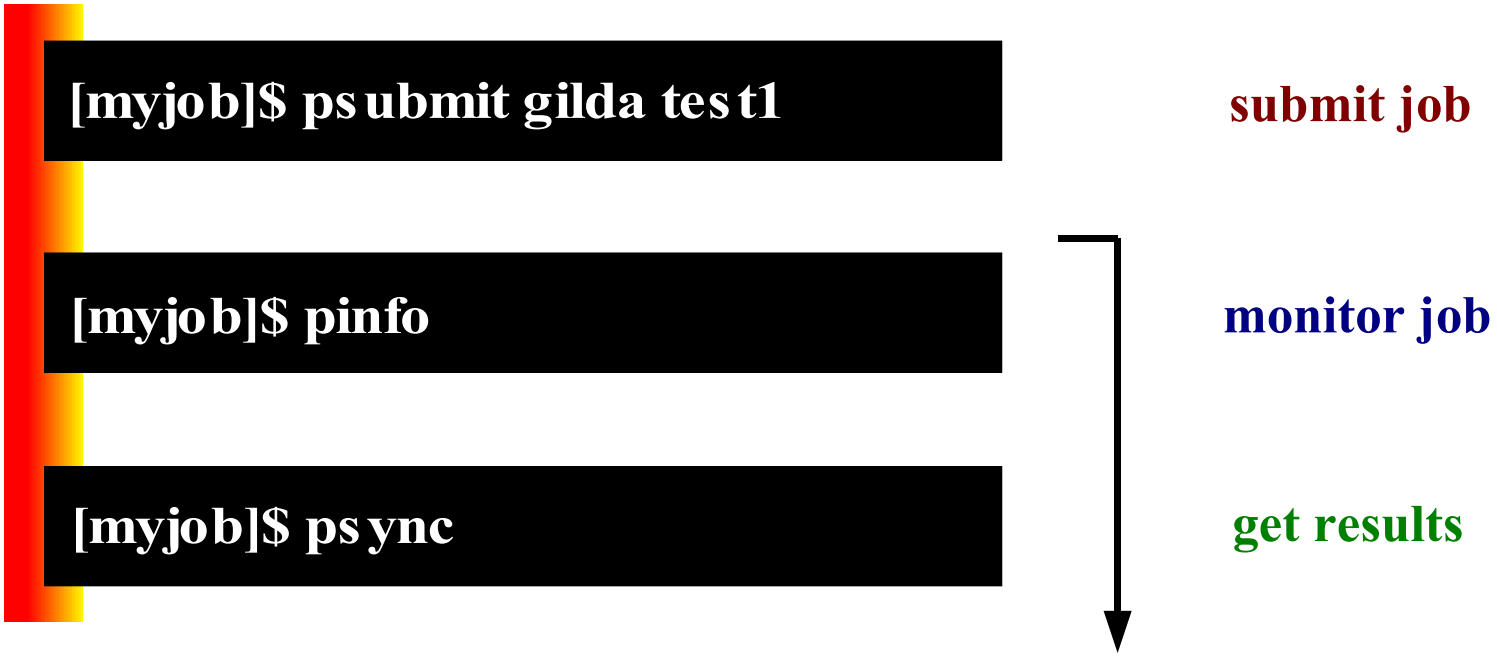
Charon System

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**

Charon System *continued*

- **Typical job flow**



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

Charon System *continued*

- **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

Charon System *continued*

- **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 (throught 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

Information Pointer

- **Uncover all details!**
  - visit available web pages at

    http://egee.cesnet.cz/en/voce/Charon.html