

Charon Extension Layer



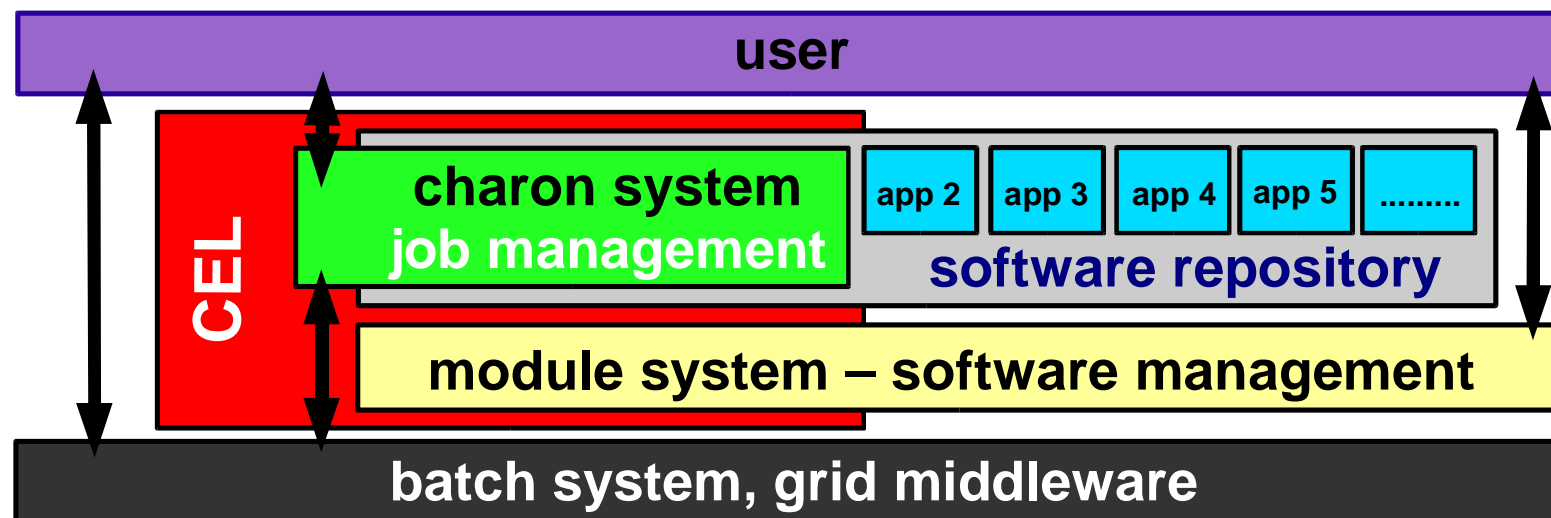
Petr Kulhánek,^{1,2} Martin Petřek,^{1,2} Jan Kmuníček^{1,3}

kulhanek@chemi.muni.cz, petrek@chemi.muni.cz, kmunicek@ics.muni.cz

- 1) CESNET z. s. p. o., Zikova 4, CZ-16000 Praha, Czech Republic
- 2) National Centre for Biomolecular Research, Faculty of Science, Masaryk University, Kotlářská 2, CZ-61137 Brno, Czech Republic
- 3) Institute of Computer Science, Masaryk University, Botanická 68a, CZ-60200 Brno, Czech Republic

- **Introduction**
- **Charon Infrastructure**
- **Application Management**
 - Module System
 - Applications in GRID Environment
- **Job Management**
 - Charon System
 - Sites
- **Conclusions**
- **Acknowledgments**

- **What is Charon?**
 - 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?**
 - 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:

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

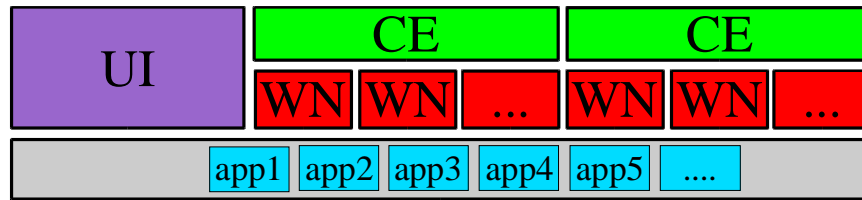
§ p4 - $N_{CPU} > MaxCPU_{s/node}$

§ shm - $1 < N_{CPU} \leq MaxCPU_{s/node}$

§ node - $N_{CPU} \leq MaxCPU_{s/node}$

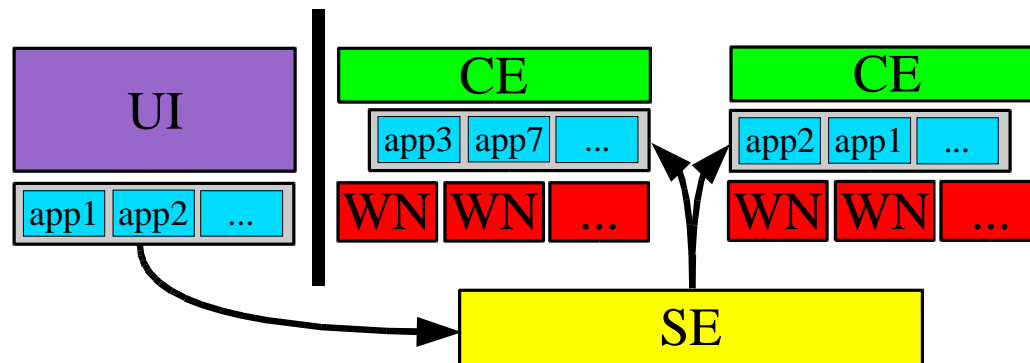
§ single - $N_{CPU}=1$

- **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 every job execution

- both modes can be carried out without any modifications of module system
- desired “non-standard” functions can be carried out by user (administrator) defined hooks (so called modactions)
- modaction is script that is executed during any action of module command
- modaction script for add action solves problems with applications in Model II
 - § it behaves differently on UI and WN
 - § it activates applications from volume on UI
 - § it downloads package from SE to WN (CE) and installs it to some temporary volume then Module System sets environment in such a way that application will be used from that volume

Advantages

all applications are available
in whole grid immediately
after their deployment to SE

Drawbacks

this approach is suitable only
for middle and long term jobs

- **Requirements**

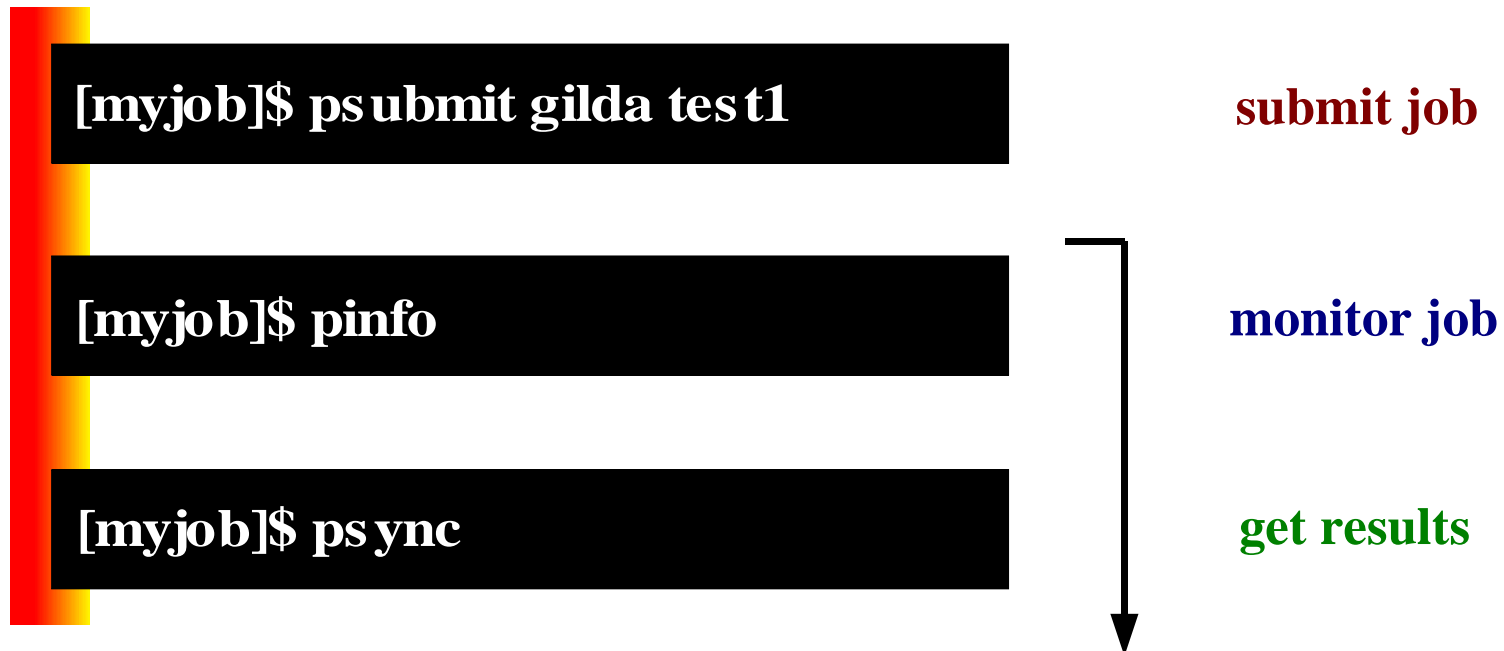
- easy job submission
- user should focus only on desired task not to all things related to submissions
- easy parallel executions of applications
- often repeated things should be process automatically
- keep information about job during execution and/or after execution



Charon System

- **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**



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

- **Job Restrictions**

- job is described by script*
- 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

- **Job autodetection***

- in some cases, user can specified input file instead of script and Charon System will prepare script for its processing
- currently autodetected jobs are: **gaussian**, **povray**, and **precycle**

- **Configuration**

- **Sync Mode** – option for data transfer between UI and WN

- § gridcopy

- *all data within job directory as input*
 - *all data within job directory as result*

- § stdout

- *all data within job directory as input*
 - *only standard output as result (other data are discarded)*

- **Resources** – identification of particular CE

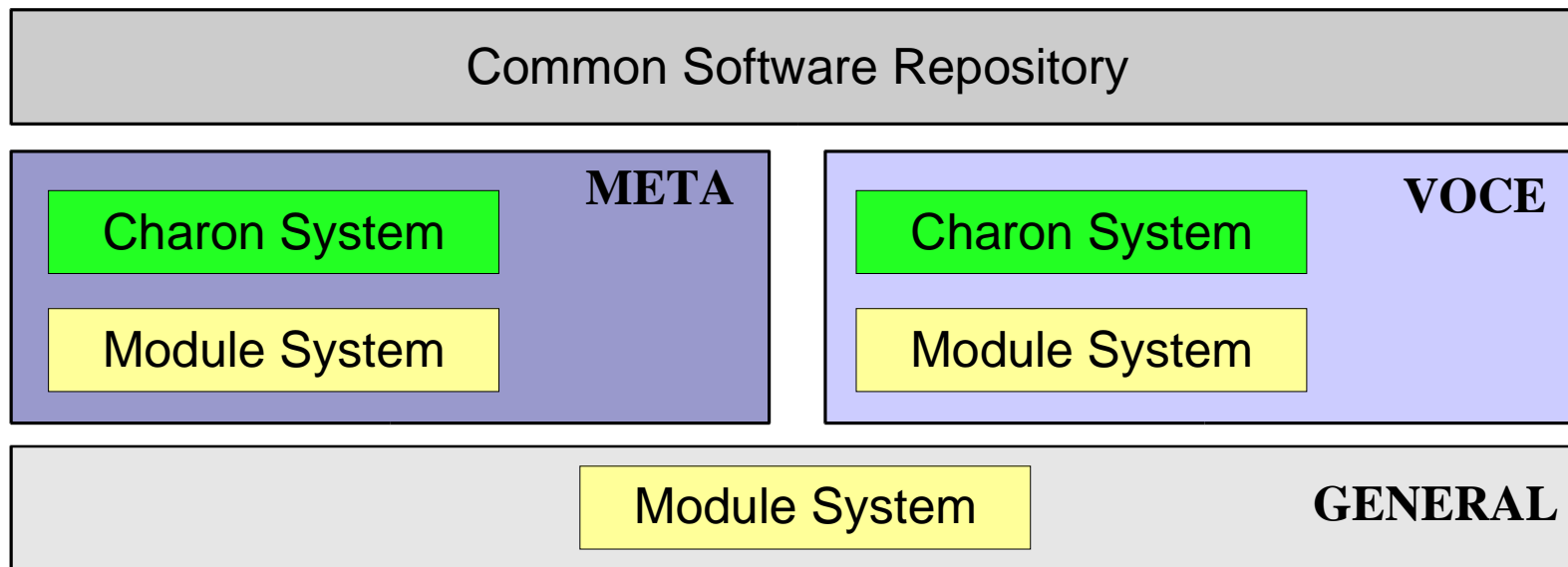
- **Properties** – fine grained selection of computational resources (through Requirements item in JDL)

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

pconfigure command serves for configuration.

It is menu driven.

- sites represent approach in utilizations of different grids (sites) from one computer
- sites are special modules within Module System
- all sites shared the same software repository but list of available applications depend on site Module System setup



- **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

- Luděk Matyska (CESNET, ICS)
- Jaroslav Koča (NCBR)
- European Commission
 - § EGEE II (contract number RI-031688)
 - § EGEE (contract number IST-2003-508833)
- Ministry of Education, Youth, and Physical Training of the Czech Republic (contract number MSM0021622413)
- Grant Agency of Czech Republic (204/03/H016)