Charon Extension Layer – Universal Toolkit for Grid Applications and Computational Jobs Maintenance

Jan Kmuníček [1], Martin Petřek [1,2], and Petr Kulhánek [2]

[1] CESNET z. s. p. o., Zikova 4, CZ-16000 Praha 6, Czech Republic; [2] National Centre for Biomolecular Research, Kotlářská 2, CZ-61137 Brno, Czech Republic
email: kmunicek@ics.muni.cz, {petrek,kulhanek}@chemi.muni.cz

Abstract

This article describes Charon Extension Layer (CEL) system – a universal framework creating a layer upon the basic Grid middleware environment and making an access to the complex Grid infrastructures much easier compared to the native middleware systems. The CEL system unifies the variability in Grid middleware (PBS, OpenPBS, LCG, gLite, *etc.*) allowing transparent access to distinct Grids. Moreover, the CEL system offers easy access and utilization of heterogeneous Grids in a unique, easy and smoothly integrated way.

Here we present a new implementation of the high-level middleware core services provided by CEL. Specific parts of the system were rewritten to enable new set of features to be available for end users utilization. First of all, present version of CEL is capable to provide multi-Grid approach (through so-called "sites") that makes everyday CEL utilization much more comfortable. Moreover, several further enhancements as incorporation of different data transfer synchronization modes, deeper personalization of the user's Charon environment, guided handling of grid middleware error states, module name completion, and others are supported now. Currently, CEL system is successfully used for end users research work within the application-generic Virtual Organization for Central Europe (VOCE) Grid infrastructure, part of the EGEE II Grid.

1 Introduction

Job submission and its subsequent management that are crucial tasks for successful utilization of cluster and/or grid environments are controlled by various batch systems (PBSPro [1], OpenPBS [2], Condor [3], LSF [4], scheduling components of grid middlewares such as Globus [5], LCG [6], gLite [7], and others). Each batch system has unique tools and different philosophy of its utilization. Moreover, the provided tools are quite raw and users have to perform many additional tasks to use computer resources properly. Since end users of clusters and/or grids are more interested in some particular scientific problems, the difficult usage of batch system might have negative impact on the efficiency of resource utilizations and therefore on studied scientific problems. The introduced Charon Extension Layer system represents one possible solution of problems associated with the utilization of low level batch system commands.

2 CEL Architecture

Charon Extension Layer (CEL) toolkit [8] provides a command-line oriented interface and is supposed for users that require a full control over their running computational jobs. CEL system provides uniform and modular approach for complex computational jobs submission and management and forms a generic system for the use of application programs in the Grid environment independently of Grid middleware present at specific fabric infrastructure. CEL can be easily used for powerful application management enabling single/parallel execution of computational jobs without the job script modification. Simultaneously, standard job management involving easy job submission, monitoring, and result retrieval can be performed without any additional hassle or requirements on users.

The complete Charon Extension Layer is combined from two distinct subsystems Module System and Charon System (see Fig. 1). Module System is used for the management of available application portfolio. It solves problems connected with execution of applications on machines with different hardware and/or operation systems and it is also capable to simplify the execution of applications in parallel environments. Charon System is a specific application managed by Module System that introduces complete solution for jobs submissions and their subsequent management.

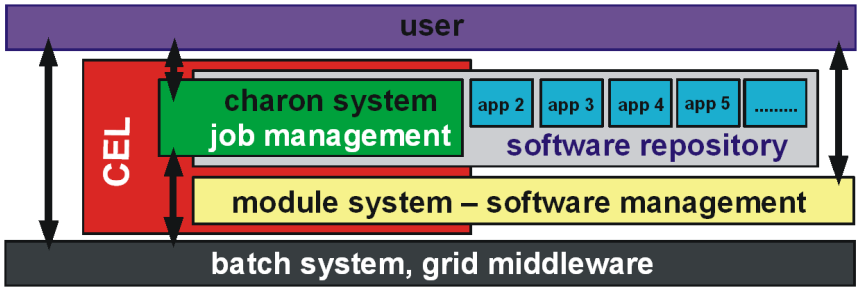

Fig. 1: A basic scheme of CEL architecture composed from two main parts – Module System and Charon System forming together a layer between a grid end user and a grid middleware.

These two parts of CEL form together a unique and consistent solution not only for job submission and administration but also for easy job preparation. Moreover, CEL does not limit original batch systems in any way. It just extends their functionality and simplifies their usage as much as possible for everyday submissions and monitoring of tens up to hundreds of complex computational jobs.

Currently Charon Extension Layer provides following set of unique features:

- easy access and utilization of heterogeneous grids in a unique, easy and smoothly integrated way
- transparent access to distinct grids
- powerful software management and administration
- comfortable enlargement/modification of available application portfolio
- easy job submission, monitoring, and results retrieval
- interactive software repository

3 Module System

The very generic requirements on a system that could be used for easy yet powerful administration of application programs within a grid environment should involve following set of features: easy application initialization, application versions conflict handling, inter-application conflicts and/or dependencies handling, very same utilization for single/parallel jobs execution, support for different levels of parallelizations. All these basic demands are currently supported by Module System that is one of the essential CEL parts serving for management of applications available in a specific grid environment. The Module System is based on similar approach as in Environment Modules Project [9] in which the application programs are activated by the modifications of shell environment. Each software package is described by a specific module in the Module System. The module contains all information which is neccessary to work with a particular software (e.g. PATH, LD_LIBRARY_PATH, additional environment setup, *etc.*). This configuration information is internally stored in XML format that makes setup of Module System very straightforward. The particular build of an application is described by so-called realization, e.g. by instructions, which describe shell environment modifications. The realization is identified by name consisting from four sections (see Fig. 2).

name[:version[:architecture[:parallelmode]]]

Fig. 2: A scheme of particular application build realization composing of four distinct parts.

User can specify only part of a realization, in that case, Module System will complete the full name of the realization in such a way that the application will best fit available computational resources. Basically, two main commands – **module** and **modconfig** – are available for managing applications under the Module System.

Main command of Module System is **module**

```
module [action] [module1 [module2] ]

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

The **modconfig** command serves for menu driven configuration of Module System (selection of vizualizations, settings of autorestored modules, *etc.*).

Module System was completely rewritten during last year taking into consideration all requirements which were arisen during its utilization in past. It is now completely written in C/C++ languages and its functionality was significantly extended. Selected new features of latest version (3.0.12) are as follows:

- multi-site approach
- easy-to-use configuration of user preferences
- user modules
- module name completion
- web browsing within module system database

3.1 Multi-Site Approach

The utilization of different computational/storage resources is very difficult task. The resources very often differ in used architectures, grid middleware, licensing politics, *etc.* As this is very unsuitable for common user, in the past we solved this problem within CEL only partially. However, the solution was limited because the setup of CEL for one particular resource was connected with one user interface. Currently, this drawback was eliminated with multi-site approach. The so-called site in CEL environment is a special module within Module System which switches the appropriate environment from one resource to other one.

The CEL site represents a virtual encapsulation of computational resources therefore allowing utilizations of different Grids (sites) from one computer. All accessible sites share the same software repository but the actual list of available applications depends on site Module System setup. This approach extends CEL in such a way that almost unlimited number of different computational resources can be handled with it on one user interface.

3.2 Supported Applications Transfer Modes

Concerning the distribution of packages of application binaries CEL currently support two different modes. The first one is a simple case when all applications are located on a shared disc volume available to all grid elements (as used in project $\mathcal{META}$ *Centrum* [10] – Czech national grid project). The situation is rather different within European-wide grid project EGEE [11] – Enabling Grids in E-Science. Within EGEE the applications can not be simply shared with

all grid elements therefore their "sharing" is provided by their deployment to a storage element (SE) once time. The only required applications are then installed on computing element (CE) during every job execution.

Both supported modes can be carried out without any modifications of Module System. Desired "non-standard" functions can be performed by user/administrator through defined hooks (so-called **modactions**). The **modaction** is a script that is executed during any action of **module** command and can be advantageously used for adding a specific action. This can be subsequently used for solving problems with applications distribution under EGEE environment utilizing that fact modaction script

- it behaves differently on user interface (UI) and worker node (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

3.3 Setting User Preferences

Enhanced version of Module System also provides easy to use configuration of the CEL user preferences by a single command which is menu driven. An user can set up the behavior of module name completion and the visualization of listed modules. The user is also allowed to select modules to be automatically loaded during specific site activation and to change priority between system and user module caches. Furthermore, the user is permitted to extend available application portfolio by own module specifications (through user modules).

3.4 Interactive Software Repository

The Module System was also extended with interactive browser of the module database containing real-time list[1] of available software realizations. This service can show the list of available applications with or without the accessible application versions. Moreover, this information is integrated with the detailed description of applications, i.e. with documentation of particular compilation and installation in the MediaWiki[2] format.

4 Charon System

The administration of a computational job through its complete lifetime within a grid environment would definitely demands easy job submission, easy parallel executions of applications. The system should naturally allow user to focus only on desired task not to all things related to submissions; often repeated things should be process automatically and it would be useful to keep information about job during its execution and/or after its execution. All of mentioned

[1] http://troll.chemi.muni.cz/whitezone/development/charon/isoftrepo/

[2] http://troll.chemi.muni.cz/whitezone/development/charon/wiki/index.php

requirements are fulfilled in the Charon System – the second major part of CEL framework.

Charon System is used for job submission and subsequent job management through an encapsulation of a single computational job resulting into minimization of overhead from direct native middleware utilization (e.g. JDL file preparation, *etc.*).

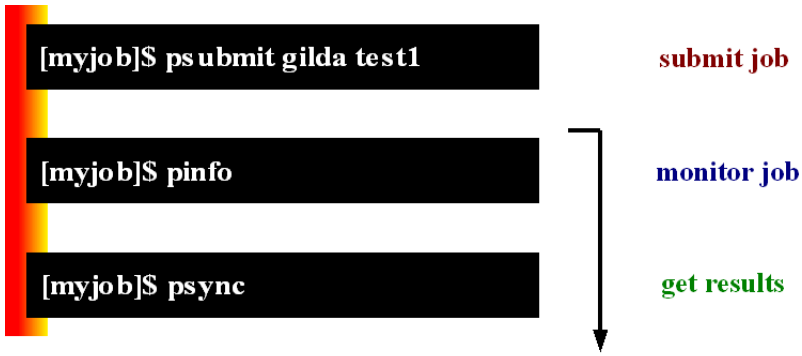

Fig. 3: A typical computational job flow available during utilization of Charon environment.

Current version of the internal part of Charon System itself was changed significantly as the multi-site approach is now supported. Besides that error messages are now more descriptive and they guide user to reasonable solution if it is applicable. There are several rules (job restrictions) that have to be taken into account while utilizing Charon System

- 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)

For specific types of jobs Charon System supports job autodetection in such a way that the user can specified input file instead of script and Charon System will subsequently prepare a script for its processing. Currently autodetected jobs are Gaussian and PovRay inputs.

Detailed configuration of Charon System allows advanced users to set particular values influencing default Charon System behavior. The user can change the job data transfer modes between UI and WN (synchronization modes) from default "gridcopy" mode (all data within job directory as input, all data within

job directory as result) to "stdout" mode (all data within job directory as input, only standard output as result – other data are discarded). Moreover, user specification of particular resources (identification of specific CE) or properties (fine grained selection of computational resources through Requirements item in JDL) is available too. All of above setup items can be uniformly combined into a single word – an alias. The precise configuration of all Charon System settings is accessible through one single menu driven command **pconfigure**.

5 Conclusion

The Charon Extension Layer framework currently provides following set of benefits in the field of single job management: encapsulation of a single computational job, minimization of overhead resulting from direct native middleware usage, easy submission and navigation during job lifetime and same usage in single/parallel executions of applications. From the application management point of view CEL provides easy application initialization seamlessly interconnected with comfortable enlargement/modification of the application portfolio available within a specific grid environment. The futher development of CEL is mainly based on the feedback from users communities requiring new features for implementation (solution of apllication licencing issues, graphical user interface for job management, utilization statistics).

Acknowledgment

The authors would like to thank personally to Luděk Matyska from CESNET (Czech Education and Scientific Network) institute and Jaroslav Koča from NCBR (National Centre for Biomoleculer Research) for allowing start of CEL development and all subsequent support. EGEE II project (contract number RI-031688) and EGEE project (contract number IST-2003-508833), respectively, are funded by European Commission. Financial support from Ministry of Education, Youth, and Physical Training of the Czech Republic (contract number MSM0021622413) and Grant Agency of Czech Republic (contract number 204/03/H016) is also gratefully acknowledged.

References

1. PBSPro, http://www.altair.com/software/pbspro.htm
2. OpenPBS, http://www.openpbs.org/
3. Condor, Condor, http://www.cs.wisc.edu/condor/
4. LSF, http://www.platform.com/Products/Platform.LSF.Family/home.htm
5. Globus, http://www.globus.org/
6. LCG, http://lcg.web.cern.ch/LCG/
7. gLite, http://glite.web.cern.ch/glite/
8. J. Kmuníček, P. Kulhánek and M. Petřek, "CHARON System – Framework for Applications and Jobs Management in Grid Environment", In *Krakow Grid Workshop 2005*, Krakow: Academic Computer Centre. 2006.

9. Environment Modules Project, http://modules.sourceforge.net/
10. $\mathcal{METACentrum}$ project, http://meta.cesnet.cz/
11. EGEE project, http://www.eu-egee.org/