A Graphical Frontend for Utilization of Grid
A CharonGUI Use Case

Vítězslav Plšek, Jan Kmuníček, and Martin Kuba

CESNET z. s. p. o., Zikova 4, 160 00 Prague 6, Czech Republic
Masaryk University, Botanická 68a, 602 00 Brno, Czech Republic
{kmunicek, makub, winsik}@ics.muni.cz

Abstract. The Charon Extension Layer (CEL) is a unique generic system for computational jobs and applications management within grid environments. It provides a sophisticated command-line interface that encapsulates all operations required to control a computational job lifetime - it allows users to submit their jobs, check jobs' statuses and get results using only three basic commands. In this paper we present CharonGUI - a graphical user interface built over the original CEL. CharonGUI adopts all advantages of the CEL with their benefits and turns them into a graphical representation. Moreover, CharonGUI simultaneously brings a set of new interesting features that offers additional comfort for the grid end users. CharonGUI is expected to complement the command-line version of CEL allowing users to have both approaches available in parallel.

1 Introduction

Communities of grid environments users have grown markedly in several last years also due to great effort of scientists and developers who work on many utilities for end users to be able to use advantages of grid environments in more effective way. Nowadays, majority of these tools and utilities provides command-line interfaces. They are advanced and effective on one hand but on the other hand they can seem to be far too complex for complete grid newcomers and for users who prefer graphical interface. One of these utilities is Charon Extension Layer (CEL). It is a unique system for computational job management in grid environment, generic enough for wide range of scientific applications. CEL system creates a layer upon the grid middleware and makes the access to the distinct grid infrastructures uniform. Original CEL system has been built as a command line interface and it unifies the provides the way of utilization multiple middlewares.

CharonGUI is an attempt to demonstrate how an existing command-line system can be enhanced in several directions by graphical extension. There was a simple request for creating utility that can help users to manage computational jobs in a more effective way in the beginning of the CharonGUI development. Therefore, the planned graphical tool was supposed to provide at least the features which can ensure minimally the same comfort as the CEL provides for its current users. The basic requested features were the following: multiplatform concept, support for distinct Grids, preservation of job life cycle characteristics, intuitive and user-friendly interface, multilingual support.

There has been a substantial amount of efforts already invested into development of user-friendly high-level middleware tools. Several categories of these tools can be distiguish - on one hand there are very purpose-specific applications/instruments written for a particular aim or to serve a particular community as AliEn[1], Moteur[2], Taverna[3], Kepler[4] or DIANE[5]. On the other hand there are well established web approaches allowing end users to utilize grid environments in a generic way through web portals as GENIUS[6] or P-GRADE[7]. Probably the most similar solution as CEL/CharonGUI is offered by frameworks as Ganga[8], GridWay[9] or gEclipse[10].

However, the CEL differs from the ones described earlier in several aspects. CEL has been built as a modular framework consisting from two main subparts - Charon system and Modules system. Charon system takes care of jobs administration itself while Modules system serves for maintenance of specific applications. The main advantage of CEL is bridging the gap among utilizing different grids using different tools. CEL makes the way of work uniform and forces users to learn only three commands to manage their jobs properly (*psubmit* for submission, *pinfo* for status checking, *psync* for results retrieval). In addition, a fourth command (*module*) is requested for exploring available applications. The complete insight can be obtained form corresponding detailed description[11, 12].

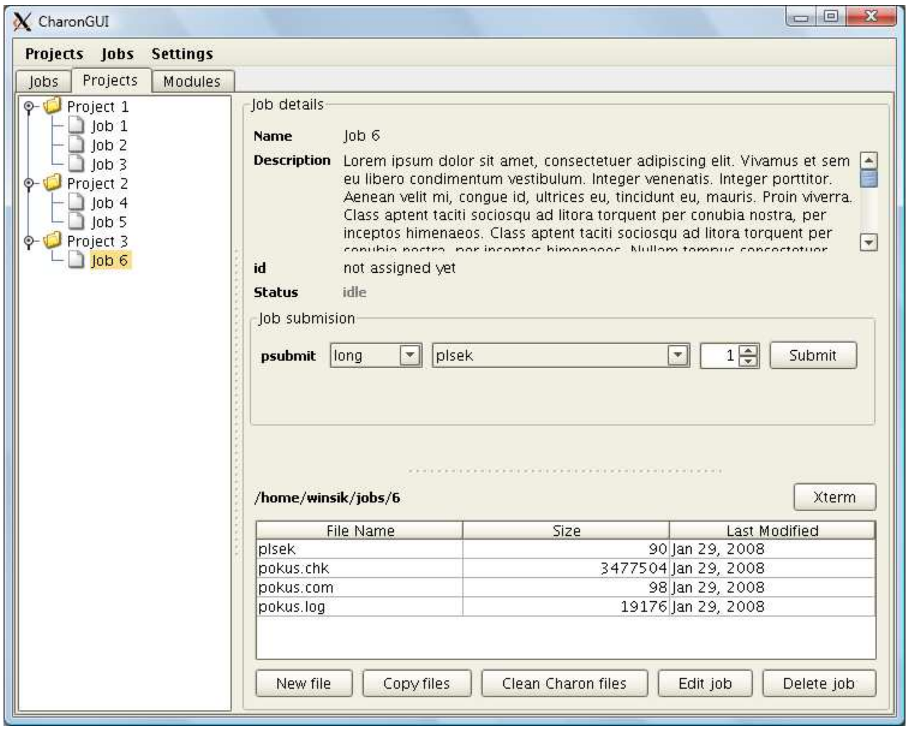
2 CharonGUI System

CharonGUI represents a graphical user interface to the Charon Extension Layer system. CharonGUI offers a simple and intuitive interface to a predefined set of options required for seamless work in a grid environment in highly useable and reliable way.

CharonGUi is a Java-based application running on the top of a user interface (UI) installation. UI is usually provided by a virtual organization (VO) as a dedicated machine from which the user can access the fabric infrastructure. In the current implementation CharonGUI runs at an user interface machine where CEL system is installed. When started remotely CharonGUI displays on a remote X-server or its emulator. The one and only one prerequisite on the server side is to have the Java Runtime Environment (JRE) installed. Security and access to grid resources is handled by lower layer of particular middleware. The channel for communication with the X-server is authenticated using standard SSH protocol. Based on the end users feedback we plan to implement full authentication features that will allow CharonGUI to operate directly at the client system.

2.1 Design

CharonGUI is implemented in Java[13]. This programming language was chosen as one of the most widely used programming languages and due to its platform independence advantage. CharonGUI does not want to reinvent the wheel therefore it does not want to implement the already implemented and debugged underlying system. This is why we have decided to create "just" a graphical extension which will use the advantages of CEL and turn it to the graphical representation. Users communicate with CEL using three basic commands *psubmit*, *pinfo*, *psync*. CharonGUI uses CEL interface in an identical way by running the commands in the background and getting their results. Manipulation with these commands is encapsulated into one class and it is separated from the main graphical interface. All future changes in CEL may not affect the whole application but only the class.



Name Job 6
Description Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Vivamus et sem eu libero condimentum vestibulum. Integer venenatis. Integer porttitor. Aenean velit mi, congue id, ultrices eu, tincidunt eu, mauris. Proin viverra. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Class aptent taciti sociosqu ad litora torquent per
id not assigned yet
Status idle

psubmit long plsek 1

/home/winsik/jobs/6

File Name	Size	Last Modified
plsek	90	Jan 29, 2008
pokus.chk	3477504	Jan 29, 2008
pokus.com	98	Jan 29, 2008
pokus.log	19176	Jan 29, 2008

Fig. 1. Overview of CharonGUI applications

CEL stores configuration information in XML files. CharonGUI extracts from these files mainly information about the present application modules and about the available queues for job submission. The structure of these files is known and the files can be parsed by competent XML parser so CharonGUI is allowed to get the desired information from them.

2.2 Architecture

Graphical user interface is created using design-tool of the Netbeans IDE[14] called Matisse[15] which allows us to create graphical interfaces by dragging and positioning graphical components using a widget toolkit for Java known as Swing [16]. The professional look and feel Alloy has been used to provide visually stunning feelings from CharonGUI utilization.

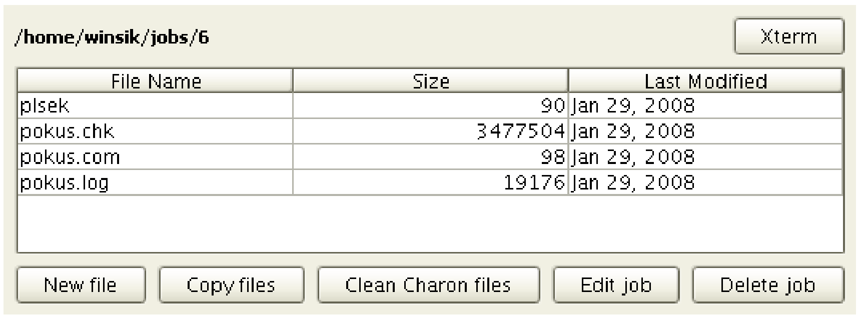
The graphical interface itself is logically divided into the three parts:

- Tab **Jobs**: This tab contains a list of all computational jobs entered into CharonGUI. It supports filtering features and performs updates.
- Tab **Projects**: This is the main tab of CharonGUI, it displays the hierarchical tree of entered jobs categorized according to logical groups: project tree (Fig. 1). The content of the panel on the right side displays a panel which depends on the type represented by a selected object in the tree. The side panel shows information about the selected project or job. It also contains a block allowing job submission, or if the job has already been submitted it contains a panel with job status and a button for status refresh.
- Tab **Modules**: This tab shows information about the application modules available in the current environment and shows them in the same form as the output of command *module*.

3 Results and Experience

CharonGUI has been successfully used and tested within distinct grid environments. CharonGUI adopts the features of CEL, executing the CEL commands in the background, whereby it provides the same functionality as was specified in the basic requirements. Users can submit computational jobs, check their status and get the results in a comfortable graphical environment.

The CharonGUI functions primarily as a laboratory book to keep track of end user's research projects and the corresponding computational jobs allowing full project and/or jobs manipulation. The researchers are allowed to freely manage, check status, sort and study computational jobs belonging to individual analyses. The interface also alows users to invoke an instance of Xtem (when the user requests an access to predefined commands) whenever it is required/needed for further detailed analysis. Moreover, a complete list of application programs already ported (as CEL application modules) to the specific environment can be explored too. CharonGUI is fully localized (currently Czech and English are available) and support a set of visualization skins.


/home/winsik/jobs/6

File Name	Size	Last Modified
plsek	90	Jan 29, 2008
pokus.chk	3477504	Jan 29, 2008
pokus.com	98	Jan 29, 2008
pokus.log	19176	Jan 29, 2008

Fig. 2. Job files management

Apart from the CharonGUI itself a set of new benefits has been recognized:

– **Job files management:** A single computational job is represented by a single directory containing all job's input files in the file system. Job can be added to the CharonGUI by choosing its directory, or creating a new one and filling it with job's input files. CharonGUI shows job files in a table similar to a generic file managers allowing standard basic file manipulations together with some added values as cleaning of CEL control jobs (Fig. 2).

– **Categorization of computational project/jobs:** We suppose a research project to be a logical group of computational jobs descriptions that can be stored in different locations even in different file systems. These descriptions refer to specific job directory and its content (Fig. 3). CharonGUI allows users to group their jobs labels into the projects and create hierarchical structure. It is possible to add an existing job label to the project, or create a new one. Removing a project will also remove the jobs labels or finished jobs from CharonGUI but not from the file system. The projects and their content are displayed as a hierarchical tree where the first level shows the projects and second level shows the jobs labels/jobs.

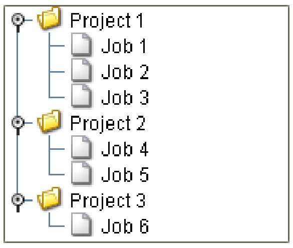
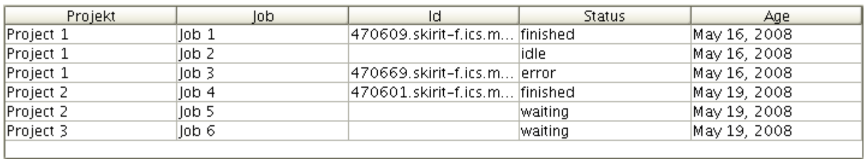


Fig. 3. Hierarchical tree with projects

– **Checking status of multiple jobs:** The biggest benefit for end users is probably the possibility to check status of multiple jobs. A complete list of jobs entered in CharonGUI is listed in the form of a synoptic table (Fig. 4). An exhaustive job overview and filtering functionality is ready to provide an outlook of the individual research project progress. Selection of predefined filters (project and time period-based) allows to define specific groups of jobs whose computational progress/status can be examined in details. The list of jobs can be sorted by relevance to its project and by the lifetime of the jobs (Fig. 5). The table refreshes status of the jobs automaticaly each 10 seconds but user is also able to refresh the table manually.



Projekt	Job	Id	Status	Age
Project 1	Job 1	470609.skirit-f.ics.m...	finished	May 16, 2008
Project 1	Job 2		idle	May 16, 2008
Project 1	Job 3	470669.skirit-f.ics.m...	error	May 16, 2008
Project 2	Job 4	470601.skirit-f.ics.m...	finished	May 19, 2008
Project 2	Job 5		waiting	May 19, 2008
Project 3	Job 6		waiting	May 19, 2008

Fig. 4. Multiple jobs status check

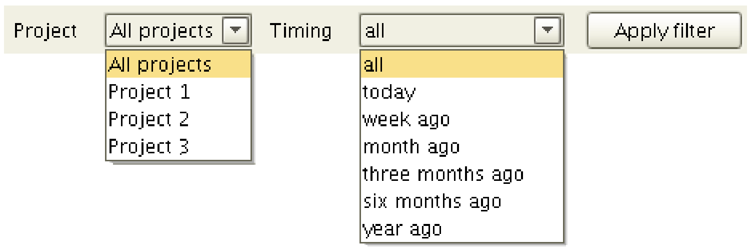


Fig. 5. Job list filter

– **Comfortable job submission**: Before retrieving the results, a job must be submitted to a specific grid environment. CharonGUI makes submitting easier by offering a dropdown menu of the CEL queues and user aliases. The selection of a script to run is also provided by a dropdown menu (Fig. 6). User can also specify the number of processors using the spinner control. After submitting the selected job CharonGUI refreshes its status and displays the information about its identification and timing (Fig. 7).

Job status is checked automatically in an exact time period, but it can be also checked on demand. CharonGUI detects following job statuses:

– *idle*: Job has not been sent to submission yet.
– *waiting*: Job is waiting for submission on the computing resources.

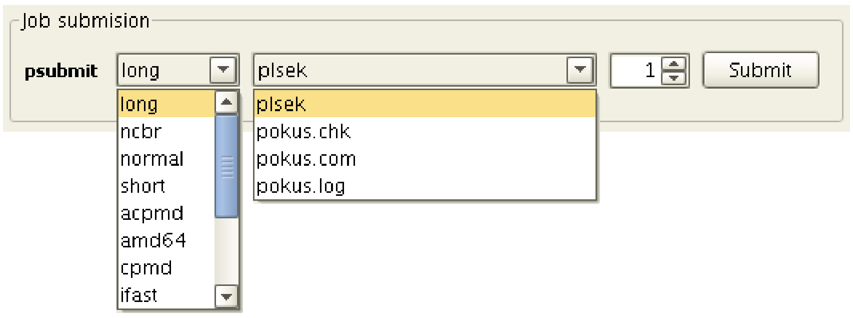
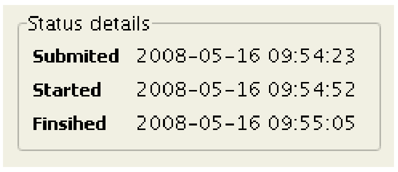


Fig. 6. Job submission panel

- *submitted*: Job has been already submitted.
- *running*: Job is being processed.
- *finished*: Job is successfully completed and finished.
- *error*: Fatal error has been detected. More information can be retrieved directly from CEL output by using xterm.



Status details

Submited	2008-05-16 09:54:23
Started	2008-05-16 09:54:52
Finsihed	2008-05-16 09:55:05

Fig. 7. Panel with job status

When a job is finished CharonGUI refreshes also its list of files and stops the automatic refresh feature for this job.

4 Conclusions and Future Work

We have implemented a graphical interface for the CEL system. This resulted into the CharonGUI application allowing comfortable usage of CEL system together with bringing several new benefits. The development of CharonGUI is expected to continue as already have several further features (configuration tool, interactive application modules list, multiple project hierarchy, client version) to be incorporated in next release version. The CharonGUI is currently available for utilization within two grid environments: *$\mathcal{META}$ Centrum* and VOCE.

- ***$\mathcal{META}$* Centrum** project covers majority of activities concerning Grids and high performance computing in general in the Czech republic. The project

  $\mathcal{META}$ Centrum[17] is one of strategic projects of CESNET institute[18] broadening the infrastructure of academic high-speed network by support for applications requiring extensive computational capacities. The aim of project *$\mathcal{META}$ Centrum* is interconnection of current computational capacity of largest academic centres of the Czech republic and its further enlargement. Nowadays, *$\mathcal{META}$ Centrum* represents the Czech national grid environment.

– **VOCE** is a virtual organization representing all EGEE Grid users within the Central Europe (CE) region willing to utilize computational resources available in the CE. VOCE[19] directly supports CE researchers by providing a computing service. This service consists of sharing data resources and computational capacities available within the CE and the installed Grid middleware and other software to solve various types of computational jobs.

5 Acknowledgements

The CharonGUI and CEL system development is finantially supported by EGEE-III project funded by the European Commission (contract number INFSO-RI-031688). The work has been supported by the research intent "Optical Network of National Research and Its New Applications" (MSM 6383917201) of the Ministry of Education of the Czech Republic.

References

1. AliEn. `http://alien.cern.ch/twiki/bin/view/AliEn/Home`.
2. Moteur. `http://rainbow.polytech.unice.fr/wiki/dokuwiki/doku.php?id=public_namespace:moteur`
3. Taverna. `http://taverna.sourceforge.net`
4. Kepler Project. `http://www.kepler-project.org/`
5. DIANE. `http://it-proj-diane.web.cern.ch/it-proj-diane/`.
6. GENIUS. `https://genius.ct.infn.it`.
7. P-GRADE. `http://www.lpds.sztaki.hu/pgrade/`.
8. Ganga. `http://ganga.web.cern.ch/ganga/index.php`.
9. GridWay Metascheduler. `http://www.gridway.org/doku.php`.
10. gEclipse. `http://www.geclipse.org`.
11. 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, 332–349, 2006.
12. J. Kmuníček, M. Petřek and P. Kulhánek, "Charon Extension Layer - Universal Toolkit for Grid Applications and Computational Jobs Maintenance", In *Krakow Grid Workshop 2006*, Krakow: Academic Computer Centre, 353–359 2007.
13. Java. `http://java.sun.com/`
14. NetBeans. `http://netbeans.org`
15. Mattisse. `http://www.netbeans.org/kb/articles/matisse.html`.
16. Swing. `http://java.sun.com/products/jfc/tsc/articles/architecture/`
17. *$\mathcal{META}$ Centrum*. `http://meta.cesnet.cz`.
18. CESNET Institute. `http://www.cesnet.cz`.
19. VOCE. `http://egee.cesnet.cz/en/voce/index.html`.