Mobile Work Environment for Grid Users. Grid Applications' Framework

Michal Kosiedowski [1], Miroslaw Kupczyk[1], Rafal Lichwala[1], Norbert Meyer[1], Bartek Palak[1], Marcin Plóciennik[1], Pawel Wolniewicz[1], and Stefano Beco [2]

[1] Poznan Supercomputing and Networking Center,
ul. Z. Noskowskiego 12/14, Poznan, Poland
{miron, syriusz, meyer, palak, marcinp, pawelw}@man.poznan.pl
[2] DATAMAT S.p.A.,
Via Laurentina, 760 - I-00143 Rome, Italy
stefano.beco@datamat.it

Abstract. In this article we aim to describe a project developing of the Migrating Desktop infrastructure for mobile users. This functionality refers to the work environment of the users who very often change their location. The user interface called the Migrating Desktop, or grid desktop, is a very useful environment that accomplishes an integrated set of services and real applications, which could be run on the grid. We introduce a framework for improving the grid application launching. The possibility of usage of interactive application in developed environment is presented.

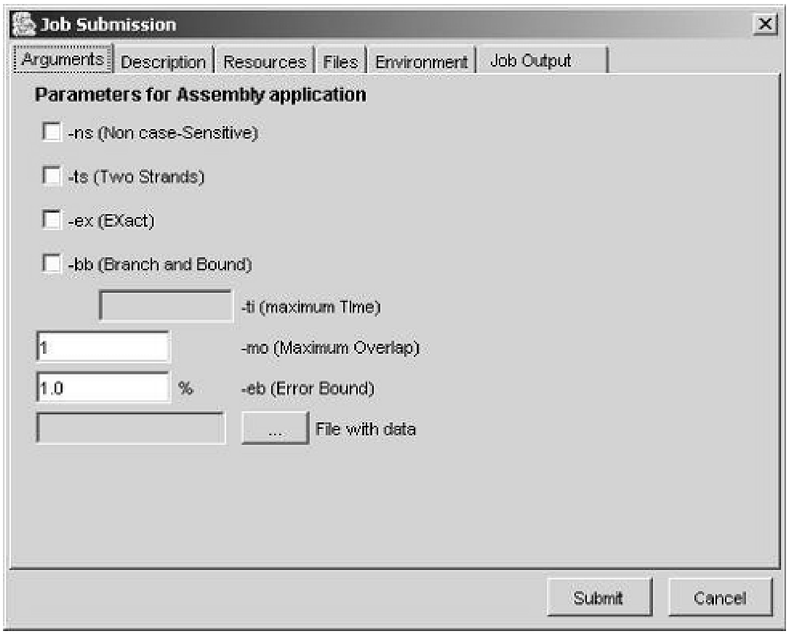
1 Introduction

What is needed for tomorrow is the proper remote and individual access to the resources, independently of the original location of the user. In the CrossGrid project (IST-2001-32243) [1] we introduce the Migrating Desktop [7]. It creates a transparent user work environment, independently of the system version and hardware. The Migrating Desktop would allow the user to access grid resources and his/her local resources from remote computers, like i.e. laptops. It will allow to run applications, manage data files, store personal settings (configuration definitions that characterise e.g. links to the user data files, links to applications, access to application portals and/or specialised infrastructure, as well as windows settings), independently of the localisation or the terminal type. In the paper we present our idea and the way of developing mechanisms for easier adjustment of the application which remains to be run on the grid. The Container and the Application Plugin paradigm will be discussed later on. We present the types of interactivity over the grid and the mechanisms for bridling it in our environment.

As an underlying layer we develop the middleware called the Roaming Access. This infrastructure is a set of modules and their interconnections hidden 'behind' the user interface. The Roaming Access and the Migrating Desktop features will not support the „moving users". It means that no special mechanisms to access the grid re sources via mobile phones, PDAs (such as palmtops, organisers, etc.) will be considered within the confines of the given project. As a bottom line of the middleware we use the Globus toolkit [3]. These facilities give us important functionality like security policy, simple remote operations, user account mapping, etc. Some elements used in this work come from the DataGrid project [2] like the idea of the grid interactive job submission, the Virtual Organisation (VO) paradigm. The architecture of the project was fully detailed in the project documentation [5] and in the paper [6]. The components of the Roaming Access were presented in [7].

2 Application Framework

The Application Plugin and the Application Container are a general view of showing application specific input and output. These allow preparing the portal framework, which is independent of application type, so that we could easily extend it and prepare for next applications. It supports the batch and interactive application as well. It is worth mentioning that this framework will soon be integrated with the GVK (Grid Visualization Kernel) [10]. We would like to emphasize that the communication methods between the distributed nodes involved in the computations are not the point here. It could be MPI jobs and the HLA based as well [9].



Parameters for Assembly application

- [ ] -ns (Non case-Sensitive)
- [ ] -ts (Two Strands)
- [ ] -ex (EXact)
- [ ] -bb (Branch and Bound)

-ti (maximum Time)

1 -mo (Maximum Overlap)

1.0 % -eb (Error Bound)

... File with data

Fig. 1. The example of the Application Wizard – submission of the Assembly Application.

The Migrating Desktop provides a wizard that the user can use to specify the job details. This wizard (see Fig. 1.) simplifies the process of specifying parameters and limits, suggesting user default or last used parameters. That wizard consists of several panels. Two panels are reserved for the application specific plugin – the Arguments Panel and the Job Output Panel. Developers of the specific application should implement the contents of these panels in term of the Application Plugin.

The Application Container is a framework for the Application Plugin. It is an abstract class that is extended by the Application Plugin and extends the graphical component (JPanel) [8], which can be placed into different dialogs, panels etc. The Application Container can be characterised in the following way:

A generic abstract class that the Application Plugin will extend.

It is application independent.

It gives the Application Container API that can be accessed from the Plugin. They are represented by a set of methods.

It controls the Application Plugin by restarting, stopping it, catching error messages and invoking appropriate actions.

It can set or get parameters within the Application Plugin.

The Application Plugin is the content of the Application Container. It extends the Application Container class and implements a set of its abstract methods. It should provide and implement two panels of the Job Submission Wizard: the input arguments and the output specification. It should be able to visualise the output of application. The Application Plugin can be characterised in the following way:

It will have a reference to Application Container API, so that, if necessary, it could invoke a function from the Application Container.

The Application Container and Plugin can be placed on the separated servers.

It should implement functions that allow the control (by the Application Container) of:

- Setting arguments,
- Getting arguments,
- Getting default job name.

The Application Plugin can be a java JApplet class if its author considers it reasonable . However, the supported plugin should define all necessary functions (should extend the Application Container class).

Generally, there will be two kinds of the Application Plugins: batch and interactive work application plugin. The application specific parameters are set in the Arguments panel (see Fig. 1.). Pre-verification of the parameters and the graphical parameters setting (eg. operations on land maps that points are input for application) will be presented on this panel, too. Graphical visualisation of results will be presented on the Job Ouput panel, if necessary. The Interactive Job Plugin is a more complicated kind but it will contain all batch work plugin functionality. It shows the interactive stream of the job output. The interactivity is described later on.

3 Interactive Applications

A Grid Batch Job (GBJ) can currently be submitted using standard queuing software: the user submits the job providing a description by a Job Description Language file and waits for the end of the job asking for its status.

When the job is completed, the user will be able to request for output results downloading in his local machine.

When the user submits a Grid Interactive Job (GIJ), he/she needs the allocation of grid resources throughout his/her "interactive session". During the whole session a bi-directional channel is opened between the user client and the application programme on a remote machine. The entire job input and output streams are exchanged with the user client via this channel: the user sends input data to the job and receives output results and error messages. Details on the mechanism we foresee to manage GIJ's are in the following sections.

There is an additional group of applications, and they consist of zillions of chunk jobs persisting several minutes. Usually, the user – the decisive element - chooses the right way of running a job. Generally speaking, we can call such job an Application Interactive Job (AIJ). In this case, the Web Server automatically submits a GIJ that can be seen just as a container or, in other words, as a "shell" that will run on allocated grid resources. It opens a direct bi-directional connection between the Portal Server and allocated resources. The user chooses the AIJ that will run on allocated computing element(s), and the input data for the chosen application job.

In other words, we can say that:

the GIJ is the container. It is a piece of software that acts as a shell/interpreter/executor of an AIJ. It is a domain-dependant application. It sits on the needed resources (defined in accordance with the AIJ needs). It shall have a function capable to interact via standard streams with the Web Server. It will be submitted using a JDL via the Scheduling Agent.

the AIJ is the content. It is the real application that the user wants to be executed over the Grid. It is a domain-dependant application. It will be interpreted/executed by the GIJ. It will be submitted using whatever language is suitable for the application (e.g. C++ scripts) directly to the GIJ.

To give an example, the Web Server performs the GIJ submission after the user chooses a DATASET for his/her purposes within a list of DATASETS available for his/her Application Virtual Organisation. The submitted GIJ contains the process called the "Interactive Session Manager" (ISM) that handles the interactive session from the point of view of the application. ISM will run as close as possible to the Storage Elements hosting a physical copy of the selected DATASET.

Then the user "submits" one or more application scripts (the AIJ's). The user can choose the Application jobs among an available list of predefined applications or he/she can define one for his/her specific needs in an editor window.

A very important feature in interactive environments is the ability to recover from unexpected failures of e.g. workstation. There is no problem in case of batch jobs, but interactive applications need to be equipped with extra functionality. It would be better not to rewrite the existing application codes. However, it should be considerable to build the additional linkable library for the grid interactive session management. There is no such possibility in the existing load facilities (queuing systems, etc...).

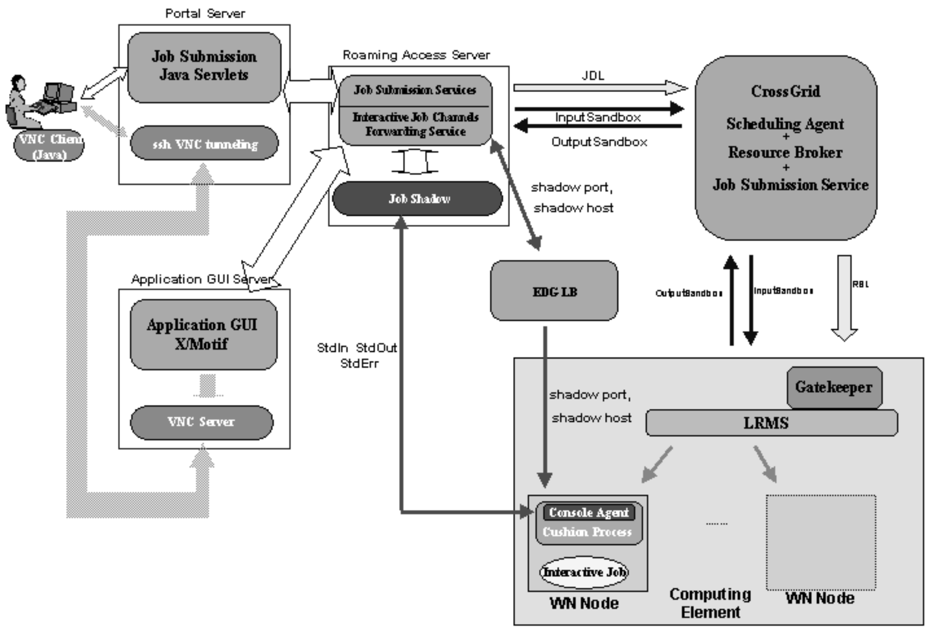


Fig. 2. How the interactive session works.

We present the solution that will be implemented for EU-CrossGrid (see Fig. 2.).

The interactive sessions [4] are handled by the Grid Console (GC) that is a system provided by Condor for getting mostly-continuous input/output from remote programs running on an unreliable network. A GC is a *split execution system* composed by two software components: an *agent* (Console Agent – CA) and a *shadow* (Console Shadow – CS or Job Shadow – JS, as it will be a slightly modified CS).
The *split execution system* is a special case of an *interposition agent*. An *interposition agent* transforms a program's operation by placing itself between the program and the operating system. When the program attempts certain system calls, the agent grabs control and manipulates the results.

In the *split execution system* an interposition agent (CA) traps some of the procedure calls of an application, and forwards them (via RPC) to a shadow process (CS) on another machine. Under this arrangement a program can run on any networked machine and yet execute exactly as if it were running on the same machine as the shadow. All the network communications are GSI-enabled.

The Console Agent runs on a Worker Node and it is a shared library that intercepts reading and writing operations on stdin, stdout, and stderr of the running job. When possible, the CA sends the output back to the CS. The shadow manages the input and output files according to the request of the agent.

If the output sending fails, CA will write it on the local disk instead. It does not matter why the input/output operation failed; the CA will keep the process running. At regular intervals it will attempt the network connection again. If the connection succeeds, it will transfer any buffered data to the shadow, and then resume normal operation.

The CA retries failed operations at regular intervals for a certain number of times, after which it will give up and kill the process. The number of times to retry and the number of seconds to pass between each retry are configurable.

The user submits his/her interactive job via the Portal Server (PS) and interacts with the remote machine where the job is running (Worker Node- WN). The Portal Server starts/stops the Job Shadow on the Submission Machine when the user opens/closes the interactive session. The PS writes in the Logging&Bookeeping (LB) and in the JDL file the ShadowPort (port number assigned to the shadow) and the ShadowHost (the host where the Job Shadow runs); this information is also used by the agent. In the JDL there will be an attribute called JobType set to "Interactive".

Before the submitted job goes in the running status, a script called *JobWrapper* prepares the environment for the job, transfers on the WN additional files within the job InputSandbox and starts the *Cushion Process*.

The *Cushion Process* is a process linked to the CA that handles the standard streams between the CA and the running job by using a named pipe. This process is needed to avoid dynamic linking between the user job process and the CA library. It also allows to avoid linking between a system library and a user process, as well as a possible failure that could appear if the user job works with other interposition agents (besides the CA).

For the standard X-Windows applications, the VNC [11] protocol will be used. It is foreseen to be integrated next year. The corresponding connections are shown in Fig. 2.

4 Summary

The Roaming Access and Migrating Desktop implement new generation tools for the end user according to the growing demands and needs. Nowadays it is not enough to have Internet or Grid access. It can be observed that the group of inexperienced users has increased significantly. On the other hand, the Grid and Internet populations are among the most mobile of all (moving between Virtual Organizations in an open network environment). Therefore, the Grid should be able to deliver a service allowing to keep and restore the users' working environment. The Job Wizard functionality is the response for easy and flexible work with the grid application, including interactivity. The mentioned functionality is a cutting edge in the grid utility. This is a goal that we approach by the delivery of the tool for the application programmers, and a uniform environment for the end-users.

The Desktop Portal Server extends the functionality of the Application Portal Server by providing a specialised advanced graphical user interface and a mechanism that allows the user to access all files stored on his/her personal machine available from other locations/machines. The mentioned functionality is developed within the CrossGrid project (IST-2001-32243, http://www.eu-crossgrid.org) and its first Prototype was released in February 2003.

References

1. Annex 1 to the "Development of Grid Environment for Interactive Applications" - EU-CrossGrid Project, IST-2001-32243, http://www.eu-crossgrid.org/CrossGridAnnex1_v31.pdf
2. Annex to the "Research and Technological Development for an International Data Grid" - EU-DataGrid Project, IST-2000-25182, http://eu-datagrid.web.cern.ch/eu-datagrid/1Y-EU-Review-Material/CD-1Y-EU-Review/3-DataGrid-TechnicalAnnex/DataGridAnnex1V5.3.pdf
3. Globus Toolkit 2.0, http://www.globus.org
4. Kupczyk, M., Lichwala, R., Meyer, N., et. al., Roaming Access and Portals: Software Requirements Specification", EU-CrossGrid Project, http://www.eu-crossgrid.org/Deliverables/M3pdf/SRS_TASK_3-1.pdf
5. Bubak, M., Malawski, M., Zajac, K., "Towards the CrossGrid Architecture", Proceedings 9th European PVM/MPI Users' Group Meeting, Linz, Austria, September/October 2002, LNCS 2474.
6. Bubak, M., Malawski, M., Zajac, K., "Current Status of the CrossGrid Architecture", Proceedings of the Cracow '02 Grid Workshop, December 11–14, 2002, Cracow, Poland
7. Kupczyk, M., Lichwala R., Meyer, N., Palak, B., Plociennik, M., Wolniewicz, P., "Roaming Access and Migrating Desktop", Proceedings of the Cracow '02 Grid Workshop, December 11–14, 2002, Cracow, Poland
8. Java Library, http://sun.java.com
9. High Level Architecture Run-Time Infrastructure RTI 1.3-Next Generation Programmer's Guide, https://www.dmso.mil/public/transition/hla/
10. Heinzlreiter, P., Kranzmueller, D., Volkert, J., "GVK – Visualization Services for the Grid", Proceedings Cracow '02 Grid Workshop, December 11–14, 2002, Cracow, Poland
11. Virtual Network Computing, http://www.uk.research.att.com/vnc/, University of Cambridge.