

# Scheduling Interactive Tasks in the Grid-based Systems

Marcin Okoń, Marcin Lawenda, Norbert Meyer, Dominik Stokłosa,  
Tomasz Rajtar, Damian Kaliszczak, Maciej Stroiński

Poznań Supercomputing and Networking Center,  
ul. Noskowskiego 12/14 Poznań, Poland  
Corresponding author: Marcin Okoń, hawky@man.poznan.pl

## Abstract

*In the user-interactive tasks the time slot reserved for running the task on a computational machine or scientific device must be synchronized with user preferences, considering specific work hours, daily schedule etc. In case of interactive experiments (eg. Virtual Laboratory project) the deciding factor is the current device availability, workload, maintenance periods etc. This kind of data can often become very non-deterministic. This solution, developed for Virtual Laboratory, was designed to be reused in any project dealing with the same problem. It has a modular architecture, all the necessary functionality and works closely with well known Grid solutions.*

## 1. Introduction

Distributed computing environments are widely used in realisation of complex and resource-demanding applications. By resources we mean CPUs, memory, disk space and graphical subsystems. The latter are mostly used in case of user-interactive data processing.

In general, in most Grid-based systems, we can distinguish two main kinds of tasks: user-interactive and batch ones. User interactive tasks can be of various types, such as: data processing, visualisation or real-time process control. Their common element is that the actual processing must be performed directly by the users, via the graphical user interface (GUI).

The scheduling and running of batch jobs is widely described and implemented in various computational systems. This cannot be said in case of interactive tasks, which are more difficult to schedule and run, because of the requirement of human presence. The key to success is to synchronize user presence and the actual task run-time.

This article describes the scheduling and management of interactive tasks (running, session prolonging, session ending). The solution was originally developed for the Virtual Laboratory system [1]. It is incorporating GRMS [4] and other Globus [5] systems. The major design requirement was to implement a platform independent system, enabling it to be easily integrated with other solutions and applications.

The Virtual Laboratory research project has been developed in Poznań Supercomputing and Networking Center since the beginning of the year 2002.

In general, the Virtual Laboratory is a distributed environment, providing a remote access to the various kind of scientific equipment and computational resources [1]. Users can submit their tasks in form of Dynamic Measurement Scenario [3] – the sets of experiments and computational tasks of different kind. These tasks form a dependencies graph – describing the connections between them and all possible flowpaths – the actual flowpath is determined upon run-time - based on results obtained on each stage. The tasks are scheduled on one-by-one (or on group) basis, after the appropriate results from the preceding tasks are obtained.

The Virtual Laboratory is not a standalone system. It was designed to cooperate with many other grid systems, providing only the purpose-specific functionality and relying on well known and reliable grid solutions. The most important system the VLab cooperates with is the Globus Toolkit [5] – in the scope of scheduling computational tasks, software services and libraries for resource monitoring, discovery, and management. All computational tasks submitted in the VLab system are transferred to the Globus via the GRMS [4] module – an important part of the GridLab project [6]. Among other external systems used by the Virtual Laboratory are: the VNC system, SZD (data management system), authentication module and RAD authorization system.

## **2. The peculiar nature of the interactive experiments**

As it was mentioned in the introduction, we can distinguish between two major types of tasks: batch and interactive.

The biggest difference (and difficulty) between those types is that - in the interactive tasks - the time slot reserved for running the task on a computational machine must be synchronized with user preferences, considering specific work hours, daily schedule etc. Another aspect is the mechanism which will present the users with the graphical interface of the actual computational (or visualization) application – which is run on dynamically assigned computational server – and allow them to perform their interactive task.

Another, very characteristic type of Virtual Laboratory tasks are the experiments. By the term experiment we mean a task, scheduled to be performed on the remote laboratory equipment, available via VLab to its users. In most cases such experiments will be an interactive processes, with users manipulating directly the remote equipment via specialized control software GUI. Experiments are difficult for formal description. Depending on a specific science domain, there can be many dependencies of external, often indeterministic factors. The device will not be non-stop available for VLab users but will be shared with local researchers (usually with higher priority than the remote users). There are also maintenance periods, in which the device is unavailable. Sometimes the presence and assistance of the device operator may be necessary – especially at the beginning of experiments.

The first scientific device incorporated into the VLab system was an NMR spectrometer. The most important problems with scheduling the NMR experiments, apart from those described above, come from the samples management. To perform an NMR experiment, an actual sample containing chemical compound has to be delivered to the NMR spectrometer and inserted into the machine. This causes a number of scheduling problems. At the task (and corresponding sample) submission point the actual NMR device has to be known and chosen, because the sample has to be sent from the remote location to the device site. The exact time of sample arrival is not known as well, making the exact task scheduling impossible until the sample arrives.

### 3. The Virtual Laboratory Architecture

The Virtual Laboratory system has a modular architecture. The modules can be grouped into three main layers. The general diagram is presented on the figure 1 below:

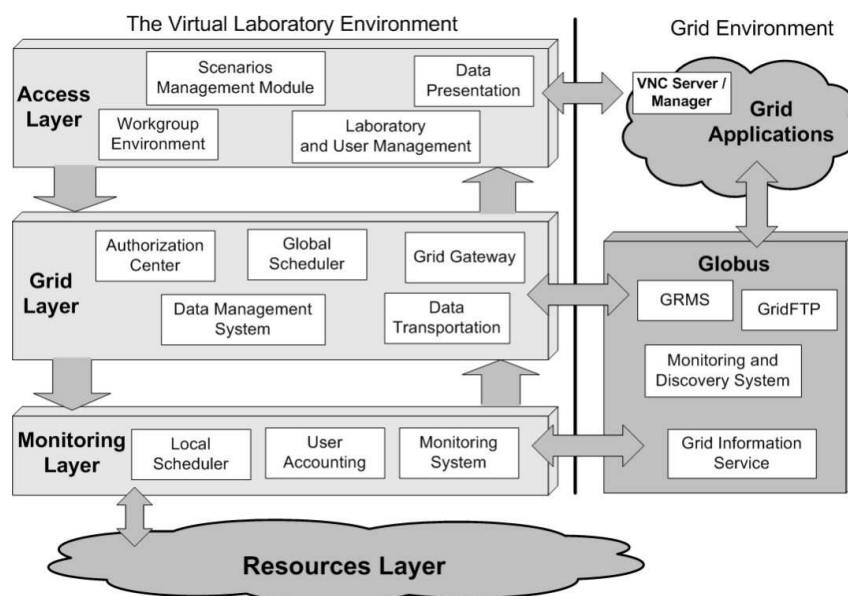


Figure 1. General architecture of the Virtual Laboratory

An Access Layer is the top layer of this structure. Basically, it contains all modules and components responsible for a VLab user access and graphical interface to the system (including a web portal), and data input interface. Below there is a Grid Layer, which communicates with external grid environment. It is responsible for user authorization and authentication, data management, general task scheduling. Modules from this layer also handle the transfer of the computational tasks to the Globus system, and gather feedback data and the computed results. The Monitoring Layer consists of lower-level modules, such as hardware dedicated schedulers, system monitoring, gathering accounting data etc. All the physical scientific devices are located in the Resources Layer, as well as modules responsible for their direct control.

On the Grid Environment side the most important element is the Globus system, with all its components (GRMS, GridFTP etc). Globus also allows to execute computational tasks (batch and interactive) on a wide variety of grid applications.

### 4. Task scheduling

As we mentioned in paragraph 1 and 2, there are two basic types of tasks in Virtual Laboratory: computational and experimental. Experiment is the specific task type for the VLab itself. It is scheduled to be run on scientific hardware and create a very challenging set of difficulties which need to be addressed (see p.2). In this paragraph, we will focus on computational tasks, batch and interactive. Interactive tasks need a definitely more sophisticated approach. The time in which the application GUI performing the task is presented to the user (or in other words: task execution time) has

to be known in advance, and synchronized with user time preferences. The mechanism responsible for displaying the interface has to be carefully designed, to address authorization and security issues. All operations performed on batch and interactive tasks are presented in the paragraphs below.

#### 4.1. Batch jobs

A general diagram describing the flowpath of batch computational tasks is presented on the Figure 2:

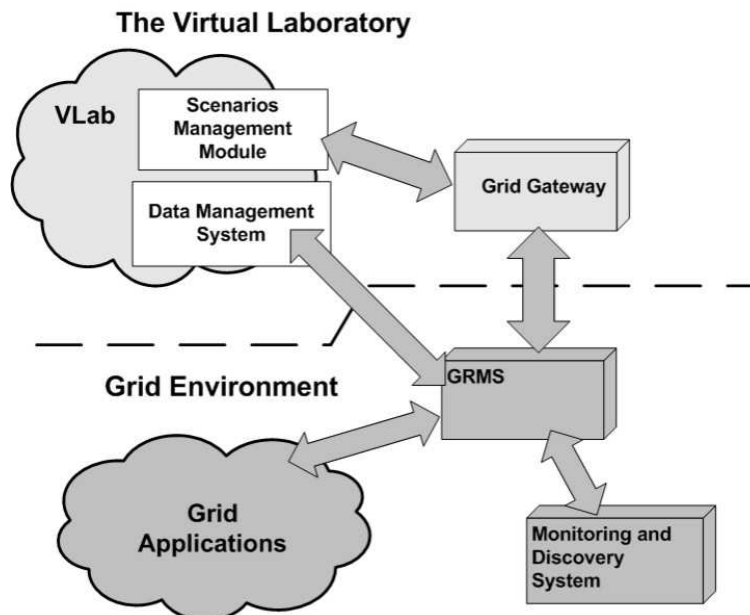


Figure 2. The batch computational tasks processing in the Virtual Laboratory System

The computational task is created in the Scenario Submission Application [3] – a part of the VLab portal, responsible for creating Dynamic Measurements Scenarios. The dynamic scenario is submitted to the Scenario Management Module (SMM). At the appropriate time, during the scenario execution phase, the SMM sends the task to the Grid Gateway module via the Global Scheduler (not shown on the diagram). Grid Gateway sends the task description to the GridLab Resource Management System [4] – an external meta scheduling system, responsible for scheduling and executing the task using the grid computational resources. The input data are passed as references to the Data Management System – the same system is used for uploading the computational results. The online task monitoring and status notification is handled by the GRMS, which contacts the Grid Gateway with the updated information about submitted tasks status.

#### 4.2. Interactive tasks

Handling the interactive tasks is more complicated than the batch ones. Different phases need to be specified here. They are described in the subparagraphs below.

#### 4.2.1. Task submission

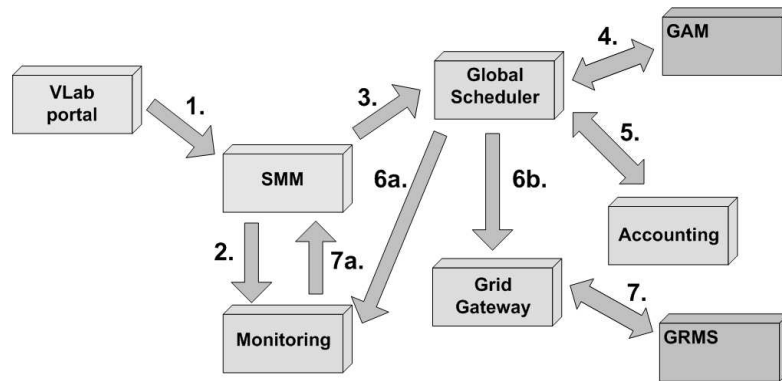


Figure 3. Interactive tasks submission

Interactive tasks are submitted into the system just as the batch ones. Each consecutive step on the diagram was marked. The detailed description is given below.

1. Task (as a one element of the measurement scenario) is sent to the Scenario Management Module from the web portal (Scenario Submission Application).
2. Task is added to the database (via the Monitoring module)
3. At the appropriate time task is sent to the Global Scheduler (GS)
4. GS authorizes the user in the Grid Authorization Module
5. GS checks with Accounting whether the user has not exceeded the limit and can submit the task
6. a) Task cannot be submitted – inform Monitoring module  
b) Task can be scheduled - send it to the Grid Gateway
7. Task is transferred to the GRMS, which schedules the task and sends the information about the results (success/failure) to the Grid Gateway. The GG sends a query about the chosen server, and signs up for notifications concerning the scheduled task.  
a) Task cannot be submitted – inform Monitoring module

#### 4.2.2. VNC session scheduling

In the previous step the interactive task was submitted into the GRMS. Now the GRMS has to decide where the interactive task should be executed. It also reserves a slot for VNC session. Information about the session schedule is returned back to the *Grid Gateway* (as notifications), which in turn sends it to the *Monitoring* module.

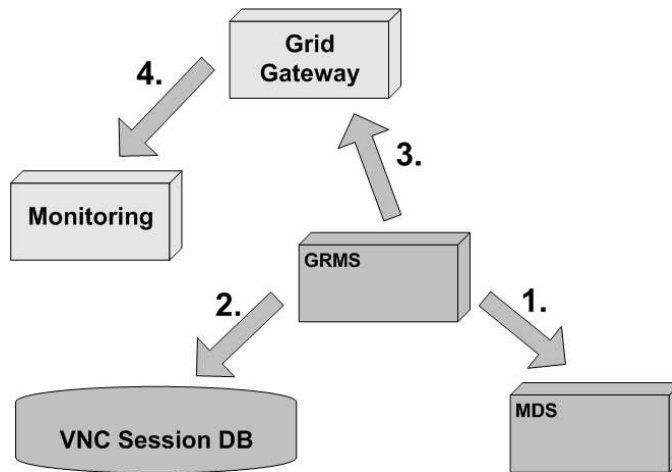


Figure 4. VNC session scheduling for interactive tasks

The detailed operations are described below:

1. The GRMS contacts the MDS system to gather information required for the connection preparation (maximum number of open VNC sessions, etc.).
2. It also contacts a *VNC Session Database* to check the actual sessions state. Taking all those information into account it will reserve a VNC session slot for interactive task invocation and update the database.
3. Scheduling information is passed as notification to the GG, which updates the task status, via the Monitoring module (point 4)

#### 4.2.3. Establishing a secure connection

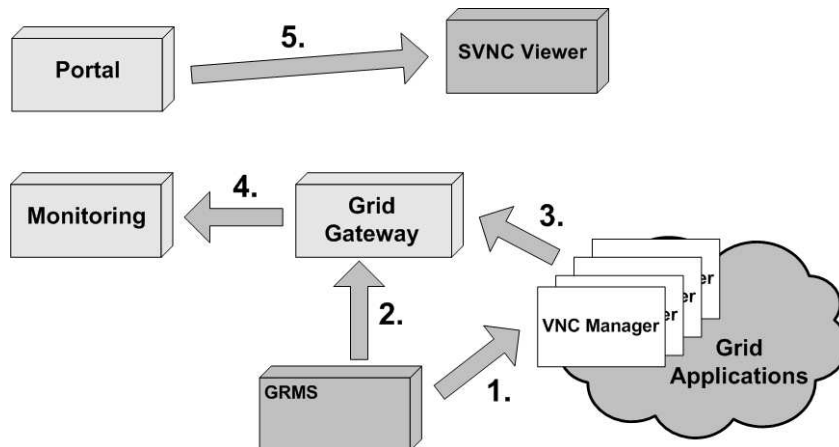


Figure 5. Establishing a secure VNC connection

The interactive task has been submitted to the system and the VNC session has been scheduled. The next step is to prepare the proper environment for the given task, launch it and wait for connection establishment from the VLab user.

1. The *GRMS* launches its scheduled task. The task is defined as an instance of the VNC Manager – which looks up the available port, runs the VNC server and the application
2. The *Grid Gateway* is notified that everything has been prepared and the session can be established
3. VNC Manager reports the port number in use and dynamically-generated password to the Grid Gateway.
4. The *Grid Gateway* propagates all gathered information to the *Monitoring* system
5. The VLab user starts the SVNC Viewer with all the connection parameters taken automatically (and transparently) from Monitoring – and therefore has a full access to the application

#### 4.2.4. Prolonging the session

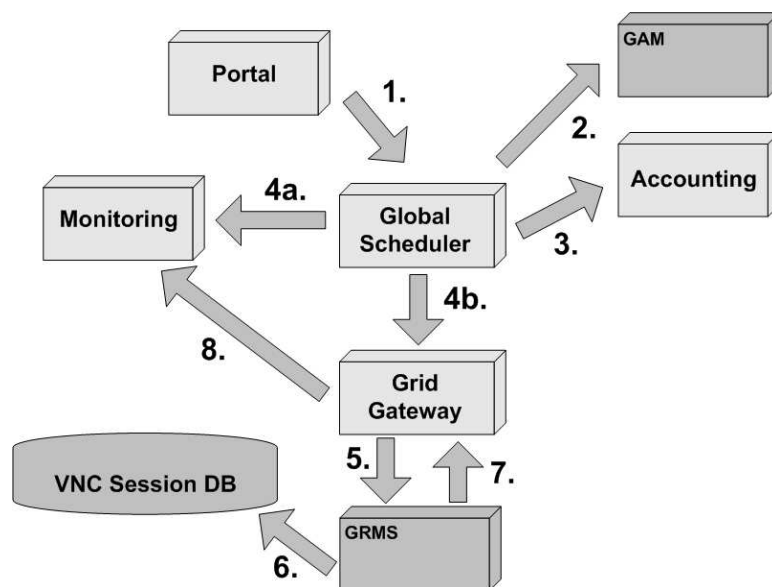


Figure 6. Prolonging the VNC Session

The interactive task can be scheduled for the certain amount of time. The time period is specified by the user during task definition and submission. It may happen that the reserved time slot is too short to complete the data processing. When the reservation period is about to expire, the VLab system displays the appropriate warning and the user is given the possibility to request the session prolongation. After evaluation the actual session state (VNC Session Database) the prolonged access is granted or the request refused.

Prolonging the VNC session step by step:

1. The request is forwarded to the Global Scheduler
2. Global Scheduler authorizes the request in the Grid Authentication Module (GAM)
3. GS checks the user limits in the Accounting module
4. a) Verification failed - the Monitoring is informed that the session extension has failed and the process ends.

- b) Verification is positive - extension request can be sent to the Grid Gateway.
- 5. The request (authorized in the VLab system) is sent to the GRMS.
- 6. Next, the actual sessions state is looked up in the VNC Session Database and new session slot is reserved (if available)
- 7. The response is sent back to the Grid Gateway (as notification). GG updates the Monitoring with the new information (point 8 in the figure)

#### 4.2.5. Finishing the VNC session by the user

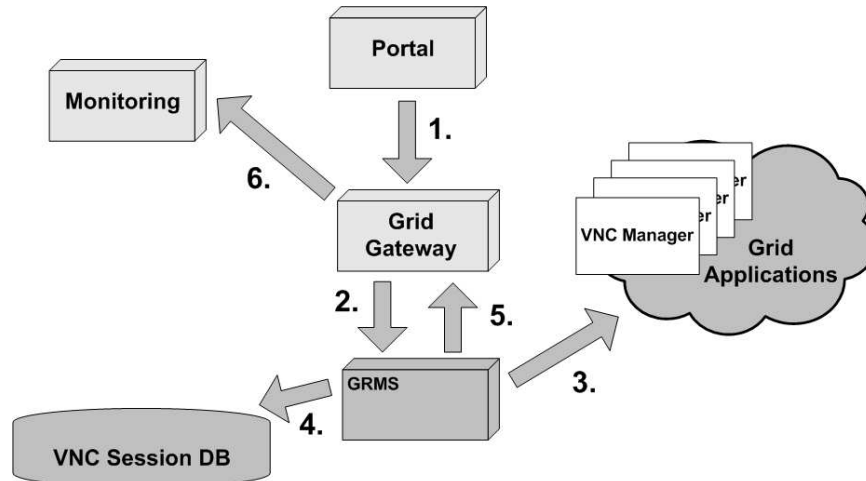


Figure 7. Ending the VNC session by the user

The VLab user has the ability to end an active VNC session at any time after the session has been started. The procedure is explained below:

1. The request is forwarded to the Grid Gateway
2. GG sends it to the GRMS
3. The GRMS system sends the appropriate signal to the instance of VNC Manager responsible for a given application. Application is closed and resources are freed.
4. GRMS updates the VNC Session Database – registration is removed.
5. The GRMS sends the result of this operation to the Grid Gateway as a notification.
6. Grid gateway updates monitoring information

#### 4.2.6. Finishing the VNC session by the system

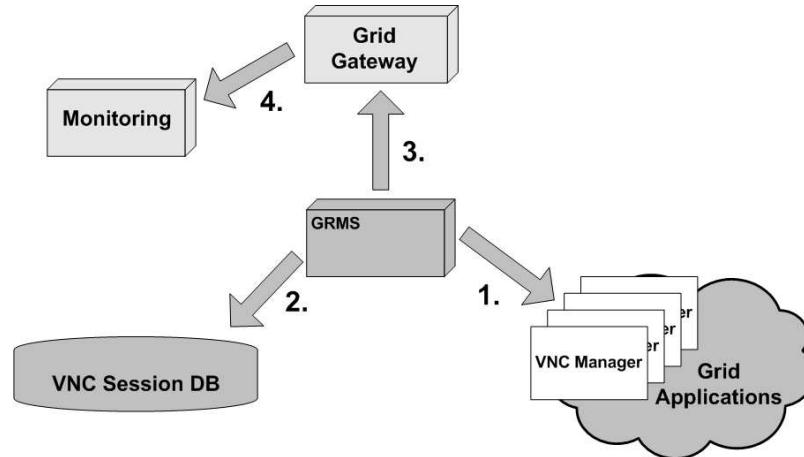


Figure 8. Ending the VNC session by the VLab system

An active VNC session can be terminated by the VLab system (and GRMS) when the reservation period expires, or for any other reason. The procedure is as follows:

1. After the reservation time expires, GRMS sends a signal to end the application (via VNC Manager).
2. GRMS updates the VNC Session Database.
3. GRMS sends information to Grid Gateway that the task was closed.
4. GG updates the task information in the Monitoring module.

## 5. Summary

Distributed Grid-based systems are undoubtedly the technology of the future. Although the described solution of scheduling interactive tasks was developed for the Virtual Laboratory System, it was designed in a way that allows it to be reused in any other project dealing with the same problem. It has a modular, easy to install architecture, all the necessary functionality and works closely with other well known Grid solutions.

At the current stage of development there are no industry standards for this kind of systems dealing with interactive tasks. Therefore, all solutions presented in this article are of experimental character. However, the first successful implementation of this system in the Virtual Laboratory, proves its useful and reliable architecture.

## References

- [1] Virtual Laboratory project <http://vlab.psnc.pl/>
- [2] Lawenda, M., Meyer, N., Rajtar, T., Okon, M., Stoklosa, D., Stroinski, M., Popenda, Ł., Gdaniec, Z., Adamiak, R.W.: General Conception of the Virtual Laboratory. International Conference on Computational Science 2004, LNCS 3038, pp. 1013-1016, Cracow, Poland, June 6-9, 2004
- [3] Lawenda, M., Meyer, N., Rajtar, T., Okon, M., Stoklosa, D., Kaliszan, D., Stroinski, M.: Dynamic Measurement Scenarios in the Virtual Laboratory system. 5th IEEE/ACM International Workshop on Grid

Computing, IEEE Computer Society Order Number P2256, ISBN 0-7695-2256-4, ISSN 1550-5510, pp. 355-359, Pittsburgh, USA, November 8, 2004

- [4] GRMS - WP9 Resource Management - GridLab Project  
<http://www.gridlab.org/WorkPackages/wp-9/>
- [5] Official Globus site: <http://www.globus.org>
- [6] GridLab project: <http://www.gridlab.org>