

Executing Workflow-Based Grid Applications with the Collaborative P-GRADE Portal

Gergely Sipos¹, Csaba Nemeth¹, Gareth J. Lewis², Vassil N. Alexandrov², Peter Kacsuk¹

¹MTA SZTAKI Computer and Automation Research Institute
H-1518 Budapest, P.O. Box 63, Hungary
{sipos, csnemeth, kacsuk}@sztaki.hu

²Advanced Computing and Emergent Technologies Centre
School of Systems Engineering, University of Reading,
Whiteknights P.O. Box 225, Reading, RG6 6AY, UK
{v.n.alexandrov, g.j.lewis}@reading.ac.uk

Abstract

The Collaborative P-GRADE Portal supports the concurrent development and cooperative execution of workflow-based grid applications. In the development phase the collaborative users can contribute to the application with their own knowledge, files and resources. In the execution phase the workflow manager uses the collaborators' delegated identities to access files and to execute jobs in the Grid. By applying different certificates during the different phases of the execution procedure, the workflow manager shares responsibility among the users. The paper introduces the issues related to the execution of concurrently developed grid applications and presents the mechanisms provided by the Collaborative P-GRADE Portal to deal with the issues either in single-grid or multi-grid environments.

1. Introduction

Almost every large production grid had been extended with a grid portal during the last few years. Most of these portals support the execution and monitoring of computational jobs[1]. Others realise workflow developer and executor environments on the top of the underlying grid infrastructures [3]. At the same time, none of the current portals support collaborative grid applications. Although there are a few "collaborative" portals already available for the Grid Community, these environments actually enable the collaboration of users without the concept of "collaborative applications".

Most of these "collaborative" portals extend the isolated application development and execution facilities with text-oriented communication tools, such as chat rooms or message boards. Other approaches enable users to apply each other's grid applications (for example workflows) as starting points to develop new applications. Although both solutions add some kind of "collaborative flavour" to the development process, the result is exactly the same kind of single-user application than in case of a non-collaborative portal. Consequently, none of the current portals

distinguish collaboratively developed applications from single-user applications during the execution phase.

Despite chat rooms and message boards are quite effective tools to share knowledge during the application development process, finally there must be one individual who integrates all the knowledge into a single application. This person has to submit the application into the Grid, has to collect and distribute the results among the interested parties. Even if a "collaborative" portal type let multiple persons contribute to the same application, there is again one individual who manages the application during the execution phase. From the point of view of the portal this person is the exclusive owner of the application. Consequently, only his/her certificates are used during the execution to call the different grid services. As a result, this individual must take all responsibility of the execution.

The centralized execution of collaboratively developed applications raises serious concerns about the practicability and scalability of the current approaches. Because of this limitation we do not consider such solutions as collaborative grid portals. In our terminology a *collaborative grid portal* is capable to provide controlled access to a grid application for multiple users during both the development and

execution processes, and it can share responsibility among these collaborators during the execution procedure.

According to the first part of the definition collaborative grid portals must be design environments specialised for the concurrent engineering of grid applications. Concurrent design environments support the development of complex entities by multiple persons simultaneously [6]. CVS [8] and [7] are probably the two most well-known concurrent design environments for the UK e-Science Community. CVS focuses on software development projects, thus it is specialised for the locking and versioning of files. BSCW realises similar functions for publication authors. We believe that in the near future grid application developers will require similar support for the concurrent engineering of grid applications. Since portals are (or will be) the primary environments for any grid-related work, this function will be probably incorporated into grid portals too. Our first results on the extension of a single-user grid portal – the P-GRADE Portal – with concurrent application engineering support have been presented in [9].

This publication focuses onto the second part of the definition, onto the execution of concurrently developed grid applications. In the remaining part of the paper concurrently engineered grid application are referred to as *collaborative applications*. The P-GRADE Grid Portal [10] is used as an example to present how the involvement of multiple participants can be handled during the execution of collaborative, workflow-based grid applications. Moreover, since the latest version of the P-GRADE Portal can be connected to multiple grids at the same time [11], the paper presents how collaborative applications can be executed with the simultaneous utilisation of multiple grids. The concepts introduced in the paper have already been implemented by the prototype version of the Collaborative P-GRADE Portal, now we are working on the integration of these results into the next P-GRADE Portal release.

The remaining part of the paper is organised as follows: Section 2 introduces the P-GRADE Grid Portal. In Section 3 an overview of collaborative P-GRADE Portal grid applications is given. Sections 4 and 5 present the execution management support given by the Collaborative P-GRADE Portal in single-grid and in multi-grid environments. Section 6 draws conclusions.

2. The P-GRADE Grid Portal

The P-GRADE Portal is a grid portal with the main goal to cover the whole lifecycle of workflow-based grid applications. It supports the development of grid workflows consisting of various types of executable components (sequential and parallel MPI and PVM jobs), executing these workflows in Globus-2 based grids relying on user credentials, and finally analysing the correctness and performance of applications by the built-in visualization facilities [10].

2.1 Grid Applications in the P-GRADE Portal

Workflow applications can be developed in the P-GRADE Portal by the graphical Workflow Editor. The Editor is implemented as a Java Web-Start application that can be downloaded and executed on the client machines on the fly. The Editor communicates only with the portal server application and it is completely independent from the portal framework and the grid middleware the server application is built on.

A P-GRADE Portal workflow is an acyclic dependency graph that connects sequential and parallel programs into an interoperating set of jobs. The nodes of such a graph are batch programs, while the arc connections define data relations among them. These arcs define the execution order of the jobs and the dependencies between them that must be resolved by the workflow manager during the execution. An ultra-short range weather forecast (Nowcast) grid application [12] is shown in Figure 1 as an example for a P-GRADE Portal workflow.

The big rectangles (labelled as *delta*, *cummu*, *visib*, *satel* and *ready*) represent the jobs of the workflow. Small rectangles (labelled by numbers) around the jobs are called ports and they represent data files that the connected jobs expect or produce (green ports represent input files, light grey ones represent output files). Directed arcs interconnect pairs of input and output files if an output file of a job serves as an input file for another job. In other words, arcs denote the necessary file transfers between jobs. Three kinds of input files can be connected to a job: indirect input files, direct local input files and direct remote input files.

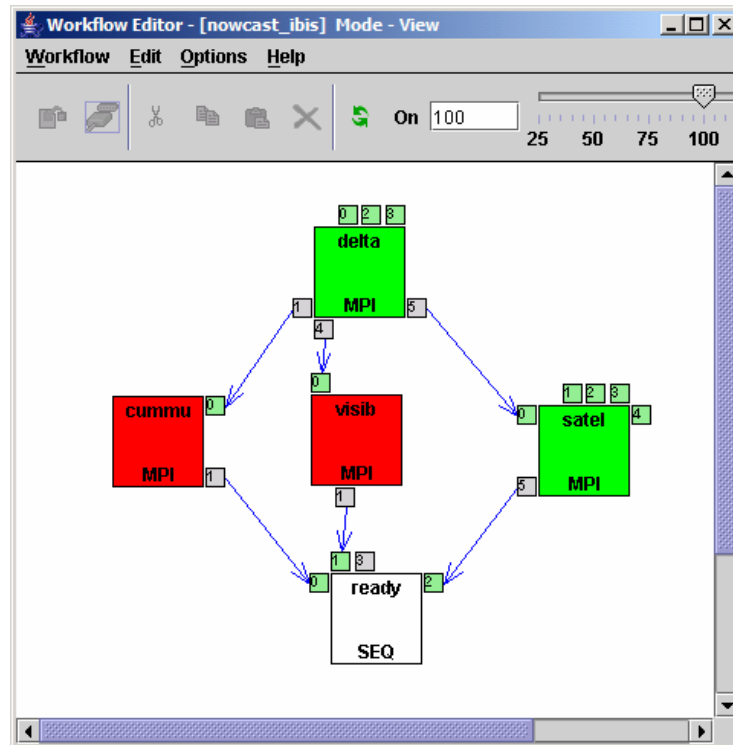


Figure 1. The Nowcast grid application in the Workflow Editor of the P-GRADE Portal

Indirect input files are produced as output files by the jobs of the workflow and are used by some other jobs of the workflow as inputs. An input port that is connected to an output port by an arc represents such a file. *Direct local input files* are provided for the jobs by the end-users. Such files must be uploaded to the portal server before the workflow is submitted into the Grid for execution. This upload procedure is performed by the Editor and the Portal Server automatically. Input ports that represent indirect input files are not connected to any output ports. *Direct remote input files* are files stored at grid storage resources. During the execution of a workflow the workflow manager subsystem of the Portal transfers such files from the storage resources to the appropriate executor sites. Input ports that represent direct remote input files are not connected to output ports either. In the Nowcast example the four input ports labelled with 0 and connected to the *commu*, *visib*, *satel* and *ready* components represent four indirect input files. On the top of that, the 1 and 2 ports of the *ready* job are indirect input files, too. The other input ports represent a mixture of direct local and direct remote files. Similarly to input files, three types of output files can be connected to the jobs of a P-GRADE Portal workflow: indirect output files, direct local output files, direct remote output files.

If an output file of a job is used only as an indirect input file for one or more other jobs, then it is called *indirect output file*. Such volatile files are automatically deleted from the Grid after the termination of the consumer jobs. The final result of a workflow consists of direct output files. *Direct local output files* must be transferred from the Grid to the portal server after the workflow terminated. *Direct remote output files* must be kept in the Grid, at the pre-defined storage resources. Notice, that an output file can be indirect output and direct output at the same time. It means that it belongs to the final result of the workflow and also serves as a data bridge between jobs. Port 5 connected to the *satel* job of the Nowcast workflow is an example for such a file. It serves both as an indirect file to transfer data from the *satel* job to the *ready* job, and also defined as an indirect remote file that must be saved at a storage resource after the termination of the workflow. The semantics of a P-GRADE Portal workflow means, that a job (node) of a workflow can be executed if, and only if all of its input files are available i.e., all the jobs that produce indirect input files for the job have successfully terminated and all the direct input files required for the job are available either on the portal server or at the pre-defined storages. For example, job *satel* can be launched when its

input file 0 was produced by job *delta*, files 1 and 2 are available on the portal server, and files 3 and 4 were available at the specified grid storages. If all the necessary input files are available for a job, then the workflow manager transfers these files – together with the binary executable – to the site where the job is allocated for execution. Therefore, the workflow describes both the control-flow and the data-flow of the application. After the job has finished, the workflow manager transfers its indirect output files to the grid sites where the consumer jobs are allocated for execution, and saves the direct local and the direct remote files on the portal server and in grid storages accordingly.

2.2 The Grid Concept of the P-GRADE Portal

Grid computing is concerned with the sharing and coordinated use of resources in distributed Virtual Organizations (VO) [13]. A uniform layer, built onto the top of the shared resources can hide the low-level differences with high-level standardised protocols, realising a sufficient foundation for distributed VOs. These high-level protocols – together with the APIs and grid services that implement them – constitute the grid middleware. This grid middleware can be created in several different ways. In fact, we can witness the fast evolution of grid middleware concepts that means a distracting feature of grids for the end users. Without a high-level user interface they should re-learn the new versions of grid middleware services, concepts and commands from time to time.

One of the main goals of the P-GRADE Portal is to hide the low-level details of the various grid systems with a technology neutral, workflow-oriented graphical desktop. To achieve high portability among the different grids, the P-GRADE Portal has been built onto the Globus middleware [14], and particularly those tools of GT-2 that are generally accepted and widely used in production grids today. Globus GridFTP, GRAM, MDS and GSI have been chosen as the basic underlying toolset for the P-GRADE Portal. Because application performance monitoring is of crucial importance in production environments, the Globus-based middleware set has been extended with the Mercury Monitor service [15].

The GridFTP file transfer service is used by the workflow manager subsystem of the Portal to transfer input, output and executable files among computational resources, among computational and storage resources and

between the portal server and the computational resources (See Figure 2.). GRAM is used by the workflow manager to start the jobs of the workflows on computational resources. GRAM assumes the presence of local job managers, and because it does not restrict them, practically any computational resource that “speaks” the GRAM protocol can be utilised by the P-GRADE Portal. MDS is the information service that stores information on the different properties of grid resources. MDS is queried by an optional element of the P-GRADE Portal, by the information system portlet. GSI is the security architecture that guarantees authentication, authorization and message-level security for GridFTP, GRAM and MDS sites. Just like MDS, the Mercury Monitor service is also an optional element of the P-GRADE Portal middleware set. Job can be executed in both Mercury-enabled and standard GT-2 grids with the P-GRADE Portal too.

The choice of this infrastructure has been justified by connecting the P-GRADE Portal to several GT-2 based Grid systems like the UK National Grid Service [16] or different VOs of the EGEE infrastructure (SEE-Grid [3] and HunGrid [17]). Notice, that most of these Grid systems use some extended versions of the GT-2 middleware. The point is that if the compulsory GRAM, GridFTP and GSI middleware set is available in a VO, then the P-GRADE Portal can immediately be used on that particular grid.

3. Collaborative Grid Applications in the P-GRADE Portal

The Collaborative P-GRADE Portal enables the concurrent engineering of grid workflows. A concurrently editable workflow belongs to a well-defined group of portal users, to a *collaborative group*.

During the concurrent development process the members of the collaborative group can define the structure of the graph, the content of the different nodes and the ports that are connected to them. The different team members can contribute to the development process simultaneously. The Portal Server and the Workflow Editor guarantee consistency, data synchronisation and protection against lost updates. For a detailed overview on the concurrent workflow development concept of the P-GRADE Portal please refer to [18].

The result of the concurrent development process is a collaborative workflow. Such a collaborative workflow contains additional information on the top of the regular graph

definition. This extra information specifies which team members defined the different components of the workflow. Moreover, since most of the components (for example a job) are described with several different parameters, this meta-information stores the relation of developers and component attributes. Table 1 demonstrates the concept by presenting a part of the meta-information that belongs to the collaborative version of the Nowcast workflow.

Table 1. Meta-information associated with the collaborative version of the Nowcast workflow. The information expresses the different user contributions.

| <i>Workflow component</i> | <i>Component attribute</i> | <i>Defined by (user ID)</i> |
|--|----------------------------|-----------------------------|
| Job "satel" | Job type (MPI) | 6 |
| | Binary executable file | 6 |
| | Command line parameters | 6 |
| | GRAM site | 8 |
| | ... | ... |
| Port "3" of job "satel" (Type: direct remote input) | Storage resource reference | 9 |
| Port "4" of job "satel" (Type: direct remote input) | Storage resource reference | 9 |
| Port "5" of job "satel" (Type: direct remote output) | Storage resource reference | 5 |
| ... | ... | ... |

The meta-information in Table 1 regards to the *satel* job and three of its ports. The type of the *satel* job, the binary executable for the job and the command line parameters of the executable file have been defined by the same user (user 6). However, it was a different group member (user 8) who has defined the GRAM resource that must be used for the execution of the job. Moreover, the location of the two direct remote input files of the job (represented by ports 3 and 4 in Figure 1) have been specified by a third group member (users 9). Finally, the last line of Table 1 shows that the storage location where the result of the job must be saved has been specified by a fourth member (user 5).

This example clearly points out that during the development phase the knowledge of multiple persons has been integrated into the workflow. Probably none of these persons were fully aware of how to define the whole application, but collaboratively they were capable for that.

One of them had an executable file. Another user knew which is the best resource to execute the job. A third person knew where the input files are located. A fourth was aware of the storage where the result must be placed.

Despite such meta-information could be associated with any kind of grid applications, currently there are no other grid tools than the P-GRADE Portal that do that.

For a different type of grid application (e.g. a service workflow) the table should contain different rows, but the idea remains the same: associate a registry with the application that stores the different collaborators' contributions. During the execution phase this registry can serve as the basis to distribute responsibilities.

4. Executing Single-Grid Collaborative Workflows

The P-GRADE Portal can be configured in two different ways. It can be connected to a single grid or it can be connected to multiple grids [HPCC/honlap]. In the single-grid mode it provides workflow execution facility for a single Globus VO. It means that every portal user and every storage and computational resource referred by the users' workflows must be the member of the same Globus VO. The aim of a collaborative group in this case is to put those people of the VO into a team that must develop and execute a grid application work together in order to deal with some issue they are all concerned of.

Imagine for example the following scenario. There are 3 persons: one software engineer and two researchers. They are members of the same VO. They want to execute a program possessed by the software engineer with the inputs of the first researcher in order to produce results for the second researcher. They want to use VO resources for the execution, but none of them wants to give access to his/her programs and data with the others.

Using the standard Globus VO management mechanisms this scenario cannot be fulfilled. Because different grid identities are mapped onto different local identities by the VO resources, the software engineer (or a client acting on his behalf) cannot access the inputs located in the storage space of the first researcher [19]. At the same time the researcher cannot specify inputs to the program stored under the software engineer's account. The problem is similar with the output files too: the second researcher is not allowed to access the result files stored under the software engineer's account, and the software engineer is not

allowed to write the files into the storage space of the researcher. The mapping of multiple grid identities onto the same local account does not provide help, since in that case every private data and program would be shared among the users.

The solution is the introduction of a trusted collaborative layer on the top of the VO. If the three persons delegate their identities to this layer, then these identities can be applied by the layer in a sophisticated manner to transfer the input data to the executor site, to start up the program, and finally to move the results into the second storage. The Collaborative P-GRADE Portal is one implementation of this layer.

In Globus environments user identities can be delegated by proxy credentials [19]. Current portals perform every step of an application execution process using the same proxy, the same user's identity. The Collaborative P-GRADE Portal can perform the different elementary steps of a workflow execution process with different proxies. By using the identity of different users during the different elementary steps, the responsibility of the execution process is shared among these individuals.

The execution of a P-GRADE Portal workflow consists of two kinds of elementary operations: file transfer and job submission. To estimate whose proxy to use to perform an elementary operation, the workflow manager system evaluates the meta-information that has been associated with a collaborative workflow during the development phase (see Table 1.) The following policy is applied during the evaluation: for job submission the proxy that belongs to the user who defined computational resource for the job is to be used. According to this policy the proxy of user 8 is used to execute the "satel" job of the Nowcast workflow.

File transfer operations require more sophisticated proxy selection method. First, the workflow manager must estimate which user's proxy must be used to access the source file. As it can be seen in Figure 2, five different file transfer scenarios are possible during a workflow execution process. 1 - A direct remote input file must be transferred from a storage resource to a computational resource. 2 - A direct remote output file must be transferred from a computational resource to a storage resource. 3 - An indirect file must be transferred from a computational resource to another computational resource. 4 - A direct local input file must be transferred from the Portal Server to a computational resource. 5 - A direct local output file must be transferred from a

computational resource to the Portal Server. The workflow manager supposes that the person who defined the source file for a transfer has the right of access.

The next step is to estimate whose proxy is needed to access the target location. If the target is a storage resource or the Portal Server, then the workflow manager has to use the proxy of the user who defined the location. If the target is a computational resource then the workflow manager has to use the same proxy than the one, which will be used to execute the consumer job on that resource. (Otherwise the input file and the job will be under different user accounts.)

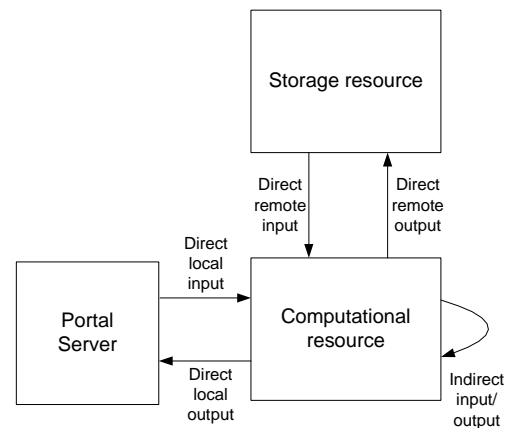


Figure 2. The elementary file transfer operations that can be contained by a P-GRADE Portal workflow execution process.

If the outcome of the evaluation process is that both the source and target locations can be accessed with the same proxy, then the file is transferred with GridFTP by the workflow manager directly. If different proxies are required at the source and the target sites, then the workflow manager transfers the file through the Portal Server indirectly. During this indirect transfer the workflow manager uses the first proxy to copy the file from the source location into a temporary storage on the Portal Server, then it uses the second proxy to copy the file from the Portal Server to the target resource. The temporary storage on the Portal Server can be accessed exclusively by the workflow manager with any of the collaborative group members' proxies.

The proxy selection policy applied by a collaborative environment actually specifies a function that assigns a user identity to every elementary execution operation. Two different environments may use two different functions, thus they may use two different proxies during the same operation. This actually means that

