Interactive In-Job Workflows*

Branislav Šimo, Ondrej Habala, Emil Gatial, and Ladislav Hluchý

Institute of Informatics, Slovak Academy of Sciences,
Dúbravská cesta 9, 845 07 Bratislava, Slovakia
branislav.simo@savba.sk

Abstract. This paper describes a new approach to interactive workflow management in the grid. By modification of existing system for management of applications composed of web and grid services an interactive workflow management system has been created, which allows users to manage complex jobs, composed of several program executions, interactively. The system uses an interactivity functionality provided by the Interactive European Grid project to forward commands from a GUI to a workflow manager running inside of a grid job. The tool is able to visualize the inner workflow of the application and the user has complete in-execution control over the job, can see its partial results, and can even alter it while it is running. This allows not only to accommodate the job workflow to the data it produces, extend or shorten it, but also to interactively debug and tune the job.

1 Introduction

The focus of current grid infrastructures like the EGEE [8] and middlewares like the gLite [6] is targeted on batch processing of computing intensive jobs, usually sequential ones. While this model is very good for e.g. parameter study applications, where the execution time of a single instance is not that important as the time required to process the whole set of jobs, there is a lot of applications where the minimization of the run rime of a single instance is important. One of the ways to achieve that goal is to parallelize the computation into cooperating processes using for example the MPI [7] messaging protocol as a means for data exchange.

The other feature lacking in currently prevalent grid infrastructures is the ability to interact with an application running in the grid. This fact stems from the focus on the high throughput aspect of the whole grid architecture. After having the high throughput grids established and deployed on the production level, it is time to support the additional types of applications.

The development in the Interactive European Grid (int.eu.grid) project [3] is focused on implementing these two missing features, intra– and inter–cluster MPI support, and interactive applications. The tools providing this functionality are discussed in the next section.

* This work was supported by projects int.eu.grid EU 6FP RI-031857, VEGA No. 2/6103/6, and INTAP (RPEU-0029-06).

Section 3 describes an interactive workflow management system of the flood forecasting application. The system was developed as a modification of system developed previously in the project K-Wf Grid [9] as a management tool for application composed of web and grid services. It allows users to manage more complex jobs, composed of several program executions, in an interactive and comfortable manner. The system uses the interactive channel of the project to forward commands from a GUI to the on-site workflow manager and to control the job during execution. While the system is used to interactively run the workflow of the flood forecasting application, it is also suitable for other applications, where the user may want to adapt their workflow execution during runtime, according to partial results or other conditions.

Section 4 shows a real use case of the system – the flood forecasting application.

2 Tools for Interactivity and MPI

In order to use the interactivity and MPI [7], the application had to be integrated with or adapted to several components of the int.eu.grid project. The application executable had to be modified to use the MPI calls and linked to MPI library. On the client side, an application specific visualization plug-in had to be created to provide customized user interface for the application in the Migrating Desktop (MD) [4] rich client framework. The MD provides an application programming interface (API) to the developer for direct connection to the application. The conceptual schema is shown on the Fig. 1. Below we give further description of these components.

The user interface client—Migrating Desktop—is a rich client framework and graphical user interface (GUI) that hides the complexities of the grid from the user. It provides basic functionality necessary for working with grid: single sign-on using user's certificate, data management (transfer of data files from a workstation to the grid and back, registration of files to the virtual directory), job management (job submission, monitoring), visualization of job results. The MD is implemented in the Java language and runs as a client on the user's machine. It is based on the Eclipse OSGi framework [10] plug-in architecture, thus allowing customization of its functionality. Plug-ins play an important role in the application support by providing application specific functionality. Input plug-ins provide custom input parameter specification, visualization plug-ins provide visualization of the application outputs and the user interface for interactive application control. A plug-in is provided in the form of a Java archive called *bundle*. It is loaded to the MD automatically upon startup, after registration into the central registry. In order to be able to control the interactive application, writing a visualization plug-in is usually necessary.

The user–application connection is realized by setting up an interactive data channel between the application running in the grid and the client plug-in. It is a data tunnel that can transfer raw binary data that are to be interpreted by the application. The channel passes all the data from the standard output of the application to the plug-in and data sent to the channel are passed to the standard

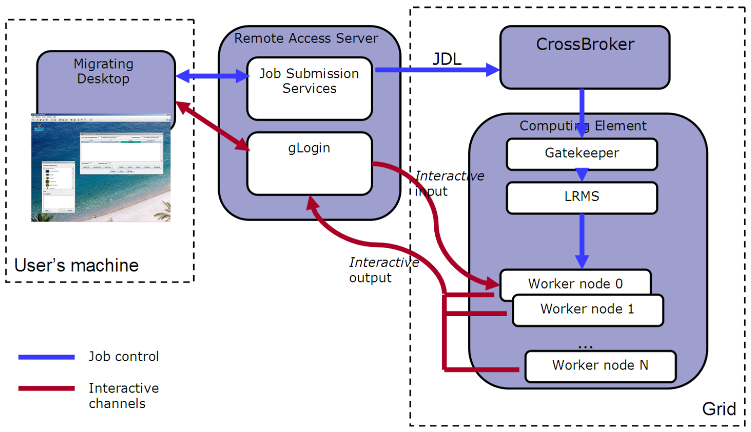

Fig. 1. Interactive channel connecting Migrating Desktop with application running in the grid

input of the application. In case of the MPI application, standard outputs of all MPI processes are merged and sent as one output stream into the channel. The standard input is available only to the master process of the application, which must then distribute any information to other processes if necessary (see Fig. 1).

Starting jobs from the MD allows the user to request a setup of the interactive channel, which is then connected to the application plug-in. The interactive channel is then set up transparently to the application. It connects the standard input, standard output and error output of the application with the MD, where they are available as separate data streams in the visualization plug-in of the application. The application plug-in has to explicitly support interactivity and support it in an application specific way, so it usually has to be implemented for each application from scratch.

Because the MD must be firewall friendly and it cannot be expected that it will have direct connection from the outside, a proxy machine is used to pass the communication from the grid to MD. The proxy machine is called Remote Access Server (RAS) and is used also for other tasks that might require traversing firewalls, e.g. file transfers. The channel between RAS and grid nodes is created using the glogin [5] tool. It uses a special setup procedure and certificate–based authentication to create a SSH tunnel. Channel between RAS and MD is currently implemented as a simple HTTP polling.

3 Interactive Workflow Management

The execution of a workflow in the grid environment usually means automatic execution of its tasks by some kind of a workflow engine. From the user's point of view the whole workflow is processed as one big job and the user can at most monitor the execution of single tasks of the workflow. In this section we describe a dynamic grid workflow execution and management system, which allows interactive monitoring and changing of a workflow running in the grid.

The difference between classical grid workflows and the one described here is that our workflow is submitted to the grid (i.e. to a resource broker [16] managing job submissions for that particular grid) as one job that will be started on one of the grid resources and then all the tasks of the workflow are executed internally as part of that workflow job. The workflow job is connected to the user interface via the interactive channel that allows the user to monitor and change the workflow and its properties. The advantage of executing the workflow in this manner is fast startup of the workflow tasks as they do not have to go through the grid resource broker.

The tool is suitable for applications, where the user may adapt their execution during runtime, according to partial results. If the need arises, another analysis may be added to process any interesting partial results that were computed. Or, if a simulation provides uninteresting data, the rest of the workflow subtree may be cancelled, and resources shifted to other parts of the job. Any application, which currently uses a shell script calling several components (binary modules or other scripts) may be easily converted to a visually controlled workflow. The workflow can then be saved, exported to an XML file, and later reused – such reuse is very simple even for non-experts.

In Section 3.1 we describe the original implementation of the workflow execution engine that was implemented in the KWf-Grid [9,2] project and then in Section 3.2 the re-implementation of the workflow engine to the grid environment of the int.eu.grid project.

3.1 Interactive Workflow Using K-Wf Grid Middleware

The main component of the Grid Application Control module, and the core of the architecture of K-Wf Grid (see Fig. 3.1) is the Grid Workflow Execution Service (GWES) [11,12]. This component is a web service, whose main function is to analyze, process, manage, and execute workflows described in a workflow description language based on Petri nets and called Grid Workflow Description Language (GWorkflowDL) [13].

The GWorkflowDL is a dialect of XML, designed specifically for controlling workflows of services, programs, grid jobs, or data transfer operations using the semantics of Petri nets. While the most widely used abstraction for workflows today is the Direct Acyclic Graph (DAG), Petri nets provide theory which is at least comparable to the theory supporting DAG operations, and enable to describe wider range of constructs, including cycles and conditional branches. Moreover, in Petri nets the data is an integral part of the whole construction (represented by so-called ”tokens”), and so the GWorkflowDL document at any stage describes the whole state of the system, which is very useful for repeating experiments and doing parameter studies. It is possible to let the workflow execute to a certain stage, then take a snapshot of its current structure into a file, and then try several executions with different parameters by simply modifying

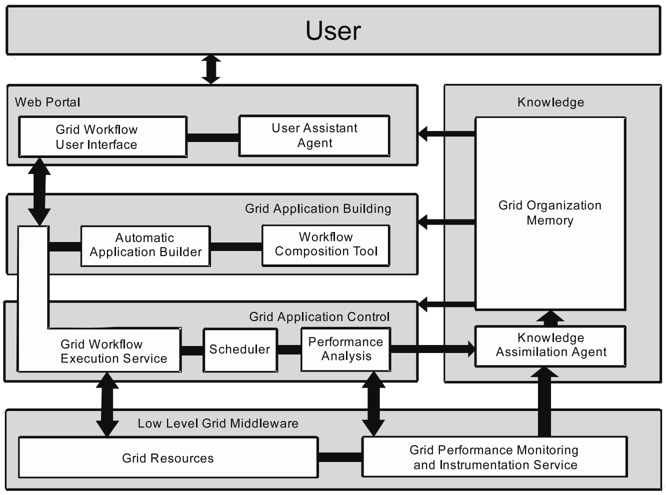
User
Web Portal
Grid Workflow User Interface
User Assistant Agent
Knowledge
Grid Application Building
Automatic Application Builder
Workflow Composition Tool
Grid Organization Memory
Grid Application Control
Grid Workflow Execution Service
Scheduler
Performance Analysis
Knowledge Assimilation Agent
Low Level Grid Middleware
Grid Resources
Grid Performance Monitoring and Instrumentation Service

Fig. 2. Architecture of components used in the K-Wf Grid project

the snapshot GWorkflowDL file. The GWES engine in K-Wf Grid is implemented as a web service, with operations that allow to

- Initiate a workflow
- Start a previously initiated workflow
- Suspend a running workflow
- Resume a suspended workflow
- Abort a running workflow (similar to suspending, but the workflow cannot be resumed)
- Restart a finished workflow
- Set and get user-readable workflow description
- Query the unique workflow identifier or its status
- Store the workflow to a preconfigured XML database
- Retrieve a stored workflow from the database
- Query any data token in a workflow
- Get or set some specific properties of a workflow.

A more detailed description of all capabilities of GWES, as well as a complete state transition diagram for GWorkflowDL-described workflow can be found in [14].

The GWES is supported by several other services and tools. In project K-Wf Grid, it is mainly the Workflow Composition Tool (WCT) and Automated Application Builder (AAB). Since GWorkflowDL supports several levels of abstraction for activities in a workflow, these tools are used to concretize an abstract place. WCT is responsible for finding an appropriate service class (non-grounded service interface description), or several service classes, for an abstract activity. AAB then finds all grounded services, which do expose the interface selected by

WCT. From these, one is picked at runtime by the scheduler (scheduling algorithms may be selected by users). These components are an integral part of the semantic support facility of the workflow construction and execution process, and they use information present in the knowledge base of the infrastructure.

Another tool supporting GWES is the Grid Workflow User Interface (GWUI). GWUI is a graphical front-end for GWES, able to visualize a workflow handled by GWES. Using GWUI, user may monitor a workflow, and perform basic interaction with it—execute it, pause, abort, query and modify data tokens in places of the Petri net. A sample of the visualization can be seen in Fig. 3.

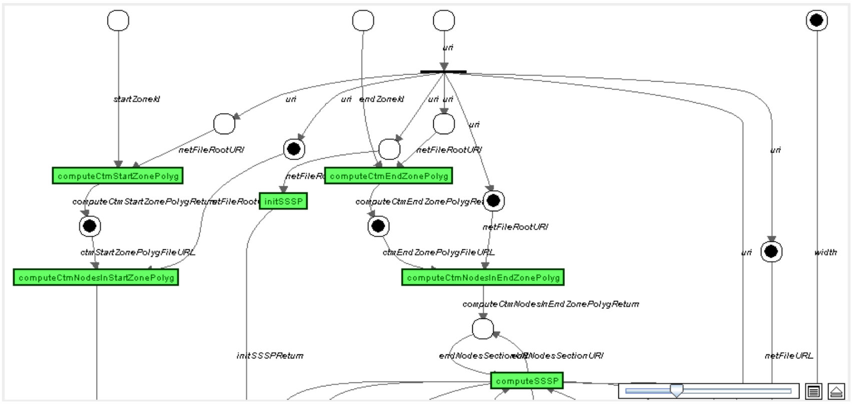

Fig. 3. A screenshot of a sample workflow visualized in GWUI

3.2 Interactive Workflows with GWES in int.eu.grid

In the project int.eu.grid, the infrastructure supporting GWES, as well as GWES itself is modified to fit into the common grid infrastructure based on the LCG [15] and the gLite [6] grid middlewares.

Since the int.eu.grid applications are not based on SOA architecture, but on more common grid jobs, GWES has been modified to be part of the core of an executable module, which is then executed as a grid job in the project's infrastructure. This job is then interactively managed by the user via GWUI embedded into the Migrating Desktop (MD) interface.

The GWES was converted into a stand-alone Java application, executable from command line. When the job starts, first executed application is GWES, with a parameter pointing to a GWorkflowDL description of the workflow to execute. Instead of a web service interface, GWES communicates through its standard input and output, which are connected to the interactive channel of int.eu.grid. At the other end of this channel is the GWUI, working as visualization plug-in in the MD. It was also modified to communicate through the interactive channel facilities of MD instead of accessing a remote web service.

The general capabilities of the GWES remain almost the same as in the K-Wf Grid. It has been extended with another job type, so it is now able to execute local programs, which are referenced by activities in the GWorkflowDL Petri net. The GWUI has received the ability to modify workflows by adding, removing, and reconnecting activities and places. The possibility to edit the data has also remained.

The WCT and AAB components are no longer present in this setup, since the workflow is not constructed from start automatically. Also, the scheduler has been replaced by a simpler module, which is able to allocate nodes to the executed activities. This is now internal part of the GWES.

The workflow job is started from MD as a special MPI interactive job. The number of nodes requested for the job must be equal or greater to the number of nodes required by any single task of the workflow, otherwise the workflow would fail. The allocation of nodes inside the workflow job is performed according to parameters set by user in the GWorkflowDL document. If there are several activities ready to fire (execute), those which cannot receive enough computational nodes wait until other activities finish and vacate their allocated nodes. If GWES during execution encounters activity, whose demands for nodes exceeds the total number of nodes allocated to the interactive job, it signals a fault to the user, and aborts the workflow.

4 Flood Forecasting Application

The flood forecasting application itself started its life in the EU project ANFAS [17]. In the beginning, it was an HPC experiment, using a hydraulic simulation model only to predict water flow in an area hit by a river flood. After ANFAS, the application has been significantly extended during the CROSSGRID [18,1] project, to contain a whole cascade of simulation models, and to use the Globus Toolkit [21], then in version 2. Since floods usually occur as a result of specific weather conditions, marked mainly by period of heavy precipitation, the simulations begin with weather prediction. From this prediction, a hydrological model computes runoff into the riverbed, and from thus prediction river level, a hydraulic simulation can predict actual flooding of the target area. With the development of the grid and incorporation of the service-oriented architecture paradigm [22], the application has also changed. In the project MEDIgRID [19], it was extended with more simulation models and visualization tools, and deployed as a set of loosely coupled WSRF [20] services, using Globus Toolkit [21] version 4.

The new architecture of what was previously called a *simulation cascade* [18] can be seen on Fig. 4). It is a set of loosely coupled models, with several possible execution scenarios. Figure 4 contains several entities, each of them having its role in our application. At the top of the figure is depicted our main data provider, the Slovak Hydrometeorological Institute (SHMI). SHMI provides us with input data for the first stage of our application, the Meteorology. The meteorological forecast is computed by the MM5 model, which operates in three

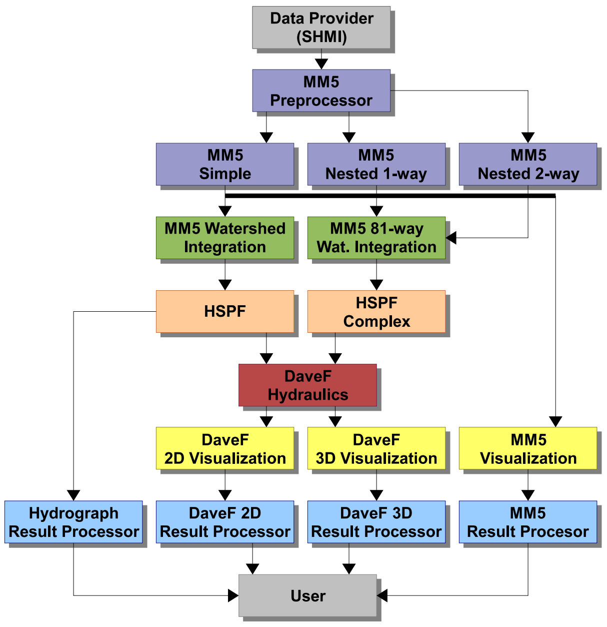

Fig. 4. Architecture of the flood forecasting application

distinct operation modes (simple, one-way nested and two-way nested). This is the forecasting step of the whole application. The predicted weather conditions are used in the Watershed integration stage to compute water runoff into the target river. This result is then further processed in the Hydrology stage, where two models - HSPF and NLC compute river levels for selected geographical points. These levels are then used to model water flow in the last, Hydraulic stage of the application. All important results are visualized and displayed to the user - if he/she requires it.

In the current implementation, the interactive workflow management system described in previous chapter is used to manage the workflow of this application inside of a job submitted to the grid.

5 Conclusions

The interactive workflow management developed for the flood forecasting application gives the user a new type of steering capabilities in terms of dynamic workflow restructuring. The application components running inside this system have no startup penalty compared to regular grid jobs, what is an advantage for workflows of short lived jobs. Because the system can be used for any other application consisting of interconnected components, we expect it to be adopted by other applications in the future.

References

1. Hluchý, L., Habala, O., Tran, V., Gatial, E., Mališka, M., Šimo, B., Slížik, P.: Collaborative Environment for Grid-based Flood Prediction. Computing and Informatics 24(1), 87–108 (2005)
2. Babík, M., Habala, O., Hluchý, L., Laclavík, M.: Semantic Services Grid in Flood-Forecasting Simulations. Computing and Informatics 26(4), 447–464 (2007)
3. Interactive European Grid project (Accessed January 2008), `http://www.interactive-grid.eu`
4. Kupczyk, M., Lichwala, R., Meyer, N., Palak, B., Plociennik, M., Wolniewicz, P.: Applications on demand as the exploitation of the Migrating Desktop. Future Generation Computer Systems 21(1), 37–44 (2005)
5. Rosmanith, H., Volkert, J.: glogin - Interactive Connectivity for the Grid. In: Juhasz, Z., Kacsuk, P., Kranzlmüller, D. (eds.) Distributed and Parallel Systems - Cluster and Grid Computing, pp. 3–11. Kluwer Academic Publishers, Budapest, Hungary (2004)
6. gLite - Next generation middleware for grid computing (Accessed January 2008), `http://glite.web.cern.ch/glite`
7. Message Passing Interface Forum (Accessed January 2008), `http://www.mpi-forum.org`
8. EGEE (Enabling grids for e-science) project (Accessed January 2008), `http://www.eu-egee.org`
9. Bubak, M., Fahringer, T., Hluchy, L., Hoheisel, A., Kitowski, J., Unger, S., Viano, G., Votis, K.: K-WfGrid Consortium: K-Wf Grid - Knowledge based Workflow system for Grid Applications. In: Proceedings of the Cracow Grid Workshop 2004, Poland, p. 39. Academic Computer Centre CYFRONET AGH (2005) ISBN 83-915141-4-5
10. Equinox - an OSGi framework implementation (Accessed January 2008), `http://www.eclipse.org/equinox`
11. Hoheisel, A., Ernst, T., Der, U.: A Framework for Loosely Coupled Applications on Grid Environments. In: Cunha, J.C., Rana, O.F. (eds.) Grid Computing: Software Environments and Tools (2006) ISBN: 1-85233-998-5
12. Hoheisel, A.: User Tools and Languages for Graph-based Grid Workflows. In: Special Issue of Concurrency and Computation: Practice and Experience. Wiley, Chichester (2005)
13. Pohl, H.W.: Grid Workflow Description Language Developer Manual. K-Wf Grid manual (2006) (Accessed January 2008), `http://www.gridworkflow.org/kwfgrid/gworkflowdl/docs/KWF-WP2-FIR-v0.2-GWorkflowDLDeveloperManual.pdf`
14. Hoheisel, A., Linden, T.: Grid Workflow Execution Service - User Manual. K-Wf Grid (2006) (Accessed January 2008), `http://www.gridworkflow.org/kwfgrid/gwes/docs/KWF-WP2-D2-FIRST-GWESUserManual.pdf`
15. LCG - LHC Computing Grid project (Accessed January 2008), `http://lcg.web.cern.ch/LCG`
16. Fernández, E., Heymann, E., Senar, M.A.: Resource Management for Interactive Jobs in a Grid Environment. In: Proc. of IEEE Int. Conf. On Cluster Computing (Cluster 2006), Barcelona, Spain, September 2006. IEEE CS Press, Los Alamitos (2006) CD-ROM edition
17. ANFAS Data Fusion for Flood Analysis and Decision Support (Accessed January 2008), `http://www.ercim.org/anfas`

18. Hluchý, L., Tran, V.D., Habala, O., Šimo, B., Gatial, E., Astaloš, J., Dobrucký, M.: Flood Forecasting in CrossGrid project. In: Dikaiakos, M.D. (ed.) AxGrids 2004. LNCS, vol. 3165, pp. 51–60. Springer, Heidelberg (2004)
19. Šimo, B., Ciglan, M., Slížik, P., Mališka, M., Dobrucký, M.: Mediterranean Grid of Multi-Risk Data and Models. In: Proc. of 1-st workshop Grid Computing for Complex Problems - GCCP 2005, VEDA, 2006, Bratislava, Slovakia, November-December 2005, pp. 129–134 (2006) ISBN 80-969202-1-9
20. Czajkowski, K., Ferguson, D.F., Foster, I., Frey, J., Graham, S., Sedukhin, I., Snelling, D., Tuecke, S., Vambenepe, W.: The WS-Resource Framework. March 5 (2004), (Accessed January 2008), `http://www.globus.org/wsrf/specs/ogsi_to_wsrf_1.0.pdf`
21. Foster, I.: Globus Toolkit Version 4: Software for Service-Oriented Systems. In: Jin, H., Reed, D., Jiang, W. (eds.) NPC 2005. LNCS, vol. 3779, pp. 2–13. Springer, Heidelberg (2005), `http://www.globus.org/toolkit`
22. Foster, I., Kesselman, C., Nick, J.M., Tuecke, S.: The Physiology of the Grid (Accessed January 2008), `http://www.globus.org/alliance/publications/papers/ogsa.pdf`