CrossGrid Integrated Workflow Management System

Martin Maliska, Branislav Simo, and Ladislav Hluchy

Institute of Informatics, Slovak Academy of Science
{martin.maliska, branislav.simo, hluchy.ui}@savba.sk

Abstract. This paper describes a workflow management system that was implemented according to the requirements of a Flood forecasting application [1] developed in the CrossGrid project [2]. The flood forecasting application contains several simulation models (meteorological, hydrological, hydraulic) implemented as the grid jobs. A need for a cascade execution of these models has been a motivation for creating workflow management system capable of executing of a cascade of simulations in the CrossGrid testbed. First implementation of the workflow system has been tied to the portal user interface, but currently has been decoupled as a standalone grid service, which can be used by the two user interfaces developed in the project.

1 Introduction

During the development of the Flood forecasting application[1] in the CrossGrid[2] project we have faced the problem of running a cascade of simulations comprising the application in the project testbed in an automatic way, so that the user will not have to submit each job separately and make the coupling of the input and output files by hand.

The application focuses on the prediction of floods, which are considered as serious problem, causing a lot of damages with many people threatened or even killed. Among the methods that can help to mitigate consequences of floods is a prediction of such an event. Our prediction system consists of several simulation models – meteorological, hydrological and hydraulic, which form the base of a simulation cascade. The cascade is actually a workflow. The meteorological model computes the precipitation in the target area, which is then processed by hydrological model for run-off computation and finally the hydraulic model computes the flow of the water in the river bed and surrounding area. More detailed description can be found in [1]. We are cooperating with experts from Slovak Hydro-Meteorological Institute, which provide us with data for the models and the relevant know-how.

These simulation models are computationally intensive and when several experts will need to run different scenarios, e. g. in the time of possible crisis or parameter studies during model calibration, the computers in the grid will provide the requested processing power.

Our task is to integrate these models so they can be executed as cascade. Each model has several parameters and because every model needs to be calibrated and a calibration has to be carried out for each target area, there should be a way how to easily run specific model(s) or the whole cascade with modified parameters for the same situation (time, space). An expert could also decide to stop the execution of

P.M.A. Sloot et al. (Eds.): EGC 2005, LNCS 3470, pp. 1032 – 1039, 2005.

current cascade after he checked the results computed so far and to execute the rest of the cascade with changed parameters or to execute a completely new run of another cascade. These are the key features which affected the design of our application.

In the context of the CrossGrid project, two different user interfaces have been implemented. The first one is a portal interface based on the portlets using Velocity[3] as a template engine. The second one is plug-in for Migrating Desktop – a Java client developed in CrossGrid.

The workflow grid service and user interfaces using it as a back end service are described in the following chapters.

2 Overview of Existing Workflow Management Systems

Several projects use workflows, because they offer them high level abstraction in building the applications. Nowadays, a research is focused on using workflows for execution of grid jobs, web services and grid services.

This research has led to development of various workflow description languages (WDLs). The WDLs have different restrictions for describing a workflow and different features. The projects like Cactus[5], Condor[6] and Unicore[7] use direct acyclic graph (DAG) to describe a workflow. The Unicore has implemented several features to eliminate disadvantages of DAG – it supports conditional and loop execution. An XML based WDL like GJobDL[8], GSFL[9], WSFL[10], SCUFL[11] and their derivation TaskGraph[12] models separately control flow and data flow. Most of them – GSFL, WSFL, SCUFL support nesting of workflow definitions.

The Pegasus[13] is an example of system which produces a concrete workflow from an abstract workflow description. Complete workflow management system P-GRADE[14] based on Condor offers tools for editing, debugging and monitoring of the workflows, performance analysis, distributed checkpointing and dynamic loadbalancing. The Fraunhofer Grid[8] uses Petri-nets for visualisation of a workflow and translates it to the GjobDL and have both explicit and implicit fault management. Another workflow framework for grid application deployment is GridAnt[15] - Apache's Ant based toolkit that specifies taks for compilation, software installation, data transfer and grid applications.

3 Why Another Workflow Management System

As it has been already mentioned, we needed a workflow system that would:

- be able to process our workflows in an interactive way, so that a user could monitor its status and the results during execution,
- not be overly complicated,
- be easily integrated with the middleware developed in CrossGrid and DataGrid[16], especially the job submission service and resource broker,
- support different types of jobs – grid jobs, local jobs to be processed on the server (quick tasks not needing the grid), java tasks to be performed as part of the workflow processing,
- allow integration with the portal user interface.

After reviewing the requirements we have decided to implement the workflow engine by ourselves. The last requirement made us to implement the first version of the engine as a part of the portal. It has worked well but it turned out that such tight coupling with the portal is not scalable and the engine cannot be used by other applications. Therefore we decided to implement it as a standalone grid service using the Globus GT3 toolkit[17]. This implementation is now accessible from both the portal and the workflow plug-in for Migrating Desktop (described later).

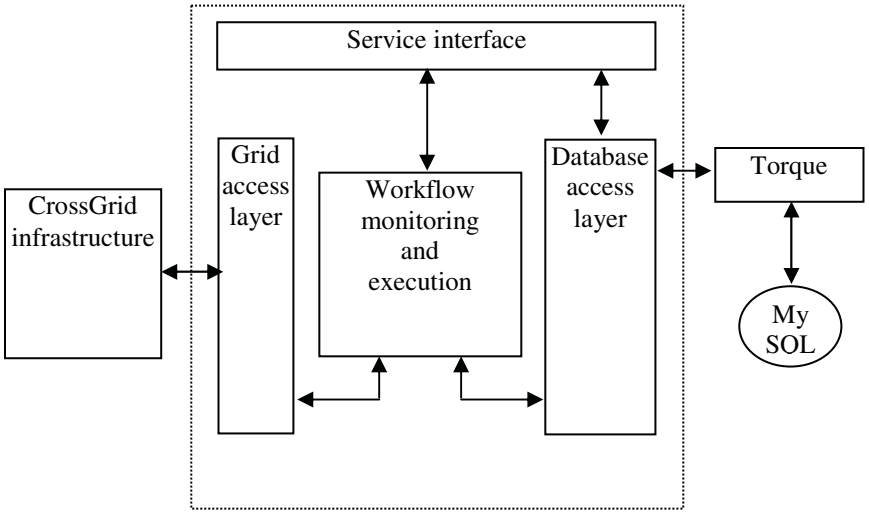

Fig. 1. Workflow service architecture

4 Implementation

The FloodGrid application consists of several simulation models implemented as the grid jobs that are coupled by input and output data. This cascade of models can be easily expressed as a set of activities with exactly defined order of execution – a workflow. A grid service was implemented to provide functionality capable of processing of the workflows as a part of the Flood forecasting application. This service has four modules: service interface, workflow execution and monitoring, grid access layer and database access layer. Figure 1 presents architecture of the service.

Instead of using a workflow description language, we decided to simply store information about the workflow templates, their content and instances to a database.

A workflow consists of several jobs (activities), it is also possible to compose the workflow from several sub-workflows. Dependencies between the jobs (activities) are described by relationships between database tables. The jobs are configurable by input and output parameters. The parameters are bounded to the resources (the files, the directories or the variables). A sharing of resource between several parameters is restricted so only one of the parameters can be an output parameter.

Every job has to have its own class inherited from class *AbstractTask,* which name is stored in database for each job. Inherited class must implement methods for determining two basic states of job – *isAborted()*, *isFinished()* and of course method *run*() that should contains implementation of what job have to do. There is one default implementation of *AbstractTask* used for executing the grid jobs – *GridTask*. This implementation uses a job attribute *param_string* to obtain name of a script file of the job that would be executed in grid.

4.1 Service Interface

The workflow service provides interface which allows these types of operations:

- create, clone, run, remove workflow instance,
- modify job parameters,
- query list of workflows and workflow instances,
- determine current job status,
- basic operations for manipulating with grid proxy certificates.

4.2 Workflow Monitoring and Execution

The workflow service contains two main threads responsible for execution and monitoring of the workflows – a *JobMonitoringThread* and an *ExecutionThread*. While there are multiple instances of the *ExecutionThread* – each running workflow has its own instance, there is only one instance of the *JobMonitoringThread*. The *JobMonitoringThread* is used to monitor state of the grid jobs, other types of jobs are not monitored by it. It has its own queue of running jobs, periodically checks a state of the all jobs from the queue and informs the appropriate *GridTask* object, which belongs to the monitored job. This job object has to decide if any change of a state occurs and whether this change is so important that the *ExecutionThread* owning the job needs to be informed about it. Map of the *ExecutionThreads* searchable by workflow ID is stored in the *JobMonitoringThread*.

An explanation about how the *ExecutionThread* works will be shown on detailed description of steps that are executed when the method *runWorkflow()* is called. At the beginning, tree of the jobs will be created, strictly speaking – tree of the objects that contain implementations of the jobs will be created. Every job object will known all its successors and predecessors and can manipulate with information about the job that belongs to this object stored in the database. Next step is to prepare the root job objects to ready-to-run state and enqueue it to the *ExecutionThread* queue of the job objects with changed state. After the notification about a queue size change, the *ExecutionThread* will wake up, if it sleeps, and process the objects from queue. A decision what to do with the job object will *ExecutionThread* made by determining the state of job. When the job is in state ready-to-run, the job will be executed. Finished job will cause that the job object will be removed from the *JobMonitoringThread* queue and a counter of unfinished jobs will be decreased. Aborted job results in cancellation of the all running jobs according to the workflow. Life cycle of *ExecutionThread* will end when the counter of unfinished jobs will drop to zero value.

4.3 Database Access Layer

All operations that manipulate with information about the workflows stored in database are encapsulated in database access layer. An interface for communication with this layer is provided by *PersistenceLayer* class. It uses object to relational database mapping tool Torque[18] instead of direct access to database. Another mapping mechanism is used for transformation of Torque database objects to objects used by the grid service interface. This mechanism is provided by *DBtoWSMapper* class.

4.4 Grid Access Layer

Grid access layer is an entry point to the CrossGrid infrastructure. A main class, *GridTools,* of this layer supplies methods for submitting a job for execution, getting job status and canceling execution of the job. All these operations are covered by accessing a CrossGrid job submission service. The job submission service uses a job description language (JDL) to specify a grid related information for submitting the job, for example: arguments, input and output sandbox, job type, standard output and input.

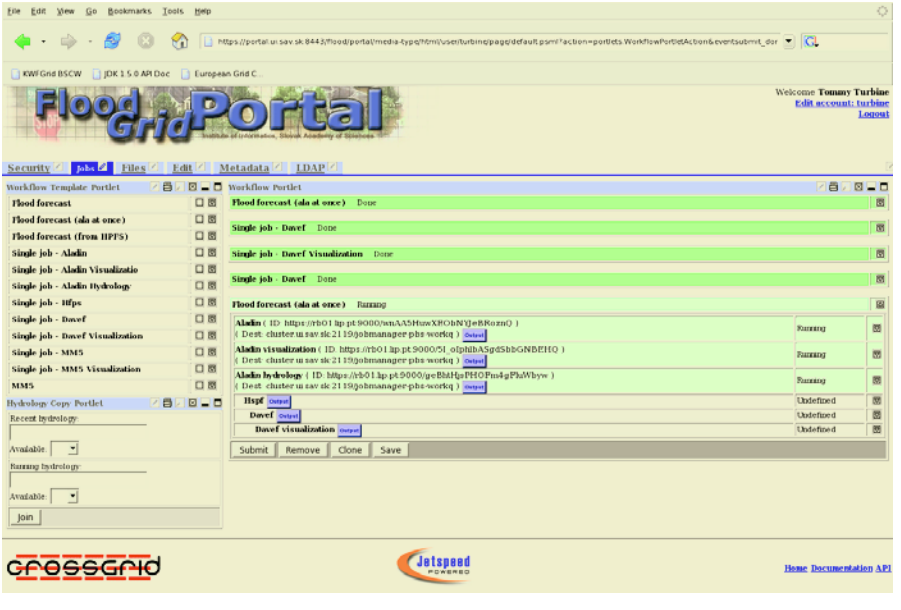

Fig. 2. Screenshot of a portal user interface

5 User Interfaces

During implementation of CrossGrid tools and services, two user interfaces - Portal and Migrating Desktop were developed. According to a need of integration with the

CrossGrid tools, the FloodGrid application was designed to be easily developed with support of both user interfaces. Workflow grid service provides unified interface that these user interfaces uses to manipulate with workflows.

5.1 FloodGrid Application Portal

Portal user interface has been built upon the Jetspeed[4] portal by using Velocity[3] template portlets. The flood application portal contains a portlet for manipulating with proxy certificate, workflow template portlet, workflow portlet, visualization portlet and metadata portlet. Figure 2 captures FloodGrid portal during a work with workflow portlet.

Workflow template portlet displays a list of workflow templates and allows user to select one which a workflow instance will be created from. Created instance and other workflow instances created by user can be monitored by workflow portlet.

This portlet can be used to explore the jobs belonging to a workflow instance and their parameters could be modified. It is possible to view results produced by each job by browsing output directories of the jobs. This functionality is covered by visualization portlet. To reflect a need of application for describing results, metadata support was implemented and its user interface provides metadata portlet, which support this operations: creation, modification, querying.

5.2 FloodGrid Plugin for Migrating Desktop

Migrating Desktop is a java based user interface developed in CrossGrid which emulates desktop environment for grid users. Every application that wants to be integrated to Migrating Desktop has to be implemented as migrating desktop plugin. Functionality of FloodGrid plugin is similar to the FloodGrid portal. Our plugin consists of these panels: workflow template panel, workflow panel, jobs panel, parameters panel and info panel. Workflow template panel and workflow panel operate in similar way as appropriate portlets on the portal except that content of a workflow is shown in a separate panel. Information panel shows relevant information about currently selected object (workflow template, workflow instance, job). Metadata operations are placed on separate tab on tabbed pane.

6 Future Work

We want to focus our effort on making stable version of portal working inside of OGCE[19] and to customize this environment so it will express the needs of expert or common users that will use our application.

Another big task is to improve fault tolerance and upgrade error reporting to the higher level. Cooperation between replica management provided by CrossGrid tools and our metadata management system has to be done so metadata system can work properly.

Although we did not expect a lot of new workflow templates and jobs in our application, it should be nice to have a workflow creation tool. We will decide to adopt an existing one or to develop new one that will fit our needs.

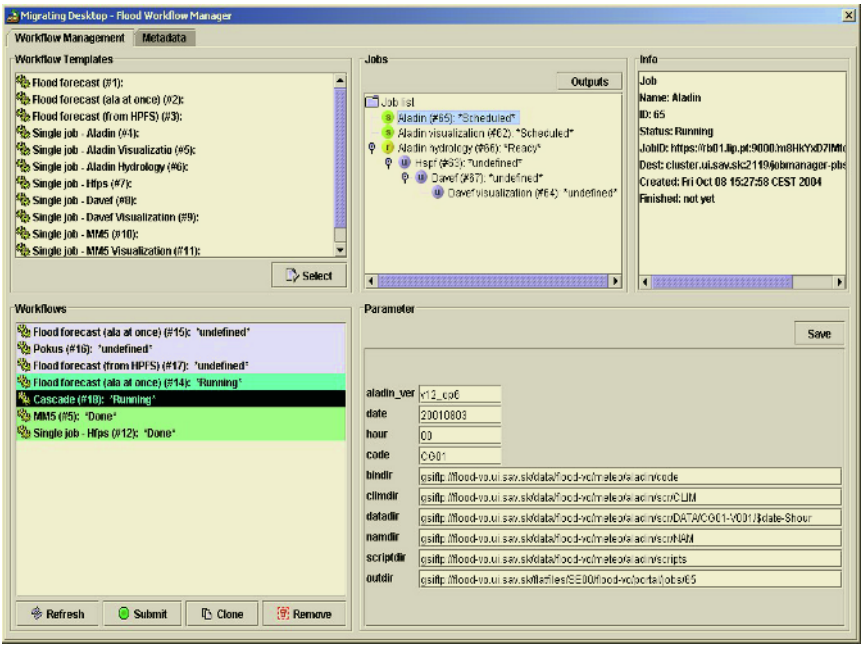

Fig. 3. Migrating Desktop plugin interface

References

[1] Hluchy L., Astalos J., Dobrucky M., Habala O., Simo B., Tran V.D.: Flood Forecasting in a Grid Environment. In: Proc. of 5-th Intl. Conf. on Parallel Processing and Applied Mathematics PPAM'2003, R.Wyrzykowski et.al. eds., 2004, LNCS 3019, Springer-Verlag, pp. 831-839, ISSN 0302-9743, ISBN 3-540-21946-3. September 2003, Czestochowa, Poland.

[2] CrossGrid - Development of Grid Environment for Interactive Applications. IST-2001-32243. http://www.crossgrid.org (visited November, 2004)

[3] Velocity Template Engine, http://jakarta.apache.org/velocity/, (visited November, 2004)

[4] Jetspeed portal framework, http://portals.apache.org/jetspeed-1/, (visited November, 2004)

[5] Cactus project, www.cactuscode.org, (visited November, 2004)

[6] The Condor project, www.cs.wisc.edu/condor/, (visited November, 2004)

[7] The UNICORE project, http://unicore.sourceforge.net/ajo.html, (visited November, 2004)

[8] Hoheisel, A., Der, U.: An XML-based Framework for Loosely Coupled Applications on Grid Environments. In: P.M.A. Sloot et al. (Eds.): ICCS 2003. LNCS 2657, Springer-Verlag, pp. 245–254

[9] S.Krishnan, P.Wagstrom, G.von Laszewski GSFL: A Workflow Framework for Grid Services

[10] Technology Report: Web Services Flow Language (WSFL), http://xml.coverpages.org/wsfl.htm, (visited November, 2004)

[11] MyGrid project homepage workflow section, http://twiki.mygrid.org.uk/twiki/bin/view/Mygrid/WorkFlow#XScufl_workflow_definition,
[12] Triana project, http://www.triana.co.uk/, (visited November, 2004)
[13] Planning for Execution in Grids, http://pegasus.isi.edu/, (visited November, 2004)
[14] P_GRADE, http://www.lpds.sztaki.hu/pgrade/, (visited November, 2004)
[15] The GridAnt, http://www-unix.globus.org/cog/projects/gridant/, (visited November, 2004)
[16] The DataGrid project, http://eu-datagrid.web.cern.ch/eu-datagrid/, (visited November, 2004)
[17] The Globus Alliance, http://www.globus.org/, (visited November, 2004)
[18] Torque, http://db.apache.org/torque/, (visited November, 2004)
[19] Open Grid Computing Environment (OGCE), http://www.collab-ogce.org/nmi/index.jsp, (visited November, 2004)