Computing and Informatics, Vol. 24, 2005, 87–108

COLLABORATIVE ENVIRONMENT FOR GRID-BASED FLOOD PREDICTION

Ladislav Hluchý, Ondrej Habala, Viet Tran, Emil Gatial
Martin Mališka, Branislav Šimo, Peter Slížik

Institute of Informatics
Slovak Academy of Sciences
Dúbravská cesta 9
845 07 Bratislava, Slovakia
e-mail: {Emil.Gatial, Martin.Maliska, hluchy.ui,
Ondrej.Habala, viet.ui}@savba.sk

Manuscript received 9 December 2004

Abstract. This paper presents the design, architecture and main implementation features of the flood prediction application of the Task 1.2 of the EU IST CROSSGRID[1] project. The paper begins with the description of the virtual organization of hydrometeorological experts, users, data providers and customers supported by the application. Then the architecture of the application is described, followed by used simulation models and modules of the collaborative environment. The paper ends with vision of future development of the application.

Keywords: Collaborative Grid Environment, Grid computing, Workflow, Data management, Portal, Flood prediction, Simulation

1 INTRODUCTION

Over the past few years, floods have caused considerable damages throughout Europe. They have affected most of the European population and they resulted in heavy material losses. The need for better flood protection has become imminent.

[1] This work is supported by EU 5FP CROSSGRID IST-2001-32243 RTD project and the Slovak Scientific Grant Agency within Research Project No. 2/3132/23.

In this paper we present the Collaborative Grid Environment for Flood Forecasting, a system intended as a support tool for hydrometeorological experts. Grid computing environments (GCEs) have increasingly gained attention in the past few years. Advances in technological infrastructure as well as a better awareness of the needs of application scientists and engineers have been the primary motivating factors. In particular, the shift in emphasis from low-level application scheduling and execution [2] to high-level problem solving signals that Grid computing is becoming increasingly important as a way of doing science. A GCE is a Problem Solving Environment (PSE) [1] with specifically formed computation kernel, using the power of Grid Computing. Good examples of some GCEs can be found in [3].

The system described herein is composed of a cascade of three simulation models – meteorological, hydrological and hydraulic model. The whole cascade is able to predict water flow in a flooded area, but users may also reduce their questions to simple weather prediction or to development of river level in a certain area (a hydrograph).

The interface of this GCE is a WWW-based portal, enabling users to run simulations and evaluate results from anywhere in the world, using a simple computer with web browser and Internet connection. The front-end is a collection of web pages with options to run the desired simulations. Behind this, a sophisticated collection of data, model codes, scripts and Grid middleware is hidden. The GCE uses public-key based authentication mechanisms, enabling secure and private data transfer, processing and storage. Furthermore, the system encompasses some collaboration tools, enabling users to exchange files and to communicate with each other.

The basic data flow is going from the storage system, through a cascade of simulations, postprocessing and visualization and then the results are displayed to the user. The whole system is controlled by the central portal interface through configuration files and/or commands to existing processes. Of course the GCE is much more complicated and versatile than just a sum of the above-mentioned components, the simulations can be run standalone, without the need for the complete cascade to compute, partial results are stored into the storage system and users can control various parameters of any of its components.

This software is developed as a part of the CrossGrid project [4]. The final product will enable much more than the first software release described in the next sections. It will include complex data management, probably a collection of concurrent models for each stage of the simulation cascade, much more sophisticated and comfortable user interface with options for prediction and warning automation, while retaining scalability and ease of use.

2 VIRTUAL ORGANIZATION FOR FLOOD FORECASTING

The scheme of the virtual organization as proposed in [5] was a general one. To actually create a real product, we have had to specify real users, data providers,

storage providers, and cycle providers. In Fig. 1 we can see actual components of the prototype (grey color).

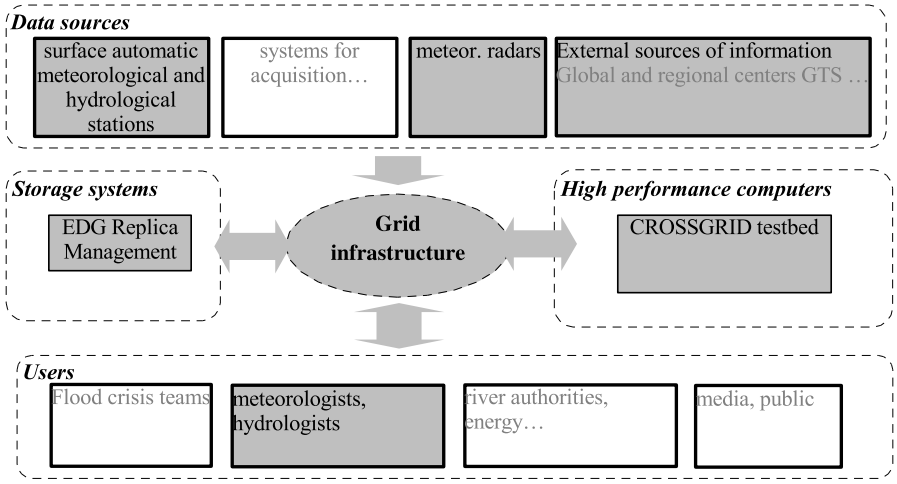

Fig. 1. Virtual organization for flood forecasting – architecture

Data providers – currently the only data provider is the Slovak Hydrometeorological Institute (SHMI). It is responsible for supplying initial boundary conditions for the meteorological simulation (these come from an external source) and for radar measurements.

Cycle providers – the system will be tested in the CrossGrid international testbed environment which will consist of 11 institutions distributed across 8 European countries. The testbed will provide applications with distributed computing and storage resources connected by high-performance network (GEANT). One of the main objectives of the CrossGrid testbed is to ensure interoperability with other grid testbeds.

Storage providers – the storage space for computation output is provided by the Institute of Informatics of the Slovak Academy of Sciences (II SAS). Temporary storage space needed to locally store input and immediate ouput data for running simulations will be provided by testbed contributors, whose processing facilities will be used.

Users – there are several groups of users in the Virtual Organization (VO) including meteorological and hydrological experts, developers and end users.

Experts – the system will be used solely by meteorological and hydrological experts from the SHMI and by trained staff from II SAS. These experts will provide configuration parameters for the simulations and they will decide which simulations need to be executed.

Developers – all development is undertaken by II SAS, who are preparing the portal, storage facilities and scripts for the simulations. They will also modify, test and deploy simulation model codes.

End users – will be, once again, experts from SHMI, who will evaluate outputs of the system. In this stage no other users will be involved.

3 CASCADE OF SIMULATIONS

The PSE is based on the cascade of three types of simulations: meteorological, hydrological and hydraulic. Its structure, basic data flows and types of the data are shown in Figure 2.

3.1 Meteorological Modelling

Forecasting of flood events requires quantitative precipitation forecasts as well as forecasting of temperature (to determine snow accumulation/melting). The simulation of atmospheric processes with the objective to predict future developments is the objective of Numerical Weather Prediction (NWP). The meteorological models are generally computationally intensive and they are usually running on the supercomputer class systems. The output of mesoscale models is used by meteorologists as a boundary condition for regional models. This nesting requires efficient management and transfers of large (tens to hundreds of megabytes) datasets. The prediction of flash floods will require the employment of high resolution storm-scale models.

Our system uses primarily the ALADIN/SLOVAKIA [6] model currently operated by SHMI. ALADIN is a LAM (Limited Area Model) developed jointly by Meteo France and cooperating countries. It can be viewed as an extension to ARPEGE, a global NWP model operated in Meteo France. The main purpose is to provide more detailed short range forecasts inside limited domain of interest. Currently the ALADIN model is operated in 13 Euro-Mediterranean countries.

The ALADIN model consists of more than 1 million lines of (mainly) Fortran 90 code. Parallel version uses MPI communication library and was successfully ported to our target platform. First tests showed that the model code has high demands on communication network and thus is not very suitable for distributed running in the Grid. The main benefit of using Grid environment could be in performing of parameter studies.

We are currently incorporating another model, the MM5 meteorological model, into the system. It will enable us to simulate weather with finer mesh (up to 2.5 km), thus reaching increased accuracy over the 7.0 km mesh of ALADIN/SLOVAKIA predictions.

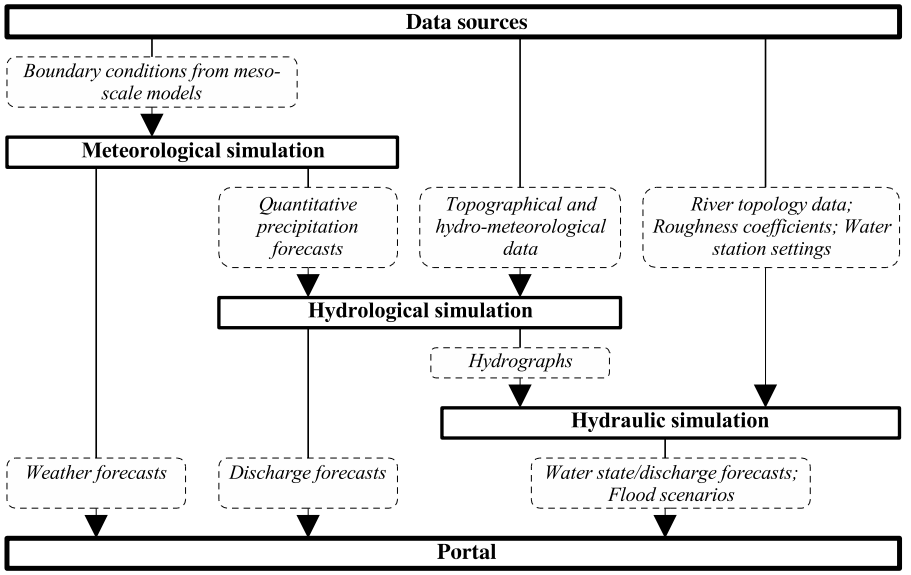

Fig. 2. Scheme of the simulation cascade of the CROSSGRID Task 1.2 application

3.2 Hydrological Modelling

We are using several hydrological simulation models, it depends on conditions and needs which will be applied model for which situation and territory; they can be also used in a combined way.

HEC-1 model is single-event-based, designed to simulate the surface runoff response of a river basin to precipitation by representing the basin as an interconnected system of hydrologic and hydraulic components [7]. The result of the modeling process is the computation of streamflow hydrographs at desired locations in the river basin.

HSPF is a program for continuous simulation of watershed hydrology and water quality for both conventional and toxic organic pollutants [8]. The HSPF model uses information such as the time history of rainfall, temperature and solar radiation; land surface characteristics such as land use patterns; and land management practices to simulate the processes that occur in a watershed. The result of this simulation is a time history of the quantity and quality of runoff from an urban or agricultural watershed.

ERM (Empirical – Regressive Model) is suited for continuous simulation; it is based on the principle of autoregressive model created by Box and Jenkins. The simplest regressive model is relational with the forecasted discharge being function of previous discharge and previous rainfalls. The function parameters are determined by regressive analysis of time series of discharges and rainfalls in

selected time interval. In case of snow, also temperature time series are taken into account and snow stage is computed.

These models are sequential programs written in FORTRAN and are not computation demanding. Thus they need not be parallelized, but are data demanding, and a lot of parameters need to be specified by model calibration. Parametric run will be used in Grid environment through portal and automatically generated scripts.

3.3 Hydraulic Modelling

One of the hydraulic models in the PSE is FESWMS (Finite Element Surface-Water Modeling System) Flo2DHi [9] which is a 2D hydrodynamic, depth averaged, free surface, finite element model. Flo2DH computes water surface elevations and flow velocities for both super- and sub-critical flow at nodal points in a finite element mesh representing a body of water (such as a river, harbor, or estuary). Effects of bed friction, wind, turbulence, and the Earth's rotation can be taken into account. In addition, dynamic flow conditions caused by inflow hydrographs, tidal cycles, and storm surges can be accurately modeled. Since Flo2DH was initially developed to analyze water flow at highway crossings, it can model flow through bridges, culverts, gated openings, and drop inlet spillways, as well as over dams, weirs, and highway embankments. Flow through bridges and culverts, and over highway embankments can be modeled as either 1D or 2D flow.

Simulation of floods is very computation-expensive. Several days of CPU-time may be needed to simulate large areas. For critical situations, e.g. when a coming flood is simulated in order to predict which areas will be threatened, and to make necessary prevention, long computation times are unacceptable. Therefore, FESWMS Flo2DH was parallelized in order to achieve better performance. In the parallel version, iterative linear solvers based on Krylov subspace methods [11] are used instead of the direct solver. These iterative solvers consist of matrix and vector operations only; thus, they offer large potential parallelism. In comparison with the direct solvers, iterative solvers are often faster and require less additional memory. The only drawback of iterative solvers is that they do not guarantee convergence to a solution. Therefore, preconditioning algorithms [10] are used to improve the convergence of iterative algorithms.

4 WORKFLOW SERVICE IMPLEMENTATION

The FloodGrid application uses the workflows for representing a cascade of simulation models. A module responsible for execution of a workflow in the FloodGrid application is a workflow service based upon a grid service standard. The workflow service is divided to four modules – interface, workflow execution and monitoring, grid access, database access. Architecture of workflow service is shown in Figure 3.

The workflow description and information about the workflow instances are stored in Mysql database.

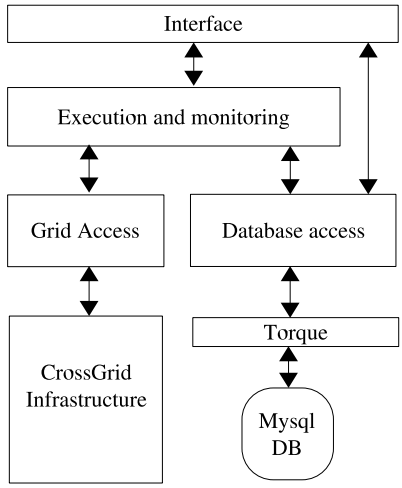

Fig. 3. Workflow service architecture

4.1 Workflow Description

Each workflow consists of several jobs (activities). Jobs can be parameterized by using output or input parameters. The parameters are bounded to the resources (resource can be a file, a directory or a variable). One resource can be shared between several parameters, but a restriction is that only one of those parameters can be an output parameter. Dependency between the jobs is simply defined as list of links between two job IDs (successor relationship). This simple definition partially eliminates restrictions of a direct acyclic graph representation of workflow, but need to be handled in job implementation to support for example conditional loops. Our definition of workflow description allows to define more than one root job, so the workflow can consist of more subworkflows. In Figure 4 a simple example of workflow is shown; below it, you can see details how the workflow description is defined in our implementation.

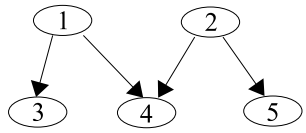

Fig. 4. Workflow example

List of root jobs: $(1, 2)$.
List of job links $(\{1, 3\}; \{1, 4\}; \{2, 4\}; \{2, 5\})$.

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

4.2 The Workflow Service Interface

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

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

4.3 Database Access

The class *PersistenceLayer* is responsible for storing all information about the workflow into the database. Instead of a direct access to database, O/R mapping tool Torque was chosen. Because Torque objects have linkage to database, they cannot be migrated. To fix this dependency on the database, a mapping mechanism is used to map Torque database object onto the our objects.

4.4 Workflow Execution and Monitoring

Workflow service contains two main threads responsible for execution and monitoring of the workflows – a *JobMonitoringThread* and a *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 the job 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 inform the appropriate *GridTask* object that belongs to the monitored job. This job object has to decide if any change of a state occurs and if this change is so relevant that the *ExecutionThread* (that owns the job) need to be informed about it. Map of the i*ExecutionThreads* searchable by workflow ID is stored in the *JobMonitoringThread*.

An explanation about how the *ExecutionThread* works will be shown on detailed description of the steps that are executed when the method *runWorkflow()* is called.

At the beginning, a tree of the jobs will be created, strictly speaking – a tree of the objects that contains implementation of the jobs will be created. Every job object will know all its successors and predecessors and can manipulate with information about the job that belongs to this object stored in the database. The next step is to prepare the root jobs objects to 'ready to run' state and enqueue them 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 me made by the *ExecutionThread* by determining the state of job. When the job is in the 'ready to run' state, the job will be executed. Finished job causes that the job object will be removed from the *JobMonitoringThread* queue and counter of unfinished jobs will be decreased. Aborted job results in cancellation of all running jobs according to the workflow. Life cycle of *ExecutionThread* will end when the counter of unfinished jobs will drop to zero value.

5 DATA MANAGEMENT

A problem solving environment of such a scale as is the flood prediction application of CROSSGRID Task 1.2 (FloodGrid) requires that its data be taken care of by a specialised software suite. This data management suite has the following roles:

- To enable delivery and storage of input data from sources outside of the PSE.
- To store all the computed (output) data, which has to be stored.
- To catalog all existing data and provide effective search facilities.
- To enable quick and straightforward access to all available data.

The data management software of CROSSGRID Task 1.2 is equipped with facilities that support all these goals. The rest of this chapter describes the facilities for transport of input data, for data cataloguing and lookup, for data storage and replication. The chapter ends with description of future development of data management software of CROSSGRID Task 1.2.

5.1 Data Management Tasks in CROSSGRID Task 1.2

General scheme of data management operations in FloodGrid is shown in Figure 5. Only the most important operations are shown there.

As we can see, the Slovak Hydrometeorological Institute (SHMI) in Bratislava provides the input data of FloodGrid – radar and satellite images, measurement stations data and input data for both ALADIN and MM5 (input data of hydrological and hydraulic stages of flood prediction are computed inside the FloodGrid). Measurement stations data is stored in a RDBMS; all other data is just flat files. The measurement stations are graphically displayed in the FloodGrid portal, and also the metadata interface of FloodGrid portal requires transfer of metadata to/from

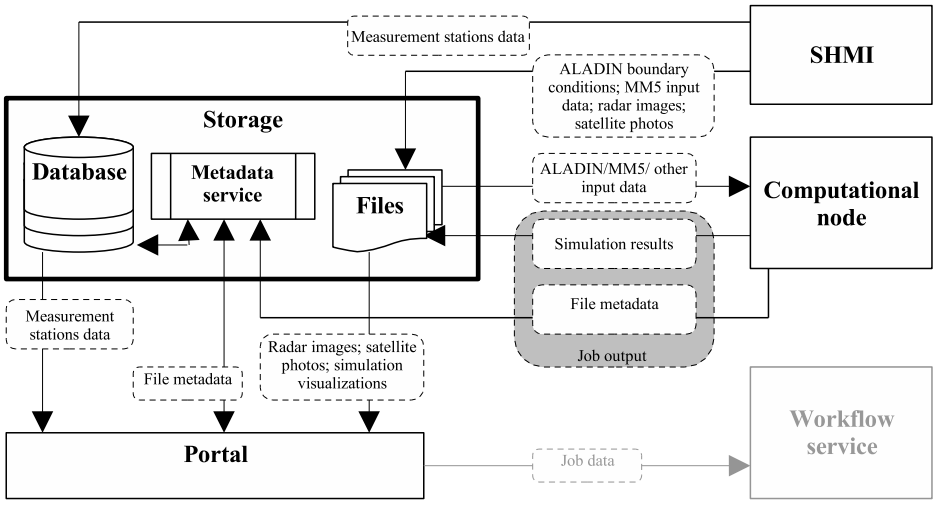

Fig. 5. Scheme of data management tasks in FloodGrid

the storage. While the measurement stations data ends its journey in the portal, the metadata is often used to find suitable data for a simulation job. Because of this connection between metadata and job definition document, also the workflow service and its connection to portal is displayed, although this part of the PSE is not incorporated in the data management suite described in this chapter (therefore it is grey in Figure 5).

The files with ALADIN and MM5 inputs are transferred (via a replica management software described later) to computational node when a job requires them. Files produced in the job are transferred back; their metadata descriptions are registered in the metadata service (also described later in this chapter).

5.2 Input Data Sources

The input data of our flood prediction problem solving environment is divided into the following groups:

- Radar images,
- Satellite photos,
- Measurement stations data,
- ALADIN boundary conditions,
- MM5 input data.

All this data was, or is currently provided by SHMI. Transport of some of this data was implemented, deployed and tested for some time and then disabled because of

SHMI Internet connection bandwidth considerations. Because all the transfers can be enabled anytime when needed, we will formulate our description in present tense to avoid confusion.

5.3 Transport of ALADIN Boundary Conditions and MM5 Input Data

The most important input data of our FloodGrid application is the ALADIN and MM5 input data, which is needed for the simulation cascade. All other simulated data is derived from one of these two sources. The data is computed at SHMI and transferred daily – uploaded to a computer inside the FloodVO. At this computer it is annotated with metadata and registered with both replication suite and metadata service. Then it is ready to be used by a simulation job.

ALADIN and MM5 inputs are actually representation of the same physical values. MM5 inputs are currently computed at SHMI from ALADIN data (this may change in the future and MM5 input data may become independent from ALADIN input data; because of this the data is transferred separately, not computed in the FloodGrid application). Both ALADIN and MM5 are meteorological models, which can be used partly interchangeably. Anyway, we use them both because of their different internal implementation and abilities. Meteorological expert may find ALADIN more useful in some simulations, while in others the choice may be MM5.

5.4 Transport of Radar Images, Satellite Photos and Measurement Stations Data

This data can be considered as 'complementary' to the ALADIN and MM5 input datasets. While it is not necessary for the simulation cascade, it may be very useful for hydrometeorological experts. The satellite and radar images provide both short-term weather information, on different scale and resolution. The measurement stations data is a source of real data, with which the computed ALADIN, MM5 and other inputs can be compared. If an expert wants to review a past situation, it is always better to have the real measured data than to rely just on simulations. As shown in Figure 5, the measurement stations data are transferred to the portal, where they are available in the form of a graph.

5.5 Replica Management

The actual storage and maintenance of a coherent dataset collection is performed by a replica management software. It keeps track of the datasets, potentially stored at multiple places duplicitly (replicated). The creation of replicas of a single dataset may be well used for better security and protection against an unwanted loss of the dataset because of a sudden storage device failure, as well as for better access to the file by making it more local to the place which requires it. Altough the term replica management may be pertinent to several areas of distributed computing research, we deal mainly with the Grid and Grid computing paradigm. For the

Grid, a replica management suite has been developed in the European DataGrid Project [14]. The software developed in work package 2 of the DataGrid [15] covers the registration, lookup, transfer and replication tasks of a mature replica management suite, with sufficiently distributed control. Its last implementation is based on the modern paradigm of web services and OGSA [13] architecture. Anyway, it is lacking a modern and scalable metadata repository.

5.6 Metadata Production and Storage

The metadata production and storage tasks are handled by three different components of the FloodGrid data management system. Metadata is produced by a set of specialised programs and scripts, which extract important values from datasets (both transferred from outside of FloodGrid and computed). The metadata is then stored in the metadata database – a RDBMS, interfaced with the rest of the FloodGrid via the metadata service. While we find it useless to describe the metadata extraction methods, which vary from one type of dataset to other and are not very complicated in general, we focus our view on the two remaining components of FloodGrid metadata system – the service and database.

5.7 Metadata Service

The metadata service (named org.crossgrid.wp12.metadata.service) is an OGSI-compliant [12] web service. It enables its user to add, remove, edit metadata descriptions of files (identified by a GUID) as well as to find registered files with certain properties. The service interface exports these methods:

- AddObject,
- RemoveObject,
- ModifyObject,
- FindObject,
- ShowStructure.

The **Object* methods allow the user to work with metadata description of a file – he/she may add a description, remove or change it or find a file (a set of files) by its (partial) description. The *ShowStructure* method is not connected to a file, but rather to the whole database. Because the system is modular and the metadata service can be used to access any database conforming to some rules (described below), this method is necessary for the service user to properly display stored data. This method shows all available metadata items, their types and allowed or available values (in case of enumerations).

The service is accessible either by a client library, available in the Java 2 programming language, or by a visual interface implemented in the FloodGrid portal.

5.8 Metadata Database

The metadata database is a structure, which supports:

- metadata items of types *String*, *Integer*, *float*, *date* (datetime) and *geometry* – geometrical shape (point, line, rectangle, polygon),
- closed enumerated sets,
- open (midifiable) enumerated sets.

The structure of the database is not hardwired into the metadata system; it is defined in a table and can be modified anytime. Simple restart of the metadata service is then needed to access the new structure. Enumerated sets are implemented via indirection and are very useful for example for string values, where frequent use of identical values is expected (like names of users, for example). Instead of storing multiple copies of the same string, only a reference to another table – holding all the defined strings – is stored. In addition, such a set of values can be locked, so the user is forced to choose only from predefined values.

5.9 Typical Data Management Usage Scenario

To better illustrate the use of the metadata suite, described in this chapter, we wil present a usage scenario, which shows the coordination of single modules of the suite.

Let us imagine a case, where the user wants to simulate a weather prediction for certain time and area. He/she logs into the FloodGrid portal and starts the task by locating the input data for his/her simulation. He/she accesses the metadata lookup portlet, enters the file description (in this case type of file – ALADIN or MM5 input data, date and geographical location of the data). The metadata lookup gives him/her one or more files. He/she accesses the files' descriptions and chooses the right one. He/she clicks on its GUID and a set of physical location URLs is displayed. He/she chooses one of these URLs for the job definition document. Second option (currently under development) would be to just enter the GUID in the job definition document. The job could then access the most convenient replica, depending on where it would be started.

After providing the job definition document via the workflow portlet (described elsewhere), the job can be submitted. Once the job is executed on a computational node, all input data is downloaded (see Figure 5) and computation can begin. Produced output data is registered and uploaded to Grid storage (CopyAndRegister function of the edg-rm replica management tool). Metadata is extracted (using the metadata extraction scripts) and sent back to the workflow service for registration into the metadata service. The upload of metadata to the workflow service is necessary because the computational nodes are not equipped with the software necessary for metadata service access. The cycle is now closed – new data is in the Grid, annotated and ready to be found and used in another computational job.

5.10 Future Development of Data Management Software in CrossGrid Task 1.2

We mentioned several 'under development' pieces of software in this paper. For example, one such future enhancement not only of the data management suite, but of the whole PSE will be better integration of replication software with simulation jobs. The user will not have to find and select the physical file URL, the software will do it automatically based on the chosen GUID.

Further modifications are expected in the metadata service. The current prototype is witout GSI security – this will change. Also a method of distribution of the (potentially widely used) metadata service is considered. More metadata services could cooperate in a transparent way – users would 'write' metadata to their local service, but lookups will be done in all available networked services. Support for more data types is also considered, altough to date such need has not arisen.

6 USER INTERFACES

There are two user interfaces that provide access to the grid for the flood application:

Application portal – flood application specific portlets in the Jetspeed [16] portal framework

Migrating Desktop – a Java based fat client emulating desktop environment and providing a user with various services making his life easier in the grid environment.

Both these interfaces access the flood workflow service (described above) in order to provide a user the ability to run, monitor and view results of his jobs regardless of the user interface he uses.

6.1 Application Portal

There is increasing need for easy to use, secure and customizable user interface from the side of users of the portal. We use the Jetspeed portal framework [16], that is open source implementation of an Enterprise Information Portal, based on Java portlet server and XML. A portal makes network resources (applications, databases and so forth) available to end-users. Jetspeed acts as the central hub where information from multiple sources are made available in an easy to use manner. A new unique session is created each time the user logs into the portal, so the portlets can exchange information within the same session.

Portlets are small pieces of Java code that can be plugged to portal and they make up the user interface. Currently there are portlets for proxy certificate retrieval (MyProxy portlet), general job submission portlet, job status portlet, metadata portlet and GridFTP portled made by other parties.

Our web portal for workflow management consists of following parts (portlets):

Workflow template portlet – shows list of defined workflow templates and it allows the user to choose desired workflow template. Jobs of the template can be examined by the button (located in the right from workflow template name, indicated as down-arrow icon). Default parameters are loaded from configuration files.

Workflow portlet – This portlet shows list of workflows of the current Jetspeed user. Each workflow and job is colored according to its current state. This portlet behaves similarly like workflow template portlet (so enables to explore jobs and parameters in the same way), except that the user is able to explore and change parameters for particular job and submit the workflow. The names of jobs are indented according to their order of execution, so it is intuitive to find out which jobs are executed in sequence or in a parallel way. The output of the job can be simply viewed by pressing the 'Output' button.

Visualization portlet – The user can browse the output directory of the job and view the content of files and pictures created by visualization postprocessing in the visualization portlet. Visualization portlet also enables to play image sequences to simulate video playback.

Metadata portlet – The user can query the metadata or add new one to metadata catalog (described above). The seaching and retrieving of replicas is realized by restrictive conditions that are sent to the metadata service. The obtained records are transparently divided into the directories, in the main panel, where they can be looked over. The user can manage directories and metadata records from main panel as well.

6.2 Migrating Desktop

Migrating Desktop (MD) is a Java based client application enabling the user to work with grid applications in desktop environment. MD enables the user to submit general application jobs, track their execution and view results, manage files locally and on the grid storage elements.

We have created a specialized plugin as an interface for the flood workflow system. Results can be viewed using MD's built-in file viewer, which can display text files and pictures.

MD contains the plugin that manages metadata records in a very similar way as in Aplication Portal. This plugin is located next to the Workflow plugin on the tabbed pane. The user can add, delete and search for the metadata records. The user can enter the restrictive conditions for metadata search in the query dialog. After some records are retrieved the user can preview the records and select any desired record. Selected records can be further processed, removed from main panel or deleted from metadata storage.

7 COLLABORATIVE TOOLS OF THE USER INTERFACE

The need of cooperation between scientists and users from many organizations in Grid projects requires sophisticated tools for collaborations in portals. The scientists need to access and share data, analyze them, and discuss with other scientists via the collaborative tools. Therefore, collaborative tools are one of the key elements of virtual organizations.

The collaboration in FloodGrid portal is based on the OGCE (Open Grid Computing Environments Collaboratory) [17] framework, which offers a grid-oriented, integrated and extensible collaborative environment via portals. Furthermore, OGCE is based on Jetspeed portal framework, which is chosen as the main platform for portals in the CrossGrid project.

OGCE collaboration portal is organized as a set of sites. Each site is a place for users to visit, work with tools and resources, be aware of who else is visiting, and work together. Each site is a collection of pages, particular configurations of tools. Each tool in OGCE portal is a portlet, an individual web component that is made accessible to users via the portal interface. Each portlet generates only a fragment of the markup that a user sees from his or her browser. Users navigate within a site by invoking the different pages and tools of the site and switching focus between these tools. Users can customize the sites (if they have permissions to do so) to control the set of tools, the tool configurations, the tools organization, and the tools layout in the sites.

One of the most important tool sets of OGCE portal are CHEF [18] teamlets that allow users to communicate with each others. There are several communication teamlets: announcements, discussion, mailing list, chat; so users can choose the suitable communication mode. The teamlets can be also customized according to the users need. Another CHEF teamlet is file hosting that allows users to share documents and data with others in the site. Users can upload files from their computers, create them online or copy them from other sites.

CHEF teamlets also offer other means of collaboration between users. There is calendar tool, where users can schedule events involving other users, times and locations. Event arrivals, event reminders and new or modified events are significant usage events and can trigger notifications. Assignment tool allow users to create tasks, describe them, and assign them to other users and user groups. Tasks may have a deadline and a set of portal resources. Task results are a set of portal resources that are created or modified by the users performing the task. A task may have a number of milestones, each of which is described as a subtask. Users can enter progress reports as they are working on a task. A presence tool shows the users who are visiting the sites.

OGCE portal also imports a large collection of grid tools from other Grid portal projects such as NPACI GridPort [19] or NCSA Alliance Portal [20]. These tools allow users to do Grid operations such as user certificate management (proxy manager portlet), accessing remote files (GridFTP portlet), browsing OGSA services (OGSA

browser), monitoring Grid resources (GPIR portlet), submitting a job to Grid (Job submission portlet).

The access to the OGCE sites is protected by membership and access right. Sites can be open to user community (i.e. everyone can visit them) or closed, that only users who have memberships of the sites, can enter the site. When a user goes to OGCE portal, he/she must provide user IDs and passwords for authentication. After that, the list of sites of which he/she has membership is displayed and he/she can visit them. Inside a site, he/she can perform only operations that are permitted according to his/her access right. The administrators of sites can give/take users' memberships and change their access rights dynamically. They can also set up new sites, add/remove tools or pages to sites, change default setting of tools, change site layouts.

8 VISUALIZATION OF SIMULATION RESULTS

The final stage of simulation processes is presentation of their results in an appropriate form. As the outputs from simulation applications usually comprise large amounts of numeric data, sophisticated ways of converting the data into easy comprehensible forms are necessary. Among these, graphical presentation takes an outstanding place, mainly for its ability to present multiple data and complex relationships, that would be very hard to express otherwise.

The aim of flood animations in the CrossGrid project was to predicate, which parts of the modeled area would be endangered in case of a natural disaster. The most straightforward way of presenting such data is a map. Though an expert in the field of meteorology would be most satisfied with data presented in classical two-dimensional form, in order to make the simulations more interesting for the general public, we created a 3D-visualization system.

8.1 Inputs and Outputs

Although we created three different visualization systems, their input and output data were nearly the same. The following data were used as inputs:

- A LIDAR terrain model. Those are terrain height data taken from a plane using the laser technology. The resolution of such data is usually very high, in our case the samples were taken in a 1×1 m grid.
- An orthophotomap, a photograph of the terrain surface taken from a plane. The resolution of the map is very high, matching the precision of the LIDAR data.
- Terrain mesh, created from the LIDAR data for the purposes of simulations. The mesh consists of triangular and polygonal shapes. The precision of the mesh was set to match the criteria of the simulation algorithm. All outputs from the simulations used this mesh as a referential structure.

- The actual results of the simulations. Though the simulation internally computed the water flow in tiny time steps, the final data were produced in 15-minutes intervals. Each time step comprised an array of numerical values, telling whether the given terrain element had been flooded or not, and what is the level of water within that element.
- Most of the input data required some kind of filtering and preprocessing, in order to get rid of small errors that have occurred in the process of their preparation, and also to make them more visually appealing. The corrections were done manually.

The VRML world files are ordinary text files, usually with the wrl extension. According to the VRML specification, they contain text (i.e., human-readable) definitions of objects to display. In order to view the VRML files, the user must have installed a VRML browser. There are plenty of free VRML browsers available for virtually all mainstream computing platforms. A VRML browser could be delivered as a self-standing application, or, more often, as a HTML-browser plugin.

As already said, we have created three different visualization systems. The first system produced simple and pertinent two-dimensional pictures, targeted mainly for experts. The output pictures were in commonly-used, platform-independent PNG or JPEG formats. The second, Blender-based system, produced visually-appealing, 3-dimensional data. The system also produced pictures in arbitrary image formats. The third, VRML-based system, produced as its output a set of VRML-world definition files.

8.2 Applied Technologies

All systems were programmed in the Python programming language. Python is a high-level programming language, providing a considerably rich standard library. It allowed the systems to be created with minimum of other additional software.

The first solution was based on the GRASS GIS system. GRASS is a free, professional geographical information system, developed and supported by an international team of experts. It offers a rich set of commands for work with raster, vector and site-oriented data. GRASS provides an environment that makes easy both the interactive use, and the creation of scripts. In our project, GRASS was used to read and convert the input terrain data, overlay them with the referential mesh and draw the flooded elements with chosen colors. The GRASS visualization system was later integrated with Migrating Desktop and Jetspeed portal.

The second solution used the Blender to create 3D pictures of the flooded area. Blender is a free, open-source 3D modeling program with support for creation of finely-rendered images, tools for creation of animations and also experimental support for creation of games. We used Blender to input the LIDAR terrain data, applying a texture onto them and overlaying the 3D world with the water mesh. Blender exposes a programming API. This made it possible to do all the necessary work with our customary Python scripts running from the inside of Blender. As

Blender is an inherently interactive application, it always needs assistance from the user, thus making the automatic workflow-driven processing impossible. The user has to choose a view, adjust the camera and a few other settings every time he or she wants to make a picture. It is significant disadvantage that made us move towards the VRML-based solution.

The third, VRML-based solution, transformed the input data into so-called VRML worlds. The VRML modeling language contains a set of primitives for describing the basic geometric object, together with the tools for creating arbitrary customary object designs. Because it is an industrial standard for describing 3D scenes, this solution is platform-independent. Although we have called the third solution VRML-based, it means only that the program produced VRML files as its output. In fact, the whole process of transforming the data from input to internal formats, processing them and transforming them to the VRML format is done purely with Python scripts. No other software is necessary, which is the biggest advantage over the previous two solutions. It makes the integration with other parts of the CrossGrid project seamless. The only requirement is a VRML browser, which has to be installed into the user's web browser.

9 CONCLUSION

We have presented a collaborative grid computing environment for flood forecasting, called FloodGrid. This environment is an application of the CROSSGRID project. It consists of a computation core with several meteorological, hydrological and hydraulic simulation models, a workflow service which steers these models, a data and metadata management suite, a user interface in two versions, including collaboration and visualization tools. The FloodGrid application is still evolving and future version will include more models and feature better usability for its end users. A two-dimensional visualization system has been incorporated into the Migrating Desktop and Jetspeed Portal user interfaces. Pictures made with this system were also presented at the regular CROSSGRID meetings. A VRML-based visualization system has been even adopted by our colleagues from the Johannes Kepler University in Linz. It was successfully modified and adjusted for usage in the CAVE system.

REFERENCES

[1] Gallopoulos, S.—Houstis, E.—Rice, J.: Computer as Thinker/Doer: Problem-Solving Environments for Computational Science. IEEE Computational Science and Engineering Magazine, Vol. 2, 1994, pp. 11–23.

[2] Foster, I.—Kesselman, C.: The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann Publishers, Inc., 1999.

[3] Ramakrishnan, N.—Watson, L. T.—Kafura, D. G.— Ribbens, C. J., Shaffer, C. A.: Programming Environments for Multidisciplinary Grid Communities. Available on: `http://citeseer.nj.nec.com/ramakrishnan01programming.html`.

[4] EU 5FP project CROSSGRID. Available on: http://www.eu-crossgrid.org.

[5] Hluchý, L.—Astaloš, J.—Habala, O.—Tran, V. D.—Šimo, B.: Concept of a Problem Solving Environment for Flood Forecasting. Recent Advances in Parallel Virtual Machine and Message Passing Interface. LNCS 2474, Springer Verlag, 2002.

[6] The International ALADIN Project. Available on: http://www.cnrm.meteo.fr/aladin/.

[7] HEC-1. Available on: http://www.bossintl.com/html/hec-1.html.

[8] Hydrological Simulation Program-Fortran. Available on: http://water.usgs.gov/software/hspf.html.

[9] FESWMS – Finite Element Surface Water Modeling System. Available on: http://www.bossintl.com/html/feswms.html.

[10] Ajmani, K.—Ng, W. F.—Liou, M. S.: Preconditioned Conjugate Gradient Methods for the Navier-Stokes Equations. Journal of Computational Physics, Vol. 110, 1994, pp. 68–81.

[11] Saad, Y.—Vorst, H.: Iterative Solution of Linear Systems in the 20th Century. Journal of Computational and Applied Mathematics, Vol. 123, 2000, pp. 1–33.

[12] Tuecke, S.—Czajkowski, K.—Foster, I.: Open Grid Services Infrastructure 1.0. Available on: http://www.ggf.org/ogsi-wg.

[13] Foster, I.—Kesselman, C.—Nick, J. M.—Tuecke, S.: The Physiology of the Grid. An Open Grid Services Architecture for Distributed Systems Integration.

[14] The DataGrid Project web site. Available on: http://www.eu-datagrid.org.

[15] Kunszt, P.—Laure, E.—Stockinger, H.—Stockinger, K: Advanced Replica Management with Reptor. In 5th International Conference on Parallel Processing and Applied Mathematics, Czestochowa, Poland, September 7-10. Springer Verlag, 2003.

[16] Apache Jetspeed Portal Framework. Available on: http://portals.apache.org/jetspeed-1/index.html.

[17] OGCE home page. Available on: http://www.ogce.org.

[18] CHEF home page. Available on: http://www.chefproject.org.

[19] NPACI GridPort home page. Available on: https://gridport.npaci.edu/.

[20] NCSA Alliance Portal. Available on: http://www.extreme.indiana.edu/xportlets/.

Ladislav Hluchý received his Dipl. Ing. (M.Sc.) degree from the Slovak Technical University Bratislava in 1975, and the C.Sc. (Ph.D.) degree in computer science from the Slovak Academy of Sciences in 1986. He is the member of IEEE Computer society, and IEEE Communication Society. His research interests include algorithms and methods for high performance computing and grid computing.

Ondrej Habala is a researcher and Ph.D. student at the Institute of Informatics of the Slovak Academy of Sciences. His primary interests are data management in grids, metadata management and DHT networks.

Viet Tran is scientist at the Institute of Informatics, Slovak Academy of Sciences. His research concentrates on numerical modeling, high performance computing, grid computing and portal technologies including collaborative environments.

Emil Gatial is researcher at the Institute of Informatics at the Slovak Academy of Sciences, Slovakia. His research is focused on applying semantic techniques and automated reasoning in knowledge management.

Martin Mališka is a Ph.D. student at the Institute of Informatics of the Slovak Academy of Sciences. His studies are concerned with the problem of workflow management in large-scale grid appication and networks of web services.

Branislav Šimo is a reasearcher and Ph.D. student at the Institute of Informatics of the Slovak Academy of Sciences. He studies mainly the usage of collaborative tools and portals in grid computing.

Peter Slížik is a Ph.D. student at the Institute of Informatics, Slovak Academy of Sciences. He is interested in visualisation of scientific data, computational peer-to-peer networks and grid computing. He works under the leadership of Dr. L. Hluchý.