Computing and Informatics, Vol. 27, 2008, 271–284

LEVERAGING INTERACTIVITY AND MPI FOR ENVIRONMENTAL APPLICATIONS

Branislav Šimo, Ondrej Habala, Emil Gatial, Ladislav Hluchý

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

Abstract. This paper describes two different approaches to exploiting interactivity and MPI support available in the Interactive European Grid project. The first application is an air pollution simulation using Lagrangian trajectory model to simulate the spread of pollutant particles released into the atmosphere. The performance of the sequential implementation of the application was not satisfactory, therefore a parallelization was planned. The MPI programming model was used because of some previous experience with it and its support in the grid infrastructure to be used. Then the interactivity enabling the user to receive visualizations of simulation steps and to exercise control over the application running in the grid was added. The user interface for interacting with the application was implemented as an plug-in into the Migrating Desktop user interface client platform. The other application is an interactive workflow management system, which is a modification of a previously developed system for management of applications 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. This tool is able to visualize the inner workflow of the application. 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.

Keywords: MPI, workflows, interactivity, grid computing, interactive european grid

1 INTRODUCTION

The focus of current grid infrastructures like EGEE [10] and middleware like gLite [7] 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 single instance is not that important as the time required to process the whole set of of jobs, there are a lot of applications where the minimization of the run time 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 [8] messaging protocol as a means for data exchange.

The other feature lacking in currently prevalent grid infrastructures is the ability to interact with the 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 [1, 4] 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 chapter. The following sections describe two different environmental applications that make use of the available MPI and interactivity functionality in order to provide better experience to the end user.

Section 3 presents an air pollution application simulating the spread of pollutant particles released to the atmosphere using the Lagrangian trajectory model. This application is very useful for study of development of past air contaminations and forecasting of behavior of possible future ones. Forecasting scenarios are important for contingency planning and preparation of e.g. evacuation plans. In the event of an accident the fast execution of the forecasting model with actual meteorological conditions is essential in proper management of the emergency. The ability to interactively increase the number of simulated particles allows faster proceeding of the simulation in the beginning when the particles are grouped together and increasing their number only after some time subject to operator's consideration.

In Section 4 we describe an interactive workflow management system of the flood forecasting application. The system was developed as a modification of a system developed previously in the project K-Wf Grid [11] 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. 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.

2 TOOLS FOR INTERACTIVITY AND MPI

In order to use the interactivity and MPI, applications 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. We use the OpenMPI [9] flavor of the MPI specification. The project also supports PACX-MPI [27] for inter-cluster parallel computing. 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) [5] 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 in Figure 1. Below we give further description of these components.

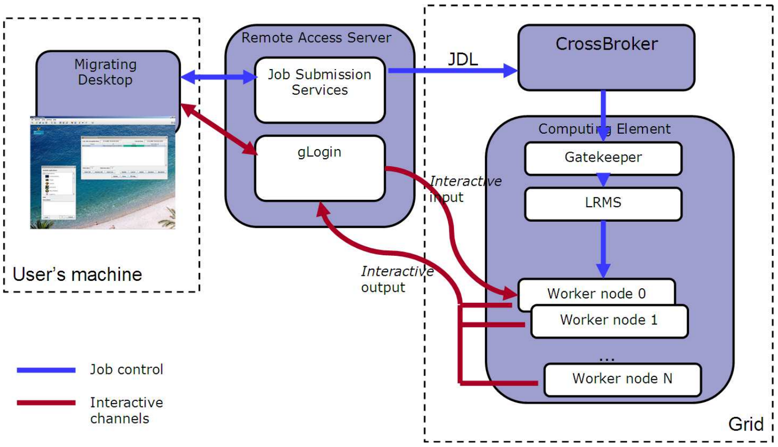


Fig. 1. Interactive channel connecting Migrating Desktop with application running in the grid

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 workstation to grid and back, registration of files to virtual directory), job management (job submission, monitoring), visualization of job results. The MD is implemented in Java language and runs as a client on the user's machine. It is based on the Eclipse OSGi framework [12] plug-in architecture, thus allowing customization of its functionality.

Plug-ins play important role in application support by providing application specific functionality. Input plug-ins provide custom input parameter specification, visualization plug-ins provide visualization of application outputs and 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 interactive application, writing a visualization plug-in is usually necessary.

The user-application connection is realized by setting up a 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 input of the application. In case of 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 Figure 1).

3 INTERACTIVE AIR POLLUTION APPLICATION

The Air pollution simulation, originally named IMS Model Suite, is a complex software system developed by MicroStep-MIS [19] primarily for environmental pollution assessment and prediction of consequences of nuclear accident or radiological emergency. It provides users with a comprehensive set of services (local, regional or continental scale dispersion and deposition modeling, data processing and visualization) as well as outputs (particle trajectories, surface and volume concentrations, calculations of the effective and equivalent doses). The architecture of the system, computational complexity and existing or potential interfaces to other systems and services make this application well suited for the Grid and virtual organization environment. The essence of all Lagrangian models is the observation of individual particles and numerical calculation of their trajectories in the atmosphere. The term "particle" denotes any air pollutant or substance (or multiple substances) in the volume of air located to certain position of the space. The particles travel with the wind and the particle trajectory and particle composition reflects natural phenomena such as turbulent diffusion, dry deposition, wet deposition caused by rain and chemical transformation or radioactive decay. The particles are independent during all lifetime, which results in independent evolution equations for each particle of the model that can be computed by a different processor in a parallel implementation. This makes Lagrangian model well suited for the parallel implementation on the computer cluster or grid. The concentration distribution is determined by counting the particles in given volume. This way, the model yields non-negative mass densities and is mass-conserving.

Lagrangian dispersion models simulating real atmosphere work in 3D grid. The scale depends on the spatial and temporal resolution of meteorological data inputs;

particle models can be used for modeling regions ranging from 20 m to thousands kilometers and for the time intervals of 10 minutes up to several weeks. Required inputs are meteorological and orographic data. The most usual ones are wind field (velocity and direction), pressure, temperature, humidity, precipitation and orographic data, e.g. type of the terrain (land or sea) and its roughness. Some parameters, such as wind fluctuations, could be generated by pre-processors. Diagnostic wind field models interpolate wind observations in measurement locations to the grid. In this way, data enter the model as a chronological order of fields. The other necessary inputs are release parameters of the source and type, molecular weight and diameter distribution of particles. The model output is a time sequence of the spatial distribution of the concentration of emitted species, its transformation products and the amount deposited.

3.1 Parallelization

At the beginning the model was a sequential application running on Windows operating system. In order to run it in the grid infrastructure provided by the int.eu.grid project, the model had to be ported to the Linux OS, what required a new implementation of certain library functionalities provided by the Borland C++ Builder – the development environment used for application development on Windows.

As already said, the particles in the model are simulated independently of each other and therefore the parallelization was quite straightforward. The particles are divided into equally sized subsets and simulated independently on individual computational nodes of the job. OpenMPI [9] flavor of MPI [8] was chosen as the implementation standard for parallelism, because it is the standard supported by the project.

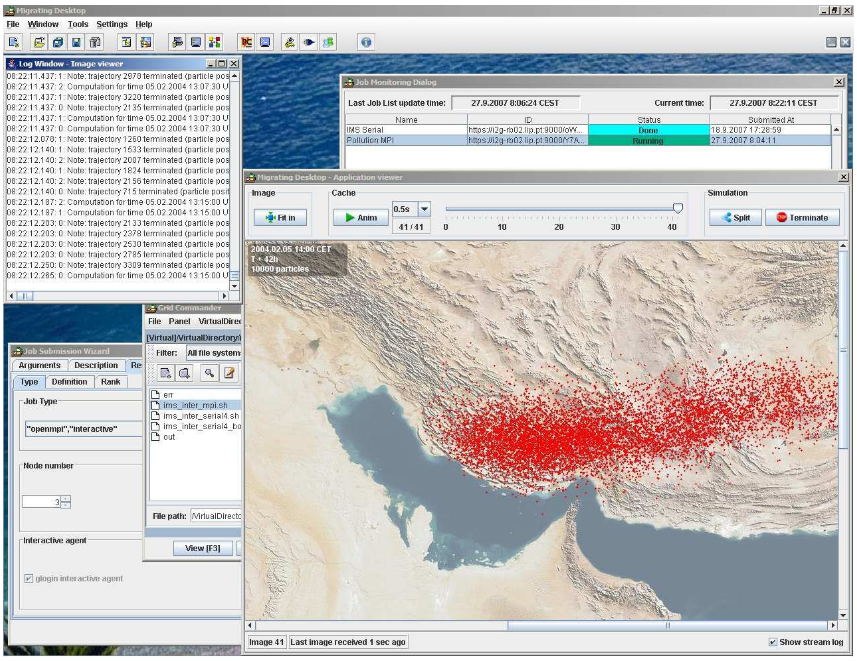
Apart from simulating just a subset of the particles, the parallel instances of the simulation run quite independently. Each one reads the common configuration files and input files by itself, simulates its share of particles and at the end of each simulation step sends the particles' positions to the master process, which collects them and dumps them to a file. The master process then executes a visualization task on the file, in order to produce an image showing current state. The image is used for example in the interactivity scenario described below. After that, the simulation continues with next simulation step.

3.2 Interactivity

Our application needs just the raw interactive channel. When considering how to express the state of the simulation to the user we decided to send pictures created by the application running on the grid that show the map of the simulated area with particles plotted on it. Such image is sent for each simulation step in the JPEG format. It is decoded by the plug-in and shown to the user. Each step of the simulation the user can send two commands back to the application – to split all the particles or to terminate the application. Particle split results in doubling the count of the particles, thus increasing the precision of the simulation from that point on, but also doubling the computational requirements. Because the interactive channel takes all the data from the standard output of the application, we had to create a simple multiplexing protocol that separates the image data and accompanying metadata from the usual text messages generated by the simulation run. The text messages are shown in separate window in MD and enable users to monitor the execution of the application from the technical point of view, i.e. environment setup, application startup, application progress. This is very convenient especially during the development stages of the application.

3.3 Experiences

The parallelization of the simulation was expected to provide significant speedup, which should have been linear in theory, as the particle set is divided into equal groups and simulated independently. However, in real simulation tests we have found out that the overhead of reading the input files, especially the files containing meteorological forecast was quite high and in scenarios with smaller number of particles (i.e. thousands) the speedup was small.



```
08:22:11.437: 1: Note: trajectory 2978 terminated (particle pos
08:22:11.437: 2: Computation for time 05.02.2004 13:07:30 U
08:22:11.437: 1: Note: trajectory 3220 terminated (particle pos
08:22:11.437: 0: Note: trajectory 2135 terminated (particle pos
08:22:11.437: 1: Computation for time 05.02.2004 13:07:30 U
08:22:11.437: 0: Computation for time 05.02.2004 13:07:30 U
08:22:12.078: 1: Note: trajectory 1260 terminated (particle pos
08:22:12.140: 1: Note: trajectory 1533 terminated (particle pos
08:22:12.140: 2: Note: trajectory 2007 terminated (particle pos
08:22:12.140: 1: Note: trajectory 1824 terminated (particle pos
08:22:12.140: 2: Note: trajectory 2156 terminated (particle pos
08:22:12.140: 0: Note: trajectory 715 terminated (particle posit
08:22:12.187: 2: Computation for time 05.02.2004 13:15:00 U
08:22:12.187: 1: Computation for time 05.02.2004 13:15:00 U
08:22:12.203: 0: Note: trajectory 2133 terminated (particle pos
08:22:12.203: 0: Note: trajectory 2378 terminated (particle pos
08:22:12.203: 0: Note: trajectory 2530 terminated (particle pos
08:22:12.203: 0: Note: trajectory 2785 terminated (particle pos
08:22:12.250: 0: Note: trajectory 3309 terminated (particle pos
08:22:12.265: 0: Computation for time 05.02.2004 13:15:00 U
```

Last Job List update time: 27.9.2007 8:06:24 CEST Current time: 27.9.2007 8:22:11 CEST

Name	ID	Status	Submitted At
IMS Serial	https://i2g-rb02.lip.pt:9000/oW...	Done	18.9.2007 17:28:59
Pollution MPI	https://i2g-rb02.lip.pt:9000/Y7A...	Running	27.9.2007 8:04:11

Job Type: "openmpi","interactive"

Node number: 3

Interactive agent: ☑ glogin interactive agent

Fig. 2. Screenshot of the Migrating Desktop showing two windows of the application plug-in on top

With the increasing number of particles the speedup was rising compared to sequential simulation as the overhead of reading input files is constant for the same scenario regardless of particle count. Because the objective is to run the simulation with as many particles as possible, the overhead poses no significant problem.

Another problem we have experienced was extreme slowdown of the file operations on cluster with shared home directories. The application uses a lot of small files that need to be unpacked at the start and are then accessed during the simulation. It has turned out that such usage of the NFS file system can cause extreme slowdown (from several second to several minutes). The performance is fine on clusters without shared home directories. The solution for clusters with shared homes is to use directory located on local disk.

The setup of interactive channel was without a problem. We have experienced occasional errors in image data, what turned out to be caused by a lack of synchronization of text output from the parallel instances of the application. This has been solved by making sure there is no writing to standard output by sibling processes while the image data are being sent to the stream by master process.

4 INTERACTIVE WORKFLOW MANAGEMENT

The execution of a workflow in grid environment usually means automatic execution of its tasks by some kind of workflow engine. From the user's point of view the whole workflow is processed as one big job and 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 [20] 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 an 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 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 following sections 4.1 and 4.2 we describe the original implementation of the workflow execution engine that was implemented in the KWf-Grid [11, 3] project and then the re-implementation of the workflow engine to the grid environment of the int.eu.grid project. Finally, Section 4.3 shows a real use case of the system – flood forecasting application.

4.1 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 Figure 3) is the Grid Workflow Execution Service (GWES) [13]. 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) [14].

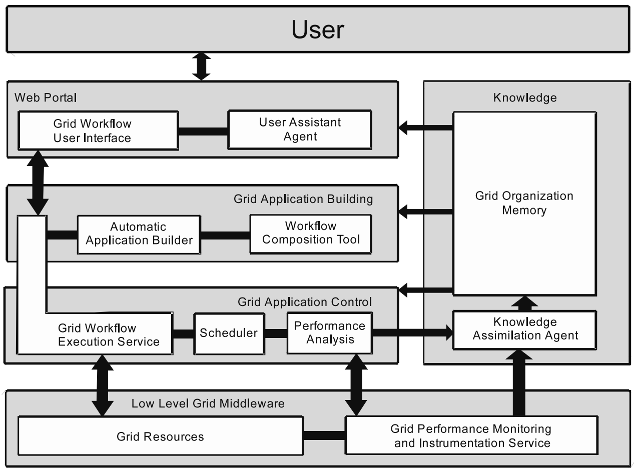

User
Web Portal
Grid Workflow User Interface
User Assistant Agent
Knowledge
Grid Organization Memory
Grid Application Building
Automatic Application Builder
Workflow Composition Tool
Grid Application Control
Grid Workflow Execution Service
Scheduler
Performance Analysis
Knowledge Assimilation Agent
Low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

Fig. 3. Architecture of components used in the K-Wf Grid project

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 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 [15].

The GWES is supported by several other services and tools. In the K-Wf Grid project, 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 the information present in the knowledge base of the infrastructure.

Another tool supporting GWES is the Grid Workflow User Interface (GWUI) [16]. 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 Figure 4.

4.2 Interactive Workflows with GWES in int.eu.grid

In the int.eu.grid project, the infrastructure supporting GWES, as well as GWES itself is modified to fit into the common grid infrastructure based on LCG [17] and gLite [7] grid middlewares.

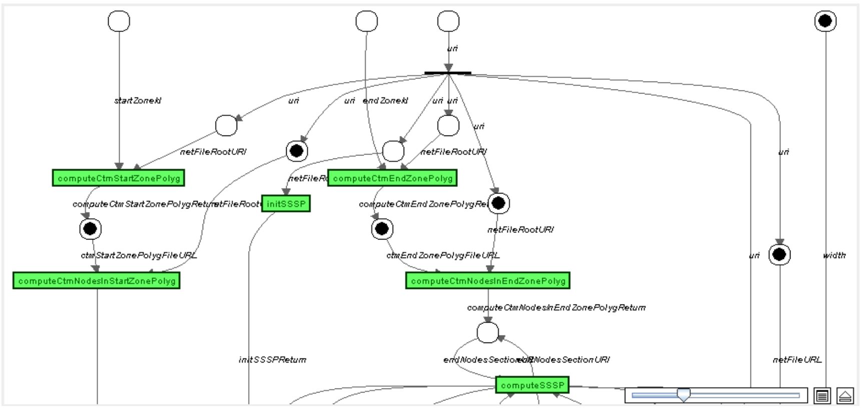


Fig. 4. A screenshot of a sample workflow visualized in GWUI

Since 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 the command line. When the job starts, the 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 ouput, which are connected to the interactive channel of int.eu.grid. At the other end of this channel is the GWUI, working as a 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 GWES remain almost the same as in 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. GWUI has received the ability to modify workflows by adding, removing, and reconnecting activities and places. The possibility to edit 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 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 to or greater than 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 encounters activity during execution, whose demand for nodes exceeds the total number of nodes allocated to the interactive job, it signals a fault to the user, and aborts the workflow.

4.3 Flood Forecasting Application

The flood forecasting application itself started its life in the EU project ANFAS [21]. In the beginning, it was an HPC experiment, using a hydraulic simulation model (FESWMS [24]) to predict water flow in an area hit by a river flood. After ANFAS, the application has been significantly extended during the CROSSGRID [22, 2] project, to contain a whole cascade of simulation models, and to use the Globus Toolkit [26], 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 predicted 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 [28], the application has also changed. In the MEDIgRID project [23], it was extended with more simulation models and visualization tools, and deployed as a set of loosely coupled WSRF [25] services, using Globus Toolkit [26] version 4.

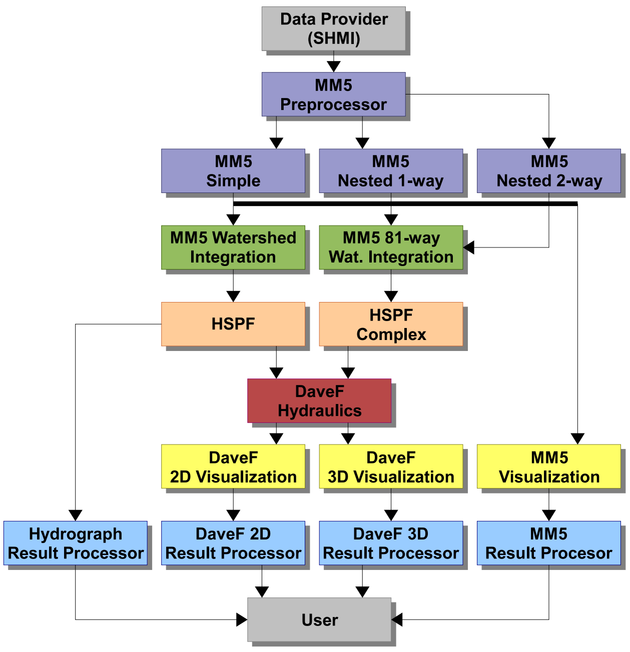


Fig. 5. Architecture of the flood forecasting application

The new architecture of what was previously called a *simulation cascade* [22] can be seen in Figure 5. It is a set of loosely coupled models, with several possible execution scenarios. Figure 5 contains several entities, each of them having its role in our application. At the top of the figure our main data provider is depicted, 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 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 the previous chapter is used to manage the workflow of this application inside of a job submitted to the grid.

5 CONCLUSIONS

We have described two different approaches to making use of the MPI and interactive features provided in the int.eu.grid project. We have shown that parallelization of a simulation model does not have to be complicated provided it is based on a convenient internal model and it gives significant speedup worth the effort. The addition of interactivity helped increase the user experience when running in grid environment by providing more convenient debugging during development and ability to steer the application. The interactive workflow management of flood forecasting application gives the user another type of steering capabilities in terms of dynamic workflow restructuring. The application components running inside this system have no startup penalty and therefore it is mostly valuable for workflows of short lived jobs. As the system can be used for any other applications consisting of an interconnecting components, we expect the real value of the system to be shown in the future.

Acknowledgement

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

REFERENCES

[1] Marco, J. et al.: The Interactive European Grid: Project Objectives and Achievements. See this volume.

[2] Hluchý, L.—Habala, O.—Tran, V.—Gatial, E.—Mališka, M.—Šimo, B.—Slížik, P.: Collaborative Environment for Grid-based Flood Prediction. In: Computing and Informatics. Vol. 24, 2005, No. 1, p. 87–108.

[3] Babík, M.—Habala, O.—Hluchý, L.—Laclavík, M.: Semantic Services Grid in Flood-Forecasting Simulations. In: Computing and Informatics. Vol. 26, 2007, No. 4, p. 447–464.

[4] Interactive European Grid project. `http://www.interactive-grid.eu`.

[5] 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, Vol. 21, 2005, No. 1, pp. 37–44.

[6] Rosmanith, H.—Volkert, J.: glogin – Interactive Connectivity for the Grid. In: Juhasz, Z., Kacsuk, P., Kranzlmüller, D.: Distributed and Parallel Systems – Cluster and Grid Computing, Proc. of DAPSYS 2004, 5th Austrian-Hungarian Workshop on Distributed and Parallel Systems, Kluwer Academic Publishers, Budapest, Hungary, September 2004, pp. 3–11.

[7] gLite – Next Generation Middleware for Grid Computing. `http://glite.web.cern.ch/glite`.

[8] Message Passing Interface Forum. `http://www.mpi-forum.org`.

[9] Gabriel, E., et al.: Open MPI: Goals, Concept, and Design of a Next Generation MPI Implementation. Proceedings of the 11th European PVM/MPI Users' Group Meeting, Budapest, Hungary, Sep 2004, Lecture Notes in Computer Science (LNCS), Vol. 3241, pp. 97–104.

[10] EGEE (Enabling grids for e-science) project. `http://www.eu-egee.org`.

[11] Bubak, M.—Fahringer, T.—Hluchy, L.—Hoheisel, A.—Kitowski, J.—Unger, S.—Viano, G.—Votis, K. and K-WfGrid Consortium: K-Wf Grid – Knowledge based Workflow system for Grid Applications. In Proceedings of the Cracow Grid Workshop 2004, p. 39, Academic Computer Centre CYFRONET AGH, ISBN 83-915141-4-5, Poland 2005.

[12] Equinox – An OSGi Framework Implementation. `http://www.eclipse.org/equinox`.

[13] Hoheisel, A.—Ernst, T.—Der, U.: A Framework for Loosely Coupled Applications on Grid Environments. In: Grid Computing: Software Environments and Tools. Cunha, J. C.; Rana, O. F. (Eds.), 2006. ISBN: 1-85233-998-5.

[14] Pohl, H. W.: Grid Workflow Description Language Developer Manual. K-Wf Grid manual, 2006, `http://www.gridworkflow.org/kwfgrid/gworkflowdl/docs/KWF-WP2-FIR-v0.2-GWorkflowDLDeveloperManual.pdf`.

[15] Hoheisel, A.—Linden, T.: Grid Workflow Execution Service – User Manual. K-Wf Grid, 2006, `http://www.gridworkflow.org/kwfgrid/gwes/docs/KWF-WP2-D2-FIRST-GWESUserManual.pdf`.

[16] Grid Workflow User Interface development site. `http://www.gridworkflow.org/kwfgrid/gwui/docs/index.html`.

[17] LCG – LHC Computing Grid project. `http://lcg.web.cern.ch/LCG`.

[18] The Visualization ToolKit (VTK). `http://www.vtk.org`.

[19] MicroStep-MIS. `http://www.microstep-mis.com`.

[20] Fernández, E.—Heymann, E.—Senar, M. A.: Resource Management for Interactive Jobs in a Grid Environment. Proc. of IEEE Int. Conf. On Cluster Computing (Cluster 2006), Barcelona (Spain), CD-ROM edition, IEEE CS Press, September, 2006.

[21] ANFAS Data Fusion for Flood Analysis and Decision Support. `http://www.ercim.org/anfas`.

[22] Hluchý, L.—Tran, V. D.—Habala, O.—Šimo, B.—Gatial, E.—Astaloš, J.—Dobrucký, M.: Flood Forecasting in CrossGrid project. In: Grid Computing, 2nd European Across Grids Conference, Nicosia, Cyprus, January 28–30, 2004, LNCS 3165, Springer-Verlag, 2004, pp. 51–60, ISSN 0302-9743, ISBN 3-540-22888-8.

[23] Šimo, B.—Ciglan, M.—Slížik, P.—Mališka, M.—Dobrucký, M.: Mediterranean Grid of Multi-Risk Data and Models. In: Proc. of 1st workshop Grid Computing for Complex Problems – GCCP 2005, Veda, 2006, pp. 129–134, ISBN 80-969202-1-9. November–December 2005, Bratislava, Slovakia.

[24] Froehlich, D. C.: HW031.D – Finite Element Surface-Water Modeling System: Two-Dimensional Flow in a Horizontal Plane. Users Manual: Federal Highway Administration Report FHWA-RD-88-177, 1989, 285 p., `http://water.usgs.gov/software/FESWMS-2DH`.

[25] 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. `http://www.globus.org/wsrf/specs/ogsi_to_wsrf_1.0.pdf`.

[26] Foster, I.: Globus Toolkit Version 4: Software for Service-Oriented Systems. IFIP International Conference on Network and Parallel Computing, Springer-Verlag LNCS 3779, pp. 2–13, 2005, `http://www.globus.org/toolkit`.

[27] Keller, R.—Gabriel, E.—Krammer, B.—Müller, M. S.—Resch, M. M.: Towards Efficient Execution of MPI Applications on the Grid: Porting and Optimization Issues. Journal of Grid Computing, Vol. 1, 2003, No. 2, pp. 133–149.

[28] Foster, I.—Kesselman, C.—Nick, J. M.—Tuecke, S.: The Physiology of the Grid. `http://www.globus.org/alliance/publications/papers/ogsa.pdf`.

Branislav Šimo is a researcher and member of the Department of Parallel and Distributed Computing at the Institute of Informatics at SAS. He participated in several EU funded projects dealing with application and user interface development in the grids. He is experienced in HPCN, parallel and distributed computing, grid computing, web services and application portals.