Int. J. of Computers, Communications & Control, ISSN 1841-9836, E-ISSN 1841-9844
Vol. VI (2011), No. 4 (December), pp. 701-712

Application Plugins for Distributed Simulations on the Grid

I.L. Muntean, A.I. Badiu

Ioan Lucian Muntean
Technical University of Cluj-Napoca
Department of Computer Science
Romania, 400027 Cluj-Napoca, 28 G. Baritiu
E-mail: ioan.lucian.muntean@cs.utcluj.ro

Alexandra Ioana Badiu
Technical University of Cluj-Napoca
Data Communication Center "Pusztai Kalman"
Romania, 400027 Cluj-Napoca, 28 G. Baritiu
E-mail: badiu@net.utcluj.ro

Abstract: Computing grids are today still underexploited by scientific computing communities. The main reasons for this are, on the one hand, the complexity and variety of tools and services existent in the grid middleware ecosystem, and, on the other hand, the complexity of the development of applications capable to exploit the grids. We address in this work the challenge of developing grid applications that keep pace with the rapid evolution of grid middleware. For that, we propose an approach based on plugins for grid applications that encapsulate a set of commonly used type of grid operations. We further propose more complex high-level functionalities, such as the plugins for remote exploration of simulation scenarios and for monitoring of the behavior of end-user applications in grids. We provide an example of a grid application constructed with these software components and evaluate based on it the performance of our approach in the context of the simulation of biological neurons. The results obtained on test and production grids demonstrate the usefulness of the proposed plugins, with a small performance overhead compared to traditional grid tools.
Keywords: Grid computing, application plugins, simulation services, GridSFEA.

1 Introduction

Simulation is acknowledged today as the third pillar of modern sciences, in addition to theory and experiments. Computational sciences are driven by simulation and provide the widest majority of applications for high-performance computing resources available in grid environments. Today, user communities of grid computing, having scientific computing background, are bounded to a large extent to a middleware traditionally employed within the respective community. Such examples are the communities formed around the projects: TeraGrid, supported by the Globus Toolkit [12], Enabling Grids for e-Science (EGEE/EGEE2) supported by the g-Lite middleware [8], or NorduGrid, where the Advanced Resource Connector [3] has emerged from. A special place in this sense is taken by the user community formed around the project Distributed European Infrastructure for Supercomputing Applications (DEISA/DEISA-2). In DEISA, the user has typically the choice between the Globus Toolkit and UNICORE [5].

The co-existence of these middleware imposes big challenges to the end-users of grids. Basically, these middleware are rather incompatible in many aspects (security, job and resource

management, etc). Therefore, users have to accommodate with different tools, languages, and operating procedures. This makes the adoption of a middleware other than the one used within his community a challenging task for the computational scientist. A great effort, supervised by the group Grid Interoperability Now of the Open Grid Forum (OGF-GIN) [20] is currently invested by the international grid community in the interoperability of these various middleware. Such examples in Europe are international consortia such as the European Grid Infrastructure (EGI) [11] or the Initiative for Globus in Europe (IGE) [12] aim at providing grid middleware components that are capable of operating in an unified grid middleware (UMD [9]). In this context, the development of grid applications that keep pace with the continuous evolution of the different middleware still remains a big challenge that complements the approach to the integration of the middleware.

In this paper, we address this latter challenge: the development of grid applications. For that, we employ the Grid Simulation Framework for Engineering Applications (GridSFEA) [15], a software package for the gridification of numerical simulation programs. Our approach is to offer out-of-the-box high-level functionalities, packed as plugins, commonly needed in such applications. The plugin mechanism available with the Eclipse Platform has been successfully used in our work. GridSFEA plugins cover for example job submission, job monitoring, file transfer, simulation explorer etc. Interactions with different middleware are possible due to low-level plugins that pack client-side functionality of these middleware. The major benefit of our work is that new applications requiring grid interactions or extensions to existing software packages can rapidly be developed without any grid-knowledge, by solely incorporating these plugins. Furthermore, end-users get a unified way of working with the grid, regardless the underlying middleware in place. In [16] we proposed a first version of the plugin-based architecture of client-side tools of GridSFEA. The evaluated prototype included only one high-level plugin, the parameter generator functionality. In the current work, we further refine the proposed architecture with additional low- and high-level plugins. The evaluation is carried out on development and production grids.

The remaining of this paper is organized as follows: next section outlines the related work. In Sec. 3, the GridSFEA framework is introduced. Eclipse plugins are briefly introduced in Sec.4. Section 5 presents the plugins for grid applications available with GridSFEA. New high-level plugins are discussed in Sec. 6. The results of this work in the context of a scenario from computational neuroscience are the topic of Sec. 7, whereas Sec. 8 outlines our conclusions.

2 Related Work

For the development of grid applications, middleware specific client-side libraries and toolkits are available in the grid community. Such examples are CoG Kit [23] for Globus Toolkit [2], GLUE [4] and CREAM [1] for g-Lite. The use of these libraries requires a thorough understanding of many technical and architectural details of each middleware stack. A great simplification in this sense is brought by the Grid Application Toolkit (GAT) [2] and its successor Simple API for Grid Application (SAGA) [14]. They provide a common programming interface to different grid operations, hiding the implementation details of each middleware. The grid know-how required for the application developer is rather low. Thus, for low-level grid operations, GAT and SAGA are a very good option. Nevertheless, they are not meant to provide high-level functionalities. Low level plugins of GridSFEA are based on community libraries such as GAT or CoG Kit. In addition to these, the framework comes with ready-to-use out-of-the-box application components providing high-level functionalities. Thus, both the application developer and the end-user of the grid need almost no knowledge about the grid.

More evolved tools for the development of grid applications are the grid frameworks such as g-Eclipse [14], Gridbus [22] , WS-GAF [19] or GFac [13]. These programs come with their

own strengths and weaknesses. Gridbus has evolved more in the direction of grid resource broker and cloud, having a rich support for parameter sweep use cases. Composition of scientific applications from Web services is the main feature of WS-GAF and GFac, being more a server-side development. A tool that focuses on the end-users is g-Eclipse. It allows operations on more than one middleware (e.g. g-Lite, GRIA [21]), and, to some extent, the development of new grid applications. The enabling technology of g-Eclipse is the plugin mechanism of the Eclipse Platform. GridSFEA considers both server-side and client-side developments. In the approach proposed in this paper, the plugins of GridSFEA are designed to be interoperable with the ones of g-Eclipse and to complement them with high-level functionalities such as remote exploration of simulation scenarios. Furthermore, the GridSFEA framework comes with its own support for simulation migration on grid computing environments.

The adoption of the grid standards is essential in the realization of grid developments. With this respect, we mention here the JSDL [18] specification of the Open Grid Forum. The end-user requests for jobs and resources can be specified independently of the grid middleware on which the jobs will run. Client tools become, to a large extent, responsible for translating the requests to the specifics of the target middleware. At the moment of writing, only a part of the JSDL specification is implemented by GridSFEA. The JSDL support complements the existing full support to GT4 resource specification language available with GridSFEA.

3 An Overview of the GridSFEA Framework

GridSFEA is a framework that enables computational scientists and engineers to gridify numerical simulation programs and to run them comfortably on the grid. The framework handles simulation scenarios on the grid and assists the user with typical tasks such as formulation of a computing job, submission of the job to the grid, retrieval of resulted simulation data, investigation of simulation results remotely and locally. Two major types of simulation jobs are especially supported by GridSFEA on the grid: parameter sweep and long running jobs.

The framework is modular and distributed, having components located on the end-user side (client-side applications), on grid nodes (simulation services), and on the grid systems where the user jobs run (the checkpoint migration tool). In Fig. 1 is depicted the architecture of GridSFEA. Simulation scenarios are annotated by the framework; at the beginning of the job execution, computation metadata of the scenario is collected by the application wrappers and provided to the simulation services. The services organize the metadata and provide it to consumer applications. Grid data exploration tools or the migration tool are examples of such applications. Based on information recorded in the metadata, the consumers can interact with the simulation data located on the grid.

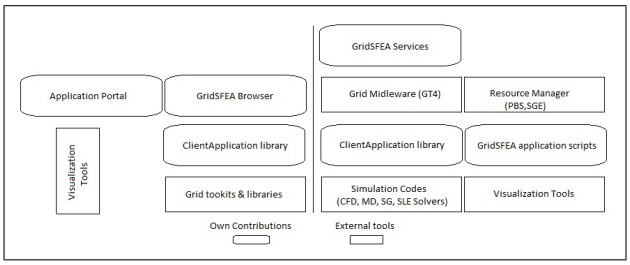

Figure 1: Modules of GridSFEA located in the user space (left) and in the grid environment (right) [15].

We briefly introduce below the role of each of the major elements of the framework. More

details about GridSFEA can be found in [15]. The simulation services are the core of GridSFEA. Their role is to store, organize, and provide scenario metadata to all modules of the framework. The Client Application Library (CAL) acts on the one hand as a client to the simulation services. On the other hand, it handles all the specific grid interactions typically required by client applications such as authentication in grid, file transfer etc. Thus, this module acts as the glue that binds the framework components and integrates the third-party grid libraries. On top of CAL client-side applications are built, such as shell or GUI-based user tools for interacting with the simulations on the grid. Such examples are portal applications, data browser programs, and command-line tool when shell interactions are required by the scientist. Gathering the metadata of the scenarios and handling of simulation checkpoints (checkpoint transfer, simulation resume, and registration of new checkpoints) are the main responsibilities of the Checkpoint Migration Tool. State-of-the-art grid libraries available in the scientific community such as CoG Kit, GAT (Globus Toolkit), DESHL (UNICORE), and CREAM (g-Lite) are employed by the framework for the implementation of grid interactions. This way, our framework allows for client-side interoperability between grids based on these middleware.

Features such as the migration of simulation scenarios or the preview of results in grid make our framework very valuable to computational scientists and engineers and give it a uniqueness touch in the field of grid applications.

4 Eclipse Plugins

In the field of software development, the Eclipse platform is widely employed, on the one hand due to its rich set of supportive tools such as compilers, debuggers, editors, and, on the other hand, due to the large number of contributions from the software community that bring many additional features to the platform. The big success of the Eclipse platform was possible mainly because of the powerful plugin mechanism it employs, which is a good example of state-of-the-art software engineering [5]. Basically, the way that the Eclipse IDE can be extended by its contributors is specified in terms of extension points. The plugins contribute with appropriate extensions to these points, without that the source code of the development environment gets changed. At runtime, these plugins are located, identified, and loaded mostly only when their functionality is required.

In [16] we have presented a detailed list of features that are important to our approach. We summarize below the most relevant ones:

- By means of extension points and extensions, plugins interact, without being familiar with each others implementation.
- The extension points define the public interface the plugin publishes and that can be implemented by further applications.
- Any number of extensions can be used for contributing with functionality to an application.
- The software life-cycle of these plugins is defined in such a way, that a plugin can be installed, updated, debugged, removed independently of the rest of the platform/hosting application.

A further advantage of the Eclipse plugin model and mechanisms is that plugins can be easily grouped together, packed and deployed as extensions of the Eclipse platform, of an existing plugin, or of a plugin-based application. The last two deployment targets are especially of interest in the context of our work on the GridSFEA framework.

5 GridSFEA Plugins for Grid Computing Applications

The mechanisms of GridSFEA are applicable to a wide range of simulation scenarios and can also be used with relatively small customization effort by further grid applications not necessarily related to computational sciences and engineering. To foster the reuse of the know-how we gathered with GridSFEA, we proposed in [16] a new plugin-based organization of the client-side modules of our framework. As enabling technology have been used the Eclipse plugins. The plugin mechanism allows for modular organization of software, for clean and rapid integration of new functionalities, for the extension of existing implementations, for the reuse of framework's features by other applications, all these without altering the existing source code of the software.

The plugins available so far in our framework provide functionalities such as:

- low-level interactions with the grid (such as authentication, job submission, file transfer);
- high-level end-user operations (such as parameter sweep, exploration of results).

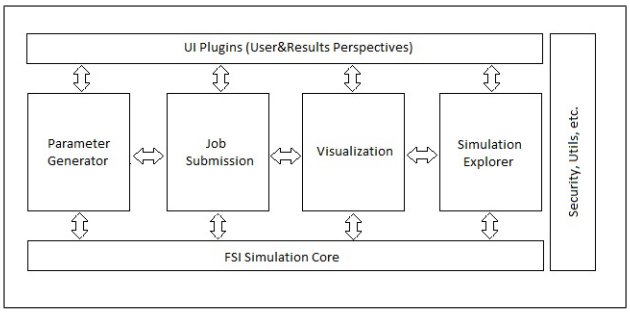

Figure 2: GridSFEA plugins for the development of grid applications [16]

Figure 2 depicts the components of GridSFEA realized with plugins, the arrows representing extension point-extension relationship between components. Several client-side functionalities of GridSFEA such as job submission, job monitoring and exploration of results have been refactored as plugins. In this paper, we further extend this architecture with two groups of plugins: First, we have designed plugins for checking the behavior of applications and services deployed on various grids. Secondly, we provide a plugin for the remote exploration of simulation results located on the grid. These new components of our framework allow the user to tackle various cases studies, such as benchmarking a grid site from an application perspective or deciding on the early stage of a long simulation weather to continue or not the respective grid computation.

Based on the proposed plugins, an application with basic grid capabilities can be rapidly created. In Fig. 3 we show the architecture of such an application. GridSFEA plugins can make contributions to all layers of classical three-layer software architectures, as long as the integration of the grid components is done using the plugin mechanism discussed in the previous section. This way, the developer can focus entirely on the functionality required by the application domain. In the upcoming sections, we give two usage scenarios for the plugins of GridSFEA: The first one is the client-side monitoring of applications and services deployed on grid sites of potential interest for a given end-user and the second one is the exploration of simulation results located on grid.

6 Novel High-level Plugins

This section introduces two novel high-level plugins of GridSFEA. Both are focused on the end-user, the computational scientist: With the first plugin, the user can formulate job requests

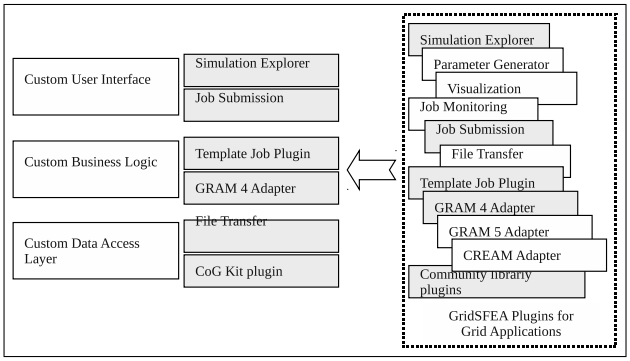

Figure 3: Enhancing a custom application (right) with grid computing capabilities by means of GridSFEA plugins (left): formulation of computation tasks, their submission to GT4-based grids, investigation and download of computation results (gray boxes)

trimmed to the needs of its favorite simulation applications and submit them to different grid sites. The simulation results are accessed with the second plugin, the Simulation Explorer, without that the user knows the actual location of the data in grid.

6.1 Plugins for monitoring grid applications

The goal of these plugins is to provide a simple yet effective mean to check to what extent jobs involving specific applications can run properly one or more grid sites. Such functionality is very useful to users that have access to several grid sites and have one or more applications to run on those sites. A set of pre-defined grid jobs is made available to the application hosting the monitoring plugins. These jobs exercise the targeted application, can use grid services such as data transfer, job submission, or information services, and are submitted to the known grid sites. The plugin collects the execution results for each test job and updates the status of the respective application and services of the grid sites.

The computational scientist, as an end-user of these plugins, can create test jobs by providing job scripts in the JSDL format. The plugins implement a subset of the JSDL specification that allows for most common operations with jobs on grids. In addition to that, the framework accepts also native WS-GRAM job scripts. By offering support for the JSDL specification of the Open Grid Forum, the framework is not bound a single grid middleware or user community. Furthermore, in order to employ these plugins, the end-users and client application of different grid middleware only need to be able to phrase JSDL requests.

The incoming JSLD requests are transformed into middleware specific job requests by means of adapter plugins, as depicted in Fig. 4. The submission of the adapted jobs to the targeted grid sites is carried out by middleware-specific libraries (e.g. CREAM for gLite, CoG Kit for Globus Toolkit 4), that become available to the framework in the form of contributing plugins. The adaptor plugins contribute to the Template Job Plugin by providing implementations of the extension point of the latter one. Community libraries such as GAT, CoG Kit are packed as plugins that export high level packages.

Our approach is lightweight and flexible for several reasons. First, the contributor plugin corresponding to a grid middleware, such as the GRAM 4, encapsulates properly the client-side specifics of that middleware, here Globus Toolkit 4. The sole interaction between the plugins is done by means of the defined extension points. Second, the monitoring plugin may be used for one or more grid sites without any changes. Furthermore, the customized test jobs can be used to benchmark the grid sites from an application perspective. The end-user of the grid can write

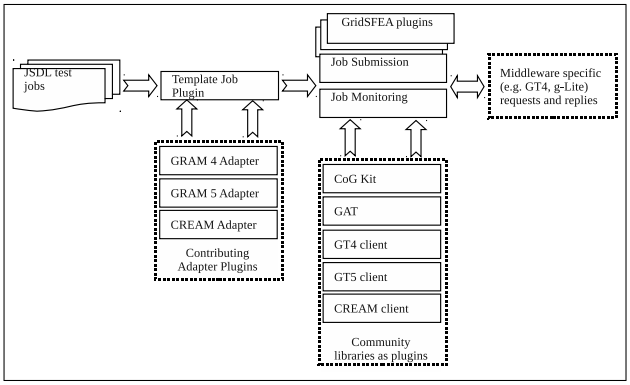

Figure 4: JSDL incoming job requests are adapted on-the-fly to middleware-specific outgoing job requests by means of plugins.

test jobs that reflect the way she uses a given application for her scenarios, the response of the monitoring plugins being thus trimmed to the user needs.

By providing access to different types of grid middleware, our plugin bundle provides client-side interoperability of grids. In a seamless way, files can be transferred between heterogeneous sites (such as from GT4 to UNICORE) and jobs can be submitted to grid sites running any of the supported grid middleware. The beneficiaries of our approach are, on the one hand, the end-users with access to multiple grid sites (for operations such as checking the existence of installed programs, running application trimmed test jobs, benchmarking sites from an application perspective). On the other hand, these plugins can be easily integrated in any Eclipse-based plugin or RCP (Rich Client Platform) application with the purpose of interacting with grid installations (such as monitoring tools).

6.2 Plugin-based exploration of simulation scenarios

A second example of usage scenario of the GridSFEA plugins contributes to the interaction of end-users with remote data located in grids. For that, a plugin for data exploration was designed based on components of GridSFEA. All data belonging to a simulation run i.e. input files, input parameters, result files, post processing results, and checkpoint files are grouped logically in a simulation scenario. Scenarios can be downloaded to the end-user space, transferred between grid sites, or used to reset/continue a simulation at the same location or at a different one.

The access to the simulation scenarios is made in a way transparent to the end user. He needs not to know the exact location of the remote data nor the way (protocol and service) to access it. With our plugin running on the user system, metadata of scenarios is retrieved from simulation services. The plugin lists all instances of a scenario, instances created during its migration for several times.

All the movements and changes of a scenario are recorded with the Simulation Services and can be displayed by tools such as the Simulation Explorer plugin (Eclipse-based plugin), the GridSFEA browser (standalone desktop application), or a thin client for mobile devices. A common feature of these tools is that the scenario data can be explored on the client side in a preview-like mode. Excerpts from scenario files can be displayed in the end-user application, as well as results of the post processing steps (images generated by VisIt, MATLAB, GNU Plot etc.).

Thus, the preview of scenarios enables the user to comfortably investigate/explore the results of the simulation without downloading the files to the end-user system. This feature gives the user the needed instrument to assist her in deciding if the results worth further investigations

or the simulation needs to be recalculated, even with changes in the input configuration of the simulation. Complete result files can also be downloaded to the end-user's machine and further explored locally with scientific visualization tools, e.g.

7 Results

The results discussed in this section have been obtained on three grid sites: one operated with GT4 (`labacal.utcluj.ro`, 48 cores), one with GT5 (`acalgrid01t.utcluj.ro`, 48 cores), and one again with GT4 (`a01.hlrb2.lrz-muenchen.de`, +8000 cores) as well. The first two sites are for the development and testing of grid applications, the third one is a production site from the DEISA infrastructure. The aim of the experiments was to show how the plugins introduced in Sec. 6 were employed to build a custom application. This evaluation was done with respect to the requirements of a simulator based on the Neocortex program [17] for investigating the dynamic properties of microcircuits of spiking neurons.

7.1 Scenario: simulation of neural microcircuits

Understanding the behavior of biological neurons, in particular of the human one, is a challenging task for the entire neuroscience community. This is an interdisciplinary area, with major contributions from biology, physics, chemistry, psychology. Since the experimental and observational means are in this particular area rather limited, the simulation becomes a very valuable instrument for gathering knowledge, along with theory and experiments. *Neurosim* is a spiking neural simulator based on the Neocortex tool, at the moment of writing still under development, having the goal to compute on HPC systems realistic models of neural microcircuits based on biological neuron models. The general aim of the investigations where *Neurosim* will be used is the evaluation of the impact the different parallelization strategies on the dynamics of neural microcircuits. In-depth descriptions of this simulation program and of the mathematical models it employs in simulations are beyond the scope of this paper. Nevertheless, we outline a few typical computational requirements of *Neurosim*: batch mode (headless operation), support for input/output files and program arguments; post-processing operations with Gnuplot; parameter-based investigations; parallel runs using hybrid parallelization techniques (MPI combined with OpenMP). The list above comprises requirements common to typical applications running on HPC systems. The execution of such applications on computing grids with HPC resources is widely supported by GridSFEA.

7.2 Custom application based on high-level plugins: GridSFEA Trotter

From the pool of plugins available with GridSFEA we have built a rich client application named here *GridSFEA Trotter*, having the kern functionalities: incoming requests for simulation jobs are formulated in JSDL, these requests are transformed into GRAM4 or GRAM5 specific job descriptions, jobs are submitted to any grid site running the middleware GT4 or GT5, results of the jobs are monitored and measured, computation results are presented to users in terms of simulation scenarios, results and post-processing files are pre-viewed remotely and further processed locally. The application comes with a set of pre-defined job templates that can be used to benchmark an application/service hosted by a grid site. These templates have been customized to the needs of *Neurosim* and have been used for describing simulations of neural microcircuits. The template jobs for Neurosim address different ways of working with this program such as performing sequential and parallel runs, staging in input files (network descriptions, initial values etc), staging out results, trigger post-processing operations. GridSFEA Trotter comprises the

high-level plugins for benchmarking grid applications and for remote exploration of simulation scenarios, and a suite of low-level plugins, such as the Template Job Plugin, GRAM 4 and GRAM 5 adaptors, File Transfer, plugins packing community libraries (CoG Kit, GT4 client, and GT5 client). By means of extension points, further adaptors can be loaded automatically at runtime by the Trotter.

7.3 Results and measurements

The goal of our experiments was to quantify the overhead introduced by our tool in comparison to the local execution of the simulation programs and to corresponding tools from the grid middleware. The job scripts perform the operations: submission of jobs without file staging, submission of jobs with file staging (in and out), and submission of jobs that require a checkpoint file handled by GridSFEA. The computation tasks are described in the JSDL format. The implementations of the JSDL specification and of its POSIX extension are partial. The suite of templates was executed on different grid sites using both our plugin-based application and the client tools of Globus Toolkit. In Table 1 is displayed the execution time of job scripts for different Linux commands contained by the templates. These commands have very small execution times, of the order of mili-seconds.

Table 1: Execution time for the JSDL job templates with GridSFEA Trotter. The size of the input and of the output file is approximately the same for one measurement.

Template	labacal[s]	acalgrid01t[s]	hlrb2[s]
/bin/ls, /bin/date, /bin/hostname	3.8	3.7	3.9
Neurosim (no processing)	6.7	6.9	7.1
Batch job	2.5	2.6	2.8

Considering the small duration of the execution of the commands employed in these tests, the results indicated in Table 1 show actually the overhead introduced by the grid middleware deployed on any of the three sites. This overhead has the order of seconds and clearly dominates the execution time of the grid job. Each site has a slightly different response time, depending on its setup. When comparing time results obtained on any of these sites, one should fairly consider the response time characteristic to the respective site. The job scripts for executing *Neurosim* require file staging operations. Table 2 shows the results related to such file operations (upload of input and checkpoint files, download of result, standard output/error, and checkpoint files). These experiments have been conducted with the GridSFEA Trotter application on the grid site labacal. The file dimensions employed here are typical for different scenarios of *Neurosim*, corresponding to neural microcircuits with different physical connections (ranging from O(1) to O (10^4) synapses for one neuron). The execution time of the simulation program has been subtracted from these time results to allow a fair comparison of the scenarios.

The time results listed in the second column of Table 2 correspond to *Neurosim* jobs where input and output files have been staged in and staged out, resp. The third column contains execution time of simulations that need to be restarted from a checkpoint and include the transfer time of that file. These values represent the total transfer time, including the contribution of GridSFEA Trotter and of the grid middleware. Results in the last column represent the time needed for the Trotter to find the actual location of a checkpoint file, information provided by the simulation services of GridSFEA. This information is extracted from the metadata about the scenario collected by the framework. The localization time must be added to the values in the third column to get the entire duration of such a job from the user perspective. The time needed

Table 2: Time measurement for executing *Neurosim* jobs with GridSFEA Trotter. The size of the input, output and checkpoint file is approximately the same for one measurement.

Scenario Size[MB]	Stage In/Out Time [s]	Checkpoint Time[s]	Localization Time[s]
0.01	6.9	5.5	9.92
0.10	5.9	5.4	9.89
1.0	5.3	5.4	9.90
10	10.2	14.1	8.72
100	53.5	98.1	9.93

to get the localization of the checkpoint does not depend on the size of the scenario (given in the first column) and is a characteristic of the framework. The job files in the GRAM 4 format corresponding to the JSDL jobs used for these measurements have been submitted to the grid with the client tool of the Globus Toolkit middleware (see Table 3).

Table 3: Time measurement for the execution of *Neurosim* jobs with client tools of GT. The size of the input, output and checkpoint file is approximately the same for one measurement.

Scenario Size[MB]	Stage In/Out Job Time [s]	Checkpoint Job Time[s]
0.01	4.4	5.5
0.10	3.9	4.6
1.0	4.2	4.8
10	6.1	9.2
100	31.3	63.8

The localization of the checkpoint has been carried out by the user, as well as the altering of the job file. Thus, these have not been considered in the measurements. We can see that the duration of the jobs is shorter than the execution of the same scenarios with GridSFEA Trotter. This is mainly due to the parallel file transfers implemented in the client tools of the Globus Toolkit. This feature is currently missing from the CoG Kit library that is employed by our application. Nevertheless, the advantages of our approach are threefold: automatic localization of checkpoint files, automatic generation of the job script that relied on checkpoints, and the use of the middleware independent job description language JSDL. The evaluation of the Simulation Explorer plugin, hosted by GridSFEA Trotter, was carried out against the tools of the GT middleware. The results were similar to the ones above.

Our conclusion reads that for small data size, below 10 MB, the overhead of our approach was small. For larger files, the download with the plugin took longer. The preview data is typically much smaller than the raw data produced by the simulation. Thus, it is expected that in real-word usage, GridSFEA Trotter will have almost the same response time as the middleware specific tools. An advantage of our approach is the integration of the simulation explorer with the rest of the plugins being capable, thus, to automatically obtain form the simulation services localization information about the scenario data, regardless where the jobs have been computed. Therefore, the investigation of the scenarios was carried out with the Trotter in a seamless way.

The performance of this approach was to some extent better than in [15], without plugins. Nevertheless, the localization time was higher here due to generally longer responses from the grid servers. The experiments conducted here showed that the plugins we provide are useful to users from the scientific computing area. We have created a sample application, GridSFEA

Trotter, which demonstrates the way our plugins are integrated and are employed for high-level operations on the grid. A strong limitation of our approach occurs when large files are involved in the operations of the end-user, the time overhead introduced by our approach being significantly high. This could be reduced by exploiting parallel file transfers, for example. A further drawback is that applications need to support the Eclipse plugin mechanism in order to host our plugins. Nevertheless, our plugin-based approach brings out-of-the-box high-level functionalities very useful to the development of grid applications.

8 Conclusions and lessons learnt

We have presented in our work on a plugin-based approach for the development of grid applications. With our approach, the realization of both high-level (focused on the grid user) and low level (oriented towards the grid) operations is possible, leading to out-of-the-box functionalities reusable in further applications. This was demonstrated here with GridSFEA Trotter. Comfortable for the grid user, the operations on grid introduce an overhead in some cases, when large data files are involved. In other cases, the performance of our approach is similar to the one of the native grid client tools, such as the ones of the Globus Toolkit, but enriches their usage within more complex scenarios such as the exploration of simulation data located on grid. Specifics of the grid middleware are packed as contributing plugins, thus, making the adoption of a further middleware a straightforward task. Moreover, the use of JSDL for formulating job requests opens the use of applications based on our plugins to scientific communities relying of different grid middleware than the Globus Toolkit. In the global context of the unification of the grid middleware, our work makes a step forward from the grid application development perspective.

Acknowledgements

This work was partly supported by National Council of Scientific Research in High Education within the PNII Human Resources program, project number 10/2009, and by the POSDRU program, financing contract POSDRU/89/1.5/S/62557.

Bibliography

[1] C. Aiftimiei, P. Andreetto, S. Bertocco S. Dalla Fina, A. Dorigo, E. Frizziero A. Gianelle, M. Marzolla, M. Mazzucato, M. Sgaravatto, S. Traldi, and L. Zangrando. Design and implementation of the gLite CREAM job. *Elsevier*, 26, 2010.

[2] G. Allen, K. Davis, T. Goodale, A. Hutanu, H. Kaiser, T. Kielmann, A. Merzky, R. van Nieuwpoort, A. Reinefeld, F. Schintke, T. Schott, E. Seidel, and B. Ullmer. The grid application toolkit: Toward generic and easy application programming interfaces for the grid. *Proceedings of the IEEE*, 93:534–550, March 2005.

[3] ARC. Advanced resource connector resources. http://www.nordugrid.org/arc, 2010.

[4] S. Burke, S. Andreozzi, and L. Field. Experiences with the GLUE information schema in the LCG/EGEE production grid. *Journal of Physics: Conference Series*, 119, 2008.

[5] E. Clayberg and R. Dan. *Eclipse: Building Commercial- Quality Plug-ins (2nd Edition).* Addison-Wesley Professional, 2006.

[6] L. Clementi, M. Rambadt, R. Menday, and J. Reetz. UNICORE deployment within the DEISA supercomputing grid infrastructure. In W. Lehner, N. Meyer, A. Streit, and C. Stewart, editors, *Euro-Par Workshops*, volume 4375 of *LNCS*, pages 264–273. Springer, 2006.

[7] EGI. Unified middleware distribution. Technical report, EGI, 2009.

[8] EGI. European Grid Infrastructure resources. http://ww.egi.eu, 2010.

[9] I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. *Intl J. Supercomputer Applications*, 11(2):115–128, 1997.

[10] gLite. Lightweight middleware for grid computing. http://glite.web.cern.ch/glite, 2010.

[11] T. Goodale, S. Jha, H. Kaiser, T. Kielmann, P. Kleijer, G. von Laszewski, C. Lee, A. Merzky, H. Rajic, and J. Shalf. SAGA: A simple api for grid applications. high-level application programming on the grid. *Computational Methods in Science and Technology*, 12(1):7–20, 2006.

[12] IGE. Initiative for Globus in Europe. http://www.ige-project.eu, 2010.

[13] G. Kandaswamy, L. Fang, Y. Huang, S. Shirasuna, S. Marru, and D. Gannon. Building web services for scientific grid applications. *IBM J. Res. Dev.*, 50(2/3):249–260, 2006.

[14] H. Kornmayer, M. Stümpert, H. Gjermundrød, and P. Wolniewicz. g-Eclipse – a contextualised framework for grid users, grid resource providers and grid application developers. In M. Bubak et al., editors, *Computational Science – ICCS 2008*, volume 5103 of *LNCS*, pages 399–408. Springer, 2008.

[15] I.L. Muntean. *Efficient Distributed Numerical Simulation on the Grid*. PhD thesis, Institut für Informatik, Technische Universität München, 2008.

[16] I.L. Muntean. Plugins for numerical simulation with GridSFEA on computing grid. In *Procs. of the 8th Intl. RoEduNet Conference Galati*, pages 51–56, 2009.

[17] R.C. Muresan and I. Ignat. The Neocortex neural simulator, a modern design. In *International Conference on Intelligent Engeneering Systems*, 2004.

[18] Open Grid Forum. Job submission description language resources. http://www.gridforum.org/documents/GFD.56.pdf, 2010.

[19] S. Parastatidis, J. Webber, P. Watson, and T. Rischbeck. WS-GAF: a framework for building grid applications using web services: Research articles. *Concurrency Computat.:Pract. Exper.*, 17(2-4):391–417, 2005.

[20] M. Riedel and et al. Interoperation of world-wide production e-science infrastructures. *Concurr. Comput.: Pract. Exper.*, 21:961–990, 2009.

[21] M. Surridge, S. Taylor, D. De Roure, and E. Zaluska. Experiences with GRIA — industrial applications on a web services grid. In *E-SCIENCE '05: Proceedings of the First International Conference on e-Science and Grid Computing*, pages 98–105, Washington, DC, USA, 2005. IEEE Computer Society.

[22] S. Venugopal, R. Buyya, and L. Winton. A grid service broker for scheduling distributed data-oriented applications on global grids. In *MGC '04: Proceedings of the 2nd workshop on Middleware for grid computing*, pages 75–80, New York, NY, USA, 2004. ACM.

[23] G. von Laszewski and M. Hategan. Workflow concepts of the Java CoG Kit. *J. Grid Computing*, 3(3-4):239–258, 2005.