

# **AutoDock gateway for user friendly execution of molecular docking simulations in cloud systems**

**Zoltán Farkas**

*MTA SZTAKI, H-1518 Budapest, Pf. 63., Hungary*

**Péter Kacsuk**

*MTA SZTAKI, H-1518 Budapest, Pf. 63., Hungary*

*University of Westminster, 115 New Cavendish Street, London W1W 6UW*

**Tamás Kiss**

*University of Westminster, 115 New Cavendish Street, London W1W 6UW*

**Péter Borsody**

*University of Westminster, 115 New Cavendish Street, London W1W 6UW*

**Ákos Hajnal**

*MTA SZTAKI, H-1518 Budapest, Pf. 63., Hungary*

**Ákos Balaskó**

*MTA SZTAKI, H-1518 Budapest, Pf. 63., Hungary*

**Krisztián Karóczkai**

*MTA SZTAKI, H-1518 Budapest, Pf. 63., Hungary*

## **List of Figures**

Figure 1: Sample WS-PGRADE workflow .....	4
Figure 2: Sample parameter sweep WS-PGRADE workflow .....	5
Figure 3: Configuration of a WS-PGRADE workflow node.....	6

Figure 4: AutoDock workflow .....	9
Figure 5: AutoDock workflow without AutoGrid .....	9
Figure 6: AutoDock Vina workflow .....	9
Figure 7: Architecture of the WS-PGRADE/gUSE and CBP integration .....	13
Figure 8: Robot certificate association .....	14
Figure 9: Multi-executable desktop grid application bundled into a self-extracting shell script .....	16
Figure 10: Template configuration .....	18
Figure 11: End user mode on the AutoDock Portal .....	19
Figure 12: Select the workflow to import .....	19
Figure 13: Configuration of a workflow in the end user mode.....	20
Figure 14: List of workflows imported .....	20
Figure 15: Workflow progress monitoring in the end user mode.....	20
Figure 16: Processing Vina inputs in 1000 jobs .....	21

## Introduction

Parameter sweep applications are frequent in scientific simulations and in other types of scientific applications. They require running the same application with a very large number of parameters and hence their execution time could take very long on a single computing resource. As a result collecting resources on demand from distributed computing infrastructures, such as clouds or grids, is a highly desired feature of any computing environment that is offered for scientists to run such applications. Cloud computing infrastructures are especially suitable for such applications due to their elasticity and easy scaling up on demand.

Molecular docking simulations are a widely utilized application area where parameter sweep scenarios are desired. Molecular docking simulation packages, for example AutoDock[1], are applied by various disciplines, such as molecular biology, computational chemistry or even psychology and require a large number of cloud resources to speed-up the computation.

In order to collect the required number of cloud resources end-users like biologists and chemists would have to learn the cloud interfaces. However, instead of learning such IT systems they would rather like to concentrate on their own scientific field and research. In order to hide this low level technology from them, high-level user interfaces like science gateways are required. WS-PGRADE[2] was designed with this idea to provide high-level, graphical workflow abstraction for users to hide the low level details of accessing the underlying cloud infrastructures. WS-PGRADE provides workflow templates that tremendously simplify the creation of parameter sweep (and other types of workflow) applications and takes care of accessing the required type and number of cloud resources. To achieve this WS-PGRADE was integrated with the CloudBroker Platform that enables accessing heterogeneous cloud resources, including Amazon EC2, IBM SmartCloud, Eucalyptus[3], OpenStack[4] and OpenNebula[5] clouds. Moreover, WS-PGRADE also supports the development of intuitive and customized end-user interfaces completely hiding the underlying complexity from the scientist.

This chapter demonstrates how parameter sweep application scenarios, such as AutoDock based molecular docking experiments on cloud computing infrastructures can be efficiently supported by the WS-PGRADE framework that is the core technology of the EU FP7 project SCI-BUS[6] aiming at developing various science gateways for a large set of different scientific user communities.

## WS-PGRADE workflows and parameter sweep application

WS-PGRADE workflows are represented as directed, acyclic graphs (DAGs), where nodes denote computational tasks (or some other workflow), and directed arcs denote data dependency between the different nodes. Figure 1 shows a workflow with 6 nodes, where different data dependencies are defined. When submitting this workflow, the first two nodes (*Copy\_A* and *Copy\_B*), as they have no prerequisites, are able to run immediately on the targeted computing infrastructure. Once *Copy\_A* has finished and produced the necessary input for the *Invert\_A* node, this job is ready for submission. Similarly, once *Copy\_B* and *Invert\_A* have both finished, WS-PGRADE can start processing the *Multi\_B* node. That is, once all the preceding nodes of a given workflow node have finished successfully, the given node is run by WS-PGRADE.

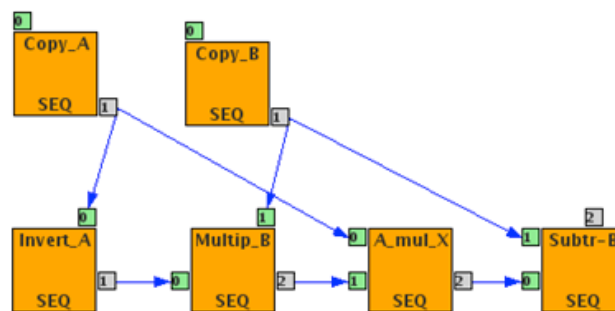
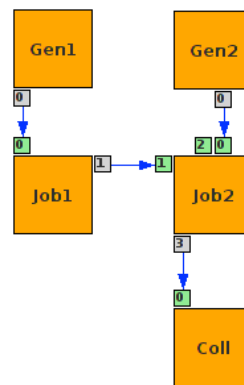


Figure 1: Sample WS-PGRADE workflow

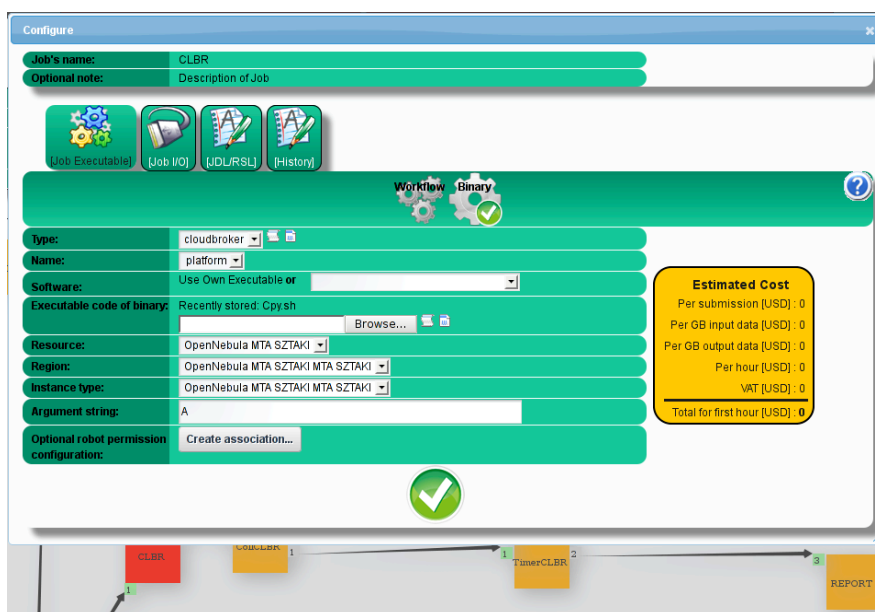
In a non-parametric workflow, every node is run only once. However, it is possible to construct parameter sweep workflows, where a node of the workflow is submitted multiple times using different input data for the different submissions. Figure 2 shows an example construct for a parameter sweep workflow.



**Figure 2: Sample parameter sweep WS-PGRADE workflow**

Figure 2 has been split into three parts: the generator phase at the top, the processing (parameter sweep) phase in the middle, and the result collecting phase at the bottom. In the generator phase WS-PGRADE runs special nodes in the workflow, the generator nodes. The task of these nodes is to produce the parameter space for the computation, for example by splitting one big input data set into smaller chunks. The generator nodes produce the different inputs for the actual computation. In the processing phase the nodes process all the inputs created by the generator nodes. If there were multiple generator nodes (as shown in Figure 2), then it is possible to specify how to pair the different inputs, using either cross or dot products of the files produced by the different generator nodes. Finally, if all of the node instances have finished in the processing phase, the collector nodes receive the results of the computations, and can process them (for example, they can search for the best result, or can produce some sort of statistics). Note that these phases can be overlapped and/or repeated within a workflow, i.e., generators and collectors can be placed into a workflow at any part of the graph without restrictions.

The configuration of workflow nodes can be performed through a simple interface, as shown in Figure 3. It is possible to define the execution resource, the application, different resource-specific settings, the command line arguments, and the data used and produced by the job.



**Figure 3: Configuration of a WS-PGRADE workflow node**

Once the configuration of the whole workflow has finished, it can be submitted. From this point on WS-PGRADE/gUSE is responsible for taking care of the nodes' execution.

### **AutoDock workflow**

This section will explain how to use the generic parameter sweep creation and execution technology described in the previous section for a concrete application, namely AutoDock. In fact, three different variants of AutoDock parameter sweep workflows have been developed and they will be described in this section.

#### ***The AutoDock application***

Traditionally, *in vitro* studies have been used to investigate the binding of receptors to their ligands and enzymes to their substrates. These wet laboratory experiments are both time consuming and expensive to carry out. An *in silico* system based on computer simulations can facilitate the modeling of receptor-ligand interactions prior to the wet laboratory experiments, and enable scientists to better focus the *in vitro* experiments using only the most promising molecules. With the advances in computer technology, it is now feasible to

screen hundreds of thousands of compounds against a single receptor to identify new inhibitors or therapeutic agents. However, the modeling programs are not user friendly, and the relationship between results obtained by molecular modeling and by *in vitro* studies and the newer biosensors, still needs to be established.

AutoDock is one example of a program which allows *in silico* modeling of intermolecular interactions. AutoDock is a suite of automated docking tools. It is designed to predict how small molecules, such as substrates or drug candidates, bind to a receptor of known 3D structure. AutoDock currently comprises of two discrete generations of software: *AutoDock 4* and *AutoDock Vina*.

AutoDock 4 is typically used to accurately model the molecular docking of a single ligand to a single receptor. In this instance the process is composed of 3 discrete stages. First a low complexity sequential pre-processing stage defines a random starting location in 3D space (termed the docking space) for both the ligand and receptor. This is achieved using a tool within AutoDockTools (ADT) called AutoGrid. The locations, which are characterised by atomic energy levels at each point within the docking space, act as a single common input to a second stage. The second stage can comprise many parallel jobs, each receiving a copy of the ligand and receptor starting locations which form the input to a genetic algorithm. The algorithm acts to randomly rotate/reposition the ligand and then determine likely docking/binding sites based upon energy levels which are calculated from the original starting locations. This process can be considered a parameter sweep, where the varied input parameter is the initial random rotation of the ligand. Finally, a single low complexity sequential post-processing stage can be used to identify the most likely binding sites by comparing energies from all jobs of the preceding stage (where minimized energies represent likely docking sites).

*AutoDock Vina* provides several enhancements over AutoDock4, increasing average

simulation accuracy whilst also being up to two orders of magnitude faster. Autodock Vina is particularly useful for virtual screening, whereby a large set of ligands can be compared for docking suitability with a single receptor. In this instance parallelism is achieved by first breaking the set of all ligands into equal sized disjoint subsets. Each compute job then uses a different subset as an input. The ligands in each subset are simulated/docked sequentially on the compute node using the single receptor, whilst a post processing stage can be used to compare the results from all compute jobs.

Researchers from the School of Life Sciences at the University of Westminster have set up a novel screening system[7] to analyze well characterized protein-ligand interactions, for example studying the interrogation of enzymes and receptors of the protozoan *Trichomonas vaginalis* (TV). TV is an important organism, with 180,000 million women affected worldwide. It is also a proven co-factor for the acquisition of HIV. Currently only one drug is available, metronidazole, and resistance has been reported. The cloning and publication of the TV genome offers new options for drug/ inhibitor detection utilizing bioinformatics and molecular modeling tools.

Westminster researchers constructed an in silico small molecule library of about 300,000 structures. Given a receptor file and the approximated position and size for the active site, the whole library was planned to be screened against the chosen receptor using the AutoDock Vina (<http://vina.scripps.edu/>) molecular docking tool. Once operational, the system could easily be utilised for other similar virtual screening experiments too.

### ***AutoDock workflows***

Three different parameter sweep workflows were developed (in the framework of the EU FP7 ER-Flow project[8]) based on the AutoDock4 and AutoDock Vina applications and the previously describe scenarios: the AutoDock workflow, the AutoDock without AutoGrid workflow, and the AutoDock Vina workflow.



The AutoDock workflow requires pdb input files (these are widely available in public databases), automatically converts these files into pdbqt format (that is required by the AutoDock application), calculates the docking space running the AutoGrid application, and docks a small ligand molecule on a larger receptor molecule structure in a Monte-Carlo simulation. Finally, it returns the required number of lowest energy level solutions. The workflow uses version 4.2.3 of the AutoDock docking simulation package. Users of this workflow are expected to provide input files for AutoGrid (molecules in pdb format), grid parameter (gpf) file, docking parameter (dpf) file, the number of simulations to be carried out and the number of required results. The workflow is shown in Figure 4. This workflow is ideal for researchers who are less familiar with the AutoDock suite and command line tools, and require high level of automation when executing their experiments.



Figure 4: AutoDock workflow

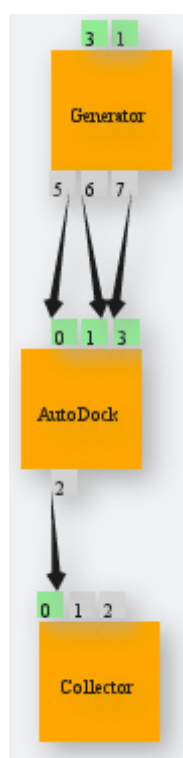


Figure 5: AutoDock workflow  
without AutoGrid



Figure 6: AutoDock Vina  
workflow

On the other hand, the AutoDock without AutoGrid workflow requires the scientist to run

scripts from the AutoGrid application on his/her own computer prior to the execution of the workflow. Although this requires specific expertise, it also gives much more flexibility to the end-user when preparing the input molecule and the docking space. As a consequence, this workflow requires pdbqt input files and the output of the AutoGrid application, and (similarly to the previous workflow) it docks a small ligand molecule on a larger receptor molecule structure in a Monte-Carlo simulation using 4.2.3 of the AutoDock docking simulation package. Users of this workflow are expected to provide a docking parameter file, a zip file containing input files for AutoDock that were generated using version 4.2.3 of AutoGrid and the former docking parameter file, the number of simulations to be carried out and the number of required results. This workflow is shown in Figure 5.

Finally, the AutoDock Vina workflow performs virtual screening of molecules using version 1.1.2 of AutoDock Vina. It docks a library of small ligands on a selected receptor molecule. Users of this workflow are expected to provide a configuration file for AutoDock Vina, an input receptor molecule in pdbqt file format, a zip file containing a number of ligands in pdbqt file format, the number of simulations to carry out and the number of required results. The workflow is shown in Figure 6.

As it can be seen in the figures, all of the workflows follow the generator node – parameter sweep node – collector node semantics. The generator nodes are executed locally, on the portal server. The parameter sweep nodes are executed in a targeted distributed computing resource, which in the first version of the gateway was the EDGeS@home volunteer desktop grid (DG). It is important to note, that for performance optimization the AutoDock and AutoDock without AutoGrid workflows are submitting one single metajob to the desktop grid server, This means that the actual parameter sweep expansion will happen on the desktop grid server, and not on the portal side (i.e. from the portal's point of view these workflows are not real parameter sweep workflows, but from the whole processing's point of view, they are, as

a number of workflow node instances will be generated on the desktop grid server). The collector nodes are executed locally, on the portal server.

The task of the generator nodes is to set up the parameter space for the middle nodes. In case of the AutoDock and AutoDock without AutoGrid workflows, this node is simply generating a metajob description for the desktop grid server, whereas in case of the AutoDock Vina workflow, the generator node distributes the ligands provided by the user in a zip file into as many packages as the number of simulations set by the user during the workflow's configuration.

The middle node of the workflows is responsible for the actual parameter sweep processing. As already mentioned, in case of the AutoDock and AutoDock without AutoGrid workflows the parameter sweep expansion happens on the desktop grid server, whereas in case of the AutoDock Vina workflow this happens on the portal server, and parameter sweep job instances will be submitted to the desktop grid server.

Finally, the task of the collector nodes is to evaluate the results of the parameter sweep executions, and collect the best dockings for the user.

### **Migrating the AutoDock workflows to the cloud**

In this section we are going to present how the AutoDock workflows have been migrated to a cloud infrastructure, including all the necessary features of WS-PGRADE/gUSE and the migration process.

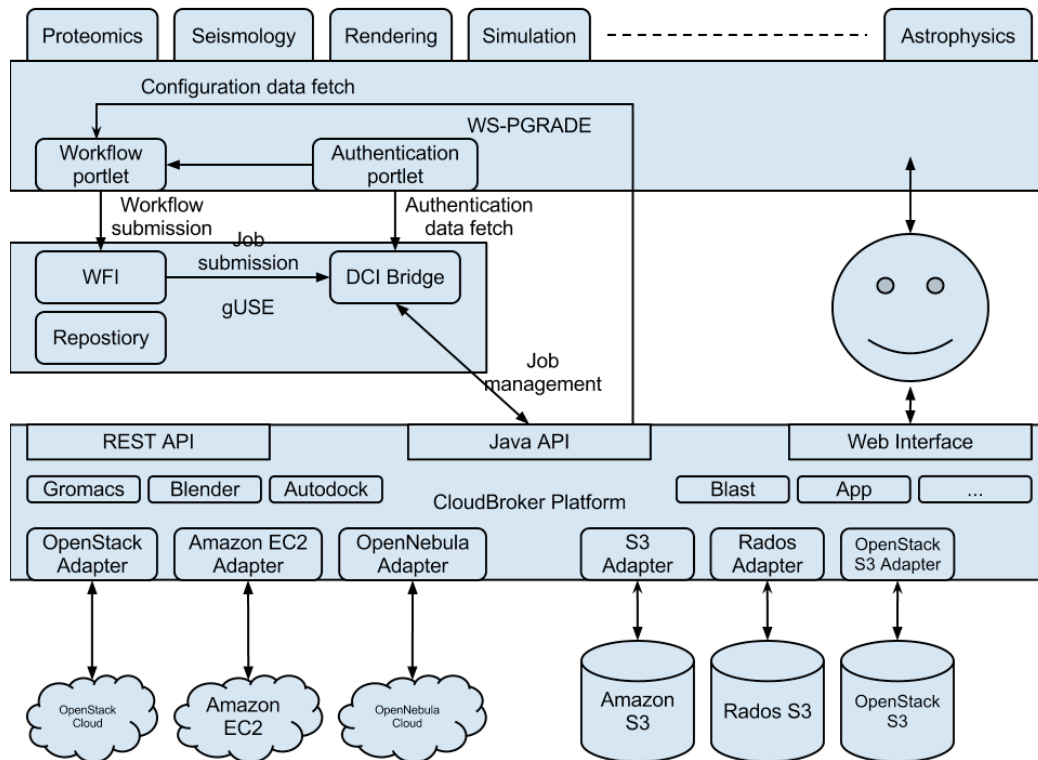
#### ***Cloud-based execution in WS-PGRADE***

WS-PGRADE offers access to a number of distributed computing infrastructures (DCIs), including clusters, grids, desktop grids and clouds. Access to cloud systems is solved with help of the CloudBroker Platform[9]. The CloudBroker Platform (CBP) offers a unified access to most major cloud infrastructures, like Amazon EC2, IBM, Eucalyptus, OpenStack

and OpenNebula at three different levels: the web interface, a RESTful web service, and a Java API. The web interface and the REST API offer access to most of the CBP functionalities, with the first one offered for users, and the second one offered for developers for integrating CBP features into their products. Finally, the Java API offers a convenient tool for accessing the majority of CBP features from Java applications. WS-PGRADE uses the Java API of CBP to access the different cloud services.

The integration of WS-PGRADE and CBP aims to hide details of the cloud infrastructure used. As shown in Figure 3, after the users have selected to use a cloud infrastructure for their workflow node, all they have to do is to select an application already deployed in the cloud (or indicate that they would like to run their own application and upload it), and select a cloud resource for the computation, for example Amazon EC2 or OpenNebula. Once the workflow has been configured and submitted, execution of the selected applications with the provided data is arranged in the background by WS-PGRADE/gUSE and the CBP.

The integration architecture of WS-PGRADE/gUSE and CBP is shown in Figure 7.



**Figure 7: Architecture of the WS-PGRADE/gUSE and CBP integration**

The top of Figure 7 represents WS-PGRADE and gUSE. Based on WS-PGRADE, a number of customized science gateways (Proteomic, Seismology, Rendering, etc.) can be created that can hide the workflow concept of WS-PGRADE through very simplified user interface. WS-PGRADE itself interacts with the CBP through two different portlets: the Workflow portlet (for creating, configuring and running workflows) and the Authentication portlet (for specifying the CBP credentials to be used by the user). The CBP credential set by the user in the Authentication portlet is used by WS-PGRADE/gUSE to communicate with the CBP service on behalf of the user (please note, that the requirement towards users to specify CBP credentials can be eliminated by assigning robot certificates to existing workflows).

Once the workflow has been configured and submitted with the help of the Workflow portlet, the set of backend components (gUSE) are responsible for arranging the workflow's execution. The Workflow Interpreter (WFI) is used to schedule nodes of the workflow for execution, and the DCI Bridge is used to actually make the different job instances run in the

selected DCI (cloud, in our case). Both WS-PGRADE and gUSE components are using the CBP Java API to access the CBP service. Once the individual job instances have been sent to the CBP by the DCI Bridge, the CBP is responsible for arranging the jobs' execution on the selected cloud service.

### ***Robot certificates in WS-PGRADE/gUSE***

As presented earlier, accessing the services of the CBP to run jobs on cloud infrastructures, assumes that the user possesses proper CBP credentials. Additionally to CBP, there are many DCIs that have such requirements (for example gLite, ARC, or UNICORE). If gateway providers would like to expose workflows for their users with nodes configured to use such infrastructures, then users will face the difficulty to manage the proper credentials for actually submitting the workflows. In order to eliminate this need, WS-PGRADE/gUSE has been extended with the robot certificate extension that enables workflow developers to assign pre-defined credentials for jobs that require some sort of authentication with the DCI they are targeted to.

Figure 8 shows the interface for setting a CBP robot credential for a workflow node set to run on a cloud infrastructure.

The screenshot displays a 'Create robot permission association' dialog box. The dialog contains the following fields and options:

- Username:** autodock\_portal@sztaki.hu
- Password:** [Masked with dots]
- Replicate settings in all Jobs:**  It is enabled only, if the Type and Grid equals for every job of the workflow.
- Buttons:** Save, Cancel

The background interface shows a configuration form for a workflow node with the following visible fields:

- Type:** cloudbroker
- Name:** platform
- Software:** Use Own Execut...
- Executable code of binary:** Recently stored: [Empty field]
- Resource:** OpenNebula MTA SZTAKI
- Region:** OpenNebula MTA SZTAKI MTA SZTAKI
- Instance type:** OpenNebula MTA SZTAKI MTA SZTAKI
- Argument string:** 10
- Optional robot permission configuration:** Create association...

On the right side of the background interface, there is a yellow box with pricing information:

- Per GB output data [USD]: 0
- Per hour [USD]: 0
- VAT [USD]: 0
- Total for first hour [USD]: 0

**Figure 8: Robot certificate association**

After the computing resource has been set, the *Create association...* button has to be clicked, and the CBP (or some other DCI-specific, depending on the target DCI) credentials had to be set in the appearing popup window. From this point on it is impossible to modify the target infrastructure and the executable of the workflow node as long as the given association is not removed.

Once a workflow with nodes set to use DCIs requiring authentication, but with proper robot certificates assigned to these nodes is exported to the local repository, users not possessing the required credentials will be able to import and actually run the workflow, with the robot certificates assigned. The usage of robot certificates will be completely hidden from the user, the only thing she will see is that the workflow can be submitted, and is running properly.

### ***Cloud-based AutoDock workflows***

We are going to present the way to migrate an existing workflow to the cloud based on the AutoDock Vina application.

The migration of the generator and collector nodes required to follow the following steps:

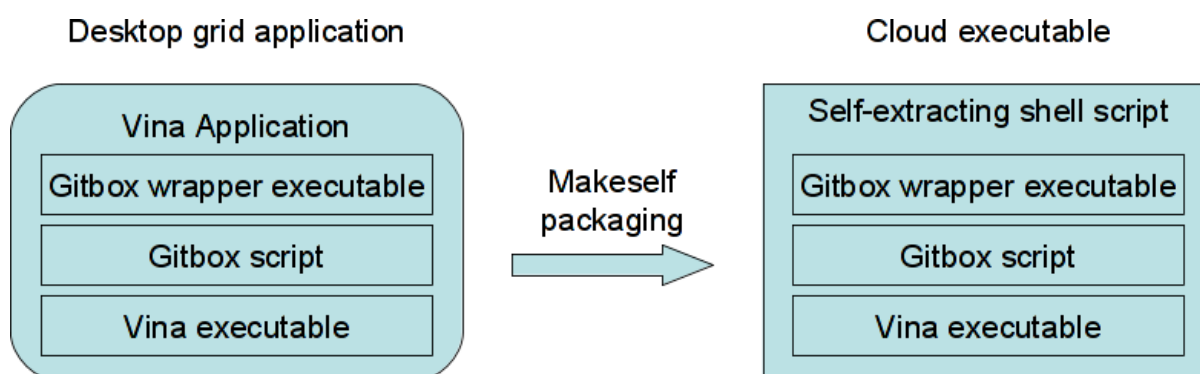
- Reconfigure the nodes to run on the cloud: for this the node's configuration window (see Figure 3) had to be opened, and the node's *type* had to be set to *cloudbroker*, the *name* to *platform*, the *resource*, *region* and *instance type* to *MTA SZTAKI*.
- Assign robot certificates to the nodes: as accessing the cloud resources requires CBP credentials, but it is not feasible to ask every end user of the customized gateway to have a valid CBP credential, a CBP robot credential had to be set for the nodes. A dedicated account for the AutoDock users in the CBP with access granted to the MTA SZTAKI OpenNebula cloud infrastructure has been created, and this account has been set during the nodes' configuration as seen in Figure 8.

The migration of the parameter sweep node was a bit more difficult, as this node of the original workflow was set to run on a volunteer desktop grid. In case of DG applications, the

executable resides on the DG server, and may consist of multiple files (one “main” executable and supporting files), like in case of the Vina DG application. The DG application is only referenced by name in the WS-PGRADE workflow’s configuration.

In order to migrate such an application to the cloud, all executable files of the DG application need to be collected from the DG server, and the workflow to be reconfigured to submit these files from the portal server. In case of a multi-file application, there are two options: either the workflow node has to be configured to run the “main” executable and additional input files representing the supporting files that have to be added to the workflow node, or a single, self-extracting archive can be created based on all the executable files, and simply this self-extracting archive should be specified as the application to run for the workflow node.

In case of the Vina application we have followed the second approach, with the help of *makeself* UNIX tool[10]. Figure 9 illustrates this multiple executable file problem and the solution. The executables belonging to the desktop grid AutoDock Vina application are bundled into a single, self-extracting shell script that can be used as the executable for the Vina node in the Vina workflow when configuring the node to run on the cloud.



**Figure 9: Multi-executable desktop grid application bundled into a self-extracting shell script**

### **AutoDock gateway**

This section will explain how to provide user friendly gateway interface for the end-user scientists who are not interested even in the workflows. They just want to use the AutoDock



workflows as black box applications, parameterize and run them in an efficient and fast way in the connected cloud systems. In the previous sections we showed how to use WS-PGRADE/gUSE to develop and configure the workflows to run in clouds. In this section we show a unique feature of our gateway technology. This is the end-user view of the WS-PGRADE gateway that can be easily created without any further programming by simply reconfiguring the gateway. In this view the end-user scientists cannot see the workflows only their parameter options through which they can specify their required input parameters. Then the gateway automatically executes the AutoDock workflows in the connected clouds in an optimal way.

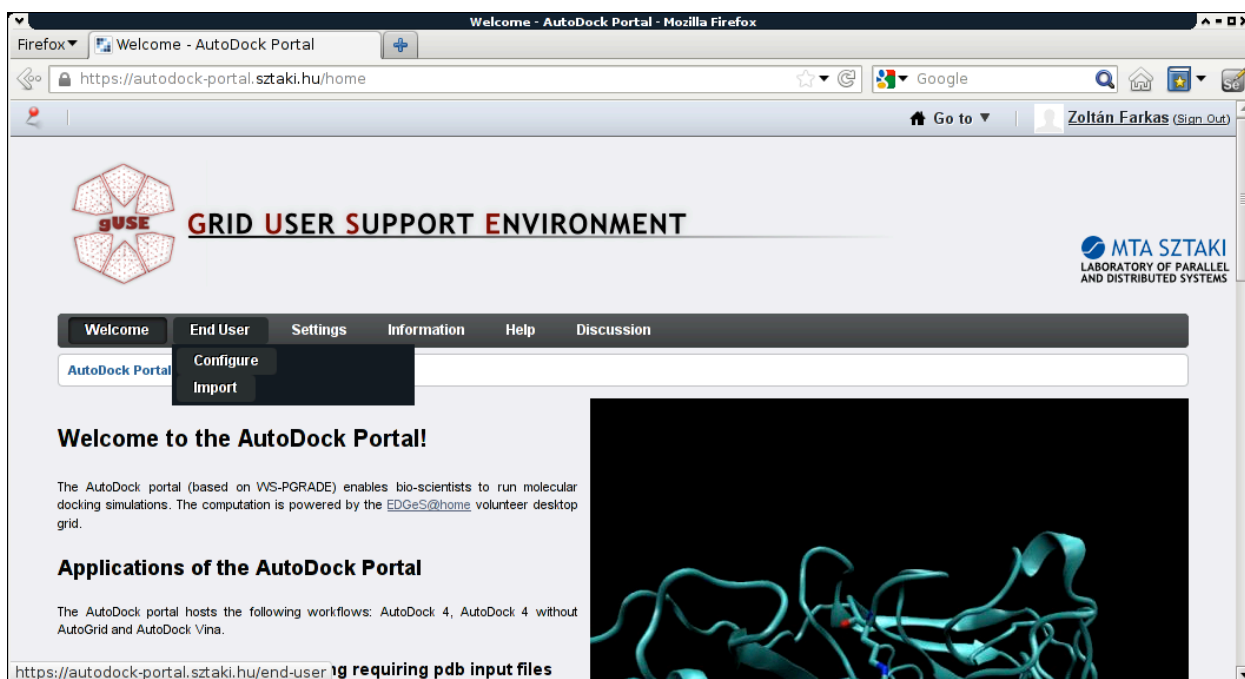
Once the AutoDock workflows are ready, it is very simple to create an end-user mode portal for them. Only two steps have to be performed: create templates of the workflows and set the end user mode as default on the portal.

The template is a special workflow type in WS-PGRADE/gUSE, containing all the properties of a workflow and restrictions on which properties can be modified and which not. Those properties that can be modified may have a short description, which can be assigned during the template's configuration. In Figure 10 we are configuring the AutoDock workflow's receptor.pdb input file (belonging to the AutoGrid job): we have set the Upload property to free, meaning that the users will be able to upload their own files to be used as receptor.pdb by the AutoGrid job. Other properties are closed, meaning the user's will not be able to modify (or even see) them. Once a template is ready, it can be exported to the internal repository, so that other users of the portal may run them.

Settings of the subsequent job with the name:	AutoGrid	Description of Job
<input type="radio"/> Close <input type="radio"/> Free	<b>Executable code of binary</b>	autodock-execute.bin
<input type="radio"/> Close <input type="radio"/> Free	<b>Job is</b>	binary
<input type="radio"/> Close <input type="radio"/> Free	[text.ctemplate.key.instancetype]	OpenNebula MTA SZTAKI MTA SZTAKI
<input type="radio"/> Close <input type="radio"/> Free	[text.ctemplate.key.software]	Wrapper 1.0
<input type="radio"/> Close <input type="radio"/> Free	[text.ctemplate.key.executable]	Wrapper 1.0 guse_wrapper.sh
<input type="radio"/> Close <input type="radio"/> Free	[text.ctemplate.key.resource]	OpenNebula MTA SZTAKI
<input type="radio"/> Close <input type="radio"/> Free	<b>Parameter of binary executable</b>	10
<input type="radio"/> Close <input type="radio"/> Free	<b>Kind of binary</b>	Sequence
<input type="radio"/> Close <input type="radio"/> Free	<b>Type of submitter</b>	cloudbroker
<input type="radio"/> Close <input type="radio"/> Free	[text.ctemplate.key.region]	OpenNebula MTA SZTAKI MTA SZTAKI
<input type="radio"/> Close <input type="radio"/> Free	<b>Grid</b>	platform
Settings of the input port named as:	receptor.pdb	Description of Port
<input type="radio"/> Close <input type="radio"/> Free	eparam	0
<input type="radio"/> Close <input type="radio"/> Free	Recently defined Input File name to <b>Upload</b>	C:/fakepath/receptor.pdb
Inherit from Job:	---	
Label:	PDB input file	
Description:	Input files for <a href="#">AutoGrid</a> (molecules in <a href="#">pdb</a> format)	
<input type="radio"/> Close <input type="radio"/> Free	<b>Internal File Name</b>	receptor.pdb
<input type="radio"/> Close <input type="radio"/> Free	<b>Dot and Cross PID</b>	0
<input type="radio"/> Close <input type="radio"/> Free	<b>Port dependent condition type</b>	0

Figure 10: Template configuration

Once all the templates are ready and have been exported into the internal repository of WS-PGRADE, the portal can be set into the end user mode. In this mode new users will receive only the End User role. The setting can be performed in Liferay's Control Panel, and the process is described in the WS-PGRADE/gUSE Admin Manual[11] in detail. Once this is set, any new user registering to the portal will be an end user, and will see only a restricted set of portlets. Figure 11 shows the portlets available for end users in case of the AutoDock portal: only workflow importing and configuration/execution is possible, workflow creation, storage browsing and other advanced features are hidden from the end users. Of course the visibility of the different portlets can be fine-tuned; this process is also described in the WS-PGRADE/gUSE Admin Manual.



**Figure 11: End user mode on the AutoDock Portal**

Execution of workflows in the end user mode is really simple: first the desired workflow has to be imported (select End User/Import in Figure 11, and selecting the desired workflow as shown in Figure 12).

Application list of selectables

Name	Notes	Exported by>Delete
<input checked="" type="radio"/> PublicAutoDock423	AutoDock 4.2.3 Application	10195
<input type="radio"/> PublicAutoDock423_noautogrid	AutoDock 4.2.3 Application without AutoGrid	10195
<input type="radio"/> Public_AutoDockVina12	AutoDock Vina 1.1.2 Application	10195

**Figure 12: Select the workflow to import**

Once the workflow is imported, then it can be configured (see End User/Configure in Figure 11), by clicking on the “Configure” button in the workflow list (see Figure 14)

After the workflow’s configuration, the workflow can be executed, and the execution can be monitored. Figure 13, Figure 14 and Figure 15 show the configuration of workflow parameters, the list of workflows imported, and the details of a workflow’s execution in the

end user mode, respectively. As it can be seen, a progress bar is presented about the workflow's execution, so users can follow their experiments' progress visually.

<b>Workflow name:</b>	PublicAutoDock423_2013-08-30-055736
<b>Note:</b>	2012-5-22
<b>receptor.pdb</b>	C:/fakepath/receptor.pdb <input type="button" value="Browse..."/>
<b>docking.gpf</b>	C:/fakepath/docking.gpf <input type="button" value="Browse..."/>
<b>docking.dpf</b>	C:/fakepath/docking.dpf <input type="button" value="Browse..."/>
<b>ligand.pdb</b>	C:/fakepath/ligand.pdb <input type="button" value="Browse..."/>
<b>Number of work units</b>	<input type="text" value="10"/>
<b>Maximum number of best results</b>	<input type="text" value="5"/>

Figure 13: Configuration of a workflow in the end user mode

Workflow name	States	Actions
PublicAutoDock423_2013-02-20-061726 2012-5-22	finished	<input type="button" value="Configure"/> <input type="button" value="Info"/> <input type="button" value="Submit"/> <input type="button" value="Delete"/> <input type="button" value="get Outputs"/>
PublicAutoDock423_2013-08-30-055736 2012-5-22	running	<input type="button" value="Details"/> <input type="button" value="Suspend all"/> <input type="button" value="get Outputs"/>
Public_AutoDockVina112_2013-04-16-115406 2012-5-22	submitted	<input type="button" value="Suspend all"/>
Public_AutoDockVina112_2013-04-17-071623 2012-5-22	error	<input type="button" value="Details"/> <input type="button" value="Configure"/> <input type="button" value="Info"/> <input type="button" value="Resume"/> <input type="button" value="Submit"/> <input type="button" value="Delete"/>

Figure 14: List of workflows imported

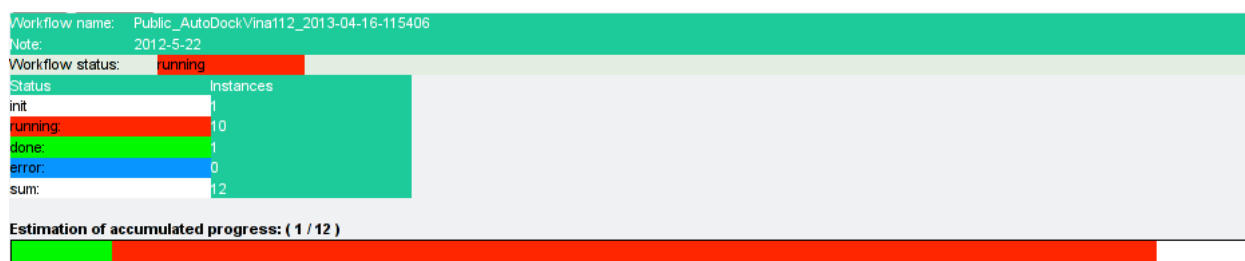


Figure 15: Workflow progress monitoring in the end user mode

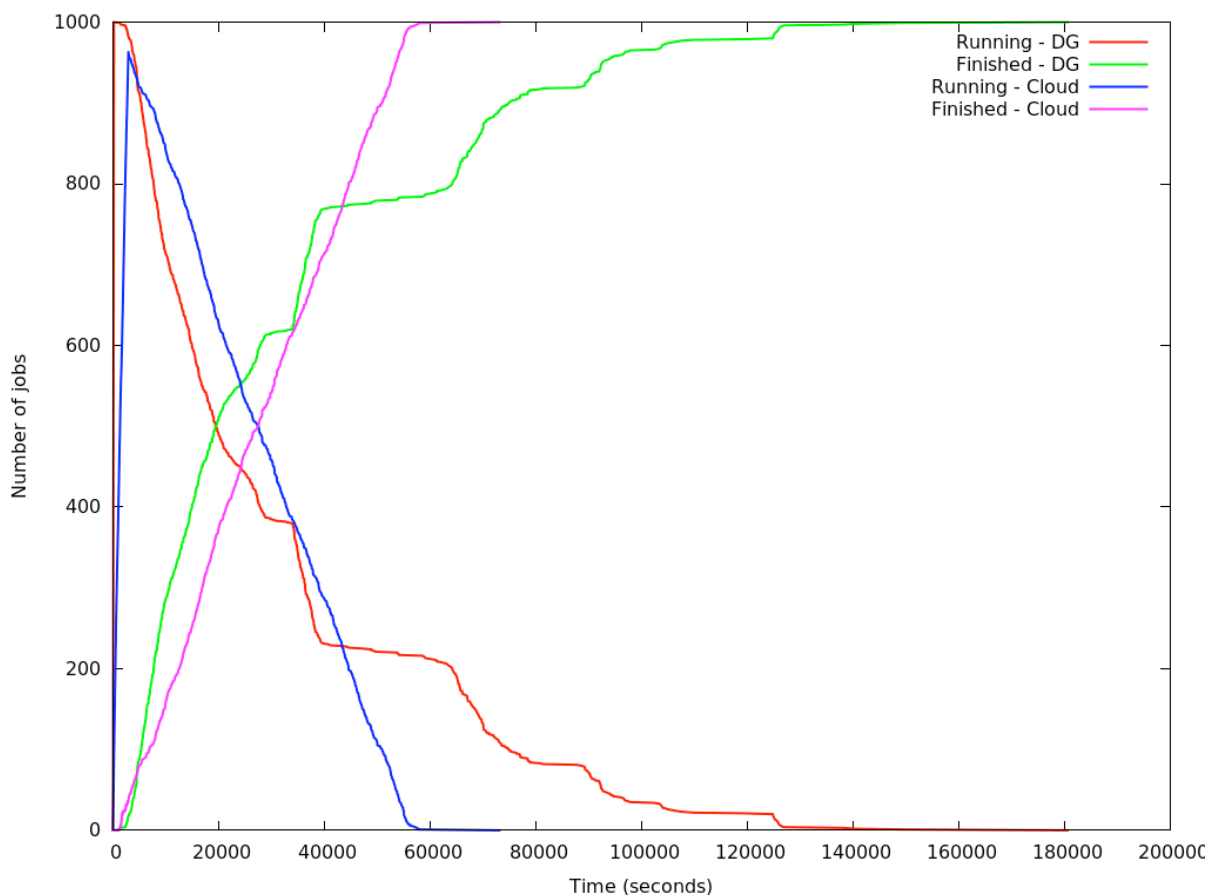
## Execution experiences, performance

This section will describe how the AutoDock workflows are actually executed on cloud systems. Performance measurements show the possible optimizations of the gateway backend mechanism and the CloudBroker Platform in order to minimize the execution time.

The selected application to perform the performance measurement was the AutoDock Vina workflow, with the input set size of 8500 ligands. The measurements have been performed by

distributing this input set among 25, 100, 250 and 1000 job. Each scenario has been executed on the desktop grid and the cloud. Although we performed the measurements in the different scenarios, we are only presenting our experiences with the 1000 job scenario, as the others show very similar results. The measurements were conducted on the SZTAKI Cloud with 25 virtual machines allocated for the experiments, and the EDGeS@home desktop grid with variable and unpredictable active clients (about 2-3 thousand at a time) available.

Figure 16 shows the case when the inputs have been split into 1000 jobs. The x axis of the figure represents the elapsed time, and the y axis represents the number of (running and finished) jobs. As it can be seen, it took more than twice as long to process the jobs on the desktop grid, as on the cloud. Both in the case of the desktop grid and the cloud execution, we can see a short “running up” period, followed by a steep processing phase, and finally, the last jobs’ processing slows down.



**Figure 16: Processing Vina inputs in 1000 jobs**

The job processing figure of the desktop grid case is a bit steeper, thus processing jobs on this DCI (with 2-3 thousand active clients) is a bit faster than in the cloud (with 25 processors). However, in the case of the volunteer desktop grid we can clearly observe the tail effect which means that the last 10-20% of the jobs require nearly as long execution time as the first 80-90% of the jobs. The tail effect is missing in the case of the cloud execution and hence the overall execution time is much shorter on the cloud. Notice that the tail effect is a well-known problem of volunteer computing and there are several ways of eliminating it [12,13]. One of the possible solutions is exactly the support of large volunteer desktop grids with relatively small dedicated clouds that run the last 10-20% of the jobs concurrently with the desktop grid. It has been presented<sup>12</sup> that with such technique the tail effect can significantly be reduced.

### **Related research**

This section compares our research with others' targeting similar objectives in distributed computing (such as grid or cloud) environments. This section clearly shows our major contributions to this field.

There are several research projects investigating how bio-molecular applications, particularly molecular docking simulations can be run on distributed computing platforms. Some examples of DCI based molecular docking simulations are detailed below. Most of these experiments are on grid computing resources with very rare exceptions currently for the utilisation of clouds.

Tantar et al [14] give an overview of current efforts how large scale parallel computing is applied to molecular simulations. The authors are also involved in the Docking@Grid project [15] that aims to define the optimal deployment architecture for grid based molecular docking simulations, and provide the accurate definition of the molecular energy surface.

The WISDOM project [16] is an international initiative to enable a virtual screening pipeline on a grid infrastructure. WISDOM was the first large scale *in silico* docking experiment on public grid infrastructure. The project has developed its own meta-middleware that utilises the EGI (European Grid Infrastructure) production infrastructure, and capable submitting and executing very large number of jobs on EGI resources. Although the WISDOM production environment is capable submitting any kind of application, the flagship application of the project is AutoDock.

Tantoso et al [17] describe a similar approach in for Globus based grids. A Web interface has been developed by the authors to execute experiments using AutoDock3.05 on target grid resources. A small workflow automates the process that includes the preparation of the receptor, creation of parameter files, calculation of grid energy, and finally the actual docking of the molecules.

Cloud-based molecular docking environments are currently hard to find in literature. The only example we know about is written by Kiss et al [18] where the authors describe the implementation of very similar molecular docking experiments on Windows Azure based clouds. However, that implementation is closely coupled with the Azure infrastructure, and the user interface is less flexible making further improvements difficult.

There are also examples for the utilization of higher level user interfaces for molecular simulations, all based on grid computing infrastructure. The Australian BioGrid portal [19] uses the DOCK [20] molecular docking software for the simulations. This work is part of the Virtual Laboratory Project that aims to utilise grid technologies for solving large-scale compute and data intensive applications in the area of molecular biology. The project uses the Nimrod Toolkit and the World Wide Grid testbed [21] to conduct the experiments.

The European Chemomentum project developed a collaborative environment based on the

UNICORE grid middleware technology to support a wide range of applications in the area of natural and life sciences [22]. Amongst other applications, the project also targeted the AutoDock docking software. The Chemomentum environment also supports the creation and execution of workflows on UNICORE resources.

## **Conclusions**

As we showed in the section on related research other AutoDock solutions are tailored to the specific grid or cloud environment. The advantage of our solution comes from its flexibility. First, it is very easy to generate various AutoDock gateway workflows for different types of users having different IT expertise and different biological simulation targets. Second, these workflows can be easily reconfigured to run in various DCIs including clusters, supercomputers, grids, desktop grids and clouds. In the current paper we showed the case when the workflows running originally on desktop grids were migrated into cloud resources. Even in the cloud environment it is extremely reconfigure the workflows to run them in various clouds like Amazon, IBM, OpenNebula and OpenStack due to the integration of WS-PGRADE/gUSE with CloudBroker Platform. This flexibility, of course, can be applied for workflows developed in other fields of science making the WS-PGRADE/gUSE gateway technology widely usable in many different areas of science and commercial activities as it is demonstrated by the various science gateways developed in the EU FP7 project SCI-BUS.

## **Acknowledgement**

The research leading to these results has been supported by the European Commission's Seventh Framework Programme (FP7/2007-2013) under grant agreement no 283481 (SCI-BUS) and 312579 (ER-Flow).

## **References**



- 
- [1] Morris, G. M., D. S. Goodsell, et al., Automated docking using a Lamarckian genetic algorithm and an empirical binding free energy function, *Journal of Computational Chemistry* 1998, **19** (14): 1639-1662
- [2] P. Kacsuk: P-GRADE portal family for grid infrastructures. In *Concurrency and Computation: Practice and Experience*, 23 (3). pp. 235-245, 2011.
- [3] <http://www.eucalyptus.com/>
- [4] <http://www.openstack.org/>
- [5] <http://opennebula.org/>
- [6] <http://www.sci-bus.eu/>
- [7] Hans Heindl, et al: High Throughput Screening for Ligands and Inhibitors of Carbohydrate Modifying Enzymes, Proceedings of the 2<sup>nd</sup> Glyco-Bioinformatics Beilstein-Institut Symposium, 27 June – 1 July, 2011, Potsdam, Germany, <http://www.beilstein-institut.de/glycobioinf2011/Proceedings/Greenwell/Greenwell.pdf>
- [8] <http://www.erflow.eu/>
- [9] <https://platform.cloudbroker.com/>
- [10] <http://megastep.org/makeself/>
- [11] [https://sourceforge.net/projects/guse/files/3.5.8/Documentation/gUSE\\_Admin\\_Manual.pdf](https://sourceforge.net/projects/guse/files/3.5.8/Documentation/gUSE_Admin_Manual.pdf)
- [12] Delamare, Simon and Fedak, Gilles and Kondo, Derrick and Lodygensky, Oleg: SpeQuloS: A QoS Service for BoT Applications Using Best Effort Distributed Computing Infrastructures. Technical report
- [13] Máté Pataki and Attila Csaba Marosi. 2013. Searching for Translated Plagiarism with the Help of Desktop Grids. *J. Grid Comput.* 11, 1 (March 2013), 149-166. DOI=10.1007/s10723-012-9224-5 <http://dx.doi.org/10.1007/s10723-012-9224-5>

- 
- [14] Tantar A-A, et al, Docking and Biomolecular Simulations on Computer Grids: Status and Trends, *Current Computer - Aided Drug Design* 2008, **4** (3): 235-249, DOI: <http://dx.doi.org/10.2174/157340908785747438>
- [15] Docking@Grid Project, <http://dockinggrid.gforge.inria.fr/index.html> [1 July 2013]
- [16] Jacq N. et al., *Grid-enabled virtual Screening against malaria, Journal of Grid Computing* 2008, **6** (1): 29-43, DOI 10.1007/s10723-007-9085-5.
- [17] Tantoso E. et al., Molecular Docking, an example of Grid enabled applications, *New Generation Computing* 2004, **22** (2): 189-190, DOI: 10.1007/BF03040958
- [18] Kiss T. et al, Large scale virtual screening experiments on Windows Azure-based cloud resources, *Concurrency and Computation*, accepted for publication, scheduled for October 2013.
- [19] Gibbins H. et al., The Australian BioGrid Portal: Empowering the Molecular Docking Research Community, Proceedings of the 3rd APAC Conference and Exhibition on Advanced Computing, Grid Applications and eResearch September 2005, Gold Coast, Australia, <http://eprints.qut.edu.au/3780/1/3780.pdf>
- [20] Ewing A. (ed.), DOCK Version 4.0 Reference Manual. University of California at San Francisco (UCSF), U.S.A., 1998, <http://www.cmp Pharm.ucsf.edu/kuntz/dock.html>
- [21] Buyya R. et al, The Virtual Laboratory: a toolset to enable distributed molecular modelling for drug design on the World-Wide Grid, *Concurrency and Computation: Practice and Experience* 2003, **15** (1): 1-25, DOI: 10.1002/cpe.704.
- [22] The Chemomentum Project, <http://www.chemomentum.org/c9m>