Enabling JChem application on grid

Miklos Kozlovszky
MTA SZTAKI
H-1111 Kende str. 13-17, Budapest, Hungary
E-mail: `m.kozlovszky@sztaki.hu`

Akos Balasko
MTA SZTAKI
H-1111 Kende str. 13-17, Budapest, Hungary
E-mail: `balasko@sztaki.hu`

Peter Kacsuk[1]
MTA SZTAKI
H-1111 Kende str. 13-17, Budapest, Hungary
E-mail: `kacsuk@sztaki.hu`

Nowadays researchers in biotechnology and at pharmaceutical industries are supported by many chemical software development platforms and desktop applications. JChem is one example of such key software platforms, which is a java based suite of integrated programs and toolkits for many cheminformatics tasks. Its components include chemical database engines, chemical structure editor and visualization tools, physicochemical property predictors and other tools for chemical structure manipulation. In collaboration with MTA SZTAKI's Application Porting Centre distributed computing infrastructure (DCI) support was added to the JChem framework to increase its processing power. Our project work addressed all issues how to combine a highly specialized chemical software development platform with a stand-alone DCI focused application development environment; namely with gUSE - grid User Support Environment. On one side the developed solution provides for JChem framework users a native application programming interface to launch time-consuming tasks transparently on the available grid and cluster infrastructure. On the other side we have opened up the gUSE with the defined generic remote access API and application specific interfaces. To demonstrate and test our solution we have created a workflow based grid application that enables the JChem software environment to do Markush searches against large datasets on gLite middleware with the help of the parameter study feature of the WS-PGRADE Portal. As a result, our solution lets JChem users to use their custom Markush searching tasks with parameter values on their local machines and to achieve significant speedup in chemical structure search by the DCI-enabled version of JChem comparing to single multi-core computers.

The International Symposium on Grids and Clouds and the Open Grid Forum
Academia Sinica, Taipei, Taiwan, March 19 - 25, 2011

[1] Speaker

1. Introduction

Chemical informatics is the one of the most relevant scientific field that contains identification of molecule structures as an important scope. As the solid structures of molecules are represented by graphs, the mathematical problem is the recognition of isomorphism of graphs, or sub-graphs. JChem is an acclaimed application developed by Chemaxon Ltd. that supports chemists to solve these kinds of problems. Since in most of the cases searching on large data set means data-, and compute-intensive challenge, solutions to decrease the computational time are highly needed. This paper shows how we have opened up the - grid User Support Environment (gUSE)[1] with the newly defined generic remote access API, how the JChem application was ported to distributed systems through a newly developed high-level programming interface that enables chemical scientist to execute JChem applications on distributed systems.

Our work was carried out mainly from the collaboration between The Grid Application Support Centre and Chemaxon Ltd. The Grid Application Support Centre (GASuC) [2] at MTA SZTAKI has already a quite long track record with successfully ported applications to DCIs. GASuC is supporting mainly large EU projects such as the EGI–InSPIRE (Integrated Sustainable Pan-European Infrastructure for Researchers in Europe) project [3], the SHIWA (SHaring Interoperable Workflows for large-scale scientific simulations on Available DCIs) project [4], and the HP-SEE (High-Performance Computing Infrastructure for South East Europe's Research Communities) project [5], where the Centre is providing knowledge transfer and development tools for DCI porting tasks. ChemAxon Ltd. provides chemical software development platforms and desktop applications for the biotechnology and pharmaceutical industries. ChemAxon's JChem application is a JAVA based cheminformatical application which supports only single computer processing (with possibility of multi-core optimization). Chemoinformatics is facing serious data processing problems recently. The used algorithms are in many cases doing searching or alignments thus crunching large amount of data received usually from structured databases. The stored amount of information in these databases have a constant growing rate, and the total processing time of a single run can last many hours or even days on a normal computer. Parallel processing on DCIs of searching and alignment problems already solved successfully in many cases. Our solution here contains a plain porting problem too, however the main achievement of our work was to define a generic remote access API for gUSE. To proof the concept as a pilot application we have targeted JChem's sequential Markush search. JChem's Markush search processing time on database (containing 1.5M molecules) takes about 24 hours, so there was a calling need to launch the same searching task in a parallel manner. To show the capabilities of the developed remote access API of gUSE, we have had to define algorithm specific workflows for the Markush search algorithm and extend JChem's programming API with a new DCI module. The developed DCI module provides a JAVA based parallel programming platform for the application developers and connects the underlying workflows with the JAVA methods. The programming API provides the programmers a transparent DCI access, and can be further extended easily with any other resource (time) consuming algorithms.

2.gUSE and WS-PGRADE

gUSE is basically a virtualization environment providing large set of high-level DCI services by which interoperation among classical service and desktop grids, clouds and clusters, unique web services and user communities can be achieved in a scalable way. gUSE has a graphical user interface, which is called WS-PGRADE. All part of gUSE is implemented as a set of Web services.

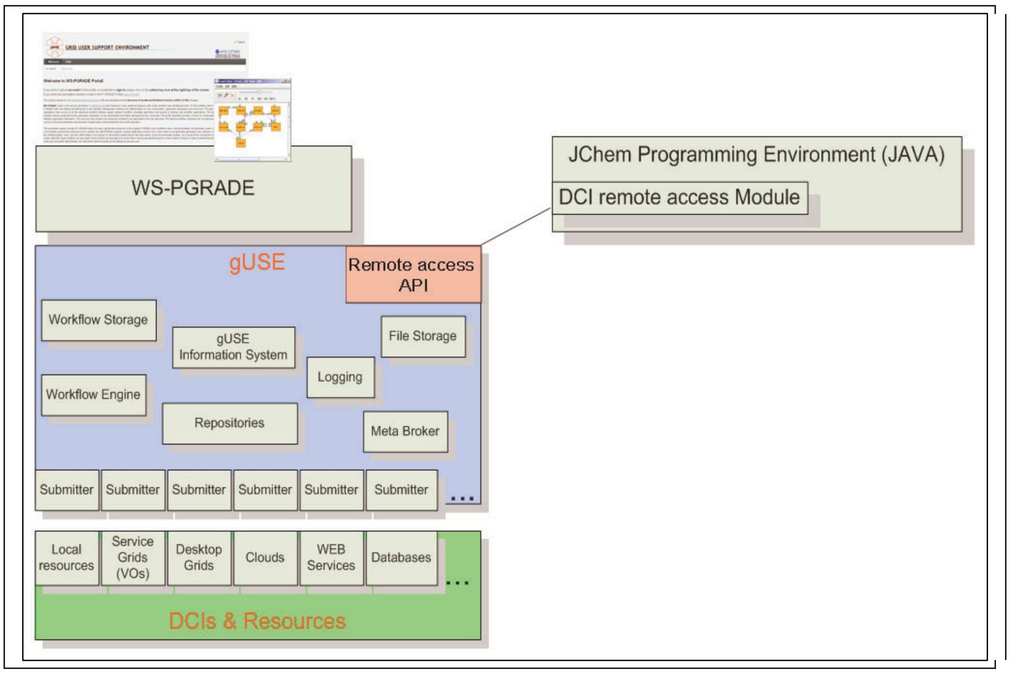

Figure 1.: WS-PGRADE and gUSE internal architecture with the remote access API

WS-PGRADE uses the client APIs of gUSE services to turn user requests into sequences of gUSE specific Web service calls. WS-PGRADE hides the communication protocols and sequences behind JSR168 compliant portlets and its GUI can be accessed via Web browsers. Before the implementation of the gUSE Remote access API, WS-PGRADE was the only interface which enabled application developers to define workflows and to submit applications into DCIs.

3.Requirements

Running cheminformatical applications is a time consuming task, so users would like to use JChem application on DCIs to shrink significantly the running time if possible. JChem end users can have access various DCIs running with different middleware, so the realization of a valuable but generic DCI submitter solution is far not trivial. Furthermore JChem is based on JAVA, which has not got generic native DCI API support yet. To give maximum support for JChem users a JAVA based solution was essential. To overcome all the issues without reinventing the wheel (or implement another DCI locked command line based solution), we have integrated the gUSE environment (which is providing seamless support for various DCIs) with the JChem environment through a JAVA API. Basically through the new JAVA based DCI

API methods in JChem, developers are able to configure and submit predefined arbitrary complex workflows with gUSE into predefined DCIs. As proof of concept the DCI support was implemented only for Markush search, however due to the generalized solution, new methods in JChem's remote DCI module are easy to be added. Important gUSE requirements were, the followings:

- Remote access API should be generic and well defined to support the integration of other frameworks.
- The developed workflows should be reusable, and located in a common repository thus developers can modify and enhance these workflows according to their needs.

In general the database handling and searching tasks should be separated or loosely coupled. This separation is one of the key requirement we have to solve distilled from the Markush search usage scenarios. The asynchronous usage of the tasks requires this separation (namely: because database update is a time consuming tasks and it should be launched on a weekly bases however the launching frequency of the searching tasks are end-user dependent and launched searching processes in several times daily can be foreseen.

3.1 JChem task execution concept

JChem provides JAVA methods to execute tasks (e.g.: Markush search against databases), and supports only single PC executions. A simple Markush searching task in JChem contains the following steps:

Step 1.: Creation of local databases (e.g.: PubChem 1.5M molecules, 4-5 GB) (required only once, after database update)

Step 2.: Searching for a specified structure

Step 3.: Getting the results

3.2 gUSE task execution concept

gUSE supports various DCIs, and its execution concept is heavily based on workflows. The definition (graph, etc.) of workflows and their jobs are stored in a local storage. Job executions on DCIs requires user level authentication, and this can be managed transparently via the WS-PGRADE.

4.Requirements

4.1 JChem's DCI Module – client side

JChem provides JAVA API, and gUSE is capable to handle only jobs and workflows, so we had to hide the underlying workflows and DCI related extra features from the JChem environment. At the client side a generic wrapper like JAVA class has been developed to access gUSE remotely. On the top of this a JChem focused JAVA API was created, which provides both generic and Markush search related methods for JChem developers.

Functions	Parameters	Return value	Descriptions
JchemSearchDCI (Constructor)	Url: url of the remote API of gUSE Password: password for authentication Application: name of the application	Created object	Creates an object that is provides information about the application, all other functions can be accessed via this object
SetQueryStructure	Query structure as a String	void	Sets the content of a local input file
SetLicenseFile	Path of the license file	void	Sets the file as an input file
UpdateDB	-	void	Submits the first generator workflow to a DCI that creates separated databases for parallel execution.
RunMarkushSearch	-	void	Submits a workflow from the client into gUSE that executes markush search functions in jChem on the separated databases
GetOutput	-	a zip file: containing the output and log files	Download the produced results of the workflow
ConvertOutput	Application Name	void	Converts the output of the application (stored by the object which provides the function) to the other one, specified in parameter

Table 1.: JChem's general DCI Module API

4.2 gUSE – server side

WS-PGRADE – gUSE's graphical user interface – was not suitable to provide server side API like communication interface and to build up a bridge between external environments and DCIs. To open up the gUSE, we have defined a remote access API, and developed the web-service-based implementation, which provides a connector service to the gUSE, and can handle job submission, job monitoring and result handling remotely. The core part of gUSE is implemented as a set of Web services so a web service based API was fitting very well in the main gUSE concept.. DCIs are requiring user level authentication, so the Remote Access API needed to support transparent certificate handling (e.g.: X509 type certificates).

4.2.1 The Remote Access API of gUSE

The gUSE Remote Access API support https/http as communication channels. Every communication is initiated by the client (the client posts data to the server). A servlet at the server side processes the incoming requests. Due to the used communication mechanism, command line solutions (curl based access wrapped in shell scripts), or a wide range of programming and scripting languages (JAVA, C, perl) can be used to realize a generic connector API at the client side. Main functions of the developed gUSE Remote Access API are the following:

Functions	Parameters	Return value	Descriptions

submit	• wfdesc: standard gUSE workflow description xml file (workflow.xml)
	• inputzip: zip file which contains the input files
	• portmapping: text file, which contains key – value pairs separated with new line, in the following format:
	if the file is an input file:
	inputfilename=WFname/JOBname/PORTnumber
	if the file is an executable:
	exename= Wfname/JOBname
	• certs.zip: zip file which contains files to authenticate to a specified grid.
	• pass: password for simpley authentication
		String:ID, which identifies the workflow.	Submits a workflow from the client into gUSE

info	• ID – the workflow runtime ID from the return value of the submit method. • pass: password for simple authentication	Workflow status: • submitted • running • finished • error • suspended • invalid	Access status information about the workflow
detailsinfo	• ID – the workflow runtime ID from the return value of the submit method. • pass: password for simple authentication	Workflow + JOB status	Access status information about the workflow's jobs.
stop	• ID – the workflow runtime ID from the return value of the submit method. • pass: password for simple authentication	• TRUE: if the abort was successful • FALSE: if an error occurred	Abort the workflow, and delete it.
download	• ID – the workflow runtime ID from the return value of the submit method. • pass: password for simple authentication	a zip file: containing the output and log files	Download the produced results of the workflow

Table 2.: Main functions of the gUSE Remote Access API

5. Parallelized Markush search concept

At the gUSE side, two separated workflows were needed for the Markush search pilot application. The developed first workflow handles the databases, updates its data and this is implemented as a generator step. The developed second workflow does database searches and it is implemented as a parameter sweep step. Basically the most important part of the concept was how we can combine the two stand-alone and separated workflows together using only the gUSE remote access API.

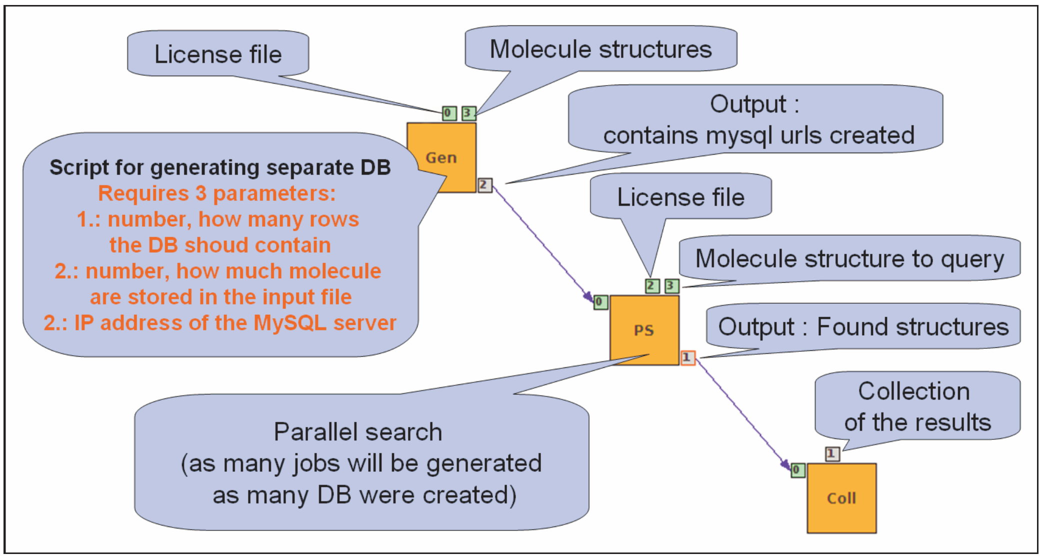
License file
Molecule structures
Output : contains mysql urls created
Gen
Script for generating separate DB
Requires 3 parameters:
1.: number, how many rows the DB shoud contain
2.: number, how much molecule are stored in the input file
2.: IP address of the MySQL server
License file
Molecule structure to query
PS
Output : Found structures
Collection of the results
Parallel search (as many jobs will be generated as many DB were created)
Coll

Figure 2.: Workflow of the parallelized Markush search

We have solved the parallelization problem of Markush search with a combination of database partitioning and the Parameter Sweep (PS) concept (shown in Figure 3.). The database was partitioned into smaller segments, these segments were distributed to different DCI nodes the searching task was launched against these database fragments at each computing nodes. At the end the results was collected and aggregated from the computing nodes. This combined solution was easy to implement with the so called parameter sweep feature of the gUSE. To explain more detailed the workflow implementation, we need to introduce here briefly the Parameter Sweep (PS) workflow concept of WS-PGRADE/gUSE.

The WS-PGRADE portal applies a DAG (directed acyclic graph) based workflow concept. In a generic workflow, nodes (shown in Figure 1 as large squares) represent jobs, which are basically batch programs to be executed on a computing element. Ports (shown here as small squares around the large ones) represent input/output files the jobs receiving or producing. Arcs between ports represent information (file) transfer operations. The basic semantics of the DAG-based workflow is that a job can be executed if and only if all of its inputs are available. This semantics is enforced by the Zen workflow manager that is used internally within the core system of the WS-PGRADE portal. Parameter Sweep (or Parameter Study) concept is useful in general, if the same task/application should be executed with different input files. Usually in such cases the task executions can be managed in a parallel manner. WS-PGRADE portal supports this kind of parallelization at a high level. The original "job" idea has been extended with Generator and Collector features to facilitate the development of PS type workflows in

WS-PGRADE portal. The Generator can generate the input files for all parallel jobs automatically, the Collector can collect all parallel outputs.

5.1.1 The Generator job – first step

The first job is in the above described Markush search workflow structure (shown in Figure 2) a Generator job, this is doing the partitioning task of the monolith database and creates small database fragments. It has two inputs the first is the database and the second is the JChem license file. The Generator receiving as command line parameters information about the required database fragmented size (maximum number of records the database fragments can contain) and about the location of the database. The job is based on normal JChem code. After the job successfully finished the references pointing to the set of database fragments are handed over to the next job, which is a Parameter Sweep type (PS) job. The Generator job needs to be launched only at first time, or at occasions when the database is appended or modified. In our final implementation we have separated this task and moved it into another loosely coupled workflow to allow users launch independently the Generator job only if needed in advance of the PS jobs.

5.1.2 Parameter Sweep job –second step

Every instance of the PS job runs on a different computing node, using the same JChem code and receives as input the query molecule description besides the database fragments and the license file. Every instance is doing a JChem based Markush search on its own database fragment in parallel. To enable independent runs, we had to modify the java.user.home environment variable for each instances. The PS job (JChem code) instances are collecting their hits in their separated result files, and at the end of the searching process these files are handled back to the gUSE server automatically by the portal as local output files.

5.1.3 Collector job – third step

The third job is basically a Collector job, which collects and merges the searching results received from the PS job instances. The output of this job is downloadable by the end user and contains the aggregated result files.

6. System requirements

The developed solution requires additional software at the client side:

- CURL, which communicates with the server side servlets.
- JChem's DCI remote access module (JAVA).

At the server side a working gUSE (v3.3 or higher) server is required with the installed Remote Access Component module.

7. Conclusions

In this paper we have described our developed software solution, which gives Distributed Computing Infrastructure (DCI) support to the JChem framework to increase its parallel processing power. Before our work grid User Support Environment (gUSE) was a closed development environment and JChem was not providing any cluster /grid/cloud support on code level. Our project work addressed all issues how to combine JChem -a highly specialized chemical software development platform- with gUSE -a stand-alone DCI focused application development environment- in a seamless way. On the client side the developed solution provides for JChem framework users a native application programming interface to launch time-consuming tasks transparently on the available grid and cluster infrastructure. On the other (server) side we have opened up the gUSE with the defined generic remote access API and application specific interfaces. To demonstrate and test our solution we have created a workflow based grid application that enables the JChem software environment to do Markush searches against large datasets on gLite middleware with the help of the parameter study feature of the gUSE's frontend (WS-PGRADE Portal). Thus even with moderate large datasets end users were able to achieve significant speedup in parallel chemical structure search with DCI-enabled version comparing to single multi-core computers. The provided solution has generic and Markush search specific parts. With the generic API methods was proved that it is easy to enable other JChem codes to run parallel on DCIs.

8. Acknowledgements

The authors would like to thank three projects supported by the European Commission's 7th Framework Programme for their financial support. This work was supported by the EGI – InSPIRE (Integrated Sustainable Pan-European Infrastructure for Researchers in Europe, under contract number: RI-261323), by the SHIWA (SHaring Interoperable Workflows for large-scale scientific simulations on Available DCIs, under contract number 261585) and by the HP-SEE (High-Performance Computing Infrastructure for South East Europe's Research Communities, under contract no. RI-261499) projects.

References

[1] Kacsuk, P. Karoczkai, K. Hermann, G. Sipos, G. Kovacs, J.; WS-PGRADE: Supporting parameter sweep applications in workflows. In: Proc. of 3rd Workshop on Workflows in Support of Large-Scale Science, In conjunction with SC 2008, Austin, TX, USA, 17 Nov. 2008, pp.1 – 10, ISBN: 978-1-4244-2827-4

[2] Accessible online at: http://www.lpds.sztaki.hu/gasuc/

[3] Accessible online : http://www.egi.eu/projects/egi-inspire/

[4] Accessible online : http://www.shiwa-workflow.eu

[5] Accessible online : http://www.hp-see.eu/