MoSGrid: Progress of Workflow driven Chemical Simulations

Georg Birkenheuer[1]*, Dirk Blunk[2], Sebastian Breuers[2], André Brinkmann[1], Gregor Fels[3], Sandra Gesing[4], Richard Grunzke[5], Sonja Herres-Pawlis[6], Oliver Kohlbacher[4], Jens Krüger[3], Ulrich Lang[7], Lars Packschies[7], Ralph Müller-Pfefferkorn[5], Patrick Schäfer[8], Johannes Schuster[1], Thomas Steinke[8], Klaus-Dieter Warzecha[7], and Martin Wewior[7]

[1]Paderborn Center for Parallel Computing, Universität Paderborn.
[2]Department für Chemie, Universität zu Köln.
[3]Department Chemie, Universität Paderborn.
[4]Zentrum für Bioinformatik, Eberhard-Karls-Universität Tübingen.
[5]Zentrum für Informationsdienste und Hochleistungsrechnen, Technische Universität Dresden.
[6]Fakultät Chemie, Technische Universität Dortmund.
[7]Regionales Rechenzentrum, Universität zu Köln.
[8]Konrad-Zuse-Institut für Informationstechnik Berlin.

ABSTRACT

Motivation: Web-based access to computational chemistry grid resources has proven to be a viable approach to simplify the use of simulation codes. The introduction of recipes allows to reuse already developed chemical workflows. By this means, workflows for recurring basic compute jobs can be provided for daily services. Nevertheless, the same platform has to be open for active workflow development by experienced users. This paper provides an overview of recent developments of the MoSGrid project on providing tools and instruments for building workflow recipes.

Contact: birke@uni-paderborn.de

1 INTRODUCTION

The BMBF funded MoSGrid project supports the computational chemistry community with an easy access to powerful compute resources. The developed MoSGrid portal[1] offers access to molecular simulation codes available on the German Grid resources. Chemical scientists are supported with instruments to handle and orchestrate complex molecular simulation methods.

The complexity of the various chemical recipes projected by the simulations are mapped to workflows. Commonly used simple workflows can be accessed and directly used by the users. A workflow editor based on WS-PGRADE allows users to develop, improve, and publish complex workflow constructs. First results are presented in [1, 2].

In this paper, we describe the recent developments for the creation of the MoSGrid infrastructure and the embedded workflow system. We start with a description of the integration of the gUSE workflow system from WS-PGRADE into the MoSGrid portal in Section 2. Parallel to the extension of WS-PGRADE and gUSE for generic workflows, MoSGrid started to implement intuitive portlets for the orchestration of specific workflows. The workflow application for Molecular Dynamics is described in Section 3, followed by the workflow application for Quantum Mechanics in Section 4. Section 5 covers the distributed data management for the workflow systems.

2 THE WORKFLOW-ENABLED GRID PORTAL IN MOSGRID

The MoSGrid portal is developed on top of WS-PGRADE [3, 4], a workflow-enabled grid portal. The chosen WS-PGRADE version is based on the open-source portal framework Liferay [5] and supports the standards JSR168 [6] and its successor JSR286 [7]. The choice of using these standards assures the sustainability of the developed portlets.

Users are enabled to create, change, invoke, and monitor workflows via WS-PGRADE, which contains a graphical workflow editor. WS-PGRADE functions as a highly flexible graphical user interface for the grid User Support Environment (gUSE). This virtualization environment provides a set of services for distributed computing infrastructures, where developers and end users can share sophisticated workflows, workflow graphs, workflow templates, and workflow applications via a repository (cf. figure 1).

gUSE contains a data-driven workflow engine and the dependencies of single steps in a workflow are represented by their connections between their input and output. The workflow engine facilitates the management of workflows, which are based on directed acyclic graphs (DAGs). DAGs allow following workflow constructs:

- Steps: A single step in the workflow describes a job with its parameters, the input and the output.

*to whom correspondence should be addressed

[1] Access to the MoSGrid portal: `http://mosgrid.de/portal`

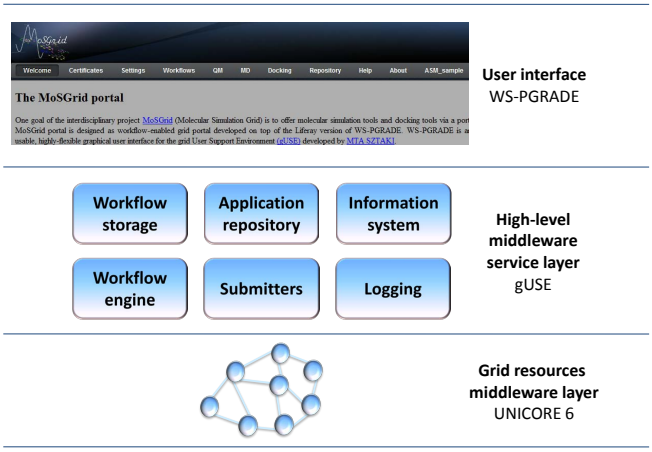

Fig. 1. The gUSE Architecture.

- Conditions: The workflow engine uses conditions to select the next step to be processed.
- Splits: A split is unconditional and delivers data for the next parallel steps.
- Joins: A join is executed after parallel steps are finished and have delivered their output.

Loops can be represented by parameter sweeps, which allow to specify varying parameters for single steps or workflows. Hence, the workflow engine invokes the same job multiple times with different parameters. Furthermore, the user can integrate generator steps and collector steps. A generator step produces multiple datasets, which are presented in the workflow graph as one output file. Hence, the input of the corresponding collector step is presented in the graph as one input file and internally the multiple files included in the input are processed by the collector step.

The workflow engine encapsulates the single steps and invokes so-called submitters (Java-based applications) for each job. Via these submitters, gUSE offers the possibility to submit jobs to grid middleware environments like Globus Toolkit and gLite, desktop grids, clouds and clusters, and unique web-services. MoSGrid extends the features of gUSE by integrating UNICORE 6 [8]. The submitter has to provide the following methods:

- actionJobSubmit: Submission of a job including authentication, authorization, and data staging
- actionJobAbort: Cancel a job
- actionJobOutput: Get the output of a job
- actionJobStatus: Query the status of a job
- actionJobResource: Return the resource, where the job was submitted to

The developed UNICORE submitter is based on the UCC (UNICORE commandline client) libraries. In contrast to programming interfaces like HiLA, the UCC libraries allow to process UNICORE workflows [9]. The adapted WS-PGRADE portal allows users to submit jobs to UNICORE [10] or to local resources. Hence, short pre-processing and/or post-processing steps of the workflow can be invoked on the gUSE server and the resource-consuming steps can be invoked in grid infrastructures.

3 WORKFLOW APPLICATION BY MOLECULAR DYNAMICS

The MoSGrid project implemented the first portlet for *Molecular Dynamic* (MD) simulations, (i) a submission interface for Gromacs portable run input (tpr) files and (ii) a molecular equilibration for protein simulations in explicit water are available. Figure 2 shows a screenshot of the portal user interface for the latter protein simulation. The design of the portlet allows an easy integration of further workflows for chemical recipes. The Gromacs simulation toolkit supports calculations with many instruments, which have to be orchestrated by a workflow system.

The UNICORE 6 infrastructure with an embedded workflow engine was already available for the implementation. Unfortunately, the use of this infrastructure was limited to the UNICORE Rich Client or the UCC. Already available APIs for accessing the UNICORE infrastructure like HiLA did not support the full workflow functionality.

In order to use the UNICORE workflow infrastructure for the MoSGrid MD portlet, we decided to implement a solution named UccAPI. The API has been implemented using the abilities of the UCC client. This strategy allows us to use well-tested code. We decided to submit even singular jobs as workflows through the MD portlet. This design decision eases the creation of the UccAPI, since only the workflow dependent commands needed to be implemented.

However, UccAPI does not need a separated UCC client, all necessary functionality is provided either embedded in UCC libraries or has been adapted, because of some parameters hidden deeply in the source-code.

Other extensions to the UCC were necessary, because it was designed to be used as stand-alone application and not as supporting library. Some of the extensions implemented the error handling.

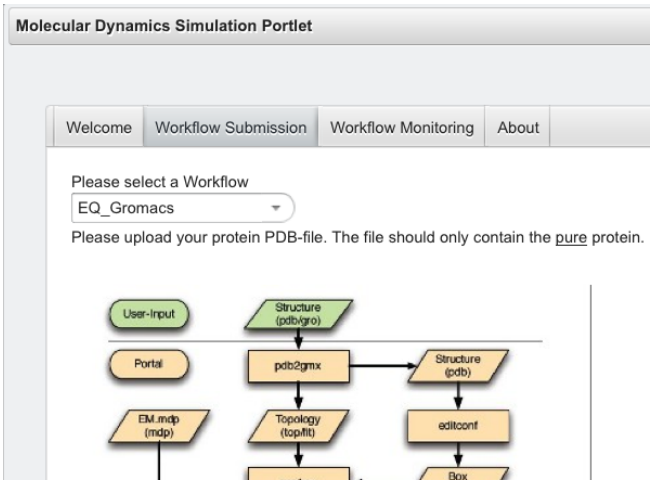

Fig. 2. A screenshot of the MD portlet.

The reason is that most errors are difficult to trace, because they occur only in complex test scenarios. Static variables were another challenge as UCC is designed to be used once for one job and not for multiple executions in parallel threads. This means that some static variables are wrongly set from the last submission or are not properly synchronized. This weakness raises the risk of errors during multiple and parallel execution.

The user of the MD portlet has most likely no UNICORE server installation available. Therefore, file uploads need to be handled separately. The upload of the files is processed as follows. Firstly, the user uploads the data for the workflow. Then, an appropriate number of compute nodes has to be chosen, the simulation length has to be set, and it has to be defined how many nanoseconds the job should simulate. At last, according to this input the configuration files were adapted.

The workflow description of UNICORE does not include a data stage-in from the client. The solution for the first prototype is to transparently define the input files in a separate job-file and transfer them by a predecessor job from the portlet to the UNICORE global storage. At a later stage of the project, an XtreemFS connection should avoid this kind of data stage-in, as described in Section 5.

The UCC API represents workflows by the use of end point references (EPRs). This is a well designed standard for service identification but does not improve readability. To avoid confusing users, it was decided to hide the EPRs and instead to show only conclusive workflow names. An exemplary name is the combination of user name, time stamp, and workflow recipe.

4 WORKFLOW APPLICATION BY QUANTUM MECHANICS

With respect to a previous user survey in the MoSGrid community, the first prototype of the *Quantum Mechanics*-Portlet [11] aimed for the implementation of Workflows for the Gaussian [12] suite, while the support for Turbomole [13], GAMESS-US [14, 15] and other relevant quantum chemical suites was shifted to a later time.

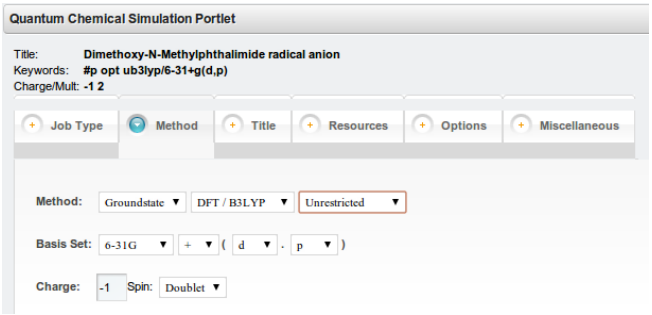

Fig. 3. A screenshot of the QM-Portlet.

Gaussian compute jobs are described in a single input file, which includes the molecular geometry and a *route card* that defines the task.

During a calculation, the progress is written to an unstructured output stream, the *log* file. In addition, calculated molecular properties are stored in a platform-dependent binary *checkpoint* file.

In contrast to its major competitors, the Gaussian program package does not allow to independently call separate executables for different tasks. Instead, a single executable parses the input file, determines the requested calculations and devises a strategy to call a series of subroutines, known as *links*. This intrinsic workflow concept, although seemingly comfortable for the end user, renders a direct override or a more granular control on the calculation level difficult.

While the *multi-step job* option of Gaussian allows the concatenation of compute jobs reusing previous results, workflows beyond this stage can only be achieved via external control, as realized in the *Quantum Mechanics*-Portlet described here.

This workflow, while at present directly implemented in the portlet, will eventually be managed through the facilities provided by WS-PGRADE.

Typically, the workflow for a quantum chemical calculation with Gaussian, as implemented in the portlet, consists of three phases, namely a pre-processing phase, the execution of the job, and a post-processing phase.

In the pre-processing phase, two basic workflows are currently available. A user may opt to upload a valid Gaussian input file previously prepared. In this workflow, no limits regarding rare keywords or exotic options exist. Less experienced users will however prefer the assistance provided by a graphical user interface in the second workflow. Here, jobs may be configured by choosing among reasonable parameters (e.g. basis sets) and tasks. Once created, these jobs are open to further editing and adjustment of parameters prior to submission.

The job execution is realized using the UNICORE command line client (UCC). A simple wrapper class encapsulates the UCC tool and emulates basic user interaction while providing the client's messages. This approach was chosen until a more sophisticated solution becomes available. This could have been the library which was developed later along with the *MD-Portlet* or, as it now is available, the framework provided by WS-PGRADE.

The results of a successful Gaussian run are retrieved and stored on the portal server. The portlet initiates different post-processing scripts.

Early attempts to process the Gaussian log file via shell scripting using tools from the Unix tool chain (e.g. *grep tr*, etc.) turned out to be tedious, ineffective and where thus soon replaced by scripts written in Python [16].

The Python scripts currently operative were written in close co-operation with colleagues from the chemical community to match their specific requirements. The total energy of each step in the course of a geometry optimisation is routinely parsed from log files. In addition to this common task in quantum chemical calculations, thermochemical data, infrared and raman absorption spectra, as well as the outcome of natural bond order (NBO) analysis are retrieved.

These results are stored to platform-independent csv files, displayed in the portlet and made available for download.

At the current stage, optimized molecular geometries can be retrieved through Python scripts from machine independent Gaussian outputs (e.g. *formatted checkpoint files*) with the use of the free Pybel [17] module, which provides Python bindings to the OpenBabel [18, 19] library.

5 DISTRIBUTED DATA MANAGEMENT IN WORKFLOWS WITH XTREEMFS AND UNICORE

5.1 Data Flow

Data management is an integral part of the MoSGrid portal. The data flow within a MoSGrid workflow originally passed four sites:

1. the WS-PGRADE portal in Tübingen,
2. the ZIH in Dresden being resource provider with the UNICORE 6 middleware,
3. frontend nodes of the D-Grid clusters and
4. compute nodes within a D-Grid cluster.

The simulation results were propagated using the reverse path. When dealing with large-scale or a large number of jobs, this introduces a lot of network traffic between clusters and the portal, which will eventually become a bottleneck.

In order to safeguard the scientific data and provide a distributed access all data is stored redundantly in the Grid file system XtreemFS [20]. Compared to the original approach, when uploading data for a simulation, XtreemFS handles the placement of replicas of the data at the according cluster instead of using less efficient data transfers via UNICORE. Additionally, the portal server does not need to store any data. However, uploads and downloads of simulation data are still realized via the portal.

In a first step the data flow using XtreemFS is as follows:

- Input data flow: A user uploads simulation input data to the Grid file system using a web interface on the portal server. The workflow engine propagates the location of these files within XtreemFS to the UNICORE 6 middleware, which then takes care of transferring the files from the frontend node to the compute nodes of a cluster.
- Output data flow: The compute nodes produce simulation results, which will be passed to XtreemFS by the UNICORE 6 middleware at the end of a simulation. Finally, the data is available for access by the user via the web interface on the portal server.

5.2 XtreemFS

XtreemFS is a distributed Grid and Cloud file system. The advantages of using XtreemFS are

- the ability to minimize data transfers especially between portal server and the grid clusters,
- the ability to manage replicated data for redundancy reasons, and
- the *Grid Security Infrastructure* (GSI) based authorization and authentication.

The requirement for deploying XtreemFS in MoSGrid is its seamless integration with UNICORE, WS-PGRADE and the software stack on D-Grid clusters.

XtreemFS is an object-based file system, i.e., file data and metadata are stored on different servers (Figure 4). The *object storage devices* (OSDs) store the contents of a file split into fixed size chunks of data (the objects). The *metadata and replica catalogs* (MRCs) contain the filename, unique file identifier, owner and directory tree, for example. XtreemFS provides packages for the most common Linux distributions and a *Filesystem in Userspace*-based (FUSE) client for seamless integration. The client acts like a local file system by translating POSIX file system calls into requests to the MRCs and OSDs. XtreemFS seems to be a large, secure and replicated local file system for its users.

XtreemFS provides custom pluggable security policies, X.509 (proxy-)certificates, Globus gridmap and UNICORE UUDB files. Furthermore, XtreemFS implements POSIX access rights and ACLs based on the *distinguished name* (DN) entries of the user X.509 certificates for authorization.

The support of GSI enables us to seamlessly integrate XtreemFS with the portal and UNICORE.

5.3 The Integration of XtreemFS in UNICORE and the Portal

Integration in WS-PGRADE: A JSR286 compliant portlet, deployed in WS-PGRADE, provides simple access to XtreemFS using a web browser. It manages the authentication and the upload of simulation input data to XtreemFS.

Integration in the UNICORE 6 middleware: XtreemFS is mounted on the frontend node of each grid cluster using the node's host certificate and the UNICORE UUDB, which is a mapping of DN entries or full public keys of certificates to local system users. On this node the *UNICORE Target System Interface* (TSI) is installed, which communicates with the batch system and makes storage available. By integrating XtreemFS with the TSI the data becomes available through UNICORE. Data transfers in the MoSGrid context are now mediated by the UNICORE 6 middleware.

6 CONCLUSIONS AND FUTURE WORK

This paper has shown an overview of recent developments of the MoSGrid project. MoSGrid has embedded chemical simulation tools for *Molecular Dynamics* and *Quantum Mechanics* in workflow recipes and allows an easy access to this instruments over the portal. With the ongoing integration of WS-PGRADE the creation

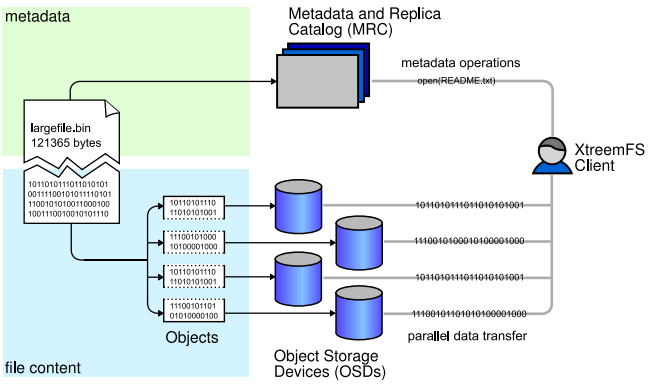

Fig. 4. The XtreemFS Architecture.

of workflows will be eased and interoperability of MoSGrids portal solution will be extended.

WS-PGRADE: At the moment, users can upload executables (e.g. shell scripts) for the configuration of single steps in WS-PGRADE. MoSGrid plans to integrate the Incarnation Database (IDB) of UNICORE to further simplify the users' interaction with the portal. The IDB contains information about the applications, which are installed on the different available UNICORE target systems. The list of available applications will be offered for selection in a drop-down menu.

MD: A portlet for the submission of *Molecular Dynamics* in MoSGrid is available. A connection to the UNICORE workflow system allows the submission of single computations or equilibration workflows. The next step of the development in MD will be to replace the UNICORE workflow environment by the gUSE/WS-PGRADE workflow management system that is interoperable with other Grid middleware or Cloud environments.

QM: In order to allow for data exchange between different computational chemistry suites, e.g. in the context of more complex workflows, a non-proprietary, highly structured data format is required. The *Chemical Markup Language* (CML) [21, 22, 23], an XML-based data format, seems a viable and promising approach.

Currently, optimized molecular geometries from machine independent Gaussian outputs (e.g. *formatted checkpoint files*) can be processed using the free Pybel [17] module. This module provides Python bindings to the OpenBabel [18, 19] library and enables to store the fundamental geometrical data in syntactically valid CML files.

The retrieval of further molecular properties, job-specific data, and their subsequent processing to CML files is under investigation.

The storage of both molecular data as well as computational recipes (i.e., workflows) in a common system-independent language and on a distributed file system, such as XtreemFS, will eventually allow for cross-domain workflows (e.g. combinations of quantum chemical calculations and *Molecular Dynamics* studies) and moreover render a sustainable data management possible.

Distributed Data Management: SAML trust delegation support for XtreemFS is currently being developed. It is planned to integrate XtreemFS and UNICORE at all participating resource providers to further minimize data traffic overhead. Support for metadata is planned, thus being able to query for simulation results. Future work includes an applet for directly uploading simulation input data to XtreemFS and the direct access from the compute nodes to XtreemFS, thus avoiding the portal and the frontend nodes for data transfer altogether.

ACKNOWLEDGEMENT

We would like to thank Bernd Schuller, István Márton, Miklos Kozlovszky, and Ákos Balaskó for the fruitful discussions and for the bug fixing for the UNICORE integration into WS-PGRADE.

Funding: This work is supported by the German Ministry of Education and Research under project grant #01IG09006 (MoSGrid) and by the European Commission FP7 Capacities Program under grant agreement nr RI-261556 (EDGI).

REFERENCES

[1]O. Niehörster, G. Birkenheuer, A. Brinkmann, B. Elsässer, D. Blunk, S. Herres-Pawlis, J. Krüger, J.Niehörster, L.Packschies, and G. Fels. Providing scientific software as a service in consideration of service level agreements. 2009.

[2]Georg Birkenheuer, Sebastian Breuers, Andé Brinkmann, Dirk Blunk, Gregor Fels, Sandra Gesing, Sonja Herres-Pawlis, Oliver Kohlbacher, Jens Krüger, and Lars Packschies. Grid-Workflows in Molecular Science. In *Proceedings of the Grid Workflow Workshop (GWW)*, February 2010.

[3]Peter Kacsuk. P-GRADE portal family for grid infrastructures. *Concurrency and Computation: Practice and Experience*, 2011. in print.

[4]Zoltan Farkas and Peter Kacsuk. P-GRADE Portal: a generic workflow system to support user communities. *Future Generation Computer Systems*, 2011. in print.

[5]Inc. Liferay. Liferay. `http://www.liferay.com`.

[6]Alejandro Abdelnur and Stefan Hepper. JSR 168: Portlet specification. `http://www.jcp.org/en/jsr/detail?id=168`, Oct 2003.

[7]M.S. Nicklous and Stefan Hepper. JSR 286: Portlet specification 2.0. `http://www.jcp.org/en/jsr/detail?id=286`, June 2008.

[8]Sandra Gesing, Istvan Marton, Georg Birkenheuer, Bernd Schuller, Richard Grunzke, Jens Krüger, Sebastian Breuers, Dirk Blunk, Georg Fels, Lars Packschies, Andre Brinkmann, Oliver Kohlbacher, and Miklos Kozlovszky. Workflow Interoperability in a Grid Portal for Molecular Simulations. In Roberto Barbera, Giuseppe Andronico, and Giuseppe La Rocca, editors, *Proceedings of the International Workshop on Science Gateways (IWSG10)*, pages 44–48. Consorzio COMETA, 2010.

[9]UNICORE Tea. High Level API for Grid Applications. `http://www.unicore.eu/community/development/hila-reference.pdf`, August 2010.

[10]A. Streit, P. Bala, A. Beck-Ratzka, K. Benedyczak, S. Bergmann, R. Breu, J. M. Daivandy, B. Demuth, A. Eifer, A. Giesler, B. Hagemeier, V. Huber S. Holl, N. Lamla, D. Mallmann, A. S. Memon, M. S. Memon, M. Rambadt, M. Riedel, M. Romberg, B. Schuller, T. Schlauch, A. Schreiber, T. Soddemann, and W. Ziegler. Unicore 6 - recent and future advancements. *JUEL-4319*, February 2010.

[11]Martin Wewior, Lars Packschies, Dirk Blunk, Daniel Wickeroth, Klaus-Dieter Warzecha, Sonja Herres-Pawlis, Sandra Gesing, Sebastian Breuers, Jens Krüger, Georg Birkenheuer, and Ulrich Lang. The MoSGrid Gaussian Portlet – Technologies for the Implementation of Portlets for Molecular Simulations. In Roberto Barbera, Giuseppe Andronico, and Giuseppe La Rocca, editors, *Proceedings of the International Workshop on Science Gateways (IWSG10)*, pages 39–43. Consorzio COMETA, 2010.

[12]M. J. Frisch, G. W. Trucks, H. B. Schlegel, G. E. Scuseria, M. A. Robb, J. R. Cheeseman, J. A. Montgomery, Jr., T. Vreven, K. N. Kudin, J. C. Burant, J. M. Millam, S. S. Iyengar, J. Tomasi, V. Barone, B. Mennucci, M. Cossi, G. Scalmani, N. Rega, G. A. Petersson, H. Nakatsuji, M. Hada, M. Ehara, K. Toyota, R. Fukuda, J. Hasegawa, M. Ishida, T. Nakajima, Y. Honda, O. Kitao, H. Nakai, M. Klene, X. Li, J. E. Knox, H. P. Hratchian, J. B. Cross, V. Bakken, C. Adamo, J. Jaramillo, R. Gomperts, R. E. Stratmann, O. Yazyev, A. J. Austin, R. Cammi, C. Pomelli, J. W. Ochterski, P. Y. Ayala, K. Morokuma, G. A. Voth, P. Salvador, J. J. Dannenberg, V. G. Zakrzewski, S. Dapprich, A. D. Daniels, M. C. Strain, O. Farkas, D. K. Malick, A. D. Rabuck, K. Raghavachari, J. B. Foresman, J. V. Ortiz, Q. Cui, A. G. Baboul, S. Clifford, J. Cioslowski, B. B. Stefanov, G. Liu, A. Liashenko, P. Piskorz, I. Komaromi, R. L. Martin, D. J. Fox, T. Keith, M. A. Al-Laham, C. Y. Peng, A. Nanayakkara, M. Challacombe, P. M. W. Gill, B. Johnson, W. Chen, M. W. Wong, C. Gonzalez, , and J. A. Pople. Gaussian 03, Revision C.02, 2004. Gaussian, Inc., Wallingford CT.

[13]TURBOMOLE V6.2 2010, a development of University of Karlsruhe and Forschungszentrum Karlsruhe GmbH, 1989-2007, TURBOMOLE GmbH, since 2007. `http://www.turbomole.com`.

[14]M. W. Schmidt, K. K. Baldridge, J.A. Boatz, S. T. Elbert, M.S. Gordon, J. H. Jensen, S. Koseki, N. Matsunaga, K. A. Nguyen, S. Su, T. L. Windus, M. Dupuis, and J.A. Montgomery. General Atomic and Molecular Electronic Structure System. *J. Comput. Chem.*, 14:1347–1363, 1993.

[15]Mark S. Gordon and Michael W. Schmidt. Advances in electronic structure theory: GAMESS a decade later. In C. E. Dykstra, G. Frenking, K. S. Kim, and G. E. Scuseria, editors, *Theory and Applications of Computational Chemistry: the first forty years*, pages 1167–1189. Elsevier, Amsterdam, 2005.

[16]The Python Language Reference. `http://docs.python.org/reference/`, 2011.

[17]Noel O'Boyle, Chris Morley, and Geoffrey Hutchison. Pybel: a Python wrapper for the OpenBabel cheminformatics toolkit. *Chemistry Central Journal*, 2(1):5–12, 2008.

[18]Rajarshi Guha, Michael T. Howard, Geoffrey R. Hutchison, Peter Murray-Rust, Henry Rzepa, Christoph Steinbeck, Jörg Wegner, and Egon L. Willighagen. The Blue Obelisk – Interoperability in Chemical Informatics. *Journal of Chemical Information and Modeling*, 46(3):991–998, 2006.

[19]Open Babel: The Open Source Chemistry Toolbox. `http://openbabel.org/`, 2011.

[20]Felix Hupfeld, Toni Cortes, Bjrn Kolbeck, Jan Stender, Erich Focht, Matthias Hess, Jesus Malo, Jonathan Marti, and Eugenio Cesario. The xtreemfs architecturea case for object-based file systems in grids. *Concurrency and Computation: Practice and Experience*, 20(17):20492060, 2008.

[21]Peter Murray-Rust and Henry S. Rzepa. Chemical Markup, XML, and the Worldwide Web. 1. Basic Principles. *Journal of Chemical Information and Computer Sciences*, 39(6):928–942, 1999.

[22]Peter Murray-Rust and Henry S. Rzepa. Chemical Markup, XML and the World-Wide Web. 2. Information Objects and the CMLDOM. *Journal of Chemical Information and Computer Sciences*, 41(5):1113–1123, 2001.

[23]Peter Murray-Rust and Henry S. Rzepa. Chemical Markup, XML, and the World Wide Web. 4. CML Schema. *Journal of Chemical Information and Computer Sciences*, 43(3):757–772, 2003.