JCTC Journal of Chemical Theory and Computation

Article

pubs.acs.org/JCTC

The MoSGrid Science Gateway – A Complete Solution for Molecular Simulations

Jens Krüger,[†,*,◆] Richard Grunzke,[‡,◆] Sandra Gesing,[§,◆] Sebastian Breuers,[∥] André Brinkmann,[⊥] Luis de la Garza,[†] Oliver Kohlbacher,[†] Martin Kruse,[∥] Wolfgang E. Nagel,[‡] Lars Packschies,[∥] Ralph Müller-Pfefferkorn,[‡] Patrick Schäfer,[#] Charlotta Schärfe,[†] Thomas Steinke,[#] Tobias Schlemmer,[▽] Klaus Dieter Warzecha,[∥] Andreas Zink,[†] and Sonja Herres-Pawlis[○]

[†]Applied Bioinformatics, University of Tübingen, Sand 14, 72076 Tübingen, Germany

[‡]Center for Information Services and High Performance Computing, Technische Universität Dresden, Zellescher Weg 12-14, 01069 Dresden, Germany

[§]Center for Research Computing, University of Notre Dame, 123 Information Technology Center, Notre Dame, Indiana 46556, United States

[∥]Regional Computing Centre, University of Cologne, Weyertal 121, 50931 Köln, Germany

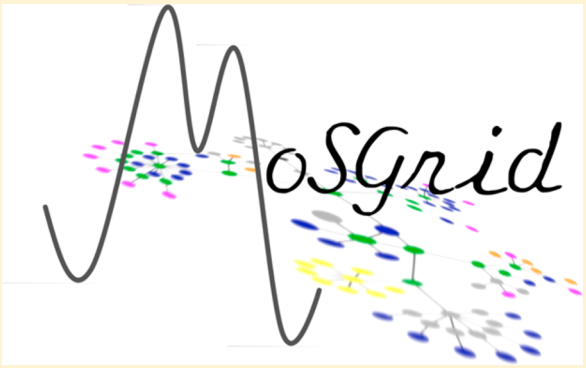
[⊥]Data Center, Johannes-Gutenberg University Mainz, Anselm-Franz-von-Bentzel-Weg 12, 55099 Mainz, Germany

[#]Distributed Algorithms and Supercomputing, Zuse Institute Berlin, Takustrasse 7, 14195 Berlin, Germany

[▽]Institute of Algebra, Technische Universität Dresden, Zellescher Weg 12-14, 01069 Dresden, Germany

[○]Department Chemie, Ludwig-Maximilians-Universität München, Butenandtstrasse 5-13, 81377 München, Germany

ABSTRACT: The MoSGrid portal offers an approach to carry out high-quality molecular simulations on distributed compute infrastructures to scientists with all kinds of background and experience levels. A user-friendly Web interface guarantees the ease-of-use of modern chemical simulation applications well established in the field. The usage of well-defined workflows annotated with metadata largely improves the reproducibility of simulations in the sense of good lab practice. The MoSGrid science gateway supports applications in the domains quantum chemistry (QC), molecular dynamics (MD), and docking. This paper presents the open-source MoSGrid architecture as well as lessons learned from its design.



INTRODUCTION

In science, computational methods have been established for years. Beginning with simple calculations and simulations, the complexity of the methods has increased tremendously as scientists are digging deeper to unveil the secrets of nature. In the fields of life sciences, materials science, and beyond, structural bioinformatics and computational chemistry have evolved to be a critical part of scientific research. Molecular dynamic methods, protein−ligand docking, and quantum chemical methods are essential and powerful techniques in the daily work of many scientists. To apply these methods, a variety of applications and software tools have been developed. Nevertheless, due to the complexity of the underlying methods their usage requires both a lot of experience and sufficient computing power. The latter can be satisfied by using Distributed Computing Infrastructures (DCIs) to gain access to appropriate computing resources like High Performance Computing (HPC) systems. The complexity of methods and resources can be circumvented by providing a user-friendly and intuitive interface to the scientists.

For the Molecular Simulation Grid (MoSGrid)[1,2] these problems have been tackled by creating a science gateway for the chemistry and materials science communities. It integrates DCIs into an e-infrastructure, which is accessible via a central portal (see Chapter Portal). Graphical user interfaces for novice and experienced users, the possibility to develop, store, reuse, and share complex workflows are some of its major computing related features.

Easy access to and the usage of the e-infrastructure is realized by a comprehensive security framework including a single-sign-on mechanism for the whole infrastructure (see Section Security). Raw and preliminary data as well as results are stored in a distributed repository, which is accessible from the science gateway (see Section Data Storage).

To ensure a structured description and representation of the produced and processed data they are specified using a Molecular

Received: February 24, 2014

   dx.doi.org/10.1021/ct500159h | J. Chem. Theory Comput. XXXX, XXX, XXX−XXX

Simulation Markup Language (MSML).[2] The metadata for this description is automatically extracted and indexed for searching (see Section Metadata).

All of these features would be in vain without the users and their applications. Thus, a major focus of MoSGrid has been community building. Currently, three domains and their applications are integrated into the science gateway: quantum chemistry with applications like Gaussian,[3] or NWChem,[4] molecular dynamics with programs like Gromacs,[5] and docking with applications like CADDSuite,[6] Autodock,[7] and FlexX[8] (see Chapter Design and Implementation as well as Chapter Applications and Results). In summary, about 200 German users of these domains are currently members of the MoSGrid community. To share its success and to widen its user base, the MoSGrid community has become part of the European projects SCI-BUS,[9] ER-flow,[10] and EDGI,[11] which address issues of science gateways, workflows, data sharing, and community building in an international context.

MoSGrid is neither the first nor the only science gateway supporting molecular simulations. Quite a few different solutions are available. WeNMR is a virtual research community bringing together structural biology and molecular dynamics.[12,13] While it is intuitive and powerful for the use of single tools, it lacks a feature to flexibly create, manage, and use workflows, which is required by the MoSGrid community. GridMACS offers a solution for GROMACS calculations relying on Globus ToolKit.[14] The architecture of GridMACS has been modularly designed allowing for the integration of further tools and infrastructures, but it does not offer a generic applicable workflow editor. MDWeb follows a similar approach but also covers Amber and NAMD while offering a guided user interface[15] and prepacked workflows. The portal offers a mature solution for Web services, whereas WS-PGRADE as the basis for the MoSGrid science gateway supports Web services and command line tools. Thus, the users can choose from a far wider range of applications in generic workflows. Dziubecki et al. created a gateway based on Vine Toolkit and UNICORE for NAMD simulations.[16] This gateway specializes on single application instead of workflows, which are a basic requirement in MoSGrid to support complex simulation chains. For Autodock a portal was created by Kiss et al. including a very intuitive result view.[17] With gUSE it uses the same basic technology, but it is focused on the Autodock application. COMPCHEM comes from the material sciences field focusing on quantum chemical calculations.[18] Access to European compute resources is provided through gLite enabling command-line access.[19] It does not support workflows nor UNICORE, which is a requirement for MoSGrid to be directly ready for HPC resources. COMPCHEM also offers the WebMO App enabling job submissions for specific tools on smartphones and a 3D editor, but it is not freely available. The Computational Chemistry Grid (CCG) is a distributed infrastructure mainly offering access to quantum chemical and molecular dynamical simulations. The interface GridChem[20] is not a portal in its closer sense but a Java desktop application integrating the software, middleware, and compute resources.[21] Even though the features are comprehensive for the domain, the workbench approach is not feasible to the goal of MoSGrid to offer a Web-based science gateway, which requires only a Web browser on the users' side and, thus, is easier to maintain. Mostly focusing on individual application domains, these solutions often represent a more specialized way of handling simulations than MoSGrid.

A common and valid criticism of scientific tool development has always been the software-centric view that often misses the scientists' needs and lacks usability.[22,23] In MoSGrid this has been addressed by integrating the user communities directly into all stages of the development process. The early provision of test environments to the users throughout the project lifetime allowed incorporating their feedback immediately during the evolution of the science gateway. This approach has led to a science gateway wholly focused on providing a productive and user-friendly working environment.

DESIGN AND IMPLEMENTATION

MoSGrid is a portal-based science gateway for molecular simulations. It can be described best as a Web-based graphical user interface for molecular simulations on distributed computing infrastructures. The portal assists with job setup, result analysis, data storage and retrieval, data visualization, user interaction, and communication. Currently, MoSGrid supports three major application domains: quantum chemistry, molecular dynamics, and docking. Applications for these domains can be accessed through portlets, specialized user interfaces implementing a unified design and user interaction concept. Users are authenticated on grid infrastructure components using assertions derived from the user's grid certificate. MoSGrid follows a design concept of high modularity and flexibility, regarding infrastructure access, user interaction, simulation software integration, and data handling. Workflows are a key component for the setup of simulations. Calculations are wrapped into configurable workflows enabling the users to customize individual tasks. For example, force fields or water models required for an MD simulation can be chosen from drop down lists. Inexperienced users, in particular, are thus supported and guided by comments and instructions during the setup of the calculation. The configured workflow can then be submitted to the grid infrastructure, and its execution can be easily monitored. Result data of workflow components is stored in a repository based on a distributed file system. Retrieval of simulation data and workflows is simplified by annotating all components with metadata. One way to achieve this is the MSML developed within this project. MSML is derived from the Chemical Markup Language (CML).[24] Job descriptions in MSML are independent of a specific program and are translated into input files for dedicated program suites. The following paragraphs provide insight into the technological details of the components of MoSGrid.

PORTAL

The MoSGrid portal offers a user-friendly interface to applications and repositories for the molecular simulation community. It is developed on top of the workflow-enabled grid portal WS-PGRADE[25] representing the highly flexible user interface of gUSE (grid User Support Environment).[26] gUSE supports various grid and cloud infrastructures, clusters, and Web services via a large set of DCI services for managing the whole life cycle of workflows and corresponding data. In general, computational workflows can be described as a sequence of connected steps in a defined order based on their control and data dependencies. Users are able to create, change, invoke, monitor, and delete workflows via graphical features in WS-PGRADE. The workflows are managed via an integrated workflow engine in gUSE, which employs so-called submitters for the job management. Each submitter is designed for the support of a specific DCI. The MoSGrid project has developed a submitter for UNICORE,[27] which provides unique features to the community. Besides the support of UNICORE jobs, it allows the use of distributed data in the cloud file system XtreemFS.[28] Furthermore, the interface of

dx.doi.org/10.1021/ct500159h | J. Chem. Theory Comput. XXXX, XXX, XXX−XXX

WS-PGRADE has been extended by an option to select a tool from an automatically generated list of tools available on the underlying UNICORE infrastructure (see Figure 1).

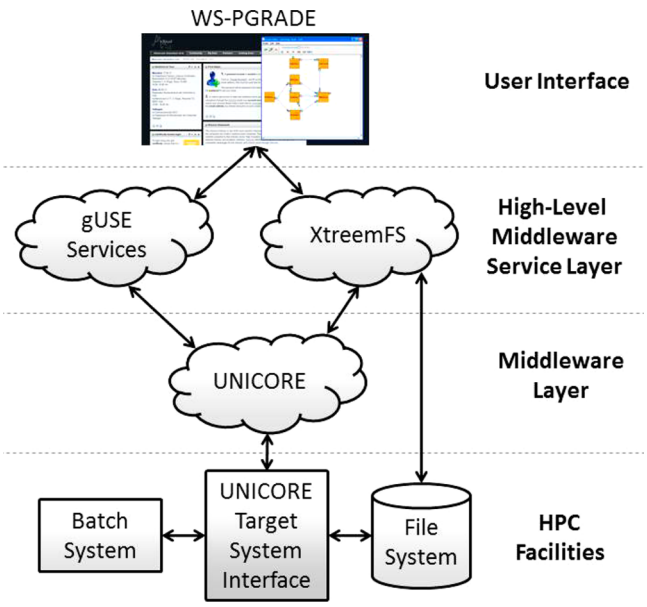


Figure 1. Underlying structure of the MoSGrid science gateway is shown. The user is shielded from the complexity while being able to use the advanced features via an intuitive and self-explanatory graphical user interface. It allows for the invocation of workflows with just a few clicks. The workflow and its contained analysis tasks are then automatically handed over to the underlying gUSE services, which in turn submit them via the UNICORE grid middleware to the HPC facilities. Here the input is made available from XtreemFS to the HPC system. Finally, the tasks are forwarded to the batch system for execution. After they are finished, the results are stored in XtreemFS, and the user is enabled to use the graphical interface to download or directly inspect the results via automatically generated plots, graphs, and 3D representations.

Additional to the generic user interface of WS-PGRADE, the MoSGrid science gateway offers domain-specific workflows and user interfaces tailored to the community's needs. For each of the application domains a specialized user interface and preconfigured workflows have been developed based on the input of the community.

SECURITY

The security features of the MoSGrid science gateway ensure the confidentiality, integrity, and availability of data considering standards for DCIs and usability aspects for the community. There are two layers of security visible to the users: the first layer is the authentication in the Web-based user interface, and the second layer regards the authentication to the underlying DCIs. Whereas no standard has evolved for cloud infrastructures yet, the security of most grid middlewares like UNICORE and Globus Toolkit[29] are based on X.509 certificates.[30]

To protect the integrity of certificates, so-called short-lived credentials have evolved, which follow the principle of single sign-on SAML (Security Assertion Markup Language)[31] assertions and GSI (Grid Security Infrastructure) proxy certificates.[32] Single sign-on allows users to authenticate just once and gain access to all connected systems without the need to manually conduct further authentication steps. The MoSGrid security infrastructure relies on the assertion concept of mutual trust, and WS-PGRADE, gUSE, and XtreemFS have been extended for its support.[1] In the MoSGrid science gateway, users obtain access to DCIs based on X.509 user certificates, and WS-PGRADE offers an intuitive user interface to manage SAML assertions for a given user certificate. Sensitive user certificates remain on the users' computers since the creation process is solved via a signed applet. Applets allow for processing data without involving the science gateway server, and signed applets additionally ensure that their origin can be proven by users. The applet has been improved by MoSGrid for the science gateway based on the feedback from the community. Thus, users only have to fill in a few mandatory fields like the location of the user certificate and the duration of the validity of the SAML assertion. Furthermore, in each of the stages, the interface presents users with only the relevant options suitable for a given step, e.g., only if a SAML assertion is uploaded to the server, its state can be browsed. Authentication into the science gateway has been implemented to be easy to use. Users are permitted to authenticate via a login-and-password mechanism, but the default configuration offers to automatically login via a certificate imported in the browser without further interaction.

DATA STORAGE

The data storage concept of the MoSGrid science gateway is based technologically on gUSE, XtreemFS, and UNICORE. MSML, as a well-structured and universal representation, serves as the central hub for the data and metadata storage. It consists of raw and preliminary simulation data as well as metadata. Furthermore, the workflow settings, the simulation input parameters, and chemical structures are stored. The data management in MoSGrid uses MSML for transfers, conversions, and analysis steps.

DISTRIBUTED FILE SYSTEM

The preliminary data and results from chemical simulations have to be stored in a persistent and reliable way, given that a single simulation can already produce vast amounts of data. This enables reproduction and reusability of experimental results of comprehensive simulations. For this purpose, XtreemFS was employed. It is a distributed file system in which the index and the raw data of the files are stored on different servers for scalability. The Object Storage Devices (OSDs) store the raw data of a file. The Metadata and Replica Catalogues (MRCs) store file related indices and a mapping of files to OSDs. The Directory Service (DIR) registers services and enables lookups by client components which provide POSIX semantics[33] for file system accesses. The client component translates file system calls to remote procedure calls, which are sent to the corresponding server. When accessing a file, its related information is retrieved from the MRC, and raw data is directly transferred from or to the OSDs. The authorization and authentication in XtreemFS is secured by the use of assertions based on grid user certificates within the MRC. XtreemFS was integrated in WS-PGRADE, gUSE, and UNICORE. At the beginning of each simulation, input data is loaded from XtreemFS. At the end of the simulation the results are written back to XtreemFS. This is automatically executed by UNICORE and the UNICORE Submitter gUSE component without any interaction from the user. Access to the file system is provided by the use of portlets in the MoSGrid science gateway. An XtreemFS installation providing 96 TiByte storage was deployed for its use in MoSGrid.

dx.doi.org/10.1021/ct500159h | J. Chem. Theory Comput. XXXX, XXX, XXX−XXX

SIMULATION REPOSITORY

The MoSGrid simulation repository was realized using gUSE and the UNICORE Metadata Service. UNICORE is a grid-middleware, which provides secure access to grid-resources. Additional features like the extended data and metadata management were added in the latest versions. The UNICORE Metadata Service is based on the open-source project Apache Lucene[34] and is used in MoSGrid for crawling metadata, indexing it, and providing a search interface to the metadata. The gUSE service provides a set of functionalities for the management of workflows. Molecular simulation recipes consist of a generic description of the simulation codes as well as the required input parameters and files. These recipes are stored in gUSE as workflows. A user chooses a predefined workflow from within the domain-specific portlets and specifies necessary input parameters and files. These are translated to an MSML-description. In the progress of a simulation the MSML description is extended by simulation results. The indexing and retrieval of this MSML data is done within the MoSGrid simulation repository. It allows for the annotation and indexing of metadata and searching it via a powerful search interface. To make the metadata in MSML format available to the repository, the MSML file is converted to JSON and stored in XtreemFS as a ".metadata"-file. This file is automatically indexed at the end of a workflow.

MOLECULAR STRUCTURE REPOSITORY

The molecular structure repository stores molecular structures from simulations using the CML format, which forms the basis of MSML. CML provides topology and geometry of the molecular structures under consideration and is thus embedded in MSML. We have implemented a parser to convert molecular structures to MSML from other structural formats. This parser currently supports the conversion of PDB and SDF to MSML. PDB support was developed based on BioJAVA,[35] whereas SDF/MOL support was realized using the Chemistry Development Kit (CDK).[36] As part of a typical workflow, a user provides a chemical structure as input in a PDB or SDF format. This file is converted to MSML by the parser and integrated with the MSML description of the job. The persistent storage of the MSML-files is handled by XtreemFS. Since the MSML-files are stored within the MSML description of the simulations, they are available for information retrieval by the UNICORE Metadata Service.

METADATA

The term metadata is used in this paper in the sense of everything related to a particular simulation that ever could be useful for a scientist. The concept can be best illustrated using the example of a small MD simulation. What molecule(s) are simulated? Which solvent and force field are used? What electrostatics, van der Waals, temperature, and pressure settings were employed? Which applications were used to prepare the molecular system and which one was used for the actual simulation? What computational resources, for example the number of cores and memory, were employed? Which output structure(s) and/or trajectories were generated? What does the potential energy over the course of simulation look like?

This quite general view can be separated into multiple levels. Within MoSGrid a hierarchical XML description named MSML was introduced covering all aspects of a particular simulation, ensuring reproducibility in the best sense of good lab practice.

The following chapter describes MSML as the central MoSGrid metadata format. It follows a description of the so-called Templatedesigner, which allows the easy creation of MSML templates. Afterward, the Generic Parser is described, which extracts information from application specific output files to fill in data in MSML files. Finally the use of UNICORE, to make metadata searchable, and the search functionality are explained.

MSML

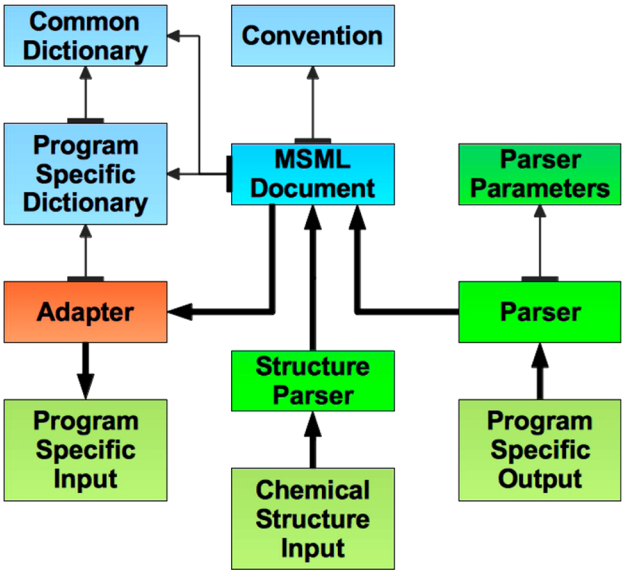
MoSGrid supports multiple simulation codes each of which produces its own output and requires specific input. To describe a complete simulation and its results independently from the actual simulation code, a common data format was introduced. This format is called MSML (Molecular Simulation Markup Language). It serves as an easy way to access all information for metadata extraction and as an intermediate format used by MoSGrid tools like the Portlet API, Templatedesigner, and the Generic Parser. MSML is a derivative of the established XML dialect called CML (Chemical Markup Language)[24] and is therefore fully compatible with CML. MSML consists of three components: the schema, conventions, and dictionaries. The schema defines which XML elements and attributes may be used in which combination. The conventions are a modular way to describe what components an XML element may, must, or must not contain. For instance, the compchem convention defines that the element *module* with the attribute *dictRef* set to *joblist* must contain at least one element of *module* with the attribute *dictRef* set to *job*. In short: One rule of the compchem convention is that a *joblist*, which describes a workflow, must contain at least one *job*. The dictionaries can be used to define the meaning of an element in MSML. For instance two different simulation codes need a number of iterations for a specific calculation. One tool uses a numeric value and the other permits only the values *few* and *many*. Two dictionaries for each simulation code have to be set up with two entries of the same identifier (e.g., *iterations*). Depending on which dictionary is referenced within the MSML template describing a certain workflow, portlets and other tools developed for MoSGrid know whether to allow numeric values or only the mentioned keywords. The original CML schema has been extended for MSML to fit the needs of computational simulations. Due to a stricter schema there was no need to set up or modify a convention for MSML. Dictionaries specific to the simulation codes supported by MoSGrid have been created (see Figure 2).

The finalization section of those MSML documents applies only to jobs which produce output that is relevant for metadata extraction. This section is created by a parser, which uses the parser configuration sections to identify the information on how to extract data from simulation code output. The adapter configuration specifies which method is used to transform the parameters from the parameter list in the initialization section to simulation code specific input. The following sections explain further the usage within MoSGrid.

TEMPLATEDESIGNER AND PARSER

MSML serves two main purposes. With MSML the so-called adapters generate simulation code specific input files and command line parameters. The domain portlets that will be introduced in later sections offer the possibility for the end user to specify parameters for a concrete simulation (e.g., maximum number of iterations, basis set, and maximum amount of memory).

dx.doi.org/10.1021/ct500159h | J. Chem. Theory Comput. XXXX, XXX, XXX−XXX




```
<?xml version="1.0" encoding="UTF-8"?>
<cml convention="convention:compchem">
        <module dictRef="compchem:jobList">
                <cmlx:parserConfiguration/>
                <module dictRef="compchem:job" id="Job1">
                        <module dictRef="compchem:environment">
                                <propertyList/>
                        </module>
                        <module dictRef="compchem:initialization">
                                <parameterList/>
                                <cmlx:adapterConfiguration/>
                                <cmlx:parserConfiguration/>
                        </module>
                        <module dictRef="compchem:finalization">
                                <propertylist/>
                        </module>
                </module>
                <module dictRef="compchem:job" id="Job2"/>
                ...
        </module>
</cml>
```

Figure 2. On the left side the relation between MSML documents and all individual components is depicted, while on the right side a simplified example of such an MSML template is shown. There is the root element *cml* which contains exactly one *joblist*. The *joblist* wraps one or more jobs and may have a *parserConfiguration* element. Each job is divided in one *environment* section, an *initialization* section, and a *finalization* section. In the *environment* section parameters for memory, CPU-count, node-count, and walltime are specified. The *initialization* section contains one or no *parameterlist*, which wraps one or more parameter of type matrix, array, or scalar. Additionally there is an *adapterConfiguration* section and another *parserConfiguration* each of which may contain one *parameterlist*. The *finalization* section contains the results of a job stored as properties within the *propertylist*.

Those values are stored in an instance of an MSML template explained above. The templates are set up by consortium users specifically for a workflow. They define which values must be supplied by the end user so that a workflow can be executed. They also contain default values that the end user may or may not change. Due to the complex and modular structure of MSML, standard XML editors proved to be unreliable as those editors cannot cope with the dictionary and convention mechanisms. Therefore, a portlet called Templatedesigner has been implemented to help consortium users creating templates. This portlet helped reducing typos and semantic errors in the MSML files significantly.

A secondary yet important purpose of MSML is to provide a standardized format to store information from a simulation. The output files from simulation codes strongly vary, and therefore the tool Generic Parser has been implemented. The Generic Parser helps consortium users to define a series of regular expressions. Those regular expressions are used as element separators, replacements, and extractors for strings within output files to extract information. Those regular expressions are stored in so-called "parser configuration files". When the generic parser is executed to extract information from output files, it loads the configuration files, which are specified in the MSML template within the parser configuration section of a job. The parser then adds a finalization block (containing the extracted data) to the job within the MSML template. The generic parser is executed as part of the workflow on grid resources as the parsing process can be very time-consuming.

INDEXING AND SEARCH

After an MSML file is created and stored in the MoSGrid repository at the end of a workflow execution, it is used to make the metadata searchable by the user. For this purpose the MSML file is transformed into a JSON[37] representation, which is stored alongside the MSML file. After a molecular simulation has finished the gUSE UNICORE submitter, developed by MoSGrid developers, instructs the UNICORE middleware to index the content of the JSON file. Afterward the metadata is available in an index managed by Apache Lucene. A search interface was added to provide an easy to use and uniform way to search over metadata stored in Lucene. It transparently issues commands to UNICORE to access Lucene, and it returns the search results to the user. Looking for e.g. a specific basis set the user just enters "6-31G". This will return links to fitting simulations. The user can select input or output files for further purposes.

PORTLET API

A portal Web site is usually composed of so-called portlets. These are pluggable software components, which can implement any kind of service. The portlet's main application logic runs on server-side while its graphical user interface (GUI) runs in a Web browser on client side. MoSGrid is based on the Liferay portal platform,[38] which by default comes with Vaadin.[39] Vaadin is a popular portlet framework, which lets JAVA developers create portlets similar to desktop applications. A large library of GUI components is available, and also the communication between client- and server-side code is handled. Each domain of MoSGrid offers its own portlet, which enables the execution of molecular simulations in the corresponding field. Despite the algorithmic divergence of these domains, the requirements for the portlets are almost identical. Thus, all domain specific portlets are built on a common Portlet API. This has several advantages for developers but also for users. In short, the API provides the most important services, e.g. gUSE workflow management, XtreemFS access, or MSML handling, but also defines the basic portlet usage and the GUI. It is still possible to customize portlets in various ways by overwriting the default implementation in one of many extension hooks. Using a domain specific portlet typically involves three workflow operations: Import, Submission, and Monitoring. The default portlet GUI thus consists of three tabs, which cover these operations.

dx.doi.org/10.1021/ct500159h | J. Chem. Theory Comput. XXXX, XXX, XXX−XXX

The inclusion of new applications within MoSGrid can be easily achieved via the Portlet API. For a new application to be included, a few tasks have to be completed. First, the application has to be installed on at least one DCI available via the science gateway and an entry made in the UNICORE Incarnation Database (IDB). The IDB is a way to make applications, on possibly many clusters, available via UNICORE. A common name, regardless of specifics of different clusters, is introduced for each application which enables UNICORE clients like the MoSGrid science gateway to automatically decide where to send jobs based on application availability. Then, in order to make a new application available via the domain portlet, a MSML dictionary tailored to the new application needs to be created. Applications will be made available via workflows. For each workflow a MSML template needs to be created to enable the Portlet API to automatically generate and display parameter masks for all jobs it contains.

IMPORT

MoSGrid has a global workflow repository in which all workflows are stored with their default settings. A user has first to import a workflow into her or his local repository in order to modify and execute it. MSML templates map workflows to domains. These are also important in other steps as it will be explained in the following sections. Each domain comes with a set of MSML templates. A template stores a unique workflow identifier to a certain workflow from the global repository. On startup, the Portlet API reads all templates, links them to the corresponding workflows, and creates a set of importable objects. Users can import such an object with a custom and unique name, which is followed by step two—submission. A template usually also contains a short workflow description and a figure of the workflow graph which is both displayed to the user.

SUBMISSION

Before users can submit an imported workflow, they have to provide some input. This may involve uploading molecule files and entering or choosing simulation parameters, for example. The Portlet API automatically creates an input mask by reading the MSML template. This process also depends on MSML dictionaries (see Section MSML). While templates store which input has to be given, dictionaries provide the corresponding meta-information for those parameters. Therefore, a template contains a list of desired input for each task in the workflow. In order to set parameters, each entry in the list may contain a default value and is linked to an entry in one of the dictionaries. The dictionary then determines the data type, value restrictions (numerical or literal), and provides tooltips. The possible combination of input fields is quite large, i.e. files can be uploaded from the user's computer or can be selected from the repository. Some files, e.g. proteins in PDB format, may need some preprocessing, which can also be done via the input mask. Parameters can be optional or mandatory, entered in a text field, or chosen from a list of predefined values, may have minimum and maximum values, or can just be a simple flag, which can be set with a checkbox. A workflow can also be linked to more than just one MSML template. This allows using one workflow with different input settings, e.g., an expert mode with more required user input and a novice mode with less parameter settings. Overall, the creation of input masks is a highly flexible process and is divided into several sources of information to avoid redundancy. After the user has filled the input mask, a validation process checks the correctness of all inputs and finally allows the submission of the workflow. Special adapters transform the input for each workflow task into any desired format, which may be a simple command-line string or even an input file in a more complex format. The user's input is also stored within the MSML template which will be passed into the workflow as a valid MSML file. The parser then takes care of this file and will extend it with results during workflow execution (see Section Templatedesigner and Parser).

MONITORING

Monitoring involves more than just checking the status of submitted workflows. Users can also abort workflows, and, even more importantly, they can browse through the results of finished workflow tasks. Besides opening results in raw text mode, the Portlet API integrates ChemDoodle Web components[40] as 3D molecule viewer and Dygraphs[41] for interactively displaying of 2D plots. These are both based on JavaScript, which means no installed or activated Java plugin is required in the user's browser (see Figure 3). Of course, it is also possible to download all result files, but they can also be kept in the MoSGrid repository where they are stored by default.

HELP AND TUTORIALS

To ease the usage of the MoSGrid science gateway, an elaborate help system was designed to pick up the user and guide him or her through the portal infrastructure. A variety of entry points were set up, first of all, the "Help" menu entry located in the horizontal menu bar. This menu gathers all help links of the science gateway in one place and is located at a position where users of desktop applications would expect it to be situated. The second entry point is on the community page being the first page view the user is presented to after login. It is enriched with iconographies, which lead the user through and to tasks necessary to perform before the science gateway can be reasonably used. These icons appear in the order of the steps that have to be accomplished (certificate generation, account activation, SAML assertion generation). Additionally, icons lead to tutorials provided for each simulation domain (docking, molecular dynamics, quantum chemistry).

Besides these general help access options, the portlets provide supportive information in the portlets themselves and, in addition, give access to the domain specific tutorial area via "Help" buttons, as mentioned before. Every computational simulation domain is handled separately within the help system. Each of the domains is explained to the user from two different perspectives. On the one hand, there is a Frequently Asked Questions (FAQ) area that creates a general idea of the domain's aims, possibilities, and limitations. On the other hand, a tutorial area eases the first steps in using the portlets. It explains the basic principles of the portlets and assists the user exemplary in using them.

This predefined help system is complemented by a community driven Wiki. Here the users can contribute with their experience and knowledge. Hence, a sustainable help resource can be created, that can grow with the user's gathered know-how. A beneficial aspect of this help structure is the shared language and viewpoint. Common problems in cross discipline projects can be the terminology and the way of thinking and approaching problems. Thus, this shared user's perspective may ease the mutual understanding.

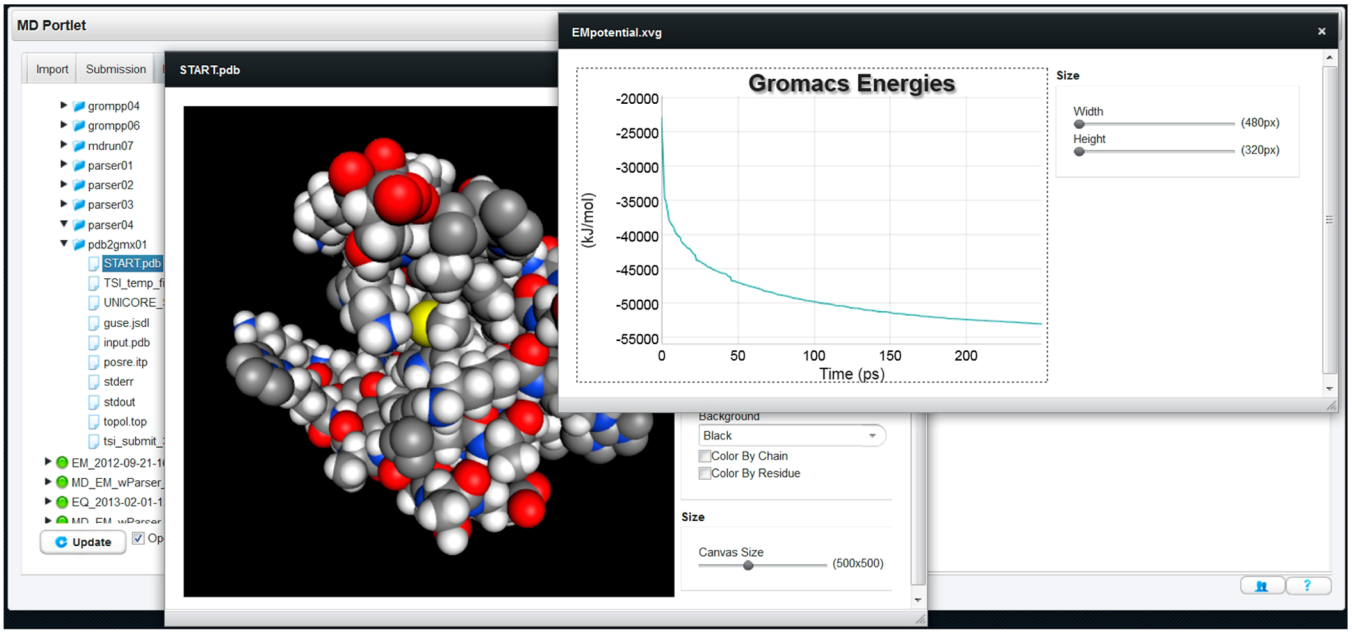


Figure 3. Visualization of molecular structure using ChemDoddle is shown, as well as the dynamic generation of plots using Dygraphs. In this way simulation results from all application domains can be analyzed directly within the portal, without immediate need for download or external postprocessing.

APPLICATIONS AND RESULTS

As outlined above, MoSGrid constitutes a unique computing platform for molecular simulations. With its modular setup of adapters and parsers it is possible to connect any scientific computing code to the MoSGrid science gateway. For the time being, some popular and well established programs from the simulation domains of quantum chemistry, molecular dynamics, and docking have been integrated, namely, Gaussian,[3] NWChem,[4] Gromacs,[5] CADDSuite,[6] FlexX,[8] and AutoDock;[7] others are to follow.

A central concept of MoSGrid is the application of workflows to perform a desired simulation. A workflow is the description of a sequence of tasks to be executed in order to reach the simulation goal. The range of complexity of such workflows is broad: It can be very straightforward, for example, just triggering a certain single-step energy calculation, but it can also be very intricate, e.g., extracting the starting structures from a database, polishing up the primary data, e.g., by protonation, charge designation, water handling, combination of different data sources, and so on.

The application of workflows offers various advantages: First, for reoccurring tasks it saves a lot of time and effort, since the single steps of a complex simulation do not have to be controlled by hand. Furthermore, it reduces the possibility of introducing careless mistakes, since the automation of the computation sequence prevents oversights. Another advantage is the possibility to apply the same procedure to a huge number of input structures with just one workflow submission. Additionally, identical simulations can be executed with different computing codes just by minimal editing of the workflow. Finally, predefined workflows enable even inexperienced users to reach elaborate simulation results, which otherwise would have been out of their scope. In combination with the documentation of the respective workflows and the MoSGrid community this also provides an innovative and new e-learning approach for computational chemistry.

In the following section, we will discuss some typical use cases of MoSGrid workflows for the different simulation domains.

QUANTUM CHEMICAL SIMULATIONS

QC simulations deal with the electronic structure of molecules. The job definition is always quite similar representing an ideal playground for the use of workflows. In a rather simple workflow, a given geometry can be calculated with a given set of methods, functional, and basis sets. The key geometric parameters are parsed and collected in tables afterward. Another use case would be the study of a complex potential hypersurface by varying one or several geometric parameter. Then, a set of similar jobs has to be submitted into the grid with the same methodical setup and varying coordinates. A suited workflow (e.g., a parameter sweep) combines the predefined coordinates with the chosen method and facilitates the whole process of submitting and collecting data. Both types of workflows are independent of the quantum chemical code. Further postprocessing can cover the addition of a solvent model, calculation of charges and frequencies, formatting of checkpoint files, and definition of new job files for subsequent time-dependent DFT calculations.

With respect to requests from the community, the geometry optimization workflows were primarily implemented for Gaussian, the well-renowned quantum chemical suite. Gaussian compute jobs are typically driven by a single input file, which includes a machine-specific section with the so-called "link0" commands, stating the number of cores and the amount of memory required, a job-specific section called route card, which describes the type of quantum chemical calculation to be performed, and finally the representation of the molecule in question, i.e., charge, spin multiplicity, and atomic coordinates. These input files are uncomfortable for scientists to work with and are inflexible in the represented structure, e.g., blank lines at certain positions are required. With the workflows for geometry optimizations, made available through the quantum chemistry portlet, users are relieved from manually constructing such an input file. Instead, the required file is generated with the help of an adapter from the user input to textboxes and listboxes with reasonable default values, taking advantage of MSML. In order to reduce the complexity of the input masks, geometry optimizations

dx.doi.org/10.1021/ct500159h | J. Chem. Theory Comput. XXXX, XXX, XXX−XXX

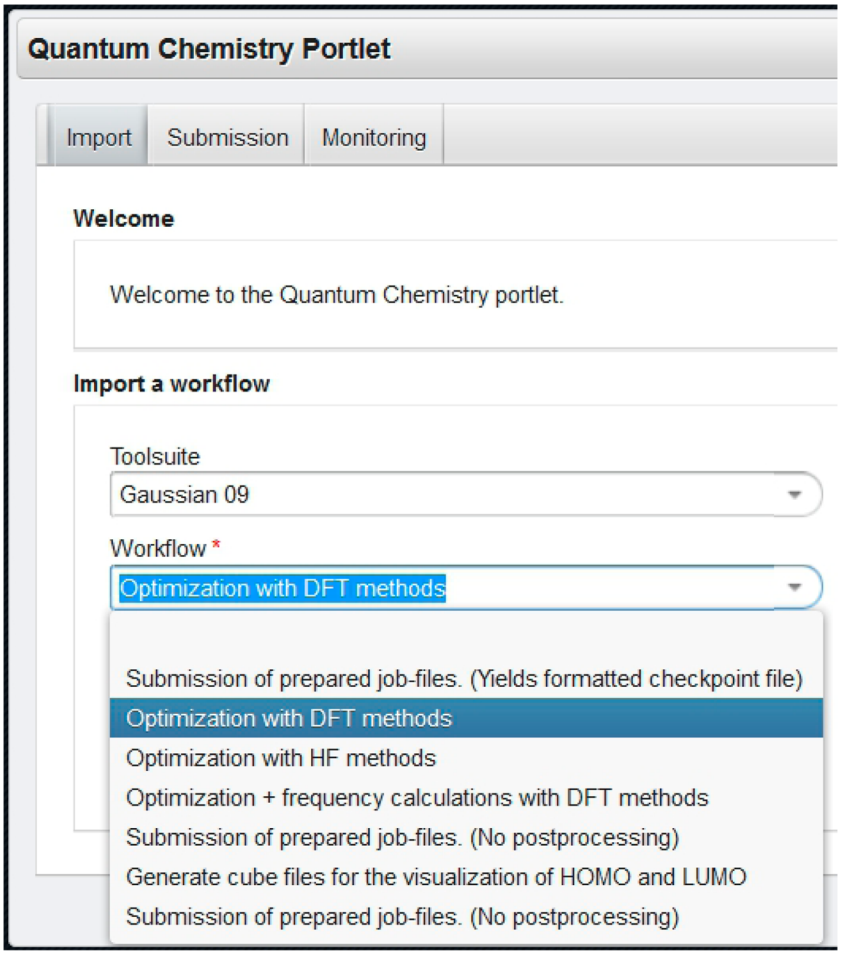


Figure 4. Variety of available workflows is shown for the Quantum Chemistry Portlet, covering Hartree−Fock and density functional methods commonly used in the field. Furthermore, different tasks aiming at a different level of user experience can be carried out by selecting the appropriate workflow.

using Hartree−Fock (HF) methods or Density functional theory (DFT) are realized through two separate workflows (see Figure 4). While the HF workflow allows to explicitly set the type of HF theory, for example ROHF or UHF for open shell species, the DFT workflow requires the DFT functional to be set. Machine-specific parameters common to both workflows, such as the number of cores, the wall time, and the required memory, are accessible through input fields. For the convenience of the users, these are preset to reasonable default values. A single listbox is populated from an MSML dictionary entry containing frequently used combinations with the Pople basis set 6-31G as the default. Unless a charged or open-shell species is to be examined, the default values for charge and spin multiplicity can be left unchanged as well. User interaction is required for the input of the molecular structure. With respect to the needs of less experienced users, the corresponding textbox supports an easily understandable four column data format, consisting of an element symbol and the Cartesian coordinates of the respective atom. When the workflow is submitted, an adapter generates the necessary input file. Once successfully calculated, the geometry optimization can be followed by several postprocessing steps. From the machine-dependent binary checkpoint file of the calculation, a new Gaussian-type input file with the optimized geometry is created. A subsequent task extracts fundamental job data, such as the type of calculation (route card), the number of orbitals, etc., to an ASCII file; another one extracts vibrational frequency data. Since the binary checkpoint file cannot be used on any other cluster, it is converted to its machine-independent and human-readable, formatted counterpart. From this formatted checkpoint file, a PDB file is generated. As a result, the user is enabled to visualize the optimized molecular geometry in the monitoring tab of the portlet through ChemDoodle.

While these workflows primarily address users needing guidance, experienced users may prefer the expert workflow. Here an input file, previously prepared by the user, may be submitted directly. A manifold of molecular properties, however, is not reflected in the molecular geometry itself. Therefore, different workflows have been devised to generate volumetric representations of their spatial distribution in the form of cube files, a file format also common to other quantum chemical packages. Currently Gaussian and NWChem workflows are available through the MoSGrid science gateway.

■ MOLECULAR DYNAMICS SIMULATIONS

MD simulations enable the users to study properties of larger molecules such as proteins and nucleic acids.[42] They deal with the impact of force fields between atoms and molecules on their physical movements. The huge number of interacting atoms as well as the required small time discretization makes it a computationally challenging task, which cannot be solved analytically for any reasonable sized molecular ensemble. Besides geometric aspects (as for example the opening and closure of a

protein binding pocket[43]), also kinetic and thermodynamic quantities can be derived, e.g., rate constants for protein folding.[44−46] Corresponding workflows can be considered as rooted graphs starting from the molecular structure provided by the user. The user input undergoes some filtering and conversion to MSML to ensure clean and error free input for the simulations. The resulting structure needs to be processed with various tools in order to create a valid simulation system including. Here the modular character of Gromacs tools is advantageous as an individual task can be assigned to a node within a workflow handling the execution of a specific tool. Although multiple different simulations are possible, all of them have at least one node pointing to the main MD executable. This program calculates the actual movement of each atom in the system and its contribution to the energy. Within the Gromacs suite this application is *mdrun*, which expects a binary combination of topology and simulation parameters as input and provides as output a trajectory, i.e. positions with respect to time, an energy file, and logs comprising simulation output. Further simulation steps can be built on this output along with various analysis tools can use it to generate scientifically relevant plots and charts. This information is directly visualized within the MD portlet. To give an example, the user may directly check output conformations using ChemDoodle or inspect root-mean-square deviation plots using Dygraphs. All relevant simulation results are annotated with metadata and stored in the MoSGrid repository.

Within MoSGrid, currently MD simulations with Gromacs are supported.[5,47] Popular programs like Desmond,[48] NAMD,[49] and AMBER[50] shall follow soon to support the user community with their research. Independent of the simulation code used, all MD programs have some general steps in common, which can be easily represented by workflows. It is noteworthy that MD workflows are mostly rather complex, and a suited workflow represents a significant help to the user who tries to overcome technical hurdles. The molecular simulation system has to be put into a periodic box, solvated with water and counterions, followed by an energy minimization. The single steps for a typical equilibration are described in detail in the following paragraph.

The first task is to identify all atoms and bonds in order to generate a correct description of the atomistic interactions, a so-called topology (topol.top). This is accomplished by using the Gromacs tool *pdb2gmx*. All protein hydrogen atoms are removed before being replaced according to the definitions of the force field (e.g., Gromos,[51] OPLS-AA,[52] Charmm,[53] or Amber[54]) used for the later simulations. A reasonable protonation state is estimated based on the default capabilities of *pdb2gmx*. The resulting structure with force field consistent atom names is put into a simulation box using *editconf*. The size of the box is chosen large enough to avoid periodicity artifacts. This is achieved by setting the distance between the solute and the box 20 percent larger than the largest cutoff. Afterward the box is filled with water using *genbox*. Here, different water models are possible (e.g., the Simple Point Charge model (SPC)[55] or the series of different Transferable Intermolecular Potential functions (TIP)[56]) (see Figure 5). The calculation of the explicit solvent water is computationally intensive, as it increases the number of interacting particles, but is usually desired to enable nativelike conditions. Implicit solvent models may be used through the submission of self-prepared job descriptions. The topology file is updated accordingly in the next step. The precompiler *grompp* is executed in conjunction with a simulation description yielding the binary file topol.tpr including all coordinates, bonds, forces, and further information to carry out an energy minimization. This calculation is carried out by calling an MPI-parallel version of *mdrun*. The output of the minimization is directly taken as a starting point for a brief position restrained equilibration run. The precompiler *grompp* is executed once again but using a different simulation description. After the restrained MD equilibration simulation the last frame of the simulation is written to a PDB file, serving as a starting point for production runs, docking studies, and/or further scientific analysis. Usually it is futile to relax the simulation system further by carrying out a nonrestrained equilibration. For variants of this simulation protocol multiple workflows are provided, reducing the tiresome system preparation to a few mouse clicks. By adding further options to the input masks that guide the user through the simulation setup process the (advanced) user will gain more control to adapt the workflow to his/her specific problem. By creating standardized workflows, in which only a small subset of parameters are modifiable by the user, different simulations of one workflow become comparable. Additionally, since all information to set up the simulation is stored inside the workflow and is inherent part of the sustainable stored file and directory structure, the simulation experiment is readily reproducible.

With a large variety of possible MD simulations, it is not possible to provide prebuilt workflows for all use cases. Therefore, users may also upload and simulate self-prepared job descriptions (Gromacs−topol.tpr), taking advantage of powerful grid resources.

DOCKING WORKFLOWS

Docking deals with the calculation of the pose of a ligand within a receptor's binding site and the estimate of the binding energy of the resulting complex. For the purpose of docking, we used the open source docking tools of CADDSuite (Computer Aided Drug Design Suite) as framework to pre- and postprocess the molecular structures. We here take advantage of several grid-enabled tools that divide docking into simple tasks.[57,58] The central docking and scoring steps can be carried out with CADDSuite,[6] Autodock,[7] or FlexX.[8]

The docking portlet provides several workflows with varying detail of user input required, ranging from the unprepared protein plus a flat file screening library to fully prepared receptor and ligand files. The complete workflow consists of four major steps: 1) target preparation, i.e., separation of the receptor structure from its ligand in the PDB file, protein protonation, elimination of irrelevant water molecules, and pocket detection; 2) ligand preparation, i.e., generation of 3D coordinates for the screening library at a specified pH; 3) docking, i.e., grid calculation and docking; and 4) rescoring of the best scoring compounds. This workflow only requires two input files: a PDB file containing a protein with a reference ligand and an SDF file containing the molecules to be docked into the protein. In addition to that, variants of the workflow are offered via the portlet that leave out parts of the preparation steps in case the user has already prepared their files using other software and only wants to perform the docking simulation on the grid. Although the user interface is kept quite clean and simple, it allows the setup for a wide range of parameters for the initial docking step and for the refinement procedures if desired (see Figure 6).

The compute time, necessary to prepare a high-level ligand database, can easily exceed the actual time needed for docking itself. Once a ligand library was properly prepared, e.g., assigning the correct protonation state and generating meaningful conformers, it may be reused anytime. As it is stored within the MoSGrid repository it easily can be selected for further

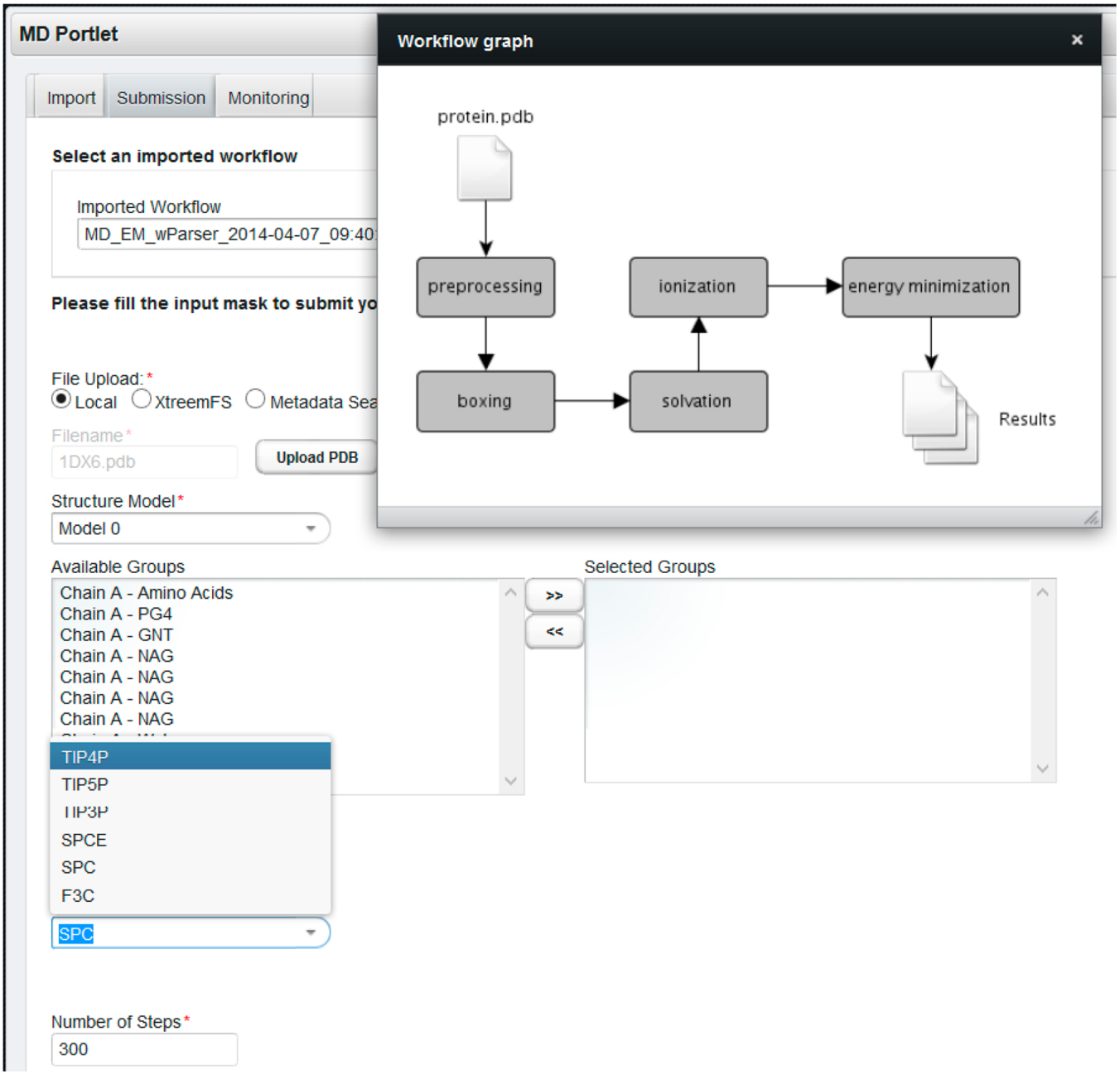


Figure 5. Part of the configuration of a Molecular Dynamics workflow is shown. In order to ensure meaningful simulations the users are obliged to select appropriate groups of the uploaded PDB file. Furthermore, details such as the force field, corresponding water model, and number of steps for the energy minimization have to be specified. For inexperienced users carefully chosen default values are preset.

simulations. It is also possible to share such a library with co-workers via the integrated user rights management.

When it comes to performing docking using libraries with several thousands of ligands, the biggest challenge is the dynamic utilization of grid resources. The workflow management service gUSE provides a simple way to parallelize jobs via generator and collector ports. Combining the use of these ports with tools in the CADDSuite, the workflow takes the ligand input SDF file and splits it into multiple smaller files. Each generated file contains a small portion of the given ligand library. Next, docking is performed on each of these single files. The workflow service will generate as many parallel jobs as files are provided. In the end, the results of each parallel docking job are merged into a file for further processing. This parallelization will provide results much faster than a pure sequential approach and consequently saves a lot of the scientist's valuable time. It is also possible to rescore docking results. Following a similar approach as for the docking step, intermediate docking results can be re-evaluated employing a different scoring function.

A major advantage for the end-users utilizing MoSGrid's docking portlet is found in the ready-to-use state of the portlet, thus preventing them from creating their own scripts to run a docking simulation. By automatizing the process of protein and ligand preparation relying on MSML, the reproducibility is guaranteed and the risks of avoidable errors minimized. More complex crystal structures of larger protein multimers may require a deeper insight into the structure of the PDB file itself. However, MoSGrid's docking portlet makes an educated guess

dx.doi.org/10.1021/ct500159h | J. Chem. Theory Comput. XXXX, XXX, XXX−XXX

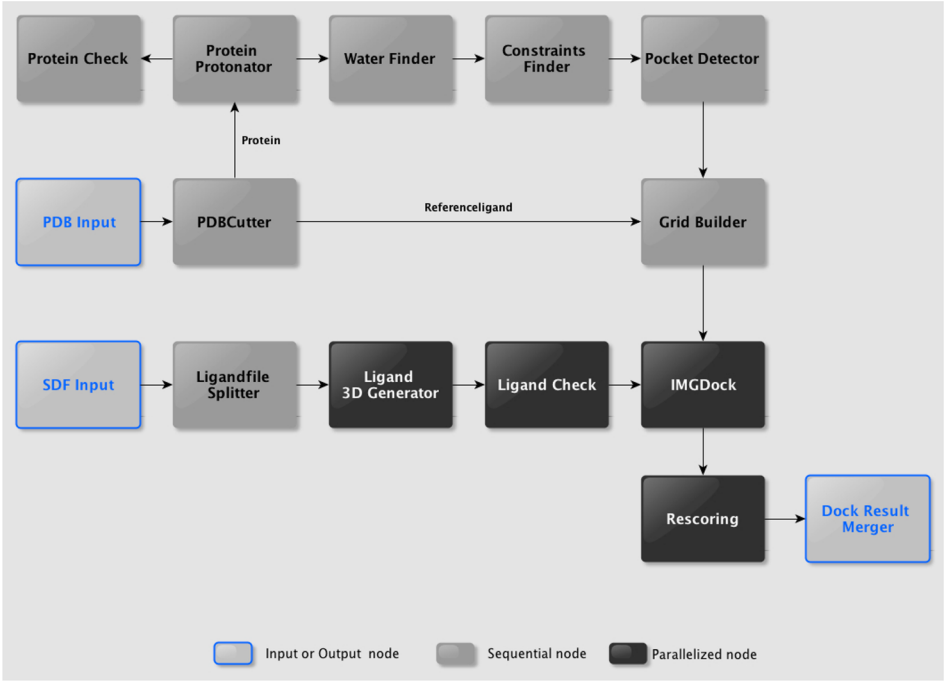


Figure 6. A typical docking workflow is shown, based on CADDSuite and its tools. The workflow handles the complete preparation and docking simulation for an input set of a target structure complex and a screening library. Workflows for other docking applications, such as FlexX and Autodock, follow the same principle and structure, although the distinct placement algorithm is used.

within the docking workflow. Furthermore, the problems occurring when converting one molecular file format into another, e.g. PDB to PDBQT, can be nearly completely eliminated due to the usage of MSML as uniform storage format, ensuring consistency of all data.

EXTRA VALUE FOR SCIENTISTS

The MoSGrid infrastructure is intended to support scientists to solve complex scientific problems by applying high-throughput and high-performance simulations in a distributed environment. Various new technical solutions were developed and implemented to ease the use of computational resources on DCI resources, to manage molecular simulation data and results, and finally to share expertise with other scientists. In that way, MoSGrid offers similar and more advanced features compared to other well-known existing infrastructures like XSEDE[59] (succeeding TeraGrid[60]), Chemomentum,[61] RESURGENCE,[62] GEMSTONE,[63] and others.

The MoSGrid science gateway manages workflows within three molecular simulation domains, quantum chemistry, molecular dynamics, and docking. The scientist can apply predefined sequences of processing steps represented in existing workflow templates as well as newly created workflows. The predefined existing workflows are validated by experts and enable a quick and easy usage of installed simulation packages across German grid resources. Newly defined workflows can be shared with other scientists in collaborative projects and for validation.

Within MoSGrid the usage of standardized representation of molecular data is fostered, being a unique feature compared to other science gateways. For the representation of structural data, simulation procedures ("recipes") and results in MoSGrid existing *defacto* standards (e.g., CML) were extended to support standardized formats as much as possible for all different kinds of data. The introduced MSML combining structural data and recipes extends CML and thus enables improved interoperability across simulation packages and sustainable data archiving solutions for each supported domain.

Based on the usage of standardized data representation, a powerful indexing and search system was implemented. Scientists can search through the science gateway for specific properties of stored data to gather information from existing accessible simulation results or molecular structures. Even large amounts of data can thus be easily handled and retrieved for reuse or evaluation.

The management of simulation data within MoSGrid is prepared for future repositories including unique references to simulation results and supplementary data. At the low-level essential files are stored in the MoSGrid repository, which handles unique identifiers for future reference.

Considering user requirements right from the beginning of development and combining established techniques has led to a reliable portal for the computational chemistry community. From a user's perspective there is no other science gateway available offering such a broad, secure, and powerful service as MoSGrid.

CONCLUSION

The MoSGrid portal demonstrates how a Web-based science gateway can solve two distinct, yet related issues: a user-friendly access to complex computational tools and the use of distributed high-performance computing infrastructures. Both issues seriously hamper the access to state-of-the-art methods in computational chemistry and computing facilities. Built upon mostly existing software infrastructures we present a general architecture suitable for the computational chemistry community. The solution sketched here is focused on three domains: quantum chemistry, molecular dynamics, and docking, targeting at a broader community than any other science gateway. It supports diverse tools and diverse selection of workflows tailored for inexperienced users and experts. MSML as generic data format for molecular simulations serves as data hub independent of formats used in tools and workflows. The close collaboration

dx.doi.org/10.1021/ct500159h | J. Chem. Theory Comput. XXXX, XXX, XXX−XXX

between the developers, providers, and the community is leading to the integration of further tools, the development of new workflows, and the support of additional domain specific features, e.g., further pre- and postprocessing steps easing the analysis of data.

The MoSGrid portal and the underlying computing infrastructure is free of charge for academic users of German research institutes. From the technical point of view it is applicable worldwide, but there are challenges due to different policies in German computing infrastructure providers. The PRACE (Partnership for Advanced Computing in Europe)[64] infrastructure is suited to help with these challenges in Europe, since it is reaching out to the whole European research community. The MoSGrid consortium is working toward the integration of PRACE within the portal. It is ideally prepared since it makes use of UNICORE and all PRACE resources offer access via UNICORE. Furthermore, the MoSGrid science gateway will be offered for the US research community as a XSEDE startup solution using the UNICORE XSEDE resources. Communities of further countries can be supported via the integration of appropriate available UNICORE resources in those countries.

Future developments shall include the extension by further infrastructures, tools, and workflows for the computational chemistry community. An extension of the setup is easily done. This integration primarily requires the conversion of data formats and the mapping of the required metadata. If the application requires input or output entirely different in nature of the workflows and tools described here, the implementation of additional portlets might be required as well. The starting point for a science gateway would be the requirement analysis for the application domain. Much can be learned from example workflows that often already exist or can be easily obtained from potential users. We thus believe that the solution described here will have impact on the whole computational chemistry community.

AUTHOR INFORMATION

Corresponding Author

*E-mail: krueger@informatik.uni-tuebingen.de.

Author Contributions

S.G., J.K., and R.G. developed the concept and implemented the portal. S.G. developed the UNICORE integration with WS-PGRADE/gUSE as discussed with O.K. P.S. and L.G. extended the UNICORE integration. S.B., J.K., M.K., L.P., and T.St. designed MSML, and R.G. and P.S. implemented the metadata management. P.S. and R.G. implemented the distributed data management, as discussed with W.E.N., R.M.P., and T.St. S.G. designed the security concept, and S.G., T.Sc., and R.G. implemented it. J.K., R.G., S.B., A.B., and P.S. integrated HPC resources with the portal. The help sites and tutorials were created by S.B. and S.G. and reviewed and improved by many others. J.K. integrated molecular dynamics codes and created related workflows. S.H.P., M.K., L.P., and K.W. worked on the integration of quantum chemistry codes and creation of QC workflows. Docking codes were integrated, and workflows were created by J.K., L.G., and C.S., as discussed with O.K. C.S. and K.W. created example workflows. A.Z. created the Portlet API concept and implemented it, as discussed with J.K. J.K., R.G., and S.G. wrote the manuscript with contributions and help from all authors.

◆S.G., J.K., and R.G. contributed equally.

Notes

The authors declare no competing financial interest.

ACKNOWLEDGMENTS

The authors would like to thank the BMBF (German Federal Ministry of Education and Research) for the opportunity to do research in the MoSGrid project (reference 01IG09006). The research leading to these results has partially been supported by the European Commission's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 283481 (SCI-BUS) and no. 312579 (ER-flow) and by the LSDMA project of the Helmholtz Association of German Research Centres. Special thanks are due to NGI-DE for managing the German Grid infrastructure and all compute centers supporting MoSGrid.

REFERENCES

(1) Gesing, S.; Grunzke, R.; Krüger, J.; Birkenheuer, G.; Wewior, M.; Schäfer, P.; Schuller, B.; Schuster, J.; Herres-Pawlis, S.; Breuers, S.; Balasko, A.; Kozlovszky, M.; Fabri, A. S.; Packschies, L.; Kacsuk, P.; Blunk, D.; Steinke, T.; Brinkmann, A.; Fels, G.; Müller-Pfefferkorn, R.; Jäkel, R.; Kohlbacher, O. A Single Sign-On Infrastructure for Science Gateways on a Use Case for Structural Bioinformatics. *J. Grid Comput.* **2012**, *10*, 769−790.

(2) Grunzke, R.; Breuers, S.; Gesing, S.; Herres-Pawlis, S.; Kruse, M.; Blunk, D.; de la Garza, L.; Packschies, L.; Schäfer, P.; Schärfe, C.; Schlemmer, T.; Steinke, T.; Schuller, B.; Müller-Pfefferkorn, R.; Jäkel, R.; Nagel, W. E.; Atkinson, M.; Krüger, J. Standards-Based Metadata Management for Molecular Simulations. *Concurr. Comput. Pract. Exp.* **2013**, http://doi.wiley.com/10.1002/cpe.3116.

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

(4) Valiev, M.; Bylaska, E. J.; Govind, N.; Kowalski, K.; Straatsma, T. P.; Van Dam, H. J. J.; Wang, D.; Nieplocha, J.; Apra, E.; Windus, T. L.; de Jong, W. A. NWChem: A Comprehensive and Scalable Open-Source Solution for Large Scale Molecular Simulations. *Comput. Phys. Commun.* **2010**, *181*, 1477−1489.

(5) Hess, B.; Kutzner, C.; van der Spoel, D.; Lindahl, E. GROMACS 4: Algorithms for Highly Efficient, Load-Balanced, and Scalable Molecular Simulation. *J. Chem. Theory Comput.* **2008**, *4*, 435−447.

(6) Kohlbacher, O. CADDSuite - a Workflow-Enabled Suite of Open-Source Tools for Drug Discovery. *J. Cheminf.* **2012**, *4*, O2+.

(7) Morris, G. M.; Goodsell, D. S.; Halliday, R. S.; Huey, R.; Hart, W. E.; Belew, R. K.; Olson, A. J. Automated Docking Using a Lamarckian Genetic Algorithm and and Empirical Binding Free Energy Function. *J. Comput. Chem.* **1998**, *19*, 1639−1662.

(8) Rarey, M.; Kramer, B.; Lengauer, T.; Klebe, G. A Fast Flexible Docking Method Using an Incremental Construction Algorithm. *J. Mol. Biol.* **1996**, *261*, 470−489.

(9) SCI-BUS. http://www.sci-bus.eu/ (accessed Apr 5, 2014).

(10) ER-flow. https://www.erflow.eu/ (accessed Apr 5, 2014).

(11) EDGI. http://edgi-project.eu/ (accessed Apr 5, 2014).

(12) WeNMR. https://www.wenmr.eu/ (accessed Apr 5, 2014).

(13) Van Dijk, M.; Wassenaar, T. A.; Bonvin, A. M. J. J. A Flexible, Grid-Enabled Web Portal for GROMACS Molecular Dynamics Simulations. *J. Chem. Theory Comput.* **2012**, *8*, 3463−3472.

(14) Chia, E.; Shamsir, M. S.; Hussein, Z. A.; Hashim, S. Z. M. GridMACS Portal: A Grid Web Portal for Molecular Dynamics Simulation Using GROMACS. In *2010 Fourth Asia International Conference on Mathematical/Analytical Modelling and Computer Simulation*; IEEE: 2010; pp 507−512.
(15) Hospital, A.; Andrio, P.; Fenollosa, C.; Cicin-Sain, D.; Orozco, M.; Gelpí, J. L. MDWeb and MDMoby: An Integrated Web-Based Platform for Molecular Dynamics Simulations. *Bioinformatics* **2012**, *28*, 1278−1279.
(16) Dziubecki, P.; Grabowski, P.; Kuczynski, T.; Kurowski, K.; Szejnfeld, D.; Tarnawczyk, D.; Wolniewicz, M. Molecular Dynamics Science Gateway with Vine Toolkit Providing UNICORE Middleware Support. In *UNICORE Summit 2011: Proceedings, 7 - 8 July 2011, Toruń, Poland*; Forschungszentrum Jülich: Toruń, Poland, 2011; pp 25−34.
(17) Farkas, Z.; Kacsuk, P.; Kiss, T.; Borsody, P.; Hajnal, Á.; Balaskó, Á.; Karóczkai, K. AutoDock Gateway for User Friendly Execution of Molecular Docking Simulations in Cloud Systems. In *Cloud Computing with E-science Applications*; CRC Press/Taylor & Francis: 2014.
(18) Rampino, S.; Faginas Lago, N.; Laganà, A.; Huarte-Larrañaga, F. An Extension of the Grid Empowered Molecular Simulator to Quantum Reactive Scattering. *J. Comput. Chem.* **2012**, *33*, 708−714.
(19) Laganà, A.; Costantini, A.; Gervasi, O.; Lago, N. F.; Manuali, C.; Rampino, S. COMPCHEM: Progress Towards GEMS a Grid Empowered Molecular Simulator and Beyond. *J. Grid Comput.* **2010**, *8*, 571−586.
(20) GridChem. https://www.gridchem.org/ (accessed Apr 5, 2014).
(21) Dooley, R.; Milfeld, K.; Guiang, C.; Pamidighantam, S.; Allen, G. From Proposal to Production: Lessons Learned Developing the Computational Chemistry Grid Cyberinfrastructure. *J. Grid Comput.* **2006**, *4*, 195−208.
(22) Todd, B. D. Development of Chemistry Portal for Grid-Enabled Molecular Science. In *First International Conference on e-Science and Grid Computing (e-Science'05)*; IEEE: 2005; pp 48−55.
(23) Byun, S.-W.; Lee, Y.-K.; Kwon, Y.-W.; Ryu, S.-H.; Jeong, C.-S. Workflow-Based Grid Portal for Quantum Mechanics. In *Grid and Cooperative Computing - GCC 2004 Workshops*; Jin, H., Pan, Y., Xiao, N., Sun, J., Eds.; Springer: Berlin, Heidelberg, 2004; pp 625−632.
(24) Murray-Rust, P.; Rzepa, H. S. Chemical Markup, XML and the World-Wide Web. 2. Information Objects and the CMLDOM. *J. Chem. Inf. Comput. Sci.* **2001**, *41*, 1113−1123.
(25) Kacsuk, P. P-GRADE Portal Family for Grid Infrastructures. *Concurr. Comput. Pract. Exp.* **2011**, *23*, 235−245.
(26) Farkas, Z.; Kacsuk, P. P-GRADE Portal: A Generic Workflow System to Support User Communities. *Future Gener. Comput. Syst.* **2011**, *27*, 454−465.
(27) Streit, A.; Bala, P.; Beck-Ratzka, A.; Benedyczak, K.; Bergmann, S.; Breu, R.; Daivandy, J. M.; Demuth, B.; Eifer, A.; Giesler, A.; Hagemeier, B.; Holl, S.; Huber, V.; Lamla, N.; Mallmann, D.; Memon, A. S.; Memon, M. S.; Rambadt, M.; Riedel, M.; Romberg, M.; Schuller, B.; Schlauch, T.; Schreiber, A.; Soddemann, T.; Ziegler, W. UNICORE 6 - Recent and Future Advancements. *JUEL-4319*; 2010.
(28) Hupfeld, F.; Cortes, T.; Kolbeck, B.; Stender, J.; Focht, E.; Hess, M.; Malo, J.; Marti, J.; Cesario, E. The XtreemFS Architecture - A Case for Object-Based File Systems in Grids. *Concurr. Comput. Pract. Exp.* **2008**, *20*, 2049−2060.
(29) Foster, I. Globus Toolkit Version 4: Software for Service-Oriented Systems. *IFIP Int. Conf. Netw. Parallel Comput.* **2006**, *Springer-V*, 2−13.
(30) Housley, R.; Polk, W.; Ford, W.; Solo, D. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile (RFC 3280). 2002. http://www.ietf.org/rfc/rfc3280.txt (accessed Apr 5, 2014).
(31) OASIS. Security Assertion Markup Language (SAML) V2.0. 2002. https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security (accessed Apr 5, 2014).
(32) Foster, I.; Kesselman, C.; Tsudik, G.; Tuecke, S. A Security Infrastructure for Computational Grids. In *CCS '98 Proceedings of the 5th ACM conference on Computer and communications security*; ACM: New York, 1998.
(33) Stender, J.; Kolbeck, B.; Hupfeld, F.; Cesario, E.; Focht, E.; Hess, M.; Malo, J.; Martí, J. Striping without Sacrifices: Maintaining POSIX Semantics in a Parallel File System. In *First USENIX Workshop on Large-Scale Computing*; USENIX Association: Berkeley, CA, USA, 2008; pp 6:1−6:8.
(34) Apache Lucene. http://lucene.apache.org/ (accessed Apr 5, 2015).
(35) Prlić, A.; Yates, A.; Bliven, S. E.; Rose, P. W.; Jacobsen, J.; Troshin, P. V.; Chapman, M.; Gao, J.; Koh, C. H.; Foisy, S.; Holland, R.; Rimsa, G.; Heuer, M. L.; Brandstätter-Müller, H.; Bourne, P. E.; Willis, S. BioJava: An Open-Source Framework for Bioinformatics in 2012. *Bioinformatics* **2012**, *28*, 2693−2695.
(36) Steinbeck, C.; Han, Y.; Kuhn, S.; Horlacher, O.; Luttmann, E.; Willighagen, E. The Chemistry Development Kit (CDK): An Open-Source Java Library for Chemo- and Bioinformatics. *J. Chem. Inf. Comput. Sci.* **2003**, *43*, 493−500.
(37) Internet Engineering Task Force. A JSON Media Type for Describing the Structure and Meaning of JSON Documents. http://tools.ietf.org/id/draft-zyp-json-schema-03.html (accessed Apr 5, 2014).
(38) Liferay. https://www.liferay.com/ (accessed Apr 5, 2014).
(39) Grönroos, M. Book of Vaadin. https://vaadin.com/book (accessed Apr 5, 2014).
(40) ChemDoodle. http://www.chemdoodle.com/ (accessed Apr 5, 2014).
(41) Dygraphs. http://dygraphs.com/ (accessed Apr 5, 2014).
(42) Karplus, M.; McCammon, J. A. Molecular Dynamics Simulations of Biomolecules. *Nat. Struct. Biol.* **2002**, *9*, 646−652.
(43) Durrant, J. D.; McCammon, J. A. Molecular Dynamics Simulations and Drug Discovery. *BMC Biol.* **2011**, *9*, 71.
(44) Scheraga, H. A.; Khalili, M.; Liwo, A. Protein-Folding Dynamics: Overview of Molecular Simulation Techniques. *Annu. Rev. Phys. Chem.* **2007**, *58*, 57−83.
(45) Seibert, M. M.; Patriksson, A.; Hess, B.; van der Spoel, D. Reproducible Polypeptide Folding and Structure Prediction Using Molecular Dynamics Simulations. *J. Mol. Biol.* **2005**, *354*, 173−183.
(46) Spoel, D.; Patriksson, A.; Seibert, M. M. Protein Folding Properties from Molecular Dynamics Simulations. In *Applied Parallel Computing. State of the Art in Scientific Computing SE - 13*; Kågström, B., Elmroth, E., Dongarra, J., Waśniewski, J., Eds.; Lecture Notes in Computer Science; Springer: Berlin, Heidelberg, 2007; Vol. *4699*, pp 109−115.
(47) Pronk, S.; Páll, S.; Schulz, R.; Larsson, P.; Bjelkmar, P.; Apostolov, R.; Shirts, M. R.; Smith, J. C.; Kasson, P. M.; van der Spoel, D.; Hess, B.; Lindahl, E. GROMACS 4.5: A High-Throughput and Highly Parallel Open Source Molecular Simulation Toolkit. *Bioinformatics* **2013**, *29*, 845−854.
(48) Bowers, K.; Chow, E.; Xu, H.; Dror, R.; Eastwood, M.; Gregersen, B.; Klepeis, J.; Kolossvary, I.; Moraes, M.; Sacerdoti, F.; Salmon, J.; Shan, Y.; Shaw, D. Scalable Algorithms for Molecular Dynamics Simulations on Commodity Clusters. In *ACM/IEEE SC 2006 Conference (SC'06)*; IEEE: 2006; pp 43−43.
(49) Phillips, J. C.; Braun, R.; Wang, W.; Gumbart, J.; Tajkhorshid, E.; Villa, E.; Chipot, C.; Skeel, R. D.; Kalé, L.; Schulten, K. Scalable Molecular Dynamics with NAMD. *J. Comput. Chem.* **2005**, *26*, 1781−1802.
(50) Salomon-Ferrer, R.; Case, D. A.; Walker, R. C. An Overview of the Amber Biomolecular Simulation Package. *Wiley Interdiscip. Rev.: Comput. Mol. Sci.* **2013**, *3*, 198−210.
(51) Oostenbrink, C.; Villa, A.; Mark, A. E.; van Gunsteren, W. F. A Biomolecular Force Field Based on the Free Enthalpy of Hydration and Solvation: The GROMOS Force-Field Parameter Sets 53A5 and 53A6. *J. Comput. Chem.* **2004**, *25*, 1656−1676.
(52) Jorgensen, W. L.; Tirado-Rives, J. The OPLS [optimized Potentials for Liquid Simulations] Potential Functions for Proteins, Energy Minimizations for Crystals of Cyclic Peptides and Crambin. *J. Am. Chem. Soc.* **1988**, *110*, 1657−1666.
(53) MacKerell, A. D.; Bashford, D.; Dunbrack, R. L.; Evanseck, J. D.; Field, M. J.; Fischer, S.; Gao, J.; Guo, H.; Ha, S.; Joseph-McCarthy, D.; Kuchnir, L.; Kuczera, K.; Lau, F. T. K.; Mattos, C.; Michnick, S.; Ngo, T.;

Nguyen, D. T.; Prodhom, B.; Reiher, W. E.; Roux, B.; Schlenkrich, M.; Smith, J. C.; Stote, R.; Straub, J.; Watanabe, M.; Wiórkiewicz-Kuczera, J.; Yin, D.; Karplus, M. All-Atom Empirical Potential for Molecular Modeling and Dynamics Studies of Proteins. *J. Phys. Chem. B* **1998**, *102*, 3586−3616.

(54) Ponder, J. W.; Case, D. A. Force Fields for Protein Simulations. *Adv. Protein Chem.* **2003**, *66*, 27−85.

(55) Berendsen, H. J. C.; Postma, J. P. M.; van Gunsteren, W. F.; Hermans, J. Interaction Models for Water in Relation to Protein Hydration. In *Intermolecular Forces*; Pullman, B., Ed.; D. Reidel Publishing Company: Dordrecht, 1981; pp 331−342.

(56) Jorgensen, W. L.; Chandrasekhar, J.; Madura, J. D.; Impey, R. W.; Klein, M. L. Comparison of Simple Potential Functions for Simulating Liquid Water. *J. Chem. Phys.* **1983**, *79*, 926.

(57) Moll, A.; Hildebrandt, A.; Lenhof, H.; Kohlbacher, O. BALLView: A Tool for Research and Education in Molecular Modeling. *Bioinformatics* **2006**, 365−366.

(58) Hildebrandt, A.; Dehof, A. K.; Rurainski, A.; Bertsch, A.; Schumann, M.; Toussaint, N. C.; Moll, A.; Stöckel, D.; Nickels, S.; Mueller, S. C.; Lenhof, H.-P.; Kohlbacher, O. BALL–Biochemical Algorithms Library 1.3. *BMC Bioinf.* **2010**, *11*, 531.

(59) XSEDE. https://www.xsede.org/ (accessed Apr 5, 2014).

(60) Wilkins-Diehr, N.; Gannon, D.; Klimeck, G.; Oster, S.; Pamidighantam, S. TeraGrid Science Gateways and Their Impact on Science. *IEEE Comput.* IEEE Press: 2008; Vol. *41*, pp 32−41.

(61) Schuller, B.; Demuth, B.; Mix, H.; Rasch, K.; Romberg, M.; Maran, U.; Sild, S.; Bala, P.; del Grosso, E.; Casalegno, M.; Piclin, N.; Pintore, M.; Sudholt, W.; Baldridge, K. K. Chemomentum - UNICORE 6 Based Infrastructure for Complex Applications in Science and Technology. In *Euro-Par 2007 Workshops: Parallel Processing (Lecture Notes in Computer Science 4854)*; Springer-Verlag: Berlin, Heidelberg, 2008; pp 94−103.

(62) Sudholt, W.; Altintas, I.; Baldridge, K. K. A Scientific Workflow Infrastructure for Computational Chemistry on the Grid. In *Computational Science - ICCS 2006, Lecture Notes in Computer Science 2006*; Springer-Verlag: Berlin, Heidelberg, 2006; pp 69−76.

(63) Baldridge, K. K.; Bhatia, K.; Greenberg, J. P.; Stearn, B.; Mock, S.; Sudholt, W.; Krishnan, S.; Bowne, A.; Amoreira, C.; Potier, Y. GEMSTONE: Grid-Enabled Molecular Science through Online Networked Environments. In *Life Sciences Grid Workshop (Satellite of Grid Asia)*; Singapore, **2005**.

(64) PRACE. http://www.prace-ri.eu/ (accessed Apr 5, 2014).