

Application of grid computing to parameter sweeps and optimizations in molecular modeling

Wibke Sudholt^{a,*}, Kim K. Baldridge^{a,b}, David Abramson^c,
Colin Enticott^c, Slavisa Garic^c, Chris Kondric^{b,c}, Duy Nguyen^{b,c}

^a *Institute of Organic Chemistry, University of Zürich, 190 Winterthurerstrasse, Zürich 8057, Switzerland*

^b *San Diego Supercomputer Center (SDSC), University of California, San Diego (UCSD),
9500 Gilman Drive, La Jolla, CA 92093-0505, USA*

^c *Center for Enterprise Distributed Systems (DSTC), Monash University, Clayton, Victoria, 3800 Australia*

Available online 5 November 2004

Abstract

In science and engineering in general and in computational chemistry in particular, parameter sweeps and optimizations are of high importance. Such parametric modeling jobs are embarrassingly parallel and thus well suited for grid computing. The Nimrod toolkit significantly simplifies the utilization of computational grids for this kind of research by hiding the complex grid middleware, automating job distribution, and providing easy-to-use user interfaces. Here, we present examples for the usage of Nimrod in molecular modeling. In detail, we discuss the parameterization of a group difference pseudopotential (GDP). Other applications are protein–ligand docking and a high-throughput workflow infrastructure for computational chemistry.

© 2004 Elsevier B.V. All rights reserved.

Keywords: Grid computing; Parameter sweep; Parameter optimization; Computational chemistry; Pseudopotential

1. Introduction

In science and engineering today, real as well as model systems tend to be complicated and highly parameterized. This can often create difficulties in the

ability to predict the influence of each of the many parameters deterministically. Therefore, there is a large need for automated parameter sweeping and optimization procedures.

With all the exciting developments in the life sciences in the past few years, parametric modeling also has become more and more important in this field of research. Biological systems are intrinsically extremely complex. In addition, the number and character of the parameters is often not well known or understood, let alone their special influence on a biological structure.

* Corresponding author.

E-mail addresses: wibke@oci.unizh.ch (W. Sudholt),
kimb@oci.unizh.ch (K.K. Baldridge), davida@csse.monash.edu.au
(D. Abramson), Colin.Enticott@csse.monash.edu.au (C. Enticott),
Salvisa.Garic@infotech.monash.edu.au (S. Garic),
ckondric@ucsd.edu (C. Kondric), ddn002@ucsd.edu (D. Nguyen).

Thus, on one hand one would like to have an easy possibility to test the importance of certain variables. On the other hand, computational parameterization of a biological system can lead to very useful models to better understand their behavior.

Our field of research is in computational chemistry and biochemistry. We computationally model the behavior of molecular systems, consisting of atoms with their associated nuclei and electrons, connected by chemical bonds. Biological macromolecules such as proteins, carbohydrates, DNA and RNA represent large and complicated assemblies of certain sets of smaller molecules or building blocks, and play important roles in life processes. However, small molecules also play important roles in our everyday life, for example, pharmaceuticals, hormones, energy transfer molecules, and so on. Of special interest in our studies are the interactions between all of these partners.

Due to their complexity, most (bio)molecular systems have many intrinsic parameters. For example, a large number of intra- and intermolecular conformations and configurations are possible by changing the positions of the atoms within each molecule or between molecular partners. The study of these numerous conformations is an integral part of molecular dynamics and Monte Carlo simulations, and in molecular ligand-protein docking calculations. Additionally, one might also want to scan particular molecular properties within a large database of compounds, in order to understand some (bio)molecular phenomenon.

The investigation of molecular systems at the computational level involves the solving of fundamental physical equations such as the Schrödinger equation in quantum mechanics and Newton's laws in classical mechanics. Due to system complexities, researchers often need to summarize complicated details of these equations by using special parameters to provide approximated, but much less resource-demanding computational models of the actual systems. This involves such constructs as basis sets, pseudopotentials, and semiempirical methods on the quantum chemical electronic-structure level, and force fields, united-atom approaches, and scoring functions on the classical atomic and molecular level.

Parameter sweeps and, to a more limited extent dependent on the type of optimization algorithm, parameter optimizations have the major advantage that

they generate lists of repetitive, independent calculations. This makes these kinds of studies embarrassingly parallel. Only at the beginning and end of each instance of a job, network transfer usually of input and output files from and to a central client machine is necessary. However, due to the complexity of the problem setup often a large number of compute-intensive calculations are necessary. Thus, a high throughput of jobs is desired, much more than can normally be executed on a single cluster or supercomputer. As such, these types of applications are perfectly suited for grid computing, i.e., the distribution onto a loosely internet-connected collection of compute resources.

In the past few years, grids in general have expanded the scope of what is possible in computational science regarding storage, computation, networking, and organization to new heights. However, it is very difficult for non-computer scientists to work directly with the grid middleware, such as the Globus toolkit [1]. Therefore, flexible software to build a bridge between the middleware and the user is necessary. The software is required to have an easy-to-use application program interface (API), and preferably a graphical user interface (GUI), and should allow a fast setup of new parametric modeling experiments. Automation of the preparation, submission, control, and collection of individual jobs should also be possible, but at the same time still be adaptable to a wide variety of application codes. All these conditions are fulfilled by the Nimrod toolkit [2], which we apply in our studies here. We should also mention that there are products analogous to Nimrod, such as the AppLeS Parameter Sweep Template (APST) [3], which are used for similar studies.

In this paper, we discuss how Nimrod enables us to solve parametric problems in quantum chemistry and molecular modeling on computational grids. We next give background on the architecture, capabilities, and usage of Nimrod as well as an overview of the grid testbeds employed in our experiments. We describe our most complete collaborative grid computing study up to now, the parameterization of a so-called group difference potential (GDP). This is followed by some ongoing work in our group in this area, concluded with a summary of the advantages and difficulties of applying grid computing to computational (bio)chemistry from our experience.

riences, plus a short future outlook for grid computing.

2. Setup—Nimrod software

Nimrod is a specialized toolkit for distributed parametric modeling. The software is actively developed in the group of David Abramson at Monash University, Australia, since 1994. Directed originally at local workstation clusters, Nimrod now targets wide-area computational grids. The main purpose is to enable scientists to design and perform parametric experiments, such as parameter sweeps and optimizations, on a computational grid quickly and easily. This means that only limited programming efforts and only minimal knowledge concerning the details of the grid setup itself should be necessary. Many research problems that are implied here would not be feasible without a grid infrastructure, but would also not be practicable without a parametric modeling system such as Nimrod (for examples, see Sections 4–7).

To reach this goal, Nimrod includes a simple declarative language to express a parametric experiment. Users simply generate a small plan or scheduling file that specifies their problem setup. This is not more difficult than writing a shell script, something computational scientists are typically familiar with. The first section of the plan file defines the names, types and domains (e.g., ranges and step sizes) of the parameters. Parameters can be floating point or integer numbers or text. In the case of parameter sweeps, the space resulting from building the cross product is scanned. The next section lists all the tasks that need to be performed in each instance of the experiment. Tasks include copying between the root machine and the grid nodes, substituting parameter placeholders by actual values in input files, executing shell commands, and behavior when an error occurs. In the case of parameter optimizations, a final section with details of the optimization algorithm follows.

The second necessary input to Nimrod is the list of local and remote compute resources available to the user on the grid. These are first generally specified by their name, and queue if necessary, and then individually selected for each parametric experiment. Furthermore, the middleware layer and corresponding security information (e.g., the Globus proxy), the type of

hardware, the maximum number of jobs, the cost for each CPU cycle and other resource details are assigned (in part automatically, e.g., from the Globus MDS) to each machine. Currently supported services include Fork, Globus, PBS, and Condor. Nimrod then provides a uniform interface for communication with each resource.

After an experiment is started as either a parameter sweep (G) or a parameter optimization (O), Nimrod continuously discovers idle resources and automatically distributes and monitors the jobs produced for each single parameter set on the computational grid. The scheduling can consider various constraints such as a soft real time deadline or the overall cost of an experiment, that is, one can run the tasks as quickly and/or as cheaply as possible. Nimrod also keeps track of failed runs and reallocates calculations to different machines if necessary.

Internally, the core of the Nimrod system consists of a three-level PostgreSQL database that stores all the details of the experiments and resources. The individual jobs are actually run by a software component called the “agent”. Agents are themselves scheduled and copied onto the various resources. The number of agents and type of binary on each machine is extracted from the resource entries in the database, and the set of commands to execute jobs determined by the middleware service (e.g., the Globus GRAM interface). Once initiated, the agents contact the database and request calculations, which are then interpreted and run sequentially on the resource. As such, any one agent can run more than one job. This design helps to hide the latency involved in the scheduling and invocation of a computation on a remote resource.

There are currently a number of implementations of Nimrod: *Nimrod/G* is the research prototype for parameter scans on a computational grid as described here, and provides the general basis for the other Nimrod tools. *EnFuzion* represents a commercial solution, based on the original Nimrod version for clusters. *Nimrod/O* enables execution of an arbitrary computational model as a core of a non-linear optimization process. The user specifies the type and space of parameters in the model, the starting point(s), possibly which constraints have to be fulfilled, and which output variable shall be minimized or maximized. *Nimrod/O* currently employs a number of built-in or interfaced optimization methods, namely domain subdivision, simplex search,

simulated annealing, BFGS, evolutionary and genetic algorithms, and additional methods can be relatively easily implemented. Jobs for cost function evaluation are then automatically distributed on a computational grid or cluster pool via the *Nimrod/G* or *EnFuzion* dispatchers.

Apart from the command line APIs, there are basically two GUIs for Nimrod: *Active Sheets* provides a spreadsheet interface for Nimrod/G, EnFuzion, and NetSolve. This GUI allows one to perform parallel computations on the grid, managed by spreadsheet software on a local desktop computer. The *Nimrod Portal* represents a web browser interface for Nimrod/G and Nimrod/O. This GUI permits users to set up, submit, and control their parametric experiments as well as to manage their resources and certificates from anywhere in the world. There is no need to port Nimrod to their internet client.

The outputs of the parametric experiments are generally sent back to the users by Nimrod in their native format, mostly as ASCII files. These files can then be used for further analysis and visualization. For example, in our studies we employed OpenDX for parameter space and QMView [4] for molecular model visualization. There is currently an interactive subjective optimization (OI) interface of Nimrod in development, requiring on-line interventions of the user into the parameter optimization process.

All non-commercial parts of the Nimrod distributed parametric modeling package are available from <http://www.csse.monash.edu.au/~nimrod/nimrodg/>. Installation procedures for Unix systems are described there in detail; the only prerequisites are installations of PostgreSQL and Python. We are also developing an automated implementation of the Nimrod toolkit into a roll distribution for the ROCKS high-performance Linux cluster management software.

3. Setup—grid testbeds

One of the major tasks in grid computing studies is to collect the necessary number of computing resources to make a distribution of the calculations reasonable. We are very fortunate to participate in the Pacific Rim Applications and Grid Middleware Assembly (PRAGMA) collaboration of institutions, which contributed to our grid infrastructure. In addition, a few machines are available for use based on local agreements. This has allowed us in our past experiments to construct large, worldwide grid testbeds, in particular for the fourth PRAGMA workshop (PRAGMA4) and the 2003 Supercomputing conference (SC2003), where most of the extended parameter sweeps for the parameterization of the group difference potential (GDP) [6] for biomolecular computations were performed.

As can be seen from the snapshots of job distributions onto machines during the execution of Nimrod experiments at PRAGMA4 and SC2003, Fig. 1, the utilized resources (workstations, clusters and supercomputers) spanned a range of countries, institutions, and administrative domains as well as hardware architectures, operating systems, and queue managers. We based our grid testbed on the Globus toolkit, which provides a uniform interface for resource management, data management, information services, and security via its GRAM, GridFTP, MDS, and GSI components. Since Nimrod needs only to be available on the portal client, the setup on each server machine typically consisted of the following steps: (1) installation and configuration of the Globus toolkit and a certificate authority (CA), if not already existing; (2) setup of the user account and Globus certificate(s) under which the experiments are performed; (3) deployment of the application software.

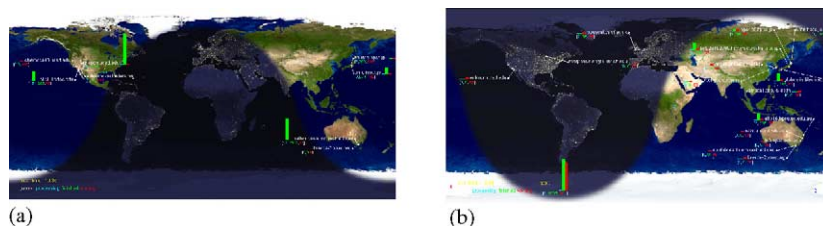


Fig. 1. Job distributions during Nimrod experiments at (a) PRAGMA4 and (b) SC 2003.

Points (1) and (2) often provide difficulties due to differences in policy and system administrator experience between the involved sites. The participation of partners in a grid organization such as PRAGMA significantly eases these steps, due to agreed upon procedures. We also experienced several problems with such things as misconfigured Globus installations and minor Globus bugs. Therefore, the setup of the Globus toolkit on the resources is a critical issue for the success of such grid computing studies. Point (3) is of course dependent on the particular software package. For the GDP parameterizations, we used the quantum chemistry code GAMESS, [7,8] which is available for various hardware platforms and could thus be widely adopted. In practice, not all grid machines actually ran jobs, either due to problems in their software configuration or due to their workload. Fortunately, the highly dynamic nature of the grid is considered in the design of Nimrod, in that the decision about which machines are used on a particular experiment is deferred until execution time. Jobs can be automatically (re)scheduled, and network and node failures, common occurrences on a large computational grid, are adequately handled.

For our ongoing grid developments, we are in the process of establishing a smaller but more permanent grid testbed, based on the Globus and Nimrod packages. The *Resurgence Grid*, established between the University of Zurich and the San Diego Supercomputer Center (SDSC), includes a large four-processor Linux machine as main client and currently three Linux clusters as main servers; further inclusion of an IBM SP3 is planned.

4. Applications and examples

The Nimrod toolkit has been applied to a range of application areas, including bioinformatics, operations research, air pollution studies, laser physics, ecological modeling, discrete optimization and meta-heuristics, electronic CAD digital simulation, ad-hoc wireless network simulation design, public health policy, computational fluid dynamics, X-ray health standards, high gain antenna design, business process simulation, astrophysics and, as illustrated in this work, quantum chemistry. There is in principle no fundamental restriction to which types of parametric modeling problems the Nimrod package can be applied. However, for full

benefit from distributing the calculations on a computational grid, the time spent in the computation part of the problem should outweigh the overhead induced by network transfer, I/O, queuing system scheduling, and the operations of Nimrod itself. Therefore, Nimrod is best suited for distributing many long-running, resource-demanding computations, as is typically the case for computational (bio)chemistry.

5. Parameterization of a group difference potential

Computational studies involving large and complex biological structures often entail numerical experimentation and hypothesis testing. When a parameter space is searched for optimal solutions, the corresponding computational requirements are amplified by several orders of magnitude. Theoretical modeling of extended molecular systems is challenging due to the rather large number of atoms, the vast conformational space which needs to be sampled dynamically, and the fine spatial and temporal resolution required, all needing sophisticated techniques to correctly describe the physical and chemical behavior.

Hybrid quantum mechanics–molecular mechanics (QM/MM) methods [9] are often used for such studies. In such techniques, the large molecular system is divided into two regions. A small, chemically “active” region is described by accurate approaches based on the Schroedinger equation (QM), while the surrounding larger, “inactive” region is treated with more approximate classical force field methods (MM). Unfortunately, the physical concepts behind the two techniques are so different that they cannot be easily coupled. In particular, when chemical bonds are cut between both parts, dangling bonds in the MM region can simply be eliminated. The outermost atoms of the QM region however, which involve the detailed nature of the electrons, would become radical in nature and behave completely different than required if the bonds were simply removed. One way to accommodate such atoms includes the first atom of the MM part as a capping atom in the QM computation, parameterized such that it correctly reflects the properties of the cut bond. This method does not require extended changes in the source code and does not lead to problems with the handling of artificial link atoms.

Zhang et al. recently developed such a “pseudobond” approach for sophisticated QM-based methods [10]. They added an effective core potential (ECP) to a fluorine atom to model the methyl group in a carbon–carbon single bond. Their parameterizations were done for the ethane molecule, CH_3CH_3 , which becomes $\text{CH}_3\text{C}_{\text{ps}}$, ($\text{C}_{\text{ps}} = \text{F}$ with pseudopotential), then tested on ethane derivatives, and later applied to investigate several enzymatic reactions using QM/MM techniques. Unfortunately, we found serious instabilities in the Zhang et al. pseudobond parameterization. Furthermore, pseudoatom ECPs appear rather difficult to parameterize due to the diversity of their target features, the multi-dimensionality of function and parameter space, and the occurrence of local minima in the optimization procedure.

Therefore, we have developed a new effective pseudoatom potential, [5,6] which only deals with the discrepancies between the isoelectronic CH_3 and F groups without exchanging the core. It is thus named “group difference potential” (GDP). The potential also provides direct information about the substituent effects of functional groups, which play a large role in applications from organic chemistry to pharmaceutical drug development. In addition, the potential may be gradually switched on or off, potentially facilitating QM free energy difference determinations. A superposition of two unmodified Gaussian functions turns out to be the most appropriate functional, a format which is already implemented in many QM programs:

$$U_{\text{eff}}(r) = A_1 \exp(-B_1 r^2) + A_2 \exp(-B_2 r^2)$$

This form corrects for differences in electron interaction and basis set through negative (attractive) and positive (repulsive) values for the amplitude coefficients A_1 and A_2 . The positive exponential prefactors B_1 and B_2 specify the radial extent of each term around the fluorine atom; the smaller their value, the less compact the corresponding function, and vice versa.

To analyze the results for each parameter set, a suitable cost function that reduces the differences between the properties of ethane and pseudoethane to a single number is required. Bond lengths, bond angle, dissociation energy, Mulliken overlap populations and atomic charges were selected as independent properties to optimize with respect to. We applied a common level of QM theory (B3LYP/6-31G(d)) using both spherical and Cartesian basis set formats. This results in four

calculations for every A_1, A_2, B_1, B_2 set, two on pseudoethane and two on the corresponding pseudomethyl radical. All computations were done with the GAMESS quantum chemical program package. The resulting 32 properties identified (hydrogen values all appear three times) have diverse units, accuracy, and importance. As such, the differences between actual and target properties x_i and X_i must be weighted appropriately for the overall cost function by applying a normalized least squares expression

$$f(A_1 A_2 B_1 B_2) = \frac{1}{\sum_{i=1}^{32} w_i} \sum_{i=1}^{32} w_i \left(\frac{x_i - X_i}{u_i} \right)^2$$

The “weighting” factors w_i correct for the number of occurrences of each carbon or hydrogen feature. The “unifying” factors u_i reflect the apparent, or desired, accuracy of each property and are chosen from chemical intuition. This equation is evaluated after completion of each tuple of GAMESS jobs, so that the w_i and u_i values can be easily adjusted to later allow straightforward pseudopotential optimizations.

To identify interesting low-cost regions and avoid trapping in local minima, we first scanned portions of the parameter space in their entirety. This task consists of huge numbers of short, uncoupled QM calculations and thus quite suited for grid computing. In the initial experiment at the PRAGMA4 workshop, the four variables were varied between -10 and 10 a.u. in steps of 1 a.u. for A_1 and A_2 , and between 0 and 10 a.u. in steps of 2 a.u. for B_1 and B_2 . Ignoring function symmetry for now, this leads to $15,876 \times 4$ jobs. At the SC'2003 event, we performed even larger parameter sweeps with $53,361 \times 4$ and $60,016 \times 4$ individual calculations.

A snapshot from the initial parameter scan at PRAGMA4 is displayed in Fig. 2. This snapshot shows the complexity of the cost function hypersurface. The isosurfaces reveal a system of connected channels with a rather “bumpy” terrain.

To find appropriate starting points for subsequent optimizations, regions with function values lower than the “zero” model (no GDP or canceling GDP red isosurface in Fig. 2) are of special interest. Several such “local” minima are scattered over the parameter space with no apparent pattern, although further analysis suggests partial linear dependence. Although we were able to find a ‘minimum’ in this particular run, all cost function

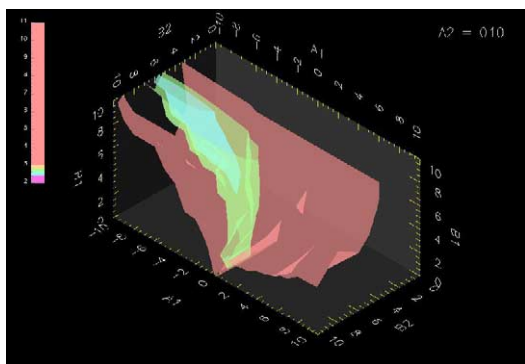


Fig. 2. A single snapshot of the GDP parameter space scanned in the PRAGMA4 experiment. A_1 , B_1 , and B_2 are displayed on the x , y , and z axes; A_2 evolves with the snapshots. Isosurfaces of the cost function f are drawn at 3620 (red), 631 (green), 398 (cyan), and 251 (magenta).

values were still too high to be acceptable. Therefore, in the SC2003 experiments we also performed sweeps with a logarithmic point distribution and a higher grid density around the most interesting region. The deepest minimum appears when a medium-size repulsive and a diffuse attractive Gaussian function are combined to build a maximum potential at the fluorine core and a shallow depression closer to the bond distance of the neighboring carbon atom. We tentatively attribute this to the larger size and smaller electron attraction of a methyl group compared to a fluoro group. With these very large data collections in hand, it turned out that the significance of the best “global” minimum could be remarkably improved when the unit factor for the bond angle was reduced, a fact that can not be easily deduced from chemical intuition. This revealed the most promising parameter region and weighting for subsequent GDP optimizations with Nimrod/O.

In subsequent studies, we used the resulting optimal parameter combinations as starting points for additional grid runs. The Nimrod/O tool, which performs automatic optimization was incorporated to explore the parameter space more intelligently with a number of search heuristics, to find the final optimized GDP. As explained above, we varied variables in the least squares procedure to generate a more funnel-like hypersurface. This will allow minimizations commencing from any remote place to travel towards the global minimum more directly. Overall, this procedure considerably reduces the development time of

GDPs for use in studies of many types of molecular systems and reaction processes using the GAMESS software.

6. Protein–ligand docking studies

In a new project in the field of molecular modeling, we have begun to apply Nimrod/G and Nimrod/O to investigations in protein–ligand docking. Such types of studies are of high importance for drug design in the pharmaceutical industry. The goal here is to scan possible conformations for combined complexes of ligand (often a small organic molecule) and protein (for example an enzyme that needs to be blocked) in space, and to find their optimal position(s) with respect to each other. The tendency for them to bind together is described by a free energy cost function. In the current setup, we use Nimrod to scan through the possible conformations of a test system based on a rotation and translation algorithm, and distribute the corresponding jobs onto a grid of machines. Unreasonable structural configurations resulting when atoms are either too far apart, or too near each other, are excluded. The electrostatic part of the binding free energy is then calculated using an adaptive Poisson–Boltzmann solver, in this case the APBS software [11]. An additional goal in this work is to refine the scoring energy function for this procedure.

7. Workflow and database investigations

The pseudopotential and docking examples demonstrate the suitability of Nimrod for high-throughput calculations in computational (bio)chemistry. Another interesting application is therefore to utilize the procedure to perform calculations for a large set of molecules, putting the results directly into a database. One of the main problems of such an endeavor is that multiple complicated steps are necessary both to prepare the input files for the calculations, as well as to analyze and store the output files into a relational database. This goes beyond the scope of the short pipelines possible to define in Nimrod plan files. However, such preparation and storage steps are not very compute-intensive, so they can be run on a local host. Therefore, we use Nimrod/G just to distribute the resource-demanding

parts of the calculations on a computational grid on basis of simple plan files, which are automatically generated from templates. The control of Nimrod/G itself is embedded in a more complex workflow infrastructure with GUI, based on the Ptolemy II [12] and Kepler [13] systems. The current setup targets the automatic calculation of jobs with GAMESS. However, the overall design is very general such that adaptation to a number of different scientific computation codes should be quite possible.

8. Conclusions

We have illustrated how the Nimrod toolkit has made it possible for us to solve parametric modeling tasks for molecular modeling on computational grids. We were able to parameterize a GDP pseudopotential [5,6] which can be used now for applications in QM/MM or free energy difference calculations, which otherwise would have been very difficult. This has encouraged us to expand our studies into the area of protein–ligand docking. Furthermore, we are currently building a workflow infrastructure on top of Nimrod, which will permit seamless generation, distribution, computation, storage, and analysis of computational (bio)chemistry jobs in an automated, high-throughput manner [14]. In this way, we hope to enable grid computing as an every-day tool for researchers interested in performing molecular modeling studies.

For scientists in general, the Nimrod toolkit simplifies the execution of parameter sweeps and optimizations on a computational grid and thus significantly enhances the usability of the global grid for such studies. Most importantly, it hides the underlying grid middleware and makes the management of the resources and the automation of the jobs user-friendly. However, it cannot and is not meant to solve all the issues associated with setting up a computational grid, such as policy differences between the different involved sites as well as installation and configuration of the grid middleware. In addition, while Nimrod considerably eases the use of grids for repetitive high-throughput calculations with existing scientific software, it is not capable of distributing high-performance calculations with intrinsically grid-parallelized codes. Yet, the diverse hardware, loose network connections, uncontrolled node

failures, and different resource management favors the use of computational grids for parameterization applications.

Acknowledgements

We thank PRAGMA and all organizations that provided computing resources, and their system administrators, in particular the SDSC ROCKs group. We thank J.P. Greenberg and K. Thompson, SDSC, for software installations. K.B. acknowledges NSF-DBI-0078296, ANI-0223043 and NIH-NBCR-RR08605. We thank D. Kurniawan at Monash for visualizations. The Nimrod project is supported by DSTC and GrangeNet, funded in part by Australian Government. D.A. acknowledges the APAC. C.K. and D.N. thank the PRAGMA-PRIME project for funding summer internships at Monash for work described here.

References

- [1] I. Foster, C. Kesselman, *Intl. J. Supercomput. Appl.* 11 (1997) 115.
- [2] Nimrod, 2004. <http://www.csse.monash.edu.au/~davida/nimrod>.
- [3] H. Casanova, G. Obertelli, F. Berman, R. Wolski, *Proceedings of the Supercomputing Conference (SC'2000)*, 2000.
- [4] K.K. Baldrige, J.P. Greenberg, *J. Mol. Graph.* 13 (1995) 63.
- [5] W. Sudholt, K.K. Baldrige, D. Abramson, C. Enticott, S. Garic, *New Generat. Comput.* 22 (2004) 137.
- [6] W. Sudholt, K.K. Baldrige, D. Abramson, C. Enticott, S. Garic, *Lect. Notes in Comput. Sci.* 3036 (2004) 6748.
- [7] M.W. Schmidt, K.K. Baldrige, J.A. Boatz, J.H. Jensen, S. Koseki, M.S. Gordon, K.A. Nguyen, T.L. Windus, S.T. Albert, *Quantum Chem. Program Exch. Bull.* 10 (1990) xx.
- [8] M. Schmidt, K.K. Baldrige, J.A. Boatz, S. Elbert, M. Gordon, J.H. Jensen, S. Koeski, N. Matsunaga, K.A. Nguyen, S.J. Su, T.L. Windus, M. Dupuis, J.A.J. Montgomery, *Comp. Chem.* 14 (1993) 1347.
- [9] J. Gao, *Acc. Chem. Res.* 29 (1996) 298.
- [10] Y. Zhang, T.-S. Lee, W.T.J. Yang, *Chem. Phys.* 110 (1999) 46.
- [11] N.A. Baker, D. Sept, S. Joseph, J.A. McCammon, *Proc. Natl. Acad. Sci. USA* 98 (2001) 10037.
- [12] Ptolemy II, 2004, <http://ptolemy.eecs.berkeley.edu/ptolemyII/>.
- [13] <http://Kepler.ecoinformatics.org/>, <http://www.Kepler-project.org/>.
- [14] K.K. Baldrige, J. Greenberg, W. Sudholt, I. Altintas, K. Bhatia, S. Mock, C. Amoreira, Y. Potier, *IEEE on Grid Comput.* 2004.



Wibke Sudholt is a postdoctoral fellow with Kim K. Baldridge at the University of Zürich, Switzerland, and also recently joined the CoLab under Petros Koumoutsakos at ETH Zürich half time. In 2001, she obtained her doctoral degree (Dr. rer. nat.) with Wolfgang Domcke at the Heinrich-Heine-University Duesseldorf, Germany, for theoretical studies of a charge-transfer process. She then was a DAAD postdoctoral fellow with J. Andrew McCammon and Kim

K. Baldridge at the University of California, San Diego. Her current interests include hybrid methodologies combined with grid computing infrastructure and scientific workflow environments for computational chemistry applications.



Kim K. Baldridge is a theoretical and computational chemist with expertise in the design, development, implementation, and application of computational methodology, for understanding chemical and biochemical reaction processes. Efforts include development of computational tools and associated grid technologies for the broader community. She has over 100 publications and is a co-author on the widely used computational chemistry software, GAMESS. Baldridge is

a Fellow of the AAAS and the APS, and holds the 2000 Agnes Fay Morgan Award. She is currently a Professor of Theoretical Chemistry and director for grid applications at the University of Zürich, and a Distinguished Scientist at the San Diego Supercomputer Center, California.



David Abramson has been involved in computer architecture and high performance computing research since 1979. Abramson is currently a Professor of Computer Science in the School of Computer Science and Software Engineering (CSSE) at Monash University, Australia. Prior to joining Monash University in 1997, he has held appointments at Griffith University, CSIRO, and RMIT. He is a project leader in the Co-operative Research Centre for Dis-

tributed Systems Nimrod Project and also Chief Investigator on two ARC funded research projects (Guard and GriddLeS). Abramson has chaired a number of international conferences. He has published over 100 papers and technical documents.



Colin Enticott completed B Comp (Hons) July, 2002 at Monash University. His project, which he did under the supervision of Professor David Abramson, “The Multi Site EnFuzion Client” dealt in the area of cluster -of-clusters computing that has lead him into Grid computing. Hired by Distributed Systems Technology Centre, the majority of his work is with assisting scientists using their applications with Nimrod and developing the user front-end of Nimrod, the Nimrod Portal.



Slavisa Garic completed a Bachelor of computer Science (Hons) degree at Monash University, Australia in November 2001. His project, Suburban Area Networks: Security, involved working on security aspects of wireless community/suburban networks. At the start of year 2002 he joined Distributed Systems Technology Centre, Melbourne Australia, where he currently works as a core Nimrod/G developer.

Chris Kondric is currently an undergraduate student at the University of California, San Deigo. Over the summer of 2004, he spent 6 weeks at the University of Monash under the PRAGMA-Prime program, working with the Nimrod team on the development of computational grid tools for molecular docking studies.

Duy Nguyen is currently an undergraduate student at the University of California, San Deigo. Over the summer of 2004, he spent 6 weeks at the Monash University under the PRAGMA-Prime program, working with the Nimrod team on the development of computational cluster application software, in particular the Nimrod ROCKS roll toolkit.