Drug Discovery Today • Volume 13, Numbers 13/14 • July 2008

# Grid computing in large pharmaceutical molecular modeling

**Brian L. Claus and Stephen R. Johnson**

Bristol-Myers Squibb Company, H23-07, P.O. Box 4000, Princeton, NJ 08543-4000, United States

**Most major pharmaceutical companies have employed grid computing to expand their compute resources with the intention of minimizing additional financial expenditure. Historically, one of the issues restricting widespread utilization of the grid resources in molecular modeling is the limited set of suitable applications amenable to coarse-grained parallelization. Recent advances in grid infrastructure technology coupled with advances in application research and redesign will enable fine-grained parallel problems, such as quantum mechanics and molecular dynamics, which were previously inaccessible to the grid environment. This will enable new science as well as increase resource flexibility to load balance and schedule existing workloads.**

Modern computational techniques in the pharmaceutical industry have placed high demands on the compute resources available in discovery. Coupled with an industry-wide push for greater cost efficiency, most major pharmaceutical companies [1–3] have employed grid computing to extract value more efficiently from their existing desktop computers. The term grid computing has evolved to include three primary types of grids: compute grids, data grids and knowledge grids. This paper will use the term grid computing to refer to compute grids [4]. Most molecular modeling applications utilizing grid computing are limited to problems that require minimal interprocess communication. Virtual screening [5], in which a large number of compounds are processed independently across several computing machines, is a good example of such a coarse-grained parallel problem. Although useful and important, there has been a dearth of other grid applications in pharmaceutical discovery. This lack of applications that benefit from coarse parallelization is a key limiting factor to the widespread use of grid computing in the pharmaceutical industry. Most applications in computational chemistry require significant communication between processors, such as that provided by compute clusters, to benefit from parallel processing power. The distinction between a cluster and a grid is somewhat ill-defined. Box 1 highlights several key differences distinguishing clusters and compute grids.

*Corresponding author:* Claus, B.L. (brian.claus@bms.com)

In addition to the lack of suitable applications, a common criticism of grid computing is the added complexity to the infrastructure that is required to perform these calculations. Indeed, porting tools to the grid infrastructure is a high activation barrier to success. Some have argued that, with the falling prices of commodity hardware, one could simplify the infrastructure through the purchase of servers and clusters to obtain an equivalent amount of processing power, or alternatively use external on-demand compute resources. In our experience, the utilization of external compute capacity in informal industrial research projects has been limited because of issues such as those highlighted in Box 2. As large corporations have already purchased desktop machines that sit mainly idle, combining these machines into a computational grid at minimal cost makes good business sense assuming the complexity can be controlled. Indeed, the corporate user greatly benefits in this scenario as the centralized IT group upgrades desktop and laptop productivity machines. These upgrades typically come without cost to the specific department running calculations on a desktop grid, and yet they result in increased performance for the grid. This, in turn, results in more compute capacity for the grid user without any increased cost. More recently, current desktop upgrades include the migration from single core to multi-core processors, which has a dramatic impact on the desktop grid capacity.

Modern grid technology has evolved in the past five years to address the perceived complexity and application issues. Modern

1359-6446/06/$ - see front matter © 2008 Elsevier Ltd. All rights reserved. doi:10.1016/j.drudis.2008.03.006

BOX 1

### Terminology: grid versus cluster

Highlighting the difference between grid computing and a more 'traditional' cluster is not as trivial as one might surmise. Several different definitions have been put forth in the literature. One simple definition put forth by Professor Coveney is 'Grid computing is distributed computing performed transparently across multiple administrative domains' [32]. Regardless of the exact definition, several primary differences between grids (see http://www-fp.mcs.anl.gov/~foster/Articles/WhatIsTheGrid.pdf) and clusters can be listed:

- *Grid*:
  1. Open standards.
  2. Decentralized control.
- *Cluster*:
  1. Closed system without access to complete suite of resources on a local network.
  2. Managed by single entity.
  3. Frequently include high speed interconnects between nodes.

It is interesting to note that the availability of high speed Ethernet has diminished the bandwidth advantage held by high speed interconnects such as Myrinet or Infiniband. Of course, latency is also crucial to the performance of applications that are truly parallelized and low latency connectivity dramatically improves performance.

approaches to grid computing no longer treat the individual processors as isolated machines; instead, they exploit key facets of private industry grids to allow problems with higher levels of intercommunication [6–8] and to address high data transfer requirements. We will discuss some of these innovations in grid technology and their implications for use in pharmaceutical discovery.

## Current applications

Grid computing has clearly been embraced by the computational chemistry community. This is especially true for research groups in academia that are taking advantage of inexpensive compute resources such as the computational chemistry grid [9]. Several manuscripts relating to the development of grid infrastructure specifically geared to use by the computational chemistry community have recently been published [9–12]. These efforts have attempted to address many of the difficulties surrounding porting modeling software to grid architectures.

Virtual screening applications are probably the best-known molecular modeling tools employed in grid computing. The Screensaver Project [5] is one of the earlier virtual screening applications using grid computing. This project, similar in many respects to the SETI@home project (http://www.setiathome.berkeley.edu/), utilizes an application that employs a scoring function based on pharmacophore matching to look for potential drug candidates. The screensaver project performed virtual screening on several targets in oncology using more than 1.5 million PCs recruited from over 200 countries.

One application of virtual screening on enterprise grids appeared in a recent report using FlexX and AutoDock to screen for inhibitors of plasmepsin [13], a potential target for the treatment of malaria. The authors report the identification of several new compound classes for the inhibition of plasmepsin, along with the identification of compounds in several well-known chemotypes. Although these *in silico* results have not yet been confirmed *in vitro*, the application demonstrates the potential utility of grid computing in virtual screening.

Typically, virtual screening approaches employ algorithms that have been designed to sacrifice accuracy in return for greater computational speed through the use of empirically generated scoring functions. Recently, there have been many publications on using scoring functions that are 'more physical' in their design. One such application is the use of MM-PBSA on an enterprise grid-computing environment at Abbott Laboratories [14,15]. The authors make use of the freely available Condor [16] grid package to scavenge unused CPU cycles from employee desktop computers to calculate protein–ligand-binding affinities. In a typical snapshot in this study, the grid is composed of 112 machines with 69 available for jobs. The remaining 43 machines were in use by their employee–owner. In their implementation the MM-PBSA job is broken down into a series of sequential steps allowing the binding energy of a single complex to be calculated in 100–200 min. At this throughput, the authors report that approximately 110 structures could be processed per wall-clock-day using their pilot grid.

Fowler *et al*. [17] report a method for calculating relative binding free energies using thermodynamic integration. The computation can be completed in less than one week by using grid computing to run the individual molecular dynamics simulations simultaneously. They also demonstrate the results were not particularly sensitive to the presence of unequilibrated data used as a result of running the simulations in parallel, as compared to the more common sequential approach. The method was applied to study a series of phosphopeptides that bind to the Src SH2 domain.

BOX 2

### Barriers to on-demand computing for industrial research

- Using on-demand computing resources incurs a real financial cost. Many research projects begin 'under the radar,' and cannot justify financial expenditure to achieve proof of concept. Cycle-scavenging on existing, internal hardware can be done for 'free' to the researcher.
- Data security presents a problem. Either a formal arrangement with the utility computing vendor must pre-exist or a scientist would be required to obtain authorization to run calculations on 3rd party hardware and be able to guarantee the data security and integrity.

These issues present too high of an activation energy barrier for most informal research projects.

## Hype and hurdles

If grids were truly and completely autonomous, selfhealing, self-provisioning, selfconfiguring, selfoptimizing infrastructure [18] as advertised, then why haven't they revolutionized research computing? The answer lies in complexity.

Grid infrastructures excel at hiding the underlying heterogeneity in a computing environment [18]. However, at some point, these heterogeneities must be taken into consideration. As a computer scientist porting an application to the grid infrastructure, the binary executable must be compiled for all operating systems and platforms to enable code execution. For some applications this is a trivial undertaking, but for others with platform-specific assumptions built into them it can represent a significant stumbling block. Fortunately for the application scientist trying to utilize the grid application, the specific hardware and operating system choices are hidden and the user is presented with a simplified view.

Troubleshooting and monitoring in a grid environment can also be more complicated relative to a homogeneous cluster or large server environment. Dealing with varying flavors of operating systems, different and frequently changing patch levels, and the impact of security updates potentially resulting in network misconfigurations can be difficult to diagnose and resolve for the grid administrator [19]. There are several aspects to monitoring that are important to consider. It is imperative to monitor the availability and status of the various compute nodes. Grids can be notorious for having machines drop out or jobs never returning. There are many reasons a machine will drop out from the grid including laptops being taken off the network, system crashes and users turning machines off at night. The recent attention on 'go green initiatives' to reduce power consumption has resulted in a significant decrease in off-hours availability of desktop machines resulting from conscientious employees turning them off. Additionally, some of these cost-cutting initiatives have focused on reducing the total number of machines supported by the corporation by ensuring employees only have either a desktop or laptop PC. Employees having the only option of selecting the laptop have increased the variability in the grid pool. There are also several nontechnical issues relating to the deployment of a grid infrastructure in an industrial setting that are discussed in Box 3 [20].

**BOX 3**

**Nontechnical barriers to corporate grid deployment**

In our experience, many issues that concern corporate IT groups with respect to grid computing are not technical, but social. These issues partially arise because corporate grid participation is not necessarily voluntary. When running on an isolated server, a scientist does not need to worry about impacting others in the organization except for their consumption of the processor cycles. When running on a cycle-scavenging desktop grid infrastructure, the scientist runs the risk of negatively impacting other users in the company. Consider:

- *Noise problems*. Newer desktop PCs have variable speed controlled fans. When they sit idle, the machines are essentially silent, however, when running grid jobs, the fans speed up and they generate more noise. Taken to the extreme of a room full of cubicles, having many computer fans turn on simultaneously can be disconcerting to the residents.
- *Executives' machines*. The grid infrastructure may not distinguish an entry level employee's computer versus the CIO's computer. It would be bad to negatively impact a presentation from an executive's laptop with a misbehaving grid job.
- *What's in it for me*? Often, the owner of the desktop PC does not have a vested interest in seeing the result of the calculation. Providing some form of positive feedback for the contribution of CPU cycles can help win over an apprehensive PC owner.
- *Blame the grid*. Any problems encountered by a desktop PC owner are blamed on the grid.

Identifying and resolving desktop PC issues can be problematic because continuous monitoring of a large number of machines at the levels required for thorough troubleshooting results in performance degradation. This requires an administrator to enable additional monitoring and logging after a problem has been reported for a client machine; however, it is frequently too late to witness the problem. Monitoring also includes the ability of the scientist to see the status of their job, view its progress and potentially even peek at the output as it is being generated. This is a capability that scientists are accustomed to the cluster environments, but it becomes much more difficult when the remote machine is an unknown entity, potentially running a different operating system and even cases where the user does not have log-in privileges. The process of porting an application to the grid must include additional layers of monitoring, error detection and resubmission, as well as result integration. It is also important for the controlling process to be able to determine whether a job has failed because of a timeout or because the client machine vanished. Infrastructures that can distinguish these failures can act more appropriately with respect to resubmitting the job if appropriate.

The tuning of application parameters presents challenges as well. The runtime of some applications span several orders of magnitude depending on the options selected by the users. Gaussian [21], a popular commercial quantum mechanics package, is an example where the input parameters dramatically affect the runtimes. Consider a single-point energy calculation versus a complete geometry optimization for the same molecule. This complicates making these types of applications generally available to a broad user community. If the users are inexperienced they can submit huge jobs with unintended results including the collapse of the grid infrastructure. The process that divides up an input set into executable pieces to be distributed out to workers also needs to know something about the potential impact of runtime options to ensure proper splitting of the input. The QCGrid [22] from the Tsukuba Advanced Computer Center has attempted to solve this problem by maintaining a database of input and output data as well as computation times for completed jobs. This allows them to estimate the required computation time for future job submissions [22]. Unfortunately, there are other circumstances where the same input parameters result in drastically different runtimes, such as the starting conformation when computing a full quantum mechanical geometry optimization.

Licensing also serves as a roadblock to moving many applications to a grid infrastructure. Typically, there are 3rd party applications that an organization may want to run on their grid, but the licensing schemes are not amenable to the grid-computing paradigm at a reasonable cost because of the large number of available processors. Common license arrangements include nodelocked licenses, named user licenses, token based or number of concurrent processor licensing, and site- or enterprise-based licensing. It is common for grid applications to be licensed using a token or concurrent processor mechanism or the site/enterprise concept. Some vendors are acknowledging the scale of grid computing and apply discounts when they license their products for these resources. However, some vendors simply take a 'per processor' license fee and multiply it by the number of grid nodes. This can lead to exorbitantly high prices for grid availability. Licensing remains a crucial component in determining whether or not an application is destined to be rolled out onto a corporate grid.

One aspect of grid computing that is not highly focused on is the behavior of the grid systems when the infrastructure becomes overloaded. Three competing factors must be balanced in the design of a well-behaved job submission: the number of units; the execution time for each unit and the data package to be delivered for each unit. The number of units will dictate the load upon the submission server. The execution time for each unit may be particularly important for grids composed of laptops that may disconnect from the grid at the end of the workday. The load placed upon the network will be related to the size of the data package required for each unit.

As an example, imagine a pairwise similarity calculation involving a million molecules. This job would require 500 billion calculations. In the extreme, this task could be divided into 500 billion separate units for submission to the grid. Such a division of labor would place an extraordinary burden on the submit server. However, each unit would have a very small data package to be distributed over the network to the compute servers, and the individual units would have short execution times. By contrast, dividing the job into two work units places almost no burden on the submit server, but would create units with very large data packages for delivery to the compute servers as well as unacceptably long runtimes. The user needs to consider these factors carefully when determining how to break the entire job into units.

Many commercial grid infrastructure packages have tried to address issues surrounding the data package burden placed upon the network infrastructure. In its simplest form, a grid job is distributed to an execution host as a selfcontained package, including the application executable and all required datasets to complete the calculation. To increase performance and reduce network congestion, some commercial infrastructure packages provide automatic on-the-fly compression of the input and output data, as well as the ability to cache frequently used input databases. When the execution scheduler tries to find a suitable location to run a job it can preferentially select a client that already has the required input data cached, eliminating the need to transfer it again. A second approach would be to ensure that applications are installed on all client machines *a priori*. This reduces data transfer and payload sizes, but can be problematic in corporate environments with heavily regulated desktop images. At the other extreme, there are approaches that incorporate data placement services to deal with the issue of staging and moving large amounts of data. This approach separates the notion of job execution from ensuring the data are readily available for the calculation [23].

Private grids inside pharmaceutical companies can employ other strategies to mitigate issues with moving large amounts of data. For instance, in a global grid situation one must pass all data back through the grid infrastructure to return it to the calculation originator. This is necessary because the grid infrastructure is the only path available to reach the compute node. In a private corporate grid one can shortcut this return mechanism and move data to its final destination directly, using simple tools such as SCP for remote file copies, the Globus Access to Secondary Storage (GASS) component of the Globus toolkit [24], or abstractions on top of the Globus toolkit such as Simple API for Grid Applications (SAGA) [6]. This can eliminate significant strain on the grid infrastructure as the grid is not responsible for relaying the data or results back to the user.

## Emerging technologies and future applications

Workflow tools have existed for decades with the intent of enhancing the user experience and facilitating the sharing of scripts, processes and best practices. There are many workflow tools available, commercial [25] and open source [26], including some focused specifically on computational chemistry integrating grid resources [11,27]. These tools aim to minimize the complexity apparent to the user of grid computing by incorporating job submission directly to the grid in a seamless fashion from existing workflows. One approach is to make the grid applications available to the workflow as web services [27]. This approach, however, may have significant difficulty in dealing with fault tolerance issues. Enhanced error handling is mitigating these problems by providing more extensive diagnostic information, such as the ability to distinguish calculation-related issues (e.g. timeout, convergence problems) from infrastructure-related issues (e.g. machines getting turned off, disks filling up, machine owner putting the job to sleep).

Message Passing Interface (MPI) is one of the standard technologies for implementing fine grain parallel algorithms that was only recently enhanced for use on grids [28]. However, the dynamic number of worker nodes inherent in grid computing presents a challenge because traditional parallel programming environments mandate the number of subprocesses that remain constant throughout an entire execution. New middleware tools, such as P2P-MPI [29] simplify these challenges for the programmer by incorporating solutions for fault tolerance in distributed applications. There are two primary approaches to this type of fault tolerance, one is to run multiple replicas of each process simultaneously and the other is to resubmit processes when they fail. Typically, this approach is relevant to independent work units that do not require any notification of their existence to a master control process for the application. For an MPI application to run multiple replicas simultaneously it must accept the result from the first completed replica and then halt or ignore results from the redundant replicas. The advantage of using simultaneous replicas is that the master process does not stall while waiting for the failed process to restart from the beginning. The obvious disadvantage is wasted resources for the replica processes that become unnecessary when there are no failures. The tuning of the number of replicas to run simultaneously, as well as the decision to run replica type algorithms versus resubmission algorithms, is clearly specific to the reliability of the nodes in each grid installation.

Theoretically, the availability of grid computing should enable one to routinely perform innovative science that is too computationally demanding in traditional computing environments based on capacity, throughput or contention for resources with other users. If the employment of grid technologies does not enable new scientific approaches, it probably makes sense to invest resources in hardware and processors that reduce the complexity of the computing environment. There is considerable progress being made in the academic research and supercomputing communities that will help push the frontiers of scientific research in the industrial molecular modeling community. Among these is the PS3Grid (http://www.ps3grid.net) that combines BOINC (http://www.boic.berkely.edu) with the cell processor to perform full-atom molecular dynamics simulations. However, this is a subtly different approach than employing existing, commodity desktop PCs to increase compute capacity.

Although there have been massive increases in computing power, there are still many types of calculations that remain untenable to calculate as an exact solution. As such, it is appealing to have a flexible mechanism to select acceptable approximations and key aspects of a calculation where additional compute time is justified in exchange for answers that are more reliable. The ability to combine different levels of rigor and types of simulation, such as molecular dynamics, quantum mechanics, molecular mechanics and molecular docking, is an industry challenge that is being addressed with a multilevel simulation framework by Sun *et al.* [7]. Currently, high-level molecular dynamics trajectories for proteins can consume a compute cluster for days. Considering most companies are pursuing multiple protein targets simultaneously and continue to be resource constrained, making appropriate approximations in exchange for reduced computational time can be advantageous. The multilevel simulation framework demonstrates and validates the approach of utilizing coarse approximations with periodic, highly rigorous calculations to generate high-quality protein structures that can be combined with small-molecule docking.

Another computationally expensive method is performing quantum mechanical calculations across an entire protein. The fragment molecular orbital (FMO) method [8] makes such a calculation more reasonable, albeit straining on compute resources. GridFMO [8] is an implementation of the GAMESS FMO calculation coupled with appropriate middleware to handle remote communications that demonstrates the ability to perform these calculations in a robust manner over distributed, heterogeneous resources. The iterative selfconsistent charge cycle utilized in the GridFMO implementation requires more communication, load balancing and synchronization during the job than is common in current molecular modeling industrial grid applications. This advance may enable full protein QM on a practical level without the need for expensive super-computer access.

Expanding compute resources are shifting the expectation from static, single image models of drug–protein interactions to models that incorporate motion and use it to help explain observations. Replica exchange is a technique to allow for more rapid and thorough sampling of an energy surface for protein dynamics. An appealing feature of replica exchange is that each replica can be computed independently of the other replicas until the exchange step is reached, making it amenable to grid computing. Such an application is being explored at Rutgers University [30] by using Comet-G with IMPACT, a molecular dynamics package from Schrodinger. Comet-G is a multiplatform infrastructure for handling grid processes, communication between processes and enabling data exchange. An interesting aspect of this approach is the reliance on Chord [31] for lookups to allow worker nodes to join and leave the simulation dynamically, providing it with fault tolerance capabilities. This is also an important aspect for situations where the grid may include a large number of laptop machines that go home at night. When they return in the morning they can rejoin the simulation and increase the throughput of the calculation.

Applications such as this one rely on increased communication between worker nodes relative to current applications. This communication is achieved through implementations of standards, such as the Globus toolkit [24]. Historically, industry has shied away from these low-level toolkits preferring a commercial, off-the-shelf solution for grid infrastructure. However, it is probably that the integration of these lower level toolkits coupled with existing production grid environments is going to be essential for unlocking the grid potential for industrial scientific computing.

## Conclusion

The future of grid computing in computer-assisted drug design (CADD) hinges on the ability of an organization to identify potential applications to be migrated to the grid to demonstrate utility and value. The historic approach on only considering ultra-coarse-grained parallel programs is too restrictive, as evidenced by the relative lack of industrial applications of grid computing appearing in the molecular modeling literature. This scarcity of industrial publication is symptomatic of the complexity and limitations of grid computing as it has been implemented in large pharmaceutical companies. Published industrial applications have yet to report a CADD example that was significantly enabled by the incorporation of grid computing beyond techniques accessible with previously existing compute resources. To reach this goal, industry needs to embrace some of the newer technology coming from academia and the broader research community to facilitate the implementation of finer grained parallel applications on their grid resources. These technologies can be layered on top of the existing grid infrastructure, enabling new applications without disrupting existing projects. Combining these technological advances with the changing hardware landscape of multicore processors will allow grid computing reach its potential of encouraging scientists within CADD to view problems in new ways.

## References

1 (2004) GlaxoSmithKline: implementing breakthrough technology for clinical development modeling and simulation. *United Devices Case Study*

2 (2005) *GARS report five: grid computing – adoption in the pharmaceutical sector*. The 451 Group

3 Rowell, J. (2003) Distributed desktop grid, PC refresh, help Novartis enhance innovation. *Intel Business Centre Case Study*

4 Konagaya, A. (2006) Trends in life science grid: from computing grid to knowledge grid. *BMC Bioinformatics* 7 (Suppl. 5), S10

5 Richards, W.G. (2002) Innovation: virtual screening using grid computing: the screensaver project. *Nat. Rev. Drug Discov.* 1, 551–555

6 Goodale, T.S. *et al.* (2006) SAGA: a simple API for grid applications. High-level application programming on the Grid. *Comput. Methods Sci. Technol.* 12, 7–20

7 Sun, Y. *et al.* (2007) Integrating multi-level molecular simulations across heterogeneous resources. *8th IEEE/ACM International Conference on Grid Computing (Grid2007)* IEEE

8 Ikegame, T. *et al.* (2007) GridFMO – quantum chemistry of proteins on the grid. *8th IEEE? ACM International Conference on Grid Computing (Grid2007)* IEEE

9 Dooley, R. *et al.* (2006) From proposal to production: lessons learned developing the computational chemistry grid cyberinfrastructure. *J. Grid Comput.* 4, 195–208

10 Sanna, N. *et al.* (2007) Gaussian grid: a computational chemistry experiment over a web service-oriented grid. *Theor. Chem. Acc.* 117, 1145–1152

11 Sild, S. *et al.* (2006) Open computing grid for molecular science and engineering. *J. Chem. Inf. Model.* 46, 953–959

12 Sun, Y. *et al.* (2007) ABCGrid: application for bioinformatics computing grid. *Bioinformatics* 23, 1175–1177
13 Kasam, V. *et al.* (2007) Design of new plasmepsin inhibitors: a virtual high throughput screening approach on the EGEE grid. *J. Chem. Inf. Model.* 47, 1818–1828
14 Brown, S.P. and Muchmore, S.W. (2006) High-throughput calculation of protein–ligand binding affinities: modification and adaptation of the MM-PBSA protocol to enterprise grid computing. *J. Chem. Inf. Model.* 46, 999–1005
15 Brown, S.P. and Muchmore, S.W. (2007) Rapid estimation of relative protein–ligand binding affinities using a high-throughput version of MM-PBSA. *J. Chem. Inf. Model.* 47, 1493–1503
16 Thain, D. *et al.* (2003) Condor and the grid. In *Grid Computing: Making the Global Infrastructure a Reality* (Berman, F. *et al.* eds), John Wiley
17 Fowler, P.W. *et al.* (2007) Rapid, Accurate, and precise Calculation of Relative Binding Affinities for the SH2 Domain Using a Computational Grid. *Journal of Chemical theory and Computation* 3 (3), 1193–1202
18 Jiménez-Peris, R. *et al.* (2007) Enterprise grids: challenges ahead. *J. Grid Comput.* 5, 283–294
19 Neocleous, K. *et al.* (2006) Grid reliability: a study of failures on the EGEE infrastructure. In *Proceedings of the CoreGRID Integration Workshop 2006* (Gorlatch, S. *et al.* eds), In pp. 61–68, Academic Computer Centre CYFRONET AGH
20 Claus, B.L. *et al.* (2007) FACT – fragment activity comparison tool. *The 8th IEEE/ACM International Conference on Grid Computing (Grid2007)*
21 Frisch, M.J. *et al.* (2004) *Gaussian 03, Revision C.02*. Gaussian, Inc.
22 Nishikawa, T. *et al.* (2003) Design and implementation of intelligent scheduler for Gaussian portal on quantum chemistry grid. In *International Conference on Computational Science – ICCS 2003 (Vol. 2659)* (Sloot, P.M.A. *et al.* eds), pp. 244–253, Springer-Verlag
23 Chervenak, A. *et al.* (2007) Data placement for scientific applications in distributed environments. *8th IEEE/ACM International Conference on Grid Computing (Grid2007)* IEEE
24 Foster, I. and Kesselman, C. (1997) Globus: a metacomputing infrastructure toolkit. *Int. J. Supercomput. Appl.* 11, 115–128
25 Pipeline Pilot. Scitegic, Inc.
26 Konstanz Information Miner (KNIME). University of Konstanz
27 Lehtovuori, P.T. and Nyroenen, T.H. (2006) SOMA – workflow for small molecule property calculations on a multiplatform computing grid. *J. Chem. Inf. Model.* 46, 620–625
28 Karonis, N.T. *et al.* (2003) MPICH-G2: a grid-enabled implementation of the message passing interface. *J. Parallel Distr. Comput.* 63, 551–563
29 Genaud, S. and Rattanapoka, C. (2007) P2P-MPI: a peer-to-peer framework for robust execution of message passing parallel programs on grids. *J. Grid Comput.* 5, 27–42
30 Li, Z. and Parashar, M. (2007) Grid-based asynchronous replica exchange. *8th IEEE/ACM International Conference on Grid Computing (Grid2007)* IEEE Computer Society Press
31 Stoica, I. *et al.* (2001) Chord: a scalable peer-to-peer lookup service for Internet applications. In *Proceedings of the SIGCOMM* pp. 149–160
32 Coveney, P.V. (2005) Scientific grid computing. *Philos. Trans. Roy. Soc. A: Math. Phys. Eng. Sci.* 363, 1707–1713