Hydra: A Self Regenerating High Performance Computing Grid for Drug Discovery

Drew Bullard,* Alberto Gobbi, Matthew A. Lardy,[†] Charles Perkins,[‡] and Zach Little[§]

Anadys Pharmaceuticals, San Diego, California 92121

Received October 31, 2007

Computer aided drug design is progressing and playing an increasingly important role in drug discovery. Computational methods are being used to evaluate larger and larger numbers of real and virtual compounds. New methods based on molecular simulations that take protein and ligand flexibility into account also contribute to an ever increasing need for compute time. Computational grids are therefore becoming a critically important tool for modern drug discovery, but can be expensive to deploy and maintain. Here, we describe the low cost implementation of a 165 node, computational grid at Anadys Pharmaceuticals. The grid makes use of the excess computing capacity of desktop computers deployed throughout the company and of outdated desktop computers which populate a central computing grid. The performance of the grid grows automatically with the size of the company and with advances in computer technology. To ensure the uniformity of the nodes in the grid, all computers are running the Linux operating system. The desktop computers run Linux inside MS Windows using coLinux as virtualization software. HYDRA has been used to optimize computational models, for virtual screening and for lead optimization.

INTRODUCTION

Tools for computer aided drug design (CADD) are an increasingly important part of modern drug discovery. Methods such as docking, shape superposition, and quantitative structure− activity relationship (QSAR) analysis, have been shown to be very useful in the identification of new leads.[1] More recently, new techniques for predicting binding energies have been developed that show better correlation with experimental values and thus are more valuable in lead optimization. These new techniques (MM-P(G)BSA,[2] FEP,[3] and QM/MM[4]) are quickly becoming tools of choice for accurately predicting ligand binding affinities. These methods are computationally expensive and generally require a grid if they are applied on a large scale. Recently, Muchmore and Brown[5,6] described the implementation of the MM-PBSA method on a compute grid using Condor[7] and Amber.[8]

So far, the access to computational tools for drug discovery has been limited to specialists. To increase the accessibility of CADD tools in the drug discovery process and create additional synergies between computational and medicinal chemists, we have implemented an infrastructure which allows all scientists to submit and analyze computational jobs directly from their desks. This infrastructure consists of Illuminator,[9] a web based user interface for preparing and submitting computational jobs, and Autocorrelator, an automated model generator and optimizer. Both of these applications demand large amounts of CPU time that a computational grid can efficiently provide.

There are several variations of computational grids[10] that were considered:

(1) a grid comprised of dedicated computers.

(2) a grid using the unused computer time on desktop computers inside the company.

(3) a combination of both.

Alternative 1 is the most easily implemented because the computers are dedicated to perform their task—no special integration or software is required. Alternative 2 does not require purchasing of new hardware and has the advantage that it grows automatically with the company. The typical corporate user will only utilize about 5% of the compute resources available on the desktop,[11] leaving 95% of the CPU resources unused. Alternative 2 also automatically takes advantage of advances in computer chip manufacturing as new computers are rolled out to end-users in the company. There are multiple frameworks available that facilitate the implementation of background computational jobs on desktop computers, examples include Condor[7] and Globus.[12] Examples of worldwide adaptation include Folding@Home[13] and Seti@Home[14] both of which use the CPU power from thousands of computers worldwide. To date, background grid computing requires the development of custom grid computing software that is designed to run together with the framework used. This prevents the easy integration of commercial software for which the source code is not available. The requirement to create custom software has kept this technology out of reach for many users.

Significant progress has been made recently in virtualization of computer hardware. Several open[15–17] and closed source[18,19] solutions are available to implement virtualization. Using virtualization, one computer can run multiple virtual machines each of which acts as a separate computer with a unique instance of an operating system. The CPU and I/O resources are shared between the virtual machines. Santosa and Schaefer[20] describe the implementation of a compute grid using Cooperative Linux[17] (coLinux). CoLinux runs as a service under the native operating system. The service

* Corresponding author. E-mail: dbullard@anadyspharma.com.
[†] Present address: Takeda San Diego, 10410 Science Center Drive, San Diego, CA 92121.
[‡] Present address: CAP, 3471 Paseo De Brisas Unit 33, Oceanside, CA, 92056.
[§] Present address: GNF, 10675 John Jay Hopkins Drive, San Diego, CA 92121.

provides a virtual computer running the Linux operating system so that any program developed for Linux can be executed.

Here we describe our implementation of alternative 3, a hybrid approach using spare cycles on users' desktops along with a set of dedicated computers. The dedicated portion of the grid is built from recycled desktop computers that were scheduled for replacement and are now dedicated to grid computing. To create a homogeneous environment, all grid computers run the Linux operating system. The dedicated computers run Linux natively. The desktop computers use coLinux to run Linux as a service in the background of the natively installed Windows operating system.

This solution results in a cost-effective grid requiring no custom programming and that is easy to manage. The CPU's of the dedicated computers in the grid are not the newest models but over time they will be replaced. The current Anadys grid has more than 165 P4 CPU's with clock rates between 1.2 and 3.6 GHz. This approach is inexpensive, and can easily be implemented in any organization.

IMPLEMENTATION

Creating a Grid of Recycled Desktop Computers. The opportunity to create a self-regenerative computational grid presented itself during a routine companywide desktop upgrade project. After evaluation of costs, manageability, return on investment and research needs the decision was made to build a proof of concept compute grid utilizing the desktops recycled during the upgrade.

Building a grid from the old desktop computers for the proof of concept was very appealing compared to other more costly alternatives. The targeted applications (OMEGA,[21] ROCS,[22] FRED,[23] SZYBKI, and GOLD[24–26])[27] are mainly CPU bound and do not require large amounts of memory or high performance disk and network I/O. Parallelization of the planned computation of molecular properties for many molecules was also expected to be trivial since each molecule can be computed independently on a separate CPU. These features make a Beowulf[28] type grid the ideal solution. We would like to emphasize that this might not be true for other applications such as quantum mechanical calculations[29] or sequence alignment[30] that can be memory and I/O intensive.

The recycled desktop computers all have single Intel P4 CPU's, with clock rates ranging from 1.2 to 2 GHz. Putting together the physical infrastructure to support the grid consisted of allocating space, provisioning power and installing several network switches. The installation of the computers in the physical grid was performed incrementally as old desktops were replaced by new ones. An automated generic installation process was used to provision old desktops as new members of the compute grid.

To enable resource utilization control, the Sun Grid Engine[31] (SGE) was implemented across the grid. SGE provides a queuing mechanism to allow a controlled distribution of compute jobs. SGE was configured with multiple queuing strategies to accommodate short running semi-interactive job submissions from Illuminator and longer running batch jobs. A job priority hierarchy was established to give jobs submitted to high priority queues (for Illuminator) faster scheduling times through the SGE scheduling process and higher operating system job priorities through

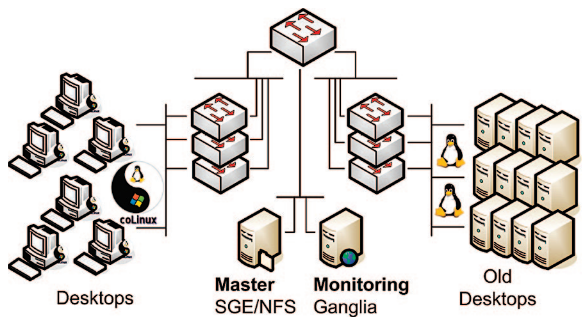

Figure 1. Grid setup.

the OS scheduler. This scheme allows for the temporary oversubscription of CPU resources on any of the grid nodes defined as part of the high priority queues. This scheme has proven adequate to allow short Illuminator jobs to complete quickly without the need to implement preemption of batch jobs when all nodes of the grid are in use. Grid nodes can be rebooted at any time, jobs that were assigned to those nodes will automatically be rescheduled by the SGE.

Additional components of the grid are an network file system (NFS) file server to house centrally located shared applications, data repositories, and home directories (Figure 1). For performance reasons the NFS storage is only used for the distribution of input files and to store the final output files. Each node has local disk space used to store temporary files for I/O intensive operations. The SGE master node also resides on the NFS file server. Grid utilization monitoring is accomplished on a separate server using Ganglia.[32]

Adding Desktop Computers to the Grid. The proof of concept implementation was very successful showing good performance and demonstrating the ease of use and maintenance. Newly developed applications like Autocorrelator and Illuminator were well received and the amount of usage made more computational resources necessary. Having just installed 80 desktop computers with new hyper-threaded 3.2 to 3.4 Ghz Intel P4 CPU's, we sought ways to harness those CPU cycles in a nonintrusive manor. Anadys had standardized on Windows XP as the company wide operating system for desktop computers. To integrate the desktop computers into the existing grid, coLinux was installed as a service in Windows XP.

The IT department at Anadys uses a desktop management system that includes a desktop deployment server to maintain a standardized desktop environment. New desktop computers are deployed with a standardized image. It was important to have the ability to include the coLinux distribution in a standardized desktop image and to be able to selectively manage which desktops would run the coLinux client service. Controlling the selection process was accomplished by storing the coLinux configuration files on a central file server and using an active directory group policy applied to each machine. The group policy checks a central repository for the existence of a coLinux configuration file that has the same name as the Windows XP host. If the configuration file exists, the group policy starts the coLinux service and sets the service to run at a low priority. Adding or removing the configuration file to the central repository activates or deactivates the use of the particular machine in the grid.

CoLinux was configured to use its own virtual network adapter that has a unique media access control (MAC)

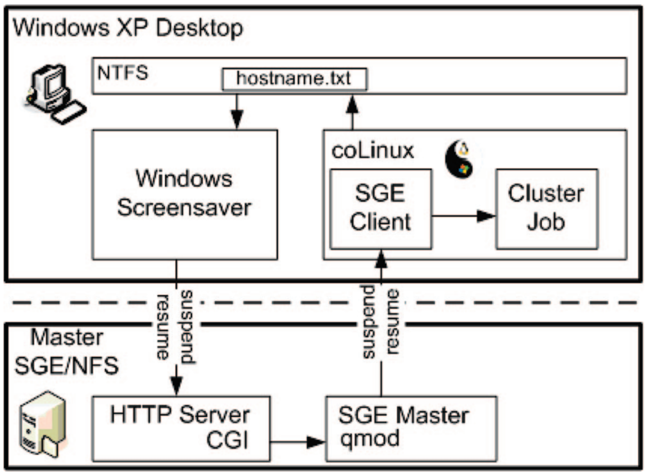

Figure 2. Screensaver suspend/resume communication flow.

address and needs its own internet protocol (IP) address. The IP addresses of coLinux clients are controlled with dynamic host configuration protocol (DHCP) reservations mapped to a unique MAC address assigned in each individual coLinux configuration file. The ending hexadecimal digits of the MAC address correspond to the ending decimal digits of the host name assigned by DHCP. Thus the central repository of coLinux configuration files is also used to map coLinux host names to Windows XP host names, which is useful for monitoring and debugging. For instance, if the configuration file hostx.xml contains MAC address 00:FF:00:00:00:0A, then DHCP assigns the IP address that maps to the host name "coclient10" in the domain name system (DNS) (0A = 10).

The coLinux clients can be seamlessly managed together with the dedicated compute grid nodes. Both types of nodes are running the CentOS 4.3 operating system.[33] Because they share the same base architecture only one set of application binaries is needed. The desktop computers have hyper-threading (HT) enabled, which allows one HT CPU to service the desktop processes and one to service grid jobs running under coLinux.

Performance monitoring and subjective user experience at the desktops confirm that there was no noticeable impact to the users. Each desktop computer has 2 GB of physical memory. CoLinux was configured to use 512 MB of memory. The 1.5 GB of memory available to Windows was found to be adequate for the general desktop applications in use at Anadys. A custom screen saver was deployed on the Windows desktop that allows the user to select whether or not grid jobs will run under coLinux while the screen saver is not active. When the screen saver is activated, a request to start computational jobs on that node is sent to the SGE Master. This is accomplished using a CGI script on the master node (Figure 2). When the screen saver is deactivated the job is suspended. To facilitate this process a coLinux boot time script is run from rc.local that writes the coLinux host name to a file in the Windows new technology file system (NTFS). The coLinux host name is then used by the customized Windows screen saver to suspend and resume SGE client jobs running on the local instance of coLinux. In practice, the background jobs had minimal impact on the desktop user experience and the screen saver is seldom used to deactivate jobs.

Table 1. Performance of Typical Computational Chemistry Applications on Different Hardware Platforms[a]

		compounds per hour			
CPU type	parallel jobs[b]	Omega	Rocs	Szybki[c]	Gold
Intel P4 3.2 GHz Native Linux	1	3536	163636	33027	781
Intel P4 3.2 GHz coLinux	1	3409	138462	31034	703
Intel P4 1.8 GHz	1	2207	72000	8391	278
Intel Pentium D 3.0 GHz	1	3345	138461	23225	532
Intel Xeon 5160 3.0 GHz	1	5270	327272	66666	902
Intel P3 900 MHz	1	1272	53731	6754	274
Intel Xeon 3.4 GHz	1	3753	150000	23076	593
hyper threaded					
Intel Xeon 3.4 GHz (2 CPU)					
each process	4	2373	85714	11356	457
total	4	9492	342857	45425	1830
Intel P4 3.2 GHz (1 CPU)					
each process	2	2236	94736	16744	444
total	2	4472	189473	33488	889
dual core					
Intel Xeon 5160 3.0 GHz (2 dual cores)					
each process	4	5270	276923	60000	777
total	4	21083	1107692	240000	3111

[a] All jobs were run ten times, and the run times were averaged. All machines were running native Linux (Cent OS v4.3) except where coLinux is specified. [b] Number of identical jobs run in parallel. [c] Minimization of the ligand occurred using a rigid Thrombin structure. The initial pose was determined as the best shape overlap with the crystallographic ligand from ROCS.

Performance. To evaluate the performance of the various hardware and operating systems, we timed computations using four applications (Omega v2.1, Rocs v.2.2, Szybki v1.4, and Gold v3.3)[21–27] across a variety of different host architectures. We selected a list of 1000 random molecules from our in-house database system, ARISE.[34,35] These compounds were conformationally enumerated with OMEGA and the conformational library was docked and superimposed. The crystallographic coordinates of Thrombin (PDB code 2c8y)[36] were used for the docking experiments. The ligand in Thrombin was used to define the binding surface for docking. The ligand was also used as the query for the superposition with ROCS. All applications were run with default parameters. Gold was run in library mode. For Gold, only the lowest energy conformation of each compound produced by Omega was used.

In Table 1, the performance when running the applications on different hardware configurations is shown. While running these experiments, no other activity was performed on the computers. In general, application performance increases roughly linearly with the clock rate of the CPU. One exception is our newest server with the 3 GHz Xeon 5160 CPU. The performance for this server is significantly higher than expected from the correlation (Figure 3). There could be multiple reasons for the good performance of the Xeon 5160 server, including a faster front side bus and newer CPU architecture. However, we did notice a slight decrease in performance when running

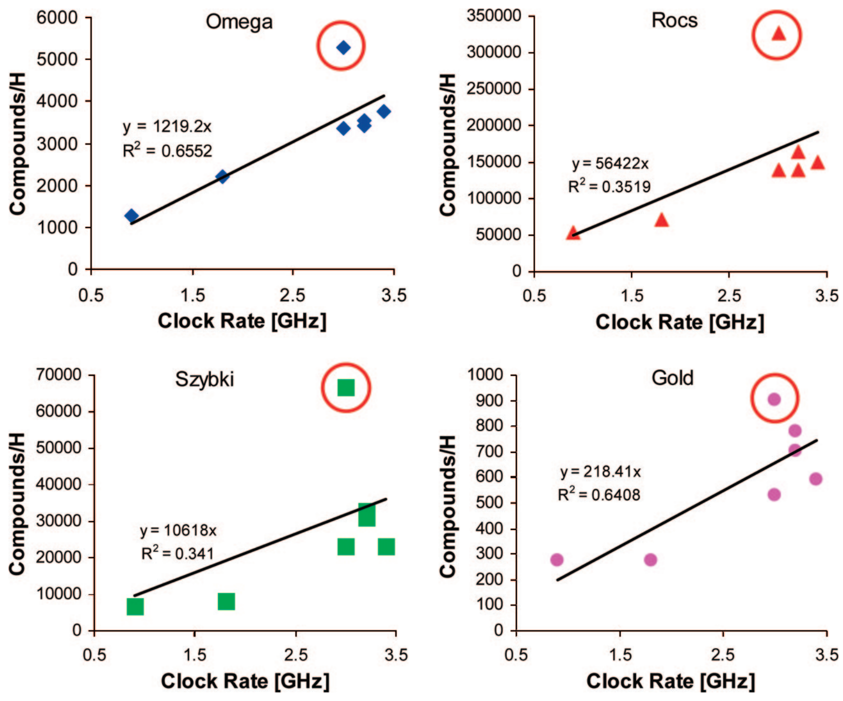

Figure 3. Performance of computational programs versus clock rate of CPU. The data point marked with a red circle was computed on the 3 GHz Xeon 5160 server. Only results of jobs running alone are included in these graphs (Table 1 rows 1–7).

Table 2. Influence of User Interaction on Jobs Running on Desktop Computers with coLinux[a]

	compounds per hour			
CPU type	Omega	Rocs	Szybki	Gold
Intel P4 3.2 GHz coLinux[b]	3409	138462	31034	703
Intel P4 3.2 GHz coLinux (live)[c]	3380	133333	31304	692
Intel P4 3.2 GHz coLinux (MEM/I/O)[d]	2167	65454	25899	497

[a] Jobs were run ten times and averaged. [b] Jobs run in coLinux in the absence of user interaction. [c] Jobs run in coLinux during regular business hours. [d] Jobs run in coLinux in parallel with an artificial memory and I/O load.

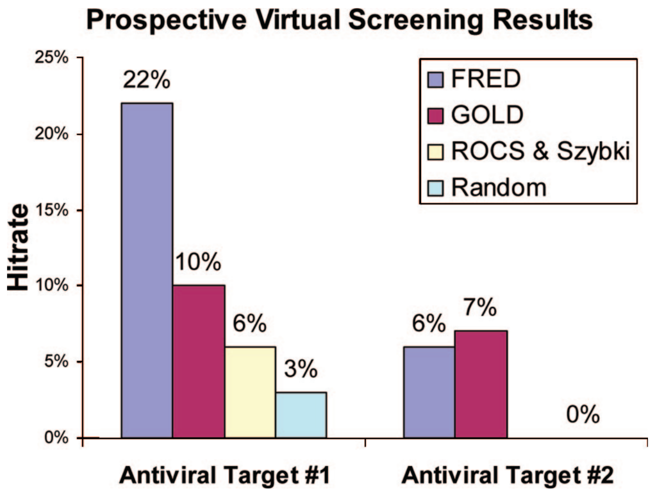

Figure 4. Results from virtual screening experiments. Compounds where considered to be hits if they had activities smaller than 50 μM and if their purity could be confirmed by LC/MS. The ROCS/Szybki method was not used for the Antiviral Target #2.

multiple identical processes on the dual core architecture. This could be due to competition for shared resources. As expected running two processes simultaneously was not as efficient on hyper-threaded CPU's as on dual core CPU's. Table 1 shows 40% loss in speed per process when running two processes per HT CPU compared to running a single process. Therefore, running two processes per HT CPU resulted in a higher overall output.

Running jobs on coLinux compares well to running the same job on an identical computer with native Linux. The jobs running on coLinux reached 85–96% of the performance of jobs on native Linux (Table 1, rows 1 and 2).

To evaluate the effects of users using the Windows XP desktop computers while computational jobs were running, the computations were repeated during regular business hours and with an artificial load (Table 2). The artificial load was generated by a Windows program that continuously read a 256 MB file into memory and then wrote it back to disk. While the effect of a real user was negligible (efficiency 96–100%), the effect of the artificial load was more significant, reducing the efficiency to 47–83%. The artificial load generated by the program represents a worst case scenario. Even in that case, a significant benefit can be achieved by running computational jobs in coLinux.

APPLICATION

Since its implementation the computational power of HYDRA was used on every drug discovery campaign at Anadys Pharmaceuticals. It was used for model optimization, virtual screening of the screening library as well as lead optimization. For projects where a crystal structure of the target was available, docking methods were used. If no crystal structure was available we applied shape similarity searches to known ligands,

Figure 4 shows the results of virtual screening experiments against two protein targets. The in-house screening library was ranked by three different computational methods:

• docking with FRED.
• docking with GOLD.
• ligand protein interaction energy from Szybki after minimization of the conformation with highest shape similarity to a known ligand.

The highest scoring 1000 compounds were then reduced to 100 by diversity selection using the DISE[37] method. Additionally, 100 compounds were selected randomly from the in-house library as a control. All compounds were then screened in biochemical assays. The results show that all computational methods have improved hit rates as compared to random selection.

CONCLUSION

The Anadys computational grid has become an essential part of the drug discovery process at Anadys Pharmaceuticals. It enables the virtual screening of hundreds of millions of compounds a day with fast methods such as ROCS. Even with more complex methods such as docking using GOLD, millions of compounds can be docked in one day. The grid also is a necessary component of the end user modeling tool, Illuminator, in that it ensures short turnaround times. It has increased the productivity of drug discovery at Anadys by increasing the initial hit rate in new projects using virtual screening. In addition it is helping medicinal chemists to prioritize compounds to be synthesized based on computational methods. The infrastructure based on recycled and actively used desktop computers has proven to be cost-effective and will continue to grow with every new employee and with every desktop computer upgrade. The use of coLinux to create a grid of Linux-based computers has proven to be very efficient and stable. The grid has been in continuous operation for the last 18 months, aside from planned maintenance. A computing grid as described in this paper provides an adequate amount of CPU time for most typical calculations used in modern computational drug discovery. The ease of implementation and maintenance should make it feasible for small and large companies.

ACKNOWLEDGMENT

We would like to acknowledge Matt Strebe and Bobby Phimmasane from Connetic for IT support. We would also like to thank Dr. Charles Kissinger for comments and the Anadys Management for supporting the implementation of the Anadys Computational Platform.

REFERENCES AND NOTES

(1) Zhong, S.; Macias, A. T.; MacKerell, A. D. Computational Identification of Inhibitors of Protein-Protein Interactions. *Curr. Topics. Med. Chem.* **2007**, *20*, 63–82.
(2) Kuhn, B.; Kollman, P. A. Binding of a Diverse Set of Ligands to Avidin and Streptavidin: An Accurate Quantitative Prediction of Their Relative Affinities by a Combination of Molecular Mechanics and Continuum Solvent Models. *J. Med. Chem.* **2000**, *43*, 3786–3791.
(3) Fujitani, H.; Tanida, Y.; Ito, M.; Jayachandran, G.; Snow, C. D.; Shirts, M. R.; Sorin, E. J.; Pande, V. S. Direct calculation of the binding free energies of FKBP ligands. *J. Chem. Phys.* **2005**, *123*, 084108.
(4) Khandelwal, A.; Lukacova, V.; Comez, D.; Kroll, D. M.; Raha, S.; Balaz, S. A Combination of Docking, QM/MM Methods, and MD Simulation for Binding Affinity Estimation of Metalloprotein Ligands. *J. Med. Chem.* **2005**, *48*, 5437–5447.
(5) Brown, S. P.; Muchmore, S. W. High-Throughput Calculation of Protein-Ligand Binding Affinities: Modification and Adaptation of the MM-PBSA Protocol to Enterprise Grid Computing. *J. Chem. Inf. Model.* **2006**, *46*, 999–1005.
(6) Brown, S. P.; Muchmore, S. W. Rapid Estimation of Relative Protein-Ligand Binding Affinities Using a High-Throughput Version of MM-PBSA. *J. Chem. Inf. Model.* **2007**, *47*, 1493–1503.
(7) Thain, D.; Tannenbaum, T.; Livny, M. Condor and the Grid. In *Grid Computing - Making the Global Infrastructure a Reality*; Berman, F.; Fox, G.; Hey, T., Eds.; John Wiley & Sons Inc.: New York, 2003; pp 1080.
(8) Pearlman, D. A.; Case, D. A.; Caldwell, J. W.; Ross, W. S.; Cheatham, I.; DeBolt, S.; Ferguson, D.; Seibel, G.; Kollman, P. AMBER, a package of computer programs for applying molecular mechanics, normal-mode analysis, molecular dynamics and free energy calculations to simulate the structural and energetic properties of molecules. *Comput. Phys. Commun.* **1995**, *91*, 1–41.
(9) Gobbi A.; Lardy M.; Showalter R.; Zhao Q.; Zhou Y. Illuminator: Increasing the Impact of Computational Tools in Drug Discovery. *Western Regional Meeting of the American Chemical Society*, San Diego, CA, Oct 9–13, 2007; poster.
(10) Chien, A.; Foster, I.; Goddette, D. Grid technologies empowering drug discovery. *Drug Discov. Today* **2002**, *7*, S177–S180.
(11) Grids - A Low Cost, High Speed Alternative to Traditional High-Performance Computing. http://www.grid.org.il/_Uploads/dbsAttachedFiles/GRID_ROI.pdf (accessed Jan 2008).
(12) Foster, I. Globus Toolkit Version 4: Software for Service-Oriented Systems. *J. Comput. Sci. Technol.* **2006**, *21*, 513–520.
(13) Pande, V. S.; Baker, I.; Chapman, J.; Elmer, S. P.; Khaliq, S.; Larson, S. M.; Rhee, Y. M.; Shirts, M. R.; Snow, C. D.; Sorin, E. J.; Zagrovic, B. Atomistic protein folding simulations on the submillisecond time scale using worldwide distributed computing. *Biopolymers* **2003**, *68*, 91–109.
(14) NET NEWS:. . .and a Search for Alien Life. *Science* **1998**, *282*, 839.
(15) The Xen virtual machine monitor. http://www.cl.cam.ac.uk/Research/SRG/netos/xen (accessed Jan 2008).
(16) QEMU. http://fabrice.bellard.free.fr/qemu (accessed Jan 2008).
(17) Aloni, D. Cooperative Linux. http://www.colinux.org/publications/Reprint-Aloni-OLS2004.pdf (accessed Jan 2008).
(18) VMware. http://www.vmware.com (accessed Jan 2008).
(19) Parallels: Cost-effective, High-Performance Virutalization Made Easy. http://www.parallels.com/files/upload/Parallels_Intel_whitepaper_on_VT_Technology.pdf (accessed Jan 2008).
(20) Santosa, M.; Schaefer, A. Build a heterogeneous cluster with coLinux and openMosix. http://www.ibm.com/developerworks/linux/library/l-colinux/ (accessed Jan 2008).
(21) Boström, J.; Greenwood, J. R.; Gottfries, J. Assessing the performance of OMEGA with respect to retrieving bioactive conformations. *J. Mol. Graphics Mod.* **2003**, *21*, 449–462.
(22) Rush, T. S.; Grant, J. A.; Mosyak, L.; Nicholls, A. A Shape-Based 3-D Scaffold Hopping Method and its Application to a Bacterial Protein-Protein Interaction. *J. Med. Chem.* **2005**, *48*, 1489–1495.
(23) Schultz-Gasch, T.; Stahl, M. Binding site characteristics in structure-based virtual screening: evaluation of current docking tools. *J. Mol. Model.* **2003**, *9*, 47–57.
(24) Jones, G.; Willett, P.; Glen, R. C. Molecular recognition of receptor sites using a genetic algorithm with a description of desolvation. *J. Mol. Biol.* **1995**, *245*, 43–53.
(25) Jones, G.; Willett, P.; Glen, R. C.; Leach, A. R.; Taylor, R. Development and Validation of a Genetic Algorithm for Flexible Docking. *J. Mol. Biol.* **1997**, *267*, 727–748.
(26) Verdonk, M. L.; Cole, J. C.; Hartshorn, M. J.; Murray, C. W.; Taylor, R. D. Improved Protein-Ligand Docking Using GOLD. *Proteins* **2003**, *52*, 609–623.
(27) OMEGA, ROCS, FRED, and SZYBKI are distributed by OpenEye Scientific Software, Santa Fe, NM: http://www.eyesopen.com (accessed Jan 2008). Gold is distributed by CCDC, Cambridge, UK: http://www.ccdc.cam.ac.uk/products/life_sciences/gold/ (accessed Jan 2008).
(28) Beowulf.org: Overview. http://www.beowulf.org/overview/index.html (accessed Jan 2008).
(29) For example using GAMESS Schmidt, M. W.; Baldridge, K. K.; Boatz, J. A.; Elbert, S. T.; Gordon, M. S.; Jensen, J. H.; Koseki, S.; Matsunaga, N.; Nguyen, K. A.; Su, S.; Windus, T. L.; Dupuis, M.; Montgomery, J. A. General Atomic and Molecular Electronic Structure System. *Comput. Chem* **1993**, *14*, 1347–1363.
(30) For example using mpiBLAST Thorsen, O.; Smith, B.; Sosa, C. P.; Jiang, K.; Lin, H.; Peters, A.; Feng, W.-C. Parallel Genomic Sequence-Search on a Massively Parallel system. In *Proceedings of the 4th international conference on Computing frontiers*, Ischia, Italy, 2007; ACM: New York, 2007; pp 59–68.

(31) Grid Engine Project Home. http://gridengine.sunsource.net/ (accessed Jan 2008).

(32) Massie, M. L.; Brent, N.; Chun, B.; Culler, D. E. The ganglia distributed monitoring system: design, implementation, and experience. *Parallel Comput.* **2004**, *30*, 817–840.

(33) CentOS: The Community Enterprise Operating System. http://www.centos.org (accessed Jan 2008).

(34) Gobbi, A.; Funeriu, S.; Ioannou, J.; Wang, J.; Lee, M.-L.; Palmer, C.; Bamford, B.; Hewitt, R. Process-Driven Information Management System at a Biotech Company: Concept and Implementation. *J. Chem. Inf. Comput. Sci.* **2004**, *44*, 964–975.

(35) Hewitt, R.; Gobbi, A.; Lee, M.-L. A Searching and Reporting System for Relational Databases Using a Graph-Based Metadata Representation. *J. Chem. Inf. Model.* **2005**, *45*, 863–869.

(36) Howard, N.; Abell, C.; Blakemore, W.; Chessari, G.; Congreve, M.; Howard, S.; Jhoti, H.; Murray, C. W.; Seavers, L. C. A.; van Monfort, R. L. M. Application of Fragment Screening and Fragment Linking to the Discovery of Novel Thrombin Inhibitors. *J. Med. Chem.* **2006**, *49*, 1346–1355.

(37) Gobbi, A.; Lee, M. DISE: Directed Sphere Exclusion. *J. Chem. Inf. Comput. Sci.* **2003**, *43*, 317–323.

CI700396B