Ligand discovery on massively parallel systems

S. R. Shave
P. Taylor
M. Walkinshaw
L. Smith
J. Hardy
A. Trew

Virtual screening is an approach for identifying promising leads for drugs and is used in the pharmaceutical industry. We present the parallelization of LIDAEUS (LIgand Discovery At Edinburgh UniverSity), creating a massively parallel high-throughput virtual-screening code. This program is being used to predict the binding modes involved in the docking of small ligands to proteins. Parallelization efforts have focused on achieving maximum parallel efficiency and developing a memory-efficient parallel sorting routine. Using an IBM Blue Gene/L™ supercomputer, runtimes have been reduced from 8 days on a modest seven-node cluster to 62 minutes on 1,024 processors using a standard dataset of 1.67 million small molecules and FKBP12, a protein target of interest in immunosuppressive therapies. Using more-complex datasets, the code scales upward to make use of the full processor set of 2,048. The code has been successfully used for the task of gathering data on approximately 1.67 million small molecules binding to approximately 400 high-quality crystallographically determined ligand-bound protein structures, generating data on more than 646 million protein–ligand complexes. A number of novel ligands have already been discovered and validated experimentally.

Introduction and background

With proteins playing a major role in almost every function carried out within a cell, the ability to control or "turn off" proteins by simply blocking their binding pockets creates exciting possibilities for controlling protein activity. Molecules (such as ions, drugs, or polypeptides) that bind to a protein of interest are referred to as *ligands* and may affect the operation of the protein within an organism. Finding specific and tightly binding ligands offers the ability to alter the behavior of proteins and plays an important role in modern medicine [1, 2]. This paper describes the use of simulation as the first stage for finding novel non-covalently bound ligands for known protein structures. *Virtual screening* (VS) [3–5] is a broad term and includes *de novo* ligand design, that is, engineering a molecule by adding and removing groups to alter its binding properties. However, for the purposes of this paper, *VS* refers to the practice of molecular docking in which a database of ligands is used for studying potential interactions with a target receptor such as a protein-binding site. This is a commonly used approach in the first stage of the drug discovery process.

Typically, the VS process involves representing the binding pocket of a protein and then executing two steps. First, the ligand is positioned (referred to as the process of *posing*), and second, the resultant complex is scored. These steps are often merged or used to direct each other. Potential ligands are extracted from a database of available compounds. Software filters can then be imposed on the potential ligands to ensure that they meet certain criteria such as Lipinski's rule of five [6] or that they are not too similar [7].

Many different codes and approaches exist for simulating the docking of a ligand to a receptor, and these approaches vary greatly in their computational complexity. As mentioned, two main aspects to docking exist [8]: the positioning and global search in which the ligand is brought into a potential binding position and the scoring in which a prediction is made about the strength of present binding interactions [9, 10].

Two approaches exist for ligand positioning: flexible body docking and rigid-body docking. Flexible body docking typically treats the ligand as flexible and the protein as rigid. Sometimes, however, both the ligand and

0018-8646/08/$5.00 © 2008 IBM

a portion of the protein are considered flexible structures. Flexible docking studies can be useful, simulating on the order of hundreds to thousands of protein–ligand complexes. The least computationally expensive of the approaches is rigid-body docking. Here, both ligands and protein receptors are rigid, and docking occurs through translation and rotation of the ligand. Many thousands of ligand–protein complexes can be simulated in this way. Combined approaches also exist and have been useful in the discovery of a novel binding trench present in the HIV (human immunodeficiency virus) integrase protein [11]. Flexible docking and rigid docking are vastly different in terms of complexity. For example, the process of exploring conformers of relatively simple molecules with three or four rotatable bonds typically requires more than 200 conformations to be sampled in order to study a full range of conformational space, even when using a broad step size. The complexity involved in treating ligands as flexible structures is a combinatorial problem that increases dramatically as conformational space is sampled more exhaustively [12]. Many different codes and approaches exist that attempt to deal with this complexity. The most widely used VS tools are GOLD (Genetic Optimization for Ligand Docking) [13], FlexX [14], DOCK [15], AutoDock [16], Glide [17, 18], and ICM (Internal Coordinate Mechanics) [19]. All of these codes carry out flexible ligand docking, using a range of approaches such as genetic algorithms, incremental construction, simulated annealing, and Monte Carlo methods in order to address the issue of computational complexity. Overviews of methods that address flexibility, along with limited comparisons between methods, are available [20, 21].

With most of the above program codes, a choice of scoring functions is available. Scoring functions are used in an attempt to estimate the binding energy between protein and ligand. Many different implementations exist, but each can be placed in one of three categories: force field-based methods, empirical scoring functions, and statistical knowledge-based methods [22]. A number of studies have shown that none of these methods properly predicts or accurately estimates the true binding energy, although each method takes a different approach for analyzing the same problem. Two studies compared seven scoring functions [8, 23], and another studied eleven [24]. For example, the scoring function of FlexX highly scores hydrogen-bonded ligands and tends to neglect lipophilic binding effects [9]. Because the differences between scoring functions are so great, results from one screening run are not directly comparable to those from another. However, a ligand may have its relative ranking compared [25]. The diversity exhibited by scoring functions has been used in consensus scoring. Using drastically different but well-performing scoring functions, the accuracy of consensus methods can be greater than that of their component parts [23, 26–28]. However, the risks associated with consensus methods are well known. "Artificial enrichment" is the main danger; that is, the scoring functions are sometimes chosen to perform well on a specific protein–ligand complex [29]. Also, various contributing functions must be well understood, and a balance must be achieved so that positive aspects of one algorithm are not diminished by another algorithm [30].

With sufficient computing time, it is possible to score protein–ligand complexes accurately using approaches derived from quantum mechanics [31–33]. The work documented in this paper is concerned with what is commonly known as *high-throughput VS* (HTVS), using simplified methods (i.e., methods that use neither quantum mechanical nor complex molecular dynamics approaches) to achieve higher simulation throughput [34] of complexes. The term *HTVS* includes program codes and approaches that test on the order of thousands of protein–ligand complexes in one program run; the simulation is not set up to produce results for just one complex.

An exhaustive study of well-known VS programs, using their default scoring functions, has shown that the program ICM, which uses metropolis Monte Carlo techniques, coupled with a force field to explore and score multiple conformations of a ligand, is the best performer (i.e., has the smallest root-mean-squared deviation from known crystallographic structures) when used to recreate the structure of known protein–ligand complexes [19]. However, this program code may not be applicable to high-performance approaches that involve the screening of large databases of small ligands.

Most program codes do not address the possibility of ligands bonding covalently to the protein receptor. The codes that do address this feature, using branches of quantum mechanics, are not high-throughput approaches. Some high-throughput codes such as FlexX can be directed to bind the ligand covalently, but this requires specific knowledge of the complex and is not performed automatically [14].

A very large increase in the number of determined protein structures (driven partly by improvements in x-ray crystallography techniques) has recently been observed. The most widely used resource for structural information on proteins is the Protein Data Bank (PDB) of the Research Collaboratory for Structural Bioinformatics (RCSB) [35]. With currently more than 40,000 protein structures in the repository, *in silico* techniques have an ever-increasing number of target structures with which to work. (Multiple entries for structures are deposited in the database while new techniques improve the resolution of those previously

submitted. Also, the proteins exist in different conformations or with different co-crystallized ligands.) Protein structure information comes in various formats, and the most widely used format is the PDB file format. This allows the representation of multiple molecules in a complex, such as a protein bound to its natural ligand.

Depending on the results obtained from the screening of thousands of potential ligands, and after further investigation, it may be worth testing the binding properties of these ligands in a biochemistry laboratory [36].

Because of the computational complexity of ligand-binding studies and the amount of high-performance computing (HPC) time typically available to biologists, only recently have VS runs been regularly breaking the one million receptor–ligand complexes "barrier" [37]. In recent years, the University of Edinburgh has been able to provide its scientists with access to teraflop computing resources. This provides researchers with a significant increase in available CPU cycles, without the complexity of applying for time with the U.K. National HPC Service. This has been achieved through the purchase of an IBM Blue Gene/L* (BG/L) supercomputer, which uses a novel architecture that is designed to provide unprecedented computing performance, coupled with very low power consumption, floor space, and cost. Designed to be highly scalable with low-latency communications, this system offers a significant breakthrough in supercomputing technology.

Although the resulting code is designed to be portable and run on any large-scale HPC platform, the target architecture for this work has been the BG/L platform. A detailed description of the architecture and its design approach have been reviewed elsewhere and will not be repeated here [38]. However, certain aspects of the architecture are important to the overall design of the parallel LIDAEUS (LIgand Discovery At Edinburgh UniverSity) code and are summarized.

The BG/L supercomputer is based on a classical massively parallel processor (MPP), distributed-memory architecture. The BG/L platform has a system-on-a-chip design, based on the PowerPC* 440 core. This feature allowed designers to have some customization capabilities without the cost of developing a totally new architecture, and it offers a customized memory system, network interface, and floating-point unit. This approach was successfully pioneered in the QCDOC (quantum chromodynamics-on-a-chip) machine, a special-purpose system for quantum chromodynamics [39]. The resulting BG/L architecture contains a large number of simple, inexpensive low-power processors connected via a custom-designed high-performance interconnect. One node (also known as a *chip*) contains two PowerPC 440 cores, with associated caches and a low clock frequency (700 MHz). The relatively low clock frequency is an important feature of the BG/L system, exploiting the fact that many HPC applications are memory bound, rather than CPU bound. By using simple, inexpensive low-clock-frequency processors, the amount of power generated is low. This in turn allows for a high physical density of processors, and thus, a relatively small amount of floor space is required. However, this does lead to two important considerations when parallelizing the LIDAEUS code. First, the code must scale to large numbers of processors or little performance benefit will be seen because of the low-clock-speed processor. Second, the code must scale well in memory, to avoid the relatively small amount of memory per processor, which limits the problem that can be studied.

The high-performance interconnect of the BG/L platform has multiple networks for different tasks. The main network is a three-dimensional (3D) torus with each node connected by six links to its nearest neighbors and utilized for point-to-point communications. These links are reasonably low latency (3.5 μs) and have a modest bandwidth (on the order of 160 MB/s) [40]. The second network is a global combining, broadcast binary tree network for collective communications (with 5-μs latency [40]). The third network is a binary tree network designed specifically for fast barrier synchronization (1.5-μs latency [40]). The two remaining networks are Ethernet networks. One is for I/O and the other is for diagnostics. The relatively low latency of the interconnect is beneficial for the LIDAEUS parallelization, helping to allow the code to scale to higher processor counts.

The hardware

The Edinburgh BG/L system is a single-rack IBM BG/L system with 1,024 chips (2,048 700-MHz PowerPC 440 processors). Installed in December 2004, this was the first BG/L system in Europe. The machine can operate either in co-processor (CO) mode, in which only one processor in each node is utilized for computation and the other is reserved for communication, or in virtual node (VN) mode, in which all processors are able to act as virtual nodes and address both computation and communication. In VN mode, the resources of the node must be shared between the two processors. Each core has a 32-KB-data, 32-KB-instruction L1 cache with no coherency between the two cores. Each node also has a very small (2 KB, 128-byte-line) L2 cache, a 4-MB shared L3 cache, and 512 MB of main memory. The system has one I/O node for every eight compute chips and four IBM BladeCenter* JS20 front-end systems for compilation and batch submission. The system runs driver release 3, with support for XL Fortran 10.1 and XL C/C++ 8.0.

LIDAEUS

LIDAEUS is a highly adaptable and modular rigid-body docking VS code written in C++ and parallelized using the Message Passing Interface (MPI). This ensures portability to a range of parallel platforms. LIDAEUS focuses on extremely high simulation speed, with the goal of a single processor being able to process a ligand in a few seconds. Extensive testing and cross-platform trials have ensured that the code is stable on a range of processors and platforms. It has been used successfully to identify a family of novel cyclin-dependent kinase inhibitors [41]. Results will be published soon that document six laboratory-confirmed cyclophilin-A binders that were found using information obtained from the parallelized code discussed here [42].

LIDAEUS job preparation is a serial process that is carried out on a local non-HPC Linux** system. Using a PDB file to represent the target protein, the first stage of the job preparation is to profile certain energy values present. These energies are written to map files, and the internal representation is a finely spaced cubic lattice with cells spaced typically 0.5 Å apart. This gives a volumetric representation of specific energies associated with the space surrounding the protein. Three maps describe van der Waals, hydrophobic, and hydrogen-bonding energies [41]. Points in the volumes have their energies defined in the following way. The van der Waals (vdw) map is defined as follows by the classic Lennard–Jones potential:

$$E(\mathrm{vdw}) = \sum_{n}^{1} \left[A/r_{\mathrm{p}}^{12} - B/r_{\mathrm{p}}^{6} \right].$$

The sum is over set n where $1 - n$ are all atoms not forming hydrogen bonds with a probe; r_{p} is the separation between the protein atom and the probe (p); and A and B are coefficients from the Amber molecular dynamics package [43]. The hydrogen-bond-donor and hydrogen-bond-acceptor energy maps are defined in a similar way as the van der Waals map but include a weighting term dependent on the deviation from ideal hydrogen-bond angles. This weighting is encoded by high energy values in the map file, which can be visualized as a halo of ideal hydrogen-bonding angles by using isosurfaces at a suitable energy level.

The next stage of job preparation involves site point generation, that is, the generation of a collection of a special set of points located near the binding site. The map files typically represent the entire volume occupied by the protein and its surrounding area. The standard method of inhibiting a protein involves binding one or more foreign molecules to key sites known as *binding sites*. A protein may have one or more defined binding sites. Often, PDB files represent the protein complexed with its natural ligand. This natural ligand directs site point generation that defines the target pocket. The positioning of these points is determined by a search through the map files in the volume defined by the natural ligand (plus an adjustable amount of additional volume) and the energy values present. Generation of these points is flexible, but by default 170 points are found in the pocket and considered as one of three types: hydrophobic, hydrogen-bond acceptor, and hydrogen-bond donor. These types denote the type of atom that would favorably sit in a certain position (from an energy standpoint). The distribution of site points is many times denser than that of the atoms in a molecule. These points offer a way to constrain the search in ligand positioning, overlaying atoms of key types onto appropriate site points.

The action of LIDAEUS during execution can be broken down into a pipeline consisting of four modules (Preen, Pose, Score, and Sort) through which the small molecules (potential ligands) are passed. These small molecules in the SDF (structure data file) format are typically put into their vacuum minimum energy conformation and docked using this conformation. For this reason, large molecules such as peptides are unsuitable for docking using LIDAEUS. The next section of this paper briefly describes the expansion of LIDAEUS to enable flexible ligand docking. In the following paragraphs, we discuss the main modules of LIDAEUS.

Preen—Atom hybridization state information is added to an internal representation of the molecule. *Hybridization* denotes characteristics of an atom that vary according to the bonding interactions and the atomic orbital state of those bound atoms.

Pose—This is the most computationally intensive part of the pipeline. The pose module takes its name from the action of positioning (i.e., "posing") a ligand against a target. Many poses may be generated by a potential ligand. Because LIDAEUS is a rigid-body docking code, suitable poses for the ligand are achieved through rotation and translation. Instead of iterating through all orientations and positions of the ligand at a coarse resolution, the pregenerated site points are used to constrain the positioning search. LIDAEUS generates a list of distances between bonded atoms in the ligand, as well as a list of distances between each site point (up to an adjustable cutoff value). A search tree is used in the following set of actions to explore combinations of fitting points. Each stage of depth within the tree matches ligand atoms to site points with increasing tolerance values. These values are user definable, but by default, the first four atoms are fitted with tolerances 0.02 Å, 0.04 Å, 0.06 Å, and 0.08 Å. The list of distances between bonded atoms in the ligand is checked against distances between site points. If a match is found, a test is carried out to see

whether the next bonded atom in the tree fits a distance present between nearby site points. If the depth of the tree of fitting atoms reaches a user-definable depth (by default four), the ligand in the correct position is able to satisfy the atom-type requirements described by the site points. Once it is known that a set of certain ligand atoms can overlay a set of site points, the 3D coordinates are extracted and LU factorization is used to obtain a combined rotation–translation matrix that overlays the key ligand atoms onto matching site points. This matrix is then stored with the representation of the ligand. Applying this matrix to the 3D coordinates of the ligand produces a pose. With ligands able to generate multiple poses, the computational complexity of the problem bears little relation to the number of ligands being tested.

Score—For each pose entry present in the representation of the potential ligand, the combined translation and rotation matrix is applied to the molecule. With the molecule oriented into a potential binding position, the energy maps are used to sum the energy contributions being made by each atom.

The resulting score has units of kcal mol^{-1}. A six-dimensional conjugate gradient energy minimization is then applied to the molecule as directed by the predicted binding score in an attempt to refine the previously generated pose. This process is repeated for each pose entry belonging to a molecule.

Sort—LIDAEUS maintains a record of the best-scoring poses across all potential ligands. It is important to note that the score denotes the energy change in binding; the more negative a binding score, the stronger the ligand is predicted to bind. A positive score predicts that energy must be added to the system in order to bind the molecule in the position dictated by its pose. Many different factors affect the range of scores achieved in the final sorted list, but a good-scoring ligand (therefore, predicted to be a strong binder) has a score less than about −20 kcal mol^{-1}.

A standard dataset for benchmarking purposes exists and contains 1.67 million small molecules (in the SDF format) that may be purchased from 24 catalogs obtained from suppliers. In this benchmark, the receptor target for these molecules is known as *FKBP12*, and it plays an important role in immunosuppressive therapies. The structure is taken from the RCSB PDB and has the PDB identifier 1FKJ. This dataset is so large that it is unmanageable in a serial or low-processor-count parallel-processing environment. Selection of a random 30,000 molecules and subsequent simplistic predictive scaling calculations show that this dataset used with its target would take 29 days to analyze on an AMD Opteron** 246 Linux system with 2 GB of RAM. Runtime lengths such as this restrict users to the screening of small datasets in an effort to obtain the results they need in the amount of time available to them. HPC systems are an attractive platform for VS runs. The ultimate goal in the parallelization of LIDAEUS is to reduce the runtime for the standard dataset, thereby providing researchers with more freedom to experiment and set up even more complex runs with larger datasets.

Parallelization

The original version of LIDAEUS ran on a seven-node openMosix** [44] cluster consisting of seven AMD XP 2600 processors, each with 1 GB of RAM and a standard Gigabit Ethernet network acting as the interconnect. The system, therefore, consisted of a cluster of standard desktop machines managed by openMosix to create a cluster capable of distributed computing. This functionality is achieved by process load balancing between nodes (machines). By starting multiple processes and pipelines of LIDAEUS on each node of the cluster and running the molecular data through a simple multiplexer program, which distributed molecules to instances of the pipeline, we allowed the operating system to migrate pipelines from nodes with a high workload to nodes that have a lesser load.

While this approach to parallelization enabled more results to be gathered in a shorter amount of time, there was still a need for a truly parallel version of the code because of the restrictiveness of the openMosix parallelization and its ability to run only in an openMosix environment. Running the code on modern HPC systems would allow the use of a large number of processors—that is, many more than the seven used in the cluster.

Parallelization of LIDAEUS for the BG/L system at the University of Edinburgh takes the form of an MPI "task farm" that uses data decomposition as the parallelization strategy. Essentially, one processor runs a master process, reading in the small molecules to be complexed with the protein and then serving them to waiting worker processes residing on other processors. The worker processes run their own Preen, Pose, and Score code. As with any parallel code, computational load balancing must be addressed. Note that the code is intended for screening more than one million molecules in a single run, and the molecule data can be served to worker processes in relatively small sets (~5 to 50 molecules at a time). This data-serving process greatly limits the scope for load imbalance.

The initial parallelization was to be specifically for describing the binding pockets of a range of proteins. A selection criteria was applied to the full RCSB PDB (as of March 9, 2005), selecting crystallographically determined structures with a resolution better than 1.7 Å that made at least four hydrogen-bonding contacts with the protein and no covalent bonds between the protein and ligand. Sequence similarity searching was then used to obtain a

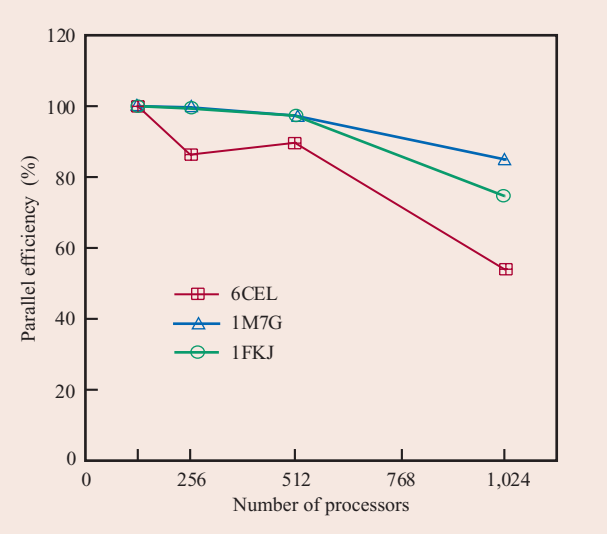

Figure 1

LIDAEUS operation without sort. Parallel efficiency is studied for different numbers of processors (with 128 as the baseline) for the standard dataset using three different target receptors. The Protein Data Bank identifiers for these receptors are given in the key.

diverse set of proteins with a sequence similarity not exceeding 90%. This specific problem with searching did not require sorting or retention of only the top results because information on the best-scoring minimized and non-minimized poses for each molecule was retained. The code has been used to screen a collection of 1.67 million molecules against a resultant selection of 387 protein-binding pockets. The results of this large computational run are stored in a database, and data-mining techniques are used to investigate the effectiveness of molecular descriptors in predicting binding affinities. **Figure 1** shows that the lack of a parallel sort, as well as any lack of interprocess communication, results in almost linear scaling and a parallel efficiency with up to 512 processors. With 1,024 processors, parallel efficiency for one receptor is as low as 54%. Further investigation is required, although a possible explanation for the low efficiency is that for 1,023 worker processes, each writing its results to a file at unpredictable intervals (with the master process reading from disk), the network switch through which the BG/L file I/O travels becomes saturated. While the total amount of data written in a run of the standard dataset is manageable at approximately 15 GB, the constant writing of a small amount of data by 1,023 streams may explain the poor parallel efficiency. The scaling data used to produce Figure 1 suggests diminishing returns beyond 1,024 processors. A version of the code described later removes the requirement for such large amounts of I/O and subsequently shows improved scaling.

As previously stated, site points offer a way to constrain the search space in the positioning of ligands. A more exhaustive search of this space can be carried out by altering a user-specifiable variable called "resol." This variable acts as a tolerance value that denotes the distance the center point of key ligand atoms must be from site points. The default value of 0.02 Å may be increased to carry out a more exhaustive search. As this value increases, the computational complexity exhibits exponential growth. Massive numbers of VS runs, dealing with millions of small molecules, are not suitable for larger tolerance values. Because the goal of VS is the discovery of interesting drug leads, higher tolerance values are useful only when using VS runs on smaller targeted subsets or classes of small molecules. However, this tolerance value offers the ability to alter the amount of computation required to process a small molecule. With this in mind, along with the consideration that the limiting factor (I/O) will remain constant with an increasing number of poses, it is possible to set up runs with a larger number of poses that scale to and beyond the limit of a single BG/L cabinet utilizing 2,048 processor cores.

In order for the parallel version of LIDAEUS to be used as an HTVS tool in the rigorous sense of the term *HTVS*, we did not collect information on the best poses for each molecule but instead implemented a sorting algorithm to produce a reduced list of the top-scoring results from all poses made by all molecules. An interesting characteristic of the BG/L architecture, specifically the current version of its compute node kernel (CNK), is that no exceptions or errors are generated when a chip has filled its local RAM and a stack overflow event occurs. The lightweight nature of the CNK means that there is also no virtual memory. Depending on the mode in which each node is run (i.e., CO or VN mode), each chip has either 512 MB or 256 MB of local memory, respectively. With the possibility of a ligand binding to a protein in a variety of poses, one small molecule may produce many positions and scores that must be retained or at least considered for retention at the sorting stage. In a restricted-memory environment such as this, a problem arises. We are attempting to sort a dataset so large that all of it cannot be held in one memory location. The sort cannot proceed on one processor. A collaborative effort among processors must be made.

Any time more complexity is added to a parallel code, consideration must be given to its effect on the overall execution time caused by the new code interaction with existing components. In addition to the two existing classes of processor—the master and the worker—we have a new class, the sink. Conceptually, the sink sits at the bottom of the task farm into which the workers dump their results. If the workers simply performed the

screening and then indiscriminately communicated their results to the sink, the performance of the code would decrease because of the communications overhead. Therefore, it is important to reduce the amount of communications generated. In order to achieve this objective, we introduce the idea of a cutoff value. At the start of the screening process, all worker processes screen their data, filling up a small buffer. When this buffer is full (typically when it contains ~50 poses), the contents are sent to the sink, which depending on their scores, adds them to its sorted list. When the requested sort size has been achieved by the sink (e.g., a size corresponding to 1,000 molecules), a cutoff value can be calculated. A new pose that does not meet the requirements of the cutoff can be disregarded, as it will never enter the sorted list. This cutoff value is then communicated back to the workers so that this new criteria can be applied to their local buffers.

Consideration must be given to the time at which this cutoff value should be communicated. One approach is to allow the worker processes to communicate their buffers to the sink and then request a new cutoff value (assuming that their data changed the cutoff). In another approach, when the worker initiates communication with the sink, a cutoff is received. The worker then removes entries from its buffer that do not meet the criteria and proceeds with the send. We chose the first pattern of suggested communication. At the start of the screening process, with the cutoff value set high to allow potentially all poses to be communicated to the sink, the code has a performance decrease. As the screening progresses, the cutoff is updated and communication to the sink is reduced. A logical approach to reducing this performance decrease would seem to be to set the cutoff to allow only very good dockings to enter the sink list. This approach may be undesirable because when a sorted list of 1,000 of the best-scoring poses is requested but fewer than 1,000 poses make it past the cutoff, the list will not contain 1,000 elements. A counterintuitive result, described later, shows that when other variables, such as the worker buffer size, are set sensibly, we may want to allow a large range of dockings to be considered in order to ensure shorter runtimes. Aside from the number of processors used for the VS run, three user-tunable variables exist that relate to the parallel sort, that is, the previously mentioned score cutoff starting value, the worker buffer size, and the number of elements the final sorted list should include. Failure to specify values results in the use of predefined defaults. Interestingly, the algorithm can be described as self-tuning. As long as the worker buffer size is not set to an unsuitably small or large value (e.g., ~1 or more than 1,000), the score cutoff value, which dictates communication, converges to a suitable value no matter how large its original starting value. As a consequence, small runs can be set up to explore the behavior of the program when a larger proportion of its runtime is spent with an unoptimized score cutoff.

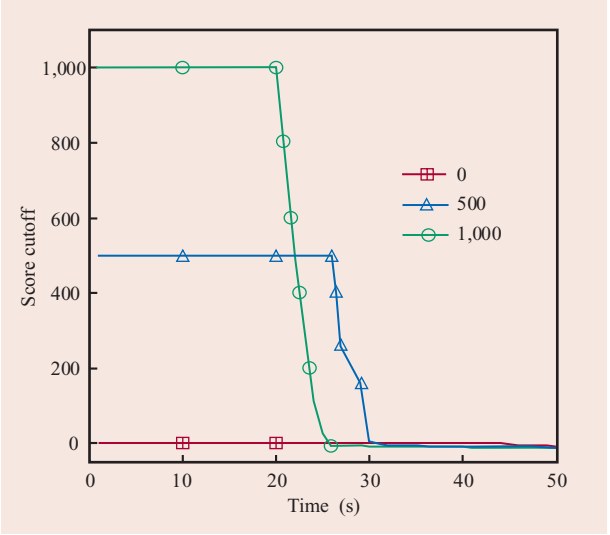

Figure 2

Using the code implementing the parallel sort, convergence of the sink score cutoff values is shown for three different initial values of score cutoff. Data was collected using the standard dataset against the receptor represented by the Protein Data Bank identifier 1FKJ.

Figure 2 represents a standard scan of 1.67 million compounds against the receptor target FKBP12. When the score cutoff is started at three drastically different values (0, 500, and 1,000), convergence of the score cutoff values occurs extremely quickly. It seems counterintuitive that the larger the score cutoff value, the slightly shorter the overall execution time. We observed runtimes of 10,983, 10,977, and 10,973 seconds, respectively, using default settings and three different starting score cutoff values, running on 128 nodes (256 processor cores). The speed of cutoff convergence can be seen in Figure 2, with the value for each scan at or less than 0 within 31 seconds of program execution. At this point, the lowest established cutoff is achieved by the scan using an initial value of 1,000. This observation for runs with a higher starting score cutoff is due to the fact that the contents of local buffers will be communicated many more times than those for runs with a lower cutoff. This higher rate of communication is caused by more molecules making it past the cutoff and filling the buffers earlier. As the scan with the higher starting score cutoff progresses, the communicated buffers will typically contain a range of results varying greatly in their score. For example, some results are just able to enter the buffer through the score cutoff, while others easily enter the buffer. In another

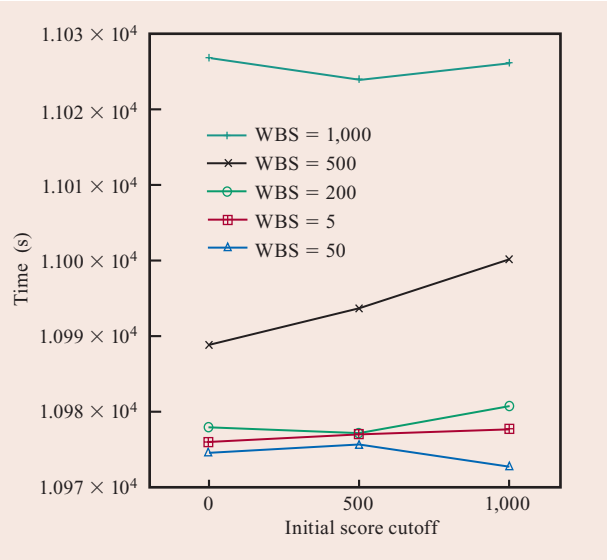

Figure 3

The execution time is shown for the standard dataset forming a complex with a receptor (Protein Data Bank identifier 1FKJ) when 256 processors are used. A range of worker buffer sizes (WBSs) and initial score cutoffs are shown.

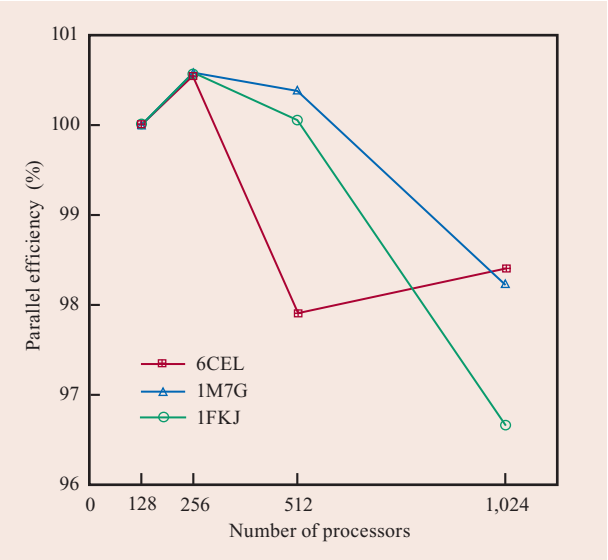

Figure 4

LIDAEUS operation with sort. Parallel efficiency is shown for a range of processors (128 as the baseline) for the standard dataset scanned against three different target receptors with Protein Data Bank identifiers shown in the key.

run with a more restrictive cutoff, the local buffers can be said to contain higher-quality results but are communicated less frequently. Essentially, results of poorer quality are being used to flush the good results in the local buffers through to the sink. A lack of communication keeps the sink cutoff values artificially unrestrictive. This flushing, and the ideal rate at which it should occur, is an artifact of the previously mentioned user-definable sorting variables. **Figure 3** shows runtimes for the standard dataset against a receptor with PDB identifier 1FKJ. A total of 256 processors are used for a range of worker buffer sizes and initial score cutoffs. It is important to note that this version of the code, which implements the parallel sort, works on a different problem than the original problem with no sort. The problem for which the initial code version was used concerned the profiling of binding pockets. Here, the best minimized and best non-minimized poses for a molecule were written to disk. This is different from the action of the code when carrying out a true VS run in which the best of all poses are to be kept or sorted. Here, every pose must be considered for output to the latter stage of sorting. One molecule has the potential to generate many poses. Thus, instead of considering just two poses per molecule, multiple poses are considered. This is reflected in profiling results; the time required for processing the same dataset in both versions of the code typically is 1.5 to 1.7 times longer when each pose generated is considered for retention. Essentially, the two codes are incomparable. As can be seen from **Figure 4**, profiling the code implementing the parallel sort has resulted in almost 100% parallel efficiency up to 1,024 processors. It is worth noting that profiling shows parallel efficiency at more than 100% in some instances. This level of efficiency is due to the fact that the value is calculated using 128 processors as the baseline, and with more processors, more worker buffers are being flushed, filling the larger sink buffer and establishing a cutoff earlier. Cache effects may also influence performance.

This result supports the previous hypotheses that the code without the parallel sort is being I/O bound at higher numbers of processors. The self-optimizing nature of the sort has ensured that most of the computation, after early convergence of the score cutoff, proceeds in almost a trivially parallel manner.

Future work concerned with the LIDAEUS code will expand the program features to enable flexible ligand docking. Development will tailor the code to the lightweight massively parallel characteristics of the BG/L architecture. We aim to ensure that the increase in execution time does not become too large. We also do not want to depart from the main goal of LIDAEUS development; that is, LIDAEUS is a tool for drug lead discovery and not for predicting the exact binding mode

of a ligand at the expense of increased and more restrictive computational runtimes. While ligand flexibility is not a new concept, the novelty of LIDAEUS lies in the scale of jobs it was designed to process. Typical LIDAEUS runs will continue to simulate more than one million receptor–ligand complexes.

Conclusion

In conclusion, LIDAEUS has been parallelized and deployed on a massively parallel system, greatly reducing the amount of computational runtime required to screen compounds against a target receptor. The initial goal of the parallelization effort was to collect information on the standard dataset of 1.67 million molecules binding to 387 binding pockets. This required the simulation of more than 646 million protein–ligand complexes, as well as 387 separate LIDAEUS runs, something unobtainable without a truly parallel version of the code and access to the massive HPC resources such as those delivered by the BG/L system. The reduced runtime of LIDAEUS jobs has been key to achieving this goal.

Each version of the parallel code has reduced HTVS runtime, previously on the order of weeks to hours. The high availability of the BG/L system has made daily runs of this magnitude a reality, enabling quicker turnaround in the identification of promising leads for useful compounds.

*Trademark, service mark, or registered trademark of International Business Machines Corporation in the United States, other countries, or both.

**Trademark, service mark, or registered trademark of Linus Torvalds, Advanced Micro Devices, Inc., or Amnon Barak in the United States, other countries, or both.

References

1. A. L. Hopkins and C. R. Groom, "The Druggable Genome," *Nat. Rev. Drug Discovery* **1**, No. 9, 727–730 (2002).
2. J. Drews, "Drug Discovery: A Historical Perspective," *Science* **287**, No. 5460, 1960–1964 (2000).
3. W. P. Walters, M. T. Stahl, and M. A. Murcko, "Virtual Screening—An Overview," *Drug Discovery Today* **3**, No. 4, 160–178 (1998).
4. D. B. Kitchen, H. Decornez, J. R. Furr, and J. Bajorath, "Docking and Scoring in Virtual Screening for Drug Discovery: Methods and Applications," *Nat. Rev. Drug Discovery* **3**, No. 11, 935–949 (2004).
5. P. D. Lyne, "Structure-Based Virtual Screening: an Overview," *Drug Discovery Today* **7**, No. 20, 1047–1055 (2002).
6. C. A. Lipinski, F. Lombardo, B. W. Dominy, and P. J. Feeney, "Experimental and Computational Approaches to Estimate Solubility and Permeability in Drug Discovery and Development Settings," *Advanced Drug Discovery Rev.* **46**, No. 1-3, 3–26 (2001).
7. J. R. Krumrine, A. T. Maynard, and C. L. Lerman, "Statistical Tools for Virtual Screening," *J. Medicin. Chem.* **48**, No. 23, 7477 (2005).
8. C. Bissantz, G. Folkers, and D. Rognan, "Protein-Based Virtual Screening of Chemical Databases. 1. Evaluation of Different Docking/Scoring Combinations," *J. Medicin. Chem.* **43**, No. 25, 4759–4767 (2000).
9. M. Stahl and M. Rarey, "Detailed Analysis of Scoring Functions for Virtual Screening," *J. Medicin. Chem.* **44**, No. 7, 1035–1042 (2001).
10. M. Vieth, J. D. Hirst, A. Kolinski, and C. L. Brooks, "Assessing Energy Functions for Flexible Docking," *J. Comput. Chem.* **19**, No. 14, 1612–1622 (1998).
11. J. R. Schames, R. H. Henchman, J. S. Siegel, C. A. Sotriffer, H. Ni, and J. A. McCammon, "Discovery of a Novel Binding Trench in HIV Integrase," *J. Medicin. Chem.* **47**, No. 8, 1879–1881 (2004).
12. O. Guner, O. Clement, and Y. Kurogi, "Pharmacophore Modeling and Three Dimensional Database Searching for Drug Design Using Catalyst: Recent Advances," *Current Medicin. Chem.* **11**, No. 22, 2991–3005 (2004).
13. G. Jones, P. Willett, R. C. Glen, A. R. Leach, and R. Taylor, "Development and Validation of a Genetic Algorithm for Flexible Docking," *J. Mol. Biol.* **267**, No. 3, 727–748 (1997).
14. B. Kramer, M. Rarey, and T. Lengauer, "Evaluation of the FLEXX Incremental Construction Algorithm for Protein–Ligand Docking," *Proteins: Structure, Function, and Bioinformatics* **37**, No. 2, 228–241 (1999).
15. T. J. A. Ewing, S. Makino, A. G. Skillman, and I. D. Kuntz, "DOCK 4.0: Search Strategies for Automated Molecular Docking of Flexible Molecule Databases," *Computer Aided Mol. Design* **15**, No. 5, 411–428 (2001).
16. D. S. Goodsell, G. M. Morris, and A. J. Olson, "Automated Docking of Flexible Ligands: Applications of AutoDock," *Computer Aided Mol. Design* **9**, No. 1, 1–5 (1996).
17. R. A. Friesner, J. L. Banks, R. B. Murphy, T. A. Halgren, J. J. Klicic, D. T. Mainz, M. P. Repasky, E. H. Knoll, M. Shelley, and J. K. Perry, "Glide: A New Approach For Rapid, Accurate Docking And Scoring. 1. Method and Assessment of Docking Accuracy," *J. Medicin. Chem.* **47**, No. 7, 1739–1749 (2004).
18. T. A. Halgren, R. B. Murphy, R. A. Friesner, H. S. Beard, L. L. Frye, W. T. Pollard, and J. L. Banks, "Glide: A New Approach for Rapid, Accurate Docking and Scoring. 2. Enrichment Factors in Database Screening," *J. Medicin. Chem.* **47**, No. 7, 1750–1759 (2004).
19. R. Abagyan, M. Totrov, and D. Kuznetsov, "ICM—A New Method for Protein Modeling and Design: Applications to Docking and Structure Prediction from the Distorted Native Conformation," *J. Comput. Chem.* **15**, No. 5, 488–506 (1994).
20. M. Vieth, J. D. Hirst, B. N. Dominy, H. Daigler, and C. L. Brooks, "Assessing Search Strategies for Flexible Docking," *J. Comput. Chem.* **16**, No. 14, 1623–1631 (1998).
21. R. Rosenfeld, S. Vajda, and C. DeLisi, "Flexible Docking and Design," *Biophysics Biomol. Structure* **24**, No. 1, 677–700 (1995).
22. I. Muegge and Y. C. Martin, "A General and Fast Scoring Function for Protein–Ligand Interactions: A Simplified Potential Approach," *J. Medicin. Chem.* **42**, No. 5, 791–804 (1999).
23. M. Jacobsson, P. Liden, E. Stjernschantz, H. Bostrom, and U. Norinder, "Improving Structure-Based Virtual Screening by Multivariate Analysis of Scoring Data," *J. Medicin. Chem.* **46**, No. 26, 5781–5789 (2003).
24. R. Wang, Y. Lu, and S. Wang, "Comparative Evaluation of 11 Scoring Functions for Molecular Docking," *J. Medicin. Chem.* **46**, No. 12, 2287–2303 (2003).
25. J. W. Raymond, M. Jalaie, and M. P. Bradley, "Conditional Probability: A New Fusion Method for Merging Disparate Virtual Screening Results," *J. Chem. Information Comput. Sci.* **44**, No. 2, 601–609 (2004).
26. M. Feher, "Consensus Scoring for Protein–Ligand Interactions," *Drug Discovery Today* **11**, No. 9-10, 421–428 (2006).
27. J. M. Yang, Y. F. Chen, T. W. Shen, B. S. Kristal, and D. F. Hsu, "Consensus Scoring Criteria for Improving Enrichment

in Virtual Screening," *J. Chem. Information Modeling* **45**, No. 4, 1134–1146 (2005).
28. M. A. Miteva, W. H. Lee, M. O. Montes, and B. O. Villoutreix, "Fast Structure-Based Virtual Ligand Screening Combining FRED, DOCK, and Surflex," *J. Medicin. Chem.* **48**, No. 19, 6012 (2005).
29. M. L. Verdonk, V. Berdini, M. J. Hartshorn, W. T. M. Mooij, C. W. Murray, R. D. Taylor, and P. Watson, "Virtual Screening Using Protein–Ligand Docking: Avoiding Artificial Enrichment," *J. Chem. Information Comput. Sci.* **44**, No. 3, 793–806 (2004).
30. L. Xing, E. Hodgkin, Q. Liu, and D. Sedlock, "Evaluation and Application of Multiple Scoring Functions for a Virtual Screening Experiment," *J. Computer Aided Mol. Design* **18**, No. 5, 333–344 (2004).
31. D. W. Zhang, Y. Xiang, A. M. Gao, and J. Z. H. Zhang, "Quantum Mechanical Map for Protein–Ligand Binding with Application to β-Trypsin/Benzamidine Complex," *J. Chem. Physics* **120**, No. 3, 1145–1148 (2004).
32. T. Xiang, F. Liu, and D. M. Grant, "Stochastic Dynamics of n-Nonane and Related Molecules in Solution Compared with Nuclear Magnetic Resonance Coupled Relaxation Times," *J. Chem. Physics* **95**, No. 10, 7576–7590 (2005).
33. K. Raha and K. M. Merz, Jr., "Large-Scale Validation of a Quantum Mechanics Based Scoring Function: Predicting the Binding Affinity and the Binding Mode of a Diverse Set of Protein–Ligand Complexes," *J. Medicin. Chem.* **48**, No. 14, 4558–4575 (2005).
34. H. J. Woo and B. Roux, "Calculation of Absolute Protein–Ligand Binding Free Energy from Computer Simulations," *Proc. Natl. Acad. Sci.* **102**, No. 19, 6825–6830 (2005).
35. H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne, "The Protein Data Bank," *Nucleic Acids Res.* **28**, No. 1, 235–242 (2000).
36. J. Bajorath, "Integration of Virtual and High-Throughput Screening," *Nat. Rev. Drug Discovery* **1**, No. 11, 882–894 (2002).
37. B. Waszkowycz, T. D. J. Perkins, R. A. Sykes, and J. Li, "Large-Scale Virtual Screening for Discovering Leads in the Postgenomic Era," *IBM Syst. J.* **40**, No. 2, 360–378 (2001).
38. G. L.-T. Chiu, M. Gupta, and A. K. Royyuru, "Preface," *IBM J. Res. & Dev.* **49**, No. 2/3, 191–194 (2005).
39. P. A. Boyle, D. Chen, N. H. Christ, M. A. Clark, S. D. Cohen, C. Cristian, Z. Dong, et al., "Overview of the QCDSP and QCDOC Computers," *IBM J. Res. & Dev.* **49**, No. 2/3, 351–365 (2005).
40. J. M. Bull, A. Gray, J. Hein, and L. Smith, "Application Performance of the Blue Gene Architecture," *Proceedings of the 2006 ACM/IEEE Conference on Supercomputing* (tutorial), Tampa, FL, 2006.
41. S. Wu, I. McNae, G. Kontopidis, S. McClue, C. McInnes, K. Stewart, S. Wang, et al., "Discovery of a Novel Family of CDK Inhibitors with the Program LIDAEUS: Structural Basis for Ligand-Induced Disordering of the Activation Loop," *Structure* **11**, No. 12, 1537–1546 (2003).
42. P. Taylor, E. Blackburn, Y. Sheng, S. Harding, K. Hsin, K. Kan, S. Shave, and M. D. Walkinshaw, "Ligand Discovery and Virtual Screening Using the Program LIDAEUS," *Br. J. Pharmacol.,* in press.
43. P. K. Weiner and P. A. Kollman, "AMBER: Assisted Model Building with Energy Refinement. A General Program for Modeling Molecules and Their Interactions," *J. Comput. Chem.* **2**, No. 3, 287–303 (1981).
44. "The *openMosix* Project," 2007; see *http://openmosix.sourceforge.net/*.

Received March 16, 2007; accepted for publication April 12, 2007; Internet publication December 11, 2007

Steven R. Shave *Institute of Structural and Molecular Biology, School of Biological Sciences, University of Edinburgh, Michael Swann Building, King's Buildings, Mayfield Road, Edinburgh, EH9 3JR (s.r.shave@sms.ed.ac.uk).* After receiving an M.Sc. degree in high-performance computing from the Edinburgh Parallel Computing Centre (EPCC) at the University of Edinburgh (2005), Mr. Shave has continued his work on the parallelization and expansion of LIDAEUS and his current pursuit of a Ph.D. degree with the Institute of Structural and Molecular Biology. His interests include massively parallel systems and the interdisciplinary crossover between computing and biology, which have been traditionally considered two distinct fields.

Paul Taylor *Institute of Structural and Molecular Biology, School of Biological Sciences, University of Edinburgh, Michael Swann Building, King's Buildings, Mayfield Road, Edinburgh, EH9 3JR (p.taylor@ed.ac.uk).* Dr. Taylor received B.Sc. (1983) and Ph.D. (1987) degrees in chemistry from the University of Edinburgh, and after 1 year of postdoctoral research using x-ray crystallography to study amino-acid/alkaloid interactions, he became the computing officer at the Department of Biochemistry at Edinburgh University, where he supported the Protein Crystallography Research Group. He subsequently became an Infomatiker with the drug design group at Sandoz Pharma A.G. in Basle, Switzerland, working on many aspects of protein structure and its interaction with small molecules, before returning to Edinburgh as a research fellow in the newly formed Structural Biochemistry Group at Edinburgh. He is currently interested in using x-ray crystallography to study interactions between proteins and small molecules and in design of software and database systems that perform *in silico* screening of large libraries of potential ligands against protein targets.

Malcolm Walkinshaw *Institute of Structural and Molecular Biology, School of Biological Sciences, University of Edinburgh, Michael Swann Building, King's Buildings, Mayfield Road, Edinburgh, EH9 3JR (m.walkinshaw@ed.ac.uk).* Professor Walkinshaw obtained B.Sc. (1973) and Ph.D. (1976) degrees from the Chemistry Department at the University of Edinburgh. After leading a structure-based drug design group in Sandoz in Switzerland for 10 years, he became the Chair of Structural Biochemistry in 1995 at the University of Edinburgh. He has published more than 200 papers on molecular recognition, protein structure, and drug discovery. His lab currently consists of 20 research fellows, Ph.D. students, and support staff who use crystallographic, biophysical, and computational approaches to study protein–ligand interactions.

Lorna Smith *Edinburgh Parallel Computing Centre (EPCC), School of Physics, University of Edinburgh, James Clerk Maxwell Building, King's Buildings, Mayfield Road, Edinburgh, EH9 3JZ (l.smith@ed.ac.uk).* Dr. Smith graduated from the University of Strathclyde, Glasgow, with B.Sc. Hons. (1994) and Ph.D. (1997) degrees in pure chemistry. Her Ph.D. focused on predicting the morphology of detergent crystals and was sponsored by Unilever Research, Port Sunlight, Liverpool. She has been at EPCC since 1997 where she has been involved in research, training, and user support. Her research interests are mainly focused on investigating new languages and models for high-performance computing (HPC) and on scaling of computer applications. Dr. Smith is the HPC Research Manager at EPCC and is responsible for a wide range of projects in the HPC and computational grid area. She manages a team of scientists who are responsible for terascaling user applications on the Blue Gene/L system at EPCC and on the U.K. National HPC System, HPCx.

Judy Hardy *Edinburgh Parallel Computing Centre (EPCC), School of Physics, University of Edinburgh, James Clerk Maxwell Building, King's Buildings, Mayfield Road, Edinburgh, EH9 3JZ (j.hardy@ed.ac.uk)*. Dr. Hardy has a B.Sc. degree in chemical physics and a Ph.D. degree in chemistry from the University of Bristol and an M.Sc. degree in software technology from Napier University in Edinburgh. Her Ph.D. research concerned microwave spectroscopy of molecules having asymmetric internal rotors. After completing her Ph.D., she worked for 10 years at Raychem Ltd., an international materials science company, in a variety of research and development roles. Dr. Hardy has worked at EPCC since 2001. She is Project Manager for a number of educational and e-learning projects and is actively involved in teaching at undergraduate and postgraduate levels within the School of Physics.

Arthur Trew *Edinburgh Parallel Computing Centre (EPCC), School of Physics, University of Edinburgh, James Clerk Maxwell Building, King's Buildings, Mayfield Road, Edinburgh, EH9 3JZ (a.trew@ed.ac.uk)*. Prior to becoming a founder member of EPCC in 1990, Professor Trew was a Research Fellow in the Department of Physics and Astronomy at the University of Edinburgh, working on computational simulations of many-body problems. He became Director of EPCC in 1997, Deputy Director of the National e-Science Centre (NeSC) in 2001, and Professor of Computational Science in 2006. He is also a Director of UOE HPCx Ltd., a wholly owned subsidiary of the University of Edinburgh, which was formed to manage the £54M and £113M HPCx and HECToR projects delivering HPC services to the U.K. academic community.