EUDOC on the IBM Blue Gene/L system: Accelerating the transfer of drug discoveries from laboratory to patient

Y.-P. Pang
T. J. Mullins
B. A. Swartz
J. S. McAllister
B. E. Smith
C. J. Archer
R. G. Musselman
A. E. Peters
B. P. Wallenfelt
K. W. Pinnow

EUDOC™ is a molecular docking program that has successfully helped to identify new drug leads. This virtual screening (VS) tool identifies drug candidates by computationally testing the binding of these drugs to biologically important protein targets. This approach can reduce the research time required of biochemists, accelerating the identification of therapeutically useful drugs and helping to transfer discoveries from the laboratory to the patient. Migration of the EUDOC application code to the IBM Blue Gene/L™ (BG/L) supercomputer has been highly successful. This migration led to a 200-fold improvement in elapsed time for a representative VS application benchmark. Three focus areas provided benefits. First, we enhanced the performance of serial code through application redesign, hand-tuning, and increased usage of SIMD (single-instruction, multiple-data) floating-point unit operations. Second, we studied computational load-balancing schemes to maximize processor utilization and application scalability for the massively parallel architecture of the BG/L system. Third, we greatly enhanced system I/O interaction design. We also identified and resolved severe performance bottlenecks, allowing for efficient performance on more than 4,000 processors. This paper describes specific improvements in each of the areas of focus.

Introduction

Pharmaceuticals may address some of the highest-priority medical challenges facing humanity, including cancers and infectious diseases. The computational screening of chemical databases for potential drug candidates has become a key component in drug discovery and has been recognized as a beneficial use of supercomputer technology.

A team of researchers at IBM and the Mayo Clinic undertook the challenge of porting, optimizing, and massively parallelizing the EUDOC** program—a molecular docking and virtual screening (VS) code developed at the Mayo Clinic—for the IBM Blue Gene/L* (BG/L) supercomputer platform. Our goal was to demonstrate that the BG/L can extend the limits of conventional VS on a commodity computing cluster and help accelerate the transfer of drug discovery from laboratory to patient. (Conventional computing clusters are termed *Beowulf clusters*—designs for high-performance parallel computing clusters on personal-computer hardware—which is commonly used to support scientific computing.) To achieve this goal, we investigated barriers to full machine utilization for screening potentially hundreds of billions of chemicals using tens of thousands of BG/L processors [1]. In this paper, we report the performance barriers that we encountered and our solutions for overcoming these barriers. The solutions and insights will be useful for porting other docking and life science applications to the BG/L system. Removal of these performance barriers led to a 34-fold speedup for screening 23,426 chemicals using the EUDOC program executed on 4,096 BG/L processors. This speedup suggests that the BG/L system is able to extend the scale of conventional VS by orders of magnitude.

0018-8646/08/$5.00 © 2008 IBM

This paper gives a brief background on VS, the EUDOC program, and the associated computer hardware. We explain, for a general audience, the computing algorithms used by EUDOC and the significance of the project. Thus, attention is given to hardware and programming details specific to EUDOC on the BG/L system, while information on the validation of the EUDOC program and its application to VS is left to other publications [2–11].

Virtual screening, EUDOC, and their dependent computer hardware

Protein structure and protein complexes

Proteins consist of linear chains of hundreds of amino acids and are usually much larger than the drug-like small molecules with which the proteins interact. Folding of the linear chains of amino acids forms three-dimensional (3D) structures with cavities. These cavities are often favorable binding sites for complexation with other molecules such as the drug-like small molecules. The intermolecular interactions between a protein and a small molecule are governed by their 3D structures and can be predicted by a computer docking program such as EUDOC, which is described below.

The completion of the Human Genome Project and the growing effort in proteomics research have recently intensified the attention being paid to 3D structures of proteins, particularly those regarded as drug targets [12, 13]. Studies of 3D protein structures advance the understanding of the genomic and proteomic information and promote the use of such insight [14, 15]. Accordingly, the Structural Genomics Initiative has been launched to determine 3D structures of globular proteins bearing unique folds [16]. To capitalize on the numerous 3D protein structures garnered from efforts of individual investigators and from the Structural Genomics Initiative, a need exists for large-scale computer docking programs. These programs search for specific conformations, positions, and orientations of two 3D structures that permit the strongest intermolecular interactions. This search is useful because the first step in essentially all biological activities is binding of one participant (a *ligand*) to a complementary, larger partner (a *receptor*). Docking programs can identify molecules with which one particular 3D structure of interest can form a complex and help researchers understand how the resulting complex elicits biological signals to other systems.

Virtual screening and the EUDOC program

Once a drug target, such as a protein target, is identified, drug discovery that identifies chemical compounds as drug candidates relies primarily on two technologies: combinatorial chemistry (including solid-phase and parallel syntheses) and high-throughput screening. The combinatorial chemistry approach is based on the premise that the greater the diversity of compounds tested, the better the chance of finding one that can be developed into a drug. High-throughput screening is a process by which many compounds can be tested automatically for activity as modulators of a particular biological target. The combination of the two technologies has been regarded as a powerful tool for drug discovery. However, the two technologies still cannot shorten the duration of the drug discovery process to the extent desired because the success rates of current high-throughput screening strategies are approximately 0.1%, and combinatorial chemistry usually requires 6 months to understand and develop reaction conditions for making structurally diverse compounds.

Computational screening (or VS), on the other hand, can be pursued by computationally docking each compound in a database into the active site of a drug target in order to identify drug candidates through evaluation of the binding affinity of the compound, controlled by charge and shape complementarity. This approach can help increase the success rate of high-throughput screening by selecting a biased subset for experimental testing. The approach also complements high-throughput screening in cases in which a particular biological assay is not suitable for robotic screening, and it affords screening of chemicals that are not yet made or not currently available.

In 1990, a computer was used to screen 10,000 chemicals in the Cambridge Crystallographic Database, leading to the identification of a haloperidol analog capable of inhibiting HIV-1 (human immunodeficiency virus 1) and HIV-2 proteases with a K_i of $\approx$100 μM [17]. (K_i is the dissociation constant of the protease-inhibitor complex.) The screening was accomplished by using a computer program called DOCK, which computationally docks each chemical into the active site of the enzymes and evaluates the shape complementarity of the docked compound relative to the active site of the two enzymes [17]. Inspired by this seminal work, the EUDOC program was devised to accomplish the following two tasks: 1) to predict ligand–receptor complexes from 3D receptor structures (referred to as *complex prediction*) and 2) to identify a subset of chemicals that is abundant with chemicals that are capable of binding to the active site of a given 3D receptor structure (referred to as *VS*). EUDOC was originally devised to perform on a commodity computing cluster of loosely connected Intel Xeon** processors [2]. It has shown success in predicting drug-bound protein complexes, identifying drug leads, and reproducing crystal structures of small-molecule complexes [3, 4, 6–11].

EUDOC is unique among molecular docking codes in that it uses conformation selection theory to address molecule flexibility. Molecular complexation in biology is best described by the conformational induction theory [18] that involves a ligand (e.g., a small molecule) binding initially to a less-compatible conformation of a receptor (e.g., a protein) and then adjusting its conformation to induce the most compatible structural conformations of the receptor. However, the conformation induction approach is not ideal for docking studies because computing the mutually dependent conformational changes of both ligand and receptor is time consuming. Alternatively, the conformation selection theory involves a scenario in which both ligand and receptor select their preformed conformations that are most compatible with each other to effect binding by shifting two equilibriums progressively from less-compatible to most-compatible conformations for both partners [19–24], where the preformed conformations are conformations at the local minima of their potential energy surfaces (i.e., local minimum conformations). When the most compatible conformers of ligand and receptor are the most prevalent, the conformation selection theory becomes the lock–key theory [18].

The conformation selection theory is ideal to computationally account for molecular flexibility in docking, because it can convert a ligand–receptor association best described by the conformational induction theory to a series of associations, each of which can be described by the lock–key theory [18]. The conformation selection theory thereby affords vast opportunities for massively parallel computing and enables a EUDOC-based docking study to be performed on thousands of BG/L processors with high efficiency [1, 2].

In order to use EUDOC, a user specifies a docking region, that is, a rectangular box (termed the *docking box*) that encloses a binding pocket of a receptor (**Figure 1**). The docking box confines the translations of the mass center of the ligand. The EUDOC program then generates different ligand–receptor complexes by a systematic combination of translations of the ligand along the *x*-, *y*- and *z*-axes and rotations of the ligand around the *x*-, *y*- and *z*-axes within the docking box with user-defined increments. These increments are expected to be between 0.25–1.5 Å and 5–30 degrees of arc, although the program handles cases outside of these ranges. As described in [2], the initial search is done by rotation at a 10-degree-of-arc increment and translation at 1.0-Å increments. A local optimization follows by rotation at a 5-degree-of-arc increment and translation at a 0.25-Å increment in a region within 20 degrees of arc and 2.0 Å. The translations and rotations are iterated with numerous conformations of both ligand and receptor. Different

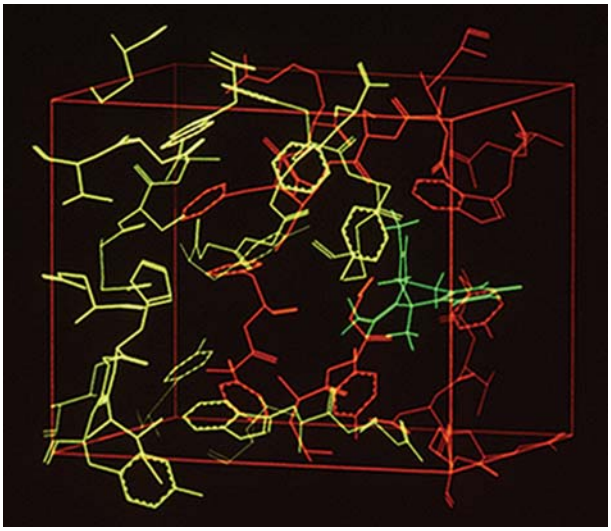

Figure 1

Example of a docking box.

conformations of the ligand and receptor are generated by a conformational search of the ligand structure and a molecular dynamics simulation of the receptor structure in water using commercially available conformational search and molecular dynamics simulation codes. The EUDOC program then calculates the intermolecular interaction energies of all the generated ligand–receptor complexes and identifies the most energetically favorable ligand–receptor complexes. The interaction energy is calculated from the potential energy of the ligand–receptor complex relative to the potential energies of the ligand and protein in their free state.

The result of the docking study is governed by the translational and rotational increments, the size of the docking box employed, and the numbers of different conformations of ligand and receptor used. The convergence of the docking results using the EUDOC program—namely that the different binding modes have been sampled sufficiently—can be confirmed by repeating the calculation with smaller translational and rotational increments.

During the outbreak of severe acute respiratory syndrome (SARS) in 2003, the EUDOC program was used to identify anti-SARS drug candidates by docking 361,413 chemicals against a computer-generated 3D model of a chymotrypsin-like cysteine proteinase (CCP) from a SARS-associated coronavirus [9, 25]. CCP is an ideal drug target for treating SARS viral infection because it is required for viral replication and transcription. The 3D model of CCP was generated from its genetic sequence by homology modeling and multiple

Table 1 Comparison of characteristics: Beowulf cluster vs. the Blue Gene/L system.

	Cluster	*Blue Gene/L system*
Processor	One to two CPU sockets (possibly multicore) per board; high-frequency Intel Xeon, AMD Opteron**, etc.	512 node cards (1,024 processors) per midplane; 700-MHz IBM PowerPC* 440
Interconnect	10–1,000 Mb/s Ethernet (e.g., Myrinet**, Quadrics**, InfiniBand**)	Five specialized networks for point-to-point, collective, or I/O tasks
File system	Local drives for each node are common	Network-attached shared file system

molecular dynamics simulations [25]. The ligand set used in the study was selected from a Mayo Clinic in-house chemical database that contained 2.5 million drug-like molecules, 70% of which are commercially available. The selection criteria were that the number of conformation-governing rotatable bonds of each selected chemical must be fewer than four and that the molecular weights of these selected chemicals must be in the range of 400–9,000 Dalton.

VS against CCP using EUDOC was performed on a commodity computing cluster using 396 Xeon 2.2-GHz processors. One of us (Y.-P. P.) devised this cluster at the Mayo Clinic; it has 235 nodes that are connected by 100-Mb Ethernet, and each node has two processors, 512 MB of memory, and an 80-GB local disk. This VS identified 12 chemicals for antiviral testing. Of the 12 chemicals tested in cell-based inhibition assays, one inhibited the human SARS-coronavirus Toronto-2 strain with an effective concentration (EC_{50}) of 23 μM, and four others exhibited 13% to 17% inhibition at a drug concentration of 32 μM [9]. The most potent inhibitor lead overlays well with a reported scissile-bond-containing substrate fragment bound in the active site of CCP [25]. These results demonstrate that, given target information at the gene level only, the VS method can identify chemicals that penetrate and rescue cells from viral infection. This is also an important validation of the VS approach using the drug-target information at the gene level.

The commodity computing cluster performance of the VS for SARS virus inhibitors summarized above is used as a baseline for much of the rest of this paper. The ligand set commonly used in this study was a 23,426-ligand subset selected from the 361,413 ligands described above. The selection criterion was that the molecular weight of each selected chemical from the 361,413-ligand set must be in the range of 400–420 Dalton. The runtime of the commodity computing cluster set the expectations for a state-of-the-art search by EUDOC at 242 minutes. However, after fully optimizing the serial code performance of EUDOC, we found it effective to use a worst-case receptor [farnesyltransferase (FTase)] and an associated selected subset test case in determining performance improvements. The meaning of *worst case* and the details of this aspect of the study are described below in the section on serial code optimizations.

Computer hardware for virtual screening

The computation for the VS described above fits the definition of an embarrassingly parallel calculation. (Using the parlance of parallel computing, an *embarrassingly parallel calculation* is one for which no particular effort is needed to segment the problem into numerous parallel tasks.) The workload is segmented into independent units of work by the nature of the problem, and these workunits can be processed in any order with no dependencies among them. Commodity clusters are well suited to VS.

However, the commodity computing cluster solution limits the scale of VS primarily because of the space limitation at machine installation sites. Extending the scale of conventional VS by orders of magnitude requires the supercomputing-level system designs that currently encompass large-scale computations [26–30]. Given the sizes of available chemical databases (containing billions of compounds) [1] and the nontrivial nature of runtime load-balancing of large-scale VS (see below), large-scale VS can be considered a supercomputer-level problem instead of a cluster-level problem. Thus, there should be motivation to migrate or port a VS code from clusters to leading-edge supercomputers such as the BG/L computer. For clarity, a short comparison of the clusters and the BG/L system is provided in **Table 1**.

Challenge of migrating a virtual screening code from clusters to the BG/L system

With VS, two processors should search through a database twice as fast as one, or 16,384 processors twice as fast as 8,192, because each ligand–receptor pairing is an independent search. While this type of screening task is trivially parallelizable over a few processors, parallelization over thousands of processors is not an easy endeavor. The runtime or computational load for each processor must be carefully balanced because the

time to complete the screening task is the time for the last processor to finish its complete set of assigned workunits. This balance is especially important when using the thousands of processors available on a BG/L system. Some basic assumptions change when moving from a cluster to the BG/L system. The single-node performance and I/O bandwidth of the BG/L system are much lower compared to a cluster, but the network bandwidth and the number of processors on the BG/L system are much higher. While each cluster node typically has a local drive, the BG/L system uses a shared file system. These differences present challenges and new opportunities that are addressed below.

Porting, optimization, and massive parallelization of EUDOC on the BG/L system

Project aims

With its architecture of many densely packed, low-cost nodes, and in view of one installation of the machine ranking as number one on the TOP500** list of most powerful supercomputers in the world in 2005, the BG/L system seemed to be an ideal platform for scaling up conventional VS by orders of magnitude. The high computing efficiency of the BG/L system in terms of dollar/flops (floating-point operations per second), watts/flops, cooling dollar/flops, and floor space/flops is reported in Reference [31]. In order to achieve our goal of demonstrating that the BG/L system can extend beyond the limits of conventional VS on commodity computing clusters and help accelerate the transfer of drug discovery from laboratory to patient, we set out to port, optimize, and massively parallelize EUDOC for the BG/L system through a collaboration between the Computer-Aided Molecular Design Laboratory of the Mayo Clinic and the Development Laboratory of IBM in Rochester, Minnesota. The specific aims of this collaboration were as follows:

1. Maintain accuracy of VS results on the BG/L system.
2. Enable much larger VS studies on the BG/L system.
3. Approach an interactive solution response for results, given sufficient BG/L nodes.
4. Identify barriers to full machine utilization on the BG/L system.

Realization of these aims offers insights into porting other docking and life science applications on the BG/L system and assistance with hardware and project planning.

Porting EUDOC to the BG/L system

The Linux** operating system-like compute node kernel on the BG/L system [32] provided most of the functionalities required by the EUDOC application, with a few exceptions such as the lack of the fork capability. The missing functionalities were easily overcome by simple modification of the EUDOC code to better match the BG/L environment, as discussed further in the following subsection. The parallelization portion of the EUDOC code had to be completely revised because the original code was devised to work on a cluster of processors connected by the 100-Mb Ethernet.

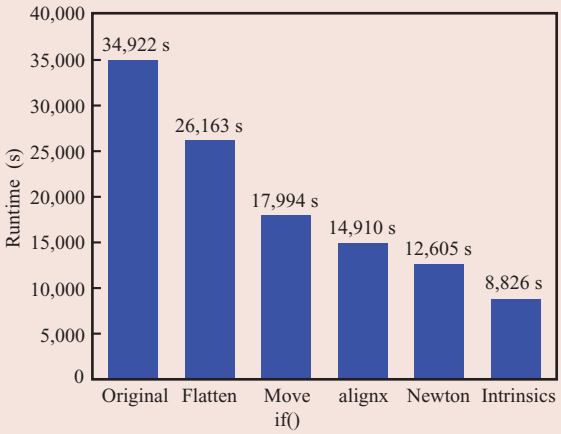

Figure 2

Single-node performance and code improvements. Note how the single-node performance improves as successive optimizations are incorporated into the EUDOC code, resulting in a four-fold improvement over the original. The *x*-axis indicates various kinds of optimizations and code revisions.

Serial code optimizations

The original EUDOC code uses a modular software design and has approximately a dozen separate programs connected with system calls. Because the BG/L kernel does not have the fork capability, the system call in C does not work. However, this problem was solved by changing each program to a subroutine.

One important preparatory step in optimizing the serial code involved changing the code to use double-precision floating-point calculations instead of single precision. This was a prerequisite for the use of the BG/L SIMD (single-instruction, multiple-data) floating-point units (FPUs), which operate on double-precision values [33]. We initially focused on the computationally intensive eudoc800 subroutine. In order to take advantage of the BG/L hardware, the code was optimized specifically for the BG/L system. **Figure 2** shows the performance improvements in eudoc800 obtained by compiler optimization options, algorithmic revisions, and code revisions. Timings shown in Figure 2 were obtained for eudoc800 runs using a ten-ligand subset of the above

Table 2 Improvements to EUDOC inner loop source code.

(a) Sample code improvements

Original code

```
For j < count
  tmp = foo[i]−bar[j]
  if (tmp < VAL)
    goto next;
```

Move if ( )

```
For j < count
  tmp[j] = foo[i]−bar[j];
For j < count
  if (tmp [j] < VAL)
    goto next;
```

__fsel

```
double jp=zero=0.0,one=1.0, diff;
for j < count
  tmp = foo[i]−bar[j]
  diff = VAL− tmp
  diff= __fsel (diff, zero, one);
  jp += diff;
if (jp > 0.0)
  goto next
```

(b) Code with full optimizations

```
const Complex double
HLIM = __cmplx(CONSTANT, CONSTANT),
one = __cmplx(1.0, 1.0),
zero = cmplx(0.0, 0.0);
Complex double tmp, cfoo, jp=zero;
double foo;
int jbatch=count−9;
cfoo= __cmplx(foo, foo);
alignx(16,bar);
#pragma unroll (5)
for(j=0;j<jbatch;j=j+2)
{
  tmp = __lfpd (&bar[j]);
  tmp = __fpsub (cfoo, tmp); //tmp=foo−bar[j]
  tmp = __fpnmsub (HLIM, tmp, tmp); //tmp=CONSTANT−tmp*tmp
  tmp = __fpsel(tmp, zero, one);
  jp  = __fpadd(jp, tmp); //Accumulate jump registers
}
if (( __creal (jp) + __cimag (jp))>0.0)
goto next:
```

described 23,426-ligand database against FTase [4, 34]. FTase and the ten ligands were chosen to reflect the computational characteristics of a worst-case docking scenario because the receptor has a huge binding pocket, which makes it challenging to find an optimal fit. Additionally, more optimization runs of eudoc800 could be performed in a short time. Each bar in Figure 2 reflects a level of performance achieved after accomplishing specific improvements as described below.

The first bar labeled "Original" shows the timing for the eudoc800 module in its base form as optimized with the `−O3` compiler option.

By focusing on the optimization of some key calculation code kernels, mainly the `vrec` reciprocal evaluation that all Lennard–Jones energy evaluations require, we minimize time spent in the library routines. Loops in the eudoc800 subroutine were combined in order to eliminate unnecessary arrays for intermediate results of computations and to improve timings by reducing the amount of traffic to and from memory. The arrays were replaced by scalar temporary variables that could be mapped by the compiler to register storage on the processor, thereby reducing the number of memory accesses. The overall improvement produced by these revisions is indicated by the second bar in Figure 2 (labeled "Flatten").

EUDOC computes intermolecular distances between the receptor and the ligand. The distance calculation loop made use of a conditional statement in the original code [see column 1 of **Table 2(a)**], preventing the use of SIMD instructions on the BG/L system. This problem was solved by separating the compute-intense portion of the loop (which contained the distance calculation) from the portion that contains the `if ( )` statement, as illustrated in column 2 of Table 2(a). In Figure 2, the timing obtained from this change is indicated by the third bar [labeled "`Move if ( )`"].

The data operands used in the calculations were aligned (forced to start on a 16-byte boundary) and the compiler informed of the alignment with the alignx directive. The resulting performance improvement is illustrated by the fourth bar (labeled "alignx").

As described in [35], the `vrec` function uses the reciprocal estimate instruction (`__fpre`) to obtain the first estimate for the reciprocal of a number (accurate to 13 bits), and then performs two Newton's iterations to refine the reciprocal to the double-precision required accuracy.

Here, Newton's method is used to solve the equation $f(x) = a - 1/x = 0$, when the reciprocal of a is desired: `x0 = __fpre(X); Xi+1 = Xi+Xi* (1.0–a*Xi)`.

In this study, we found that eudoc800 retained enough precision without two iterations of Newton's method because the VS results using the revised eudoc800 were the same as those obtained using the original eudoc800 code. This observation is consistent with the example given in [33]. The improvement of this revision is shown by the fifth bar (labeled "Newton").

We further improved the eudoc800 code by using the data-select instruction as shown in the "`__fsel`" column of Table 2(a). The `__fsel` function shown here is equivalent to `if (diff >=0) diff=one else diff=zero`.

The `__fsel` built-in function maps directly to an IBM PowerPC processor assembly instruction. This replaces the final loop containing the `if ( )` with a single `if ( )` statement that tests whether the branch is taken. Consequently, the compiler can utilize the SIMD instructions to improve dataflow through the caches and to pipeline this loop much more efficiently than the original code.

Additional optimizations of the loop are shown in **Table 2(b)**. The tmp and jp variables are now `__Complex` double scalars, which map directly to the primary register–secondary register pairs in the register file of the processor. The loop is unrolled by five. (When we use the phrase "loop is unrolled," we refer to the process in which the instructions that are called in multiple iterations of the loop are combined into a single iteration.) This matches the floating-point latency for multiply–adds (and most other SIMD floating-point instructions). When unrolling by five, the two floating-point registers, used for storing five instructions previously, are available to be accessed when needed for the next operation, because the compiler has placed four other instructions between these register references during the loop unrolling process. The compiler still manages the floating-point register usage, memory references, and instruction scheduling for this loop.

Newer XL compiler levels also contribute to performance. Upgrading the XL C compiler from version 7.0 to 8.0 decreased runtimes by 21%. The timing associated with these changes is shown by the sixth bar (labeled "Intrinsics").

The changes described in this section yielded a fourfold improvement compared with the original eudoc800 code compiled with the `–O3` optimization option. Compared to the "Flatten" bar in Figure 2, hand-tuning yielded a threefold improvement, underscoring the value of hand-tuning as a complement to state-of-the-art compiler technology.

Parallel code optimization

The EUDOC program uses a master/worker scheme implemented by a message-passing interface (MPI) on a cluster of up to 800 Xeon processors connected with the 100-Mb Ethernet. The master node dispenses workunits on request from the worker nodes. The size of the workunit is a predetermined number of ligands sent to each worker node for screening. In the EUDOC code for the cluster implementation, the workunit size was 65 ligands. This size provided a balance between best-case total workload completion time and prevention of master overload due to network bottleneck conditions.

The workunit size was examined on the BG/L system to understand how to take advantage of the BG/L architecture, which can have up to 65,536 dual-core nodes connected with a high-speed network [36]. We found that load balance could be greatly improved by using small workunits because the interprocessor communication network on the BG/L system is specifically designed for high bandwidth and readily accommodates more master/worker interchange traffic than the network on a commodity computing cluster. Communication bottlenecks, which would have been typical on a cluster running with data partitioned into smaller workunits, were never encountered on the BG/L system even when the workunit size was reduced to one ligand.

The four plots in **Figure 3(a)** illustrate the improvement in the balance of runtimes across processors in the parallel execution of EUDOC jobs as the workunit size varies. A flat graph is indicative of processors finishing work at similar times and represents full machine and processors utilization. At a workunit of 60 ligands, which is comparable to the cluster implementation, we observed significant variations in processor runtimes. Decreasing the workunit size on the BG/L system provided a significantly improved balance between processing time and prevention of master node overload. Machine utilization is measured by the ratio of the average total runtime for all processors to the maximum total runtime of any single processor. A significant difference indicates that the majority of processors have completed their entire set of assigned workunits and become idle, while a small number of processors are still doing work. This indicates that completion time could be shortened if tasks could be allocated more evenly among processors. According to this measurement, the machine utilization on the BG/L system was 53% for the 60-ligand workunit case and 92% for the 3-ligand workunit scenario.

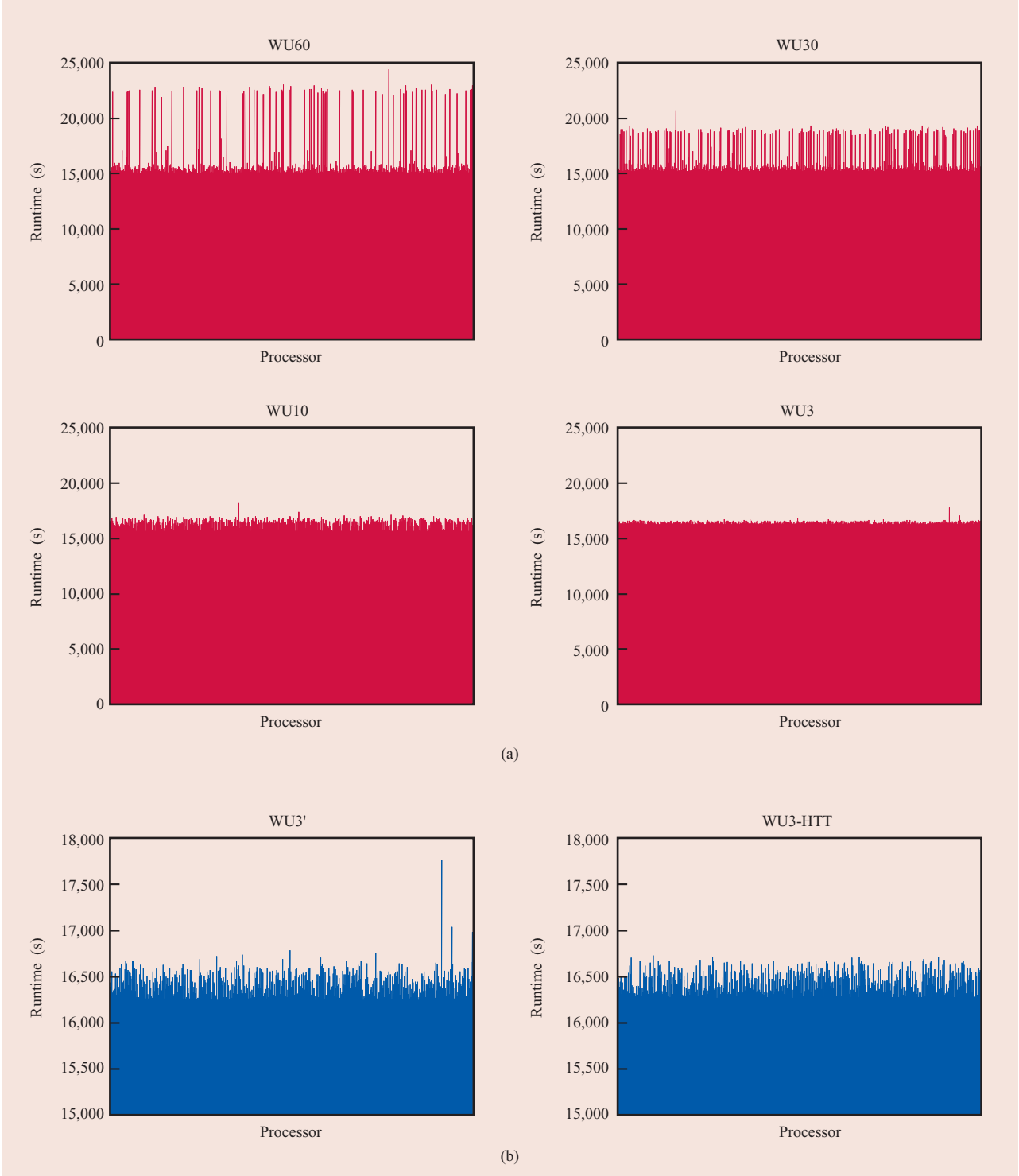

Figure 3

Improving processor load balance. (a) Improvement in load balance through reduction in workunit (WU) size. Smaller granularity dispatches of work make the processor runtimes more similar. Note the more balanced distribution of runtimes as WU size is reduced from 60 to 3. Job runtime is determined by the longest running processor time, which drops from 24,500 seconds to 17,800 seconds with WU3—a performance improvement of 27%. (b) Additional refinement in load balance is achieved through an HTT pre-sort scheme for ligand input data file. See text for details. The *x*-axis corresponds to 1 through 512 processors.

We observed that even using a three-ligand workunit size, a few processors completed their runs with wall-clock times noticeably longer than other processors. This problem could be solved by using a one-ligand workunit on the BG/L system. However, this solution would increase network traffic. In this study, we observed the runtime for docking each ligand on a BG/L processor to be generally proportional to the number of atoms in the ligand. Therefore, a head-tail-tail (HTT) scheme was used to improve the load balance by sorting the database in order of descending number of atoms in the ligand and then distributing one ligand from the head of the sorted list (corresponding less runtime) and two from the tail of the list (corresponding to more runtime) to each worker. As shown in **Figure 3(b)**, this scheme improved the machine utilization from 92% to 98% for the best-case performance of the three-ligand workunit case without increasing network traffic. The plot on the left is an expanded scale for the workunit size of 3 that is shown in Figure 3(a). Note how the remaining spike of processor runtime is eliminated by the HTT algorithm, as shown in the plot on the right in Figure 3(b), achieving a near-flat set of balanced runtimes. Final runtime is lowered to 16,700 seconds, for an overall performance gain of 32% compared with the 60-workunit original case.

Sorting a database according to the number of atoms in the ligand can be done before the VS and is a one-time investment. Therefore, the HTT, head-tail, or head scheme is recommended for performing massively parallel calculations at fine granularities. Other load-balancing approaches [37, 38] are worth exploring as well in follow-on work. Smaller workunits generally produce better parallelism and lead to better machine utilization. The workunit size on the BG/L system can be as small as one ligand because of the high-bandwidth communication interconnect of the BG/L system. This fine granularity in assignment of workunits provides the greatest opportunity for balancing the CPU loads across all processors. However, in this mode of operation, we recommend sorting a database according to the number of atoms in the ligand prior to VS because of the large differences in ligand size, which has a significant effect on the runtimes for the one-ligand workunits.

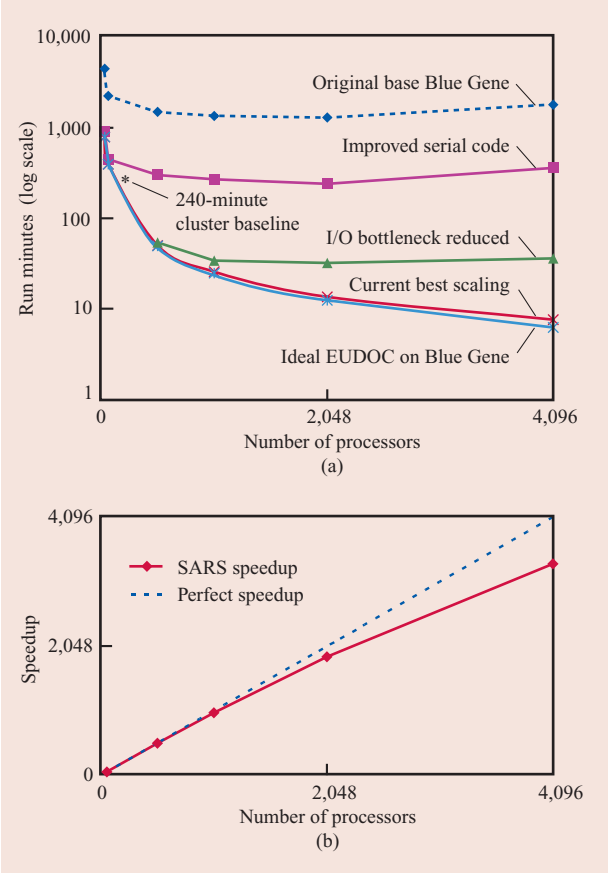

Figure 4

EUDOC runtime improvements. (a) SARS virus 23,426-ligand test-case runtimes. Good scaling at 82% efficiency or better was eventually obtained for EUDOC. The curve labeled "current best scaling" illustrates runtimes for this version of the application, with best performance at 4,096 processors, corresponding to the rightmost point of the curve. The shared file system exhibits bottlenecks at larger configurations. Initial versions of the code demonstrated elapsed times no better than the 396 Xeon node cluster baseline comparison, even with enhancements as provided in "improved serial code." Minimizing I/O allowed more processors to contribute to speedup by reducing load on the centralized shared file system. (b) Resulting SARS virus 23,426-ligand test-case speedup, with excellent efficiency demonstrated through 4,096 processors.

I/O barrier to massive parallelization

The serial and parallel code optimizations made the revised EUDOC code scale reasonably well for up to 512 BG/L processors. Profiling the revised code on large numbers of BG/L processors identified the I/O bottleneck that caused poor scaling of EUDOC over more than 512 BG/L processors. The scaling characteristics of several revisions of the EUDOC code are summarized in **Figure 4**. The curve denoting the original base Blue Gene* shows that the original EUDOC code ran slower on the BG/L system (1,300 minutes) than on a commodity computing cluster (242 minutes) and that it scaled poorly on the BG/L system. Although the single-processor runtimes were significantly improved, as shown by the curve denoting improved serial code, the revised code still ran no faster on the BG/L system than on a commodity computing cluster regardless of the number of BG/L processors used. It was only when the I/O loads were reduced significantly, as shown by the curve labeled "I/O

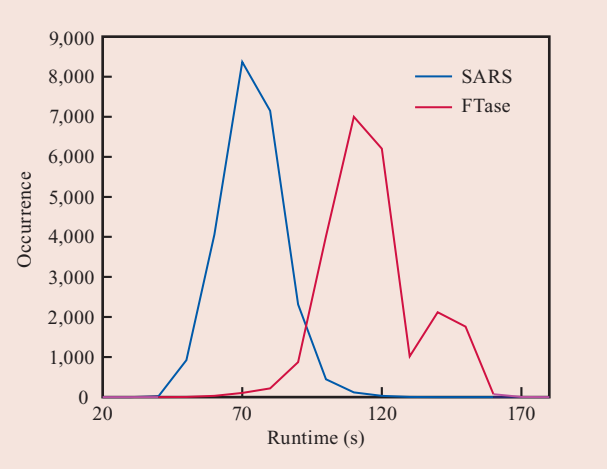

Figure 5

Histogram of runtimes for the 23,426-ligand database run against the SARS and FTase receptors. Both cases scale similarly when the CPU load is balanced through single-ligand workunit distribution.

bottleneck reduced," that EUDOC on the BG/L system scaled to well more than 1,024 BG/L processors and ran faster on the BG/L (32 minutes) system than on a commodity computing cluster (242 minutes). Further reduction of the I/O loads—by replacing dynamic memory allocation with static array declaration and other significant code revisions specific to EUDOC—improved the scalability of the EUDOC code on the BG/L system from 1,024 to 4,096 BG/L processors and, most importantly, reduced the wall-clock (elapsed) time of screening 23,426 ligands against CCP for anti-SARS drug candidates from 242 minutes using 396 Xeon processors (2.2 GHz) on a commodity computing cluster to 13 and 7 minutes using 2,048 and 4,096 PowerPC 440 processors (700 MHz) on the BG/L system, respectively, as shown by the "Current best scaling" curve.

Similar scalability was observed when screening 23,426 ligands from the same database against FTase. Because FTase has a larger binding site than CCP, the runtime for each ligand against FTase is generally 50% longer than that of CCP (**Figure 5**). This alleviates the I/O loads for screening against FTase because of the relatively low frequency of workunit dispatching. A similar scalability was also observed when the database was increased from 23,426 to several hundred thousand ligands by concatenating several instances of this database.

The EUDOC program was originally designed to execute in stages on a commodity computing cluster with independent file systems. For each phase of execution, all outputs for a succeeding stage are sent to disk. For large-scale EUDOC runs on a BG/L system with a shared file system, this means the runtime for each processor depends on how fast the shared file system can handle concurrent accesses to hundreds of thousands of independent files in the 23,426 ligand test case. The number of these files can increase to the millions for a larger database. EUDOC adjustments were made reducing the amount of I/O produced. Consequently, the application scaled well up to configurations of 4,096 processors, but it required additional modifications to scale further. Such modifications, employing the unique internal networks of the BG/L system and not possible on commodity computing clusters, would lead to embarrassingly parallel status through file system I/O reduction.

A distributed file system such as the IBM General Parallel File System* (GPFS*) can in theory improve the scaling further. However, because of the modular nature of VS (i.e., a large-database screen can be divided into small-database screens, each of which can be performed on one or two racks of a BG/L system with up to 4,096 processors), the current best-scaling version of the EUDOC code on the BG/L system offers a practical solution to the large-scale VS problem.

The I/O barrier identified herein offers an insight into porting other docking and life science applications to a BG/L system. Porting embarrassingly parallel codes, which run well on commodity computing clusters that have multiple file systems, to the BG/L system incorporating a shared file system requires attention to the I/O bottleneck. As demonstrated above, the EUDOC code runs slower on the BG/L system than on a commodity computing cluster if no improvements are made to the I/O implementation, to the compute kernels, and to the load balance of the program.

The BG/L system: An ideal platform for large-scale virtual screening

The study described in this paper offers a proof of concept that the BG/L system can provide significant speed improvement for VS using the EUDOC program optimized for the BG/L system, allowing larger databases to be screened in a shorter time. With further work on I/O improvement, EUDOC would better fit the archetype of a trivially parallel VS program and search potentially hundreds of billions of chemicals in real time, given a sufficiently large BG/L configuration.

Much of this study focused on a real-life VS of 361,413 chemicals against CCP, which identified a small-molecule CCP inhibitor exhibiting an EC_{50} of 23 μM in a cell-based assay [9]. Screening a subset of 23,426 chemicals from the 361,413-chemical database that also identified the 23-μM inhibitor required 242 minutes on a commodity computing cluster using 396 Xeon 2.2-GHz processors. The BG/L implementation described herein

was able to reduce this time to 7 minutes using 4,096 BG/L processors, representing a 34-fold speedup compared with the commodity computing cluster baseline and a 186-fold speedup compared with the initial EUDOC code ported to the BG/L system (1,300 minutes).

Given the outstanding performance of VS on the BG/L system and the modular nature of VS, the BG/L system is an ideal platform for large-scale VS that enables the genome-to-drug-lead approach—that is, identifying drug candidates by using only genomic information as exemplified by the identification of the CCP inhibitor [1]. The genome-to-drug-lead approach empowered by the BG/L system helps to reduce the manpower requirements of searching for drug leads—which is one of the greatest limitations to drug discovery—and exploits the extensive availability of drug targets at the gene level, ultimately enhancing the possibility of accelerating progress from laboratory to patient.

Acknowledgments

We gratefully acknowledge the support from the Defense Advanced Research Projects Agency (DAAD19-01-1-0322), the U.S. Army Research Office (DAAD19-03-1-0318), the U.S. Army Medical Research Acquisition Activity (W81XWH-04-2-0001), the National Institutes of Health (5R01AI054574-03), the IBM Blue Gene/L Life Sciences Center of Excellence, the Mayo Clinic–IBM Center of Excellence, the High Performance Computing Modernization Program of the U.S. Department of Defense, the San Diego Supercomputing Center, the University of Minnesota Supercomputing Institute, the Compaq Medical Sciences Group, the Jay and Rose Phillips Family Foundation, and the Mayo Foundation for Medical Education and Research.

References

1. Y.-P. Pang, "In Silico Drug Discovery: Solving the 'Target-Rich and Lead-Poor' Imbalance Using the Genome-to-Drug-Lead Paradigm," *Clin. Pharmacol. Ther.* **81**, No. 1, 30–34 (2007).
2. Y.-P. Pang, E. Perola, K. Xu, and F. G. Prendergast, "EUDOC: A Computer Program for Identification of Drug Interaction Sites in Macromolecules and Drug Leads from Chemical Databases," *J. Comput. Chem.* **22**, No. 15, 1750–1771 (2001).
3. Q. Wang and Y.-P. Pang, "Accurate Reproduction of 161 Small-Molecule Complex Crystal Structures Using the EUDOC Program: Expanding the Use of EUDOC to Supramolecular Chemistry," *PLoS ONE* **2**, No. 6, e531 (2007).
4. E. Perola, K. Xu, T. M. Kollmeyer, S. H. Kaufmann, F. G. Prendergast, and Y.-P. Pang, "Successful Virtual Screening of a Chemical Database for Farnesyltransferase Inhibitor Leads," *J. Med. Chem.* **43**, No. 401–408 (2000).
5. M. A. Miller, "Chemical Database Techniques in Drug Discovery," *Nat. Rev. Drug Discov.* **1**, No. 3, 220–227 (2002).
6. M. L. Raves, M. Harel, Y.-P. Pang, I. Silman, A. P. Kozikowski, and J. L. Sussman, "Structure of Acetylcholinesterase Complexed with the Nootropic Alkaloid, (–)-Huperzine A," *Nat. Struct. Biol.* **4**, No. 1, 57–63 (1997).
7. Y.-P. Pang, K. Xu, T. M. Kollmeyer, E. Perola, W. J. McGrath, D. T. Green, and W. F. Mangel, "Discovery of a New Inhibitor Lead of Adenovirus Proteinase: Steps Toward Selective, Irreversible Inhibitors of Cysteine Proteinases," *FEBS Lett.* **502**, No. 3, 93–97 (2001).
8. H. Sun, Y.-P. Pang, O. Lockridge, and S. Brimijoin, "Re-engineering Butyrylcholinesterase as a Cocaine Hydrolase," *Mol. Pharmacol.* **62**, No. 2, 220–224 (2002).
9. A. J. Dooley, N. Shindo, B. Taggart, J. G. Park, and Y.-P. Pang, "From Genome to Drug Lead: Identification of a Small-Molecule Inhibitor of the SARS Virus," *Bioorg. Med. Chem. Lett.* **16**, No. 4, 830–833 (2006).
10. J. G. Park, P. C. Sill, E. F. Makiyi, A. T. Garcia-Sosa, C. B. Millard, J. J. Schmidt, and Y.-P. Pang, "Serotype-Selective, Small-Molecule Inhibitors of the Zinc Endopeptidase of Botulinum Neurotoxin Serotype A," *Bioorg. Med. Chem.* **14**, No. 2, 395–408 (2006).
11. J. Tang, J. G. Park, C. B. Millard, J. J. Schmidt, and Y.-P. Pang, "Computer-Aided Lead Optimization: Improved Small-Molecule Inhibitor of the Zinc Endopeptidase of Botulinum Neurotoxin Serotype A," *PLoS ONE* **2**, No. 8, e761 (2007).
12. J. C. Venter, M. D. Adams, E. W. Myers, P. W. Li, R. J. Mural, G. G. Sutton, H. O. Smith, et al., "The Sequence of the Human Genome," *Science* **291**, No. 5507, 1304–1351 (2001).
13. R. E. Banks, M. J. Dunn, D. F. Hochstrasser, J. C. Sanchez, W. Blackstock, D. J. Pappin, and P. J. Selby, "Proteomics: New Perspectives, New Biomedical Opportunities," *Lancet* **356**, No. 9243, 1749–1756 (2000).
14. Y.-P. Pang, "Novel Acetylcholinesterase Target Site for Malaria Mosquito Control," *PLoS ONE* **1**, No. 1, e58 (2006).
15. Y.-P. Pang, "Species Marker for Developing Novel and Safe Pesticides," *Bioorg. Med. Chem. Lett.* **17**, No. 197–199 (2007).
16. S. K. Burley, S. C. Almo, J. B. Bonanno, M. Capel, M. R. Chance, T. Gaasterland, D. W. Lin, A. Sali, F. W. Studier, and S. Swaminathan, "Structural Genomics: Beyond the Human Genome Project," *Nat. Genet.* **23**, No. 2, 151–157 (1999).
17. R. L. DesJarlais, G. L. Seibel, I. D. Kuntz, P. S. Furth, J. C. Alvarez, P. R. Ortiz de Montellano, D. L. DeCamp, L. M. Babe, and C. S. Craik, "Structure-Based Design of Nonpeptide Inhibitors Specific for the Human Immunodeficiency Virus 1 Protease," *Proc. Natl. Acad. Sci. USA* **87**, No. 17, 6644–6648 (1990).
18. D. E. Koshland, "The Key–Lock Theory and the Induced Fit Theory," *Angew. Chem. Int. Ed. Engl.* **33**, No. 23–24, 2375–2378 (1995).
19. A. S. Burgen, "Conformational Changes and Drug Action," *Fed. Proc.* **40**, No. 13, 2723–2728 (1981).
20. R. F. Bruns, "Conformational Induction Versus Conformational Selection: Evidence from Allosteric Enhancers," *Trends Pharmacol. Sci.* **17**, No. 5, 189–191 (1996).
21. T. Kenakin, "Receptor Conformational Induction Versus Selection: All Part of the Same Energy Landscape: Agonists Can Differentially Stabilize Multiple Active States of Receptors," *Trends Pharmacol. Sci.* **17**, No. 190–191 (1996).
22. Y.-P. Pang, N. D. Silva, C. Hydock, and F. G. Prendergast, "Docking Studies on the Complexed and Uncomplexed FKBP12 Structures with Bound and Unbound Ligands: An Implication of Conformational Selection Mechanism for Binding," *J. Mol. Model.* **3**, No. 240–248 (1997).

23. Q. Wang and Y.-P. Pang, "Preference of Small Molecules for Local Minimum Conformations when Binding to Proteins," *PLoS ONE* **2**, No. 9, e820 (2007).
24. Q. Wang and Y.-P. Pang, "Normal-Mode-Analysis-Monitored Energy Minimization Procedure for Generating Small-Molecule Bound Conformations," *PLoS ONE* **2**, No. 10, e1025 (2007).
25. Y.-P. Pang, "Three-Dimensional Model of a Substrate-Bound SARS Chymotrypsin-Like Cysteine Proteinase Predicted by Multiple Molecular Dynamics Simulations: Catalytic Efficiency Regulated by Substrate Binding," *Proteins* **57**, No. 747–757 (2004).
26. A. Gara, M. A. Blumrich, D. Chen, G. L.-T. Chiu, P. Coteus, M. E. Giampapa, R. A. Haring, et al., "Overview of the Blue Gene/L System Architecture," *IBM J. Res. & Dev.* **49**, No. 2/3, 195–212 (2005).
27. W. Pulleyblank, "How to Build a Supercomputer," *IEE Rev.* **50**, No. 1, 48–52 (2004).
28. V. Salapura, R. Walkup, and A. Gara, "Exploiting Workload Parallelism for Performance and Power Optimization in Blue Gene," *Micro IEEE* **26**, No. 5, 67–81 (2006).
29. J. Moreira, "Delivering Teraflops: An Account of How Blue Gene Was Brought to Life," presented at Modern Computing, IEEE John Vincent Atanasoff 2006 International Symposium, 2006.
30. A. A. Bright, M. R. Ellavsky, A. Gara, R. A. Haring, G. V. Kopcsay, R. F. Lembach, J. A. Marcella, M. Ohmacht, and V. Salapura, "Creating the Blue Gene/L Supercomputer from Low Power Systom-on-a-Chip ASICs," *IEEE International Solid-State Circuits Conference, Digest of Technical Papers,* February 2005, pp. 188–189.
31. F. Allen, G. Almási, W. Andreoni, D. Beece, B. J. Berne, A. Bright, J. Brunheroto, et al., "Blue Gene: A Vision for Protein Science using a Petaflop Supercomputer," *IBM Syst. J.* **40**, No. 2, 310–327 (2001).
32. J. E. Moreira, G. Almasi, C. Archer, R. Bellofatto, P. Bergner, J. R. Brunheroto, M. Brutman, et al., "Blue Gene/L Programming and Operating Environment," *IBM J. Res. & Dev.* **49**, No. 2/3, 367–376 (2005).
33. S. Chatterjee, L. R. Bachega, P. Bergner, K. A. Dockser, J. A. Gunnels, M. Gupta, F. G. Gustavson, et al., "Design and Exploitation of a High-Performance SIMD Floating-Point Unit for Blue Gene/L," *IBM J. Res. & Dev.* **49**, No. 2/3, 377–392 (2005).
34. Y.-P. Pang, K. Xu, J. El Yazal, and F. G. Prendergast, "Successful Molecular Dynamics Simulation of the Zinc-Bound Farnesyltransferase Using the Cationic Dummy Atom Approach," *Protein Sci.* **9**, No. 1857–1865 (2000).
35. B. Walkup, "Blue Gene/L System and Optimization Tips"; see *http://scv.bu.edu/documentation/presentations/BGL/BG_optimization.pdf*.
36. G. Almási, C. Archer, J. G. Castaños, M. Gupta, X. Martorell, J. E. Moreira, W. Gropp, S. Rus, and B. Toonen, "MPI on BlueGene/L: Designing an Efficient General Purpose Messaging Solution for a Large Cellular System," *Proceedings of the 10th European PVM/MPI Users' Group Meeting,* Venice, Italy, September 29–October 2, 2003; see *https://asc.llnl.gov/computing_resources/bluegenel/pdf/mpi.pdf*.
37. C. C. Lee and D. T. Lee, "A Simple On-line Bin-Packing Algorithm," *J. ACM* **32**, No. 3, 562–572 (1985).
38. A. C.-C. Yao, "New Algorithms for Bin-Packing," *J. ACM* **27**, No. 2, 207–227 (1980).

Received March 15, 2007; accepted for publication April 22, 2007; Internet publication December 19, 2007

Yuan-Ping Pang *Computer-Aided Molecular Design Laboratory, Mayo Clinic, 200 First Street SW, Rochester, Minnesota 55905 (pang@mayo.edu)*. Dr. Pang received his B.S. degree in physical chemistry at Amoy University in China, his M.S. degree training in neuroscience and biochemistry at the Shanghai Institute of Physiology in China (with Li-Jun Chen), and his Ph.D. degree in synthetic chemistry at the University of Pittsburgh (with Alan P. Kozikowski). He then embarked on a 1-year sabbatical study in computational chemistry at the University of California, San Francisco (with the late Peter A. Kollman). Since 1991, he has been working on the development and application of special-purpose computer hardware and software, as well as novel methods for drug discovery at the Mayo Clinic, and he is currently a professor of biophysics and pharmacology and the Director of the Computer-Aided Molecular Design Laboratory at the Mayo Clinic. His research is supported by the Defense Advanced Research Projects Agency (DARPA), the U.S. Army Medical Research Acquisition Activity (USAMRAA), the U.S. Army Research Office of the U.S. Department of Defense, the National Institutes of Health, the High Performance Computing Modernization Program, the State of Minnesota, and the Mayo Foundation for Medical Education and Research.

Timothy J. Mullins *IBM Systems and Technology Group, Development Laboratory, 3605 Highway 52 North, Rochester, Minnesota 55901 (mullins@us.ibm.com)*. Mr. Mullins is a member of the IBM Systems and Technology Group Development Laboratory in Rochester, Minnesota. He is involved with collaborations that develop high-performance computing life sciences applications for advanced technologies such as the IBM Blue Gene/L supercomputer and the Cell Broadband Engine** accelerator chip. He has a breadth of experience in computer systems design and development, especially in areas relating to performance modeling and analysis of products involving supercomputers, large server systems, network processors, and system-on-a-chip designs. He has held positions involving system design and architecture, CPU design and timing analysis, uniprocessor/multiprocessor/non-uniform memory access (NUMA) MP architectures and microarchitectures, system I/O buses, disk storage subsystems, and I/O controllers. Mr. Mullins joined IBM in 1977, and he holds an M.S.E.E. degree from the University of Minnesota as well as a B.S.E.E. degree from the University of California, Berkeley.

Brent A. Swartz *IBM Systems and Technology Group, Development Laboratory, 3605 Highway 52 North, Rochester, Minnesota 55901 (swartzbr@us.ibm.com)*. In May 1987, Mr. Swartz received a B.S. degree in computer science, physics, and mathematics from the University of Wisconsin–Eau Claire. He currently works on Blue Gene supercomputers at IBM in Rochester, Minnesota. His job responsibilities include performance testing for Blue Gene systems and applications porting and optimization. Since 1987, he has focused on the optimization of codes for high-performance computers. He previously worked on high-performance computing at NCube, Inc., the EPA National Environmental Supercomputer Center (NESC), the AFRL Maui High Performance Computing Center (MHPCC), and Cray, Inc.

Jeff S. McAllister *IBM Systems and Technology Group, Development Laboratory, 3605 Highway 52 North, Rochester, Minnesota 55901 (jsmcalli@us.ibm.com)*. Mr. McAllister received an M.S. degree in computer science from the University of Alaska, Fairbanks in 1998. His thesis concerned the reengineering of scientific codes. Since then, he has assisted many scientific and business projects worldwide. He has worked as a liaison between supercomputer hardware administrators and users at the Arctic

Region Supercomputing Center and as a coder, bringing mathematical ideas to peak performance implementations on clusters up to the largest machines available. He currently works for IBM in Rochester, Minnesota, facilitating the development of emerging high-performance applications on Blue Gene and Cell Broadband Engine systems.

Brian E. Smith *IBM Systems and Technology Group, Development Laboratory, 3605 Highway 52 North, Rochester, Minnesota 55901 (smithbr@us.ibm.com)*. Mr. Smith received an M.S. degree in computer engineering from Iowa State University in January 2005. He received B.S. degrees in both computer engineering and electrical engineering from Iowa State University in 2000. He currently works at IBM in Rochester, Minnesota, on the Blue Gene supercomputers. His job responsibilities include the communications stack for Blue Gene systems (both MPI and ARMCI/GA) and applications porting and optimizing. He previously worked on high-performance computing at the USDOE Ames Laboratory Research Laboratory.

Charles J. Archer *IBM Systems and Technology Group, 3605 Highway 52 North, Rochester, Minnesota 55901 (archerc@us.ibm.com)*. Mr. Archer is a software engineer working on the Blue Gene and Road Runner projects. He received a B.S. degree in chemistry and a B.A. degree in mathematics from Minnesota State University at Moorhead, and an M.S. degree in chemistry from Columbia University. He is currently a graduate student in computer science at the University of Minnesota. Mr. Archer has worked on the OS/400* PASE project and grid computing, and was the message-passing team lead for the Blue Gene family of supercomputers. His current role is development, optimization, and maintenance of the Road Runner and Blue Gene message-passing software stack.

Roy G. Musselman *IBM Systems and Technology Group, Development Laboratory, 3605 Highway 52 North, Rochester, Minnesota 55901 (mussel@us.ibm.com)*. Mr. Musselman works as the Blue Gene System Performance Team Leader at IBM in Rochester, Minnesota. He received a B.S. degree in electrical engineering from Iowa State University in 1980 and an M.S. degree from the University of Minnesota in 2005. He began his employment with IBM at Endicott, New York, in 1980, where he worked as a logic designer and product engineer for midrange computer systems. From 1985 to 2005, Mr. Musselman was integrally involved with the design and support of IBM internal use of hardware simulation accelerators and emulators. This included a 2-year temporary assignment with Supercomputer Systems, Inc., in Eau Clare, Wisconsin, before transferring to IBM in Rochester, Minnesota, in 1992. Mr. Musselman has authored several patents pertaining to hardware emulation and parallel communications.

Amanda E. Peters *IBM Systems and Technology Group, Development Laboratory, 3605 Highway 52 North, Rochester, Minnesota 55901 (apeters@us.ibm.com)*. Ms. Peters received a B.S. degree in both computer science and physics from Duke University in 2005. She currently works at IBM in Rochester, Minnesota, on the Blue Gene supercomputers. She is involved with the porting, validating, and optimizing of life science applications.

Brian P. Wallenfelt *14439 Fairway Drive, Eden Prairie, Minnesota 55344 (wallenfe@gmail.com)*. Mr. Wallenfelt received an M.S. degree in computer science from the University of Minnesota in March 2007. He received a B.S. degree in computer engineering from the University of Illinois at Urbana Champaign. He currently works for GeoDigm Corporation in Chanhassen, Minnesota. He previously worked on the Blue Gene system performance, applications, and kernel development teams at IBM in Rochester, Minnesota.

Kurt W. Pinnow *2415 Summit Drive NE, Rochester, Minnesota 55906 (kwp@area51online.net)*. Mr. Pinnow received his B.S. degree in applied math and physics from the University of Wisconsin in January 1969. Mr. Pinnow is now retired from IBM after having worked more than 38 years in the computer performance area. More than 35 of these years were spent working for IBM, where Mr. Pinnow's work has primarily focused on software performance. His first performance work with IBM concerned the parallel computer system software used in conjunction with the anti-ballistic missile system to track and plan attacks against incoming missiles in a real-time parallel computing environment. More recently Mr. Pinnow's contributions concerned the iSeries* components involved with the system file server, the SQL database engine and optimizer, and TCP/IP communications. For the last 3 years prior to his retirement in 2006, Mr. Pinnow worked on the Blue Gene/L system when he and his team successfully ensured the performance of Blue Gene. In his Blue Gene/L system work, the span of his efforts encompassed hardware, operating system software, and performance-oriented application programming.