JCTC Journal of Chemical Theory and Computation

Toward Large Scale Parallelization for Molecular Dynamics of Small Chemical Systems: A Combined Parallel Tempering and Domain Decomposition Approach

Henk A. Slim and Mark R. Wilson*

Department of Chemistry, University of Durham, South Road, Durham, DH1 3LE, U.K.

Received June 27, 2008

Abstract: A combined parallel tempering (replica exchange) and domain decomposition approach is presented, which allows for the effective use of large numbers of processor cores (>256) on modest sized simulations of chemical systems (~5000 sites). The approach is implemented in the **gbmoldd** molecular dynamics program for the simulation of coarse-grained molecular systems composed of combinations of isotropic and/or anisotropic particles. Benchmark results are presented for two test systems: a C_{24} united atom chain and a coarse-grained system of spherocylinders.

1. Introduction

In recent years, parallel tempering has become a powerful technique to improve sampling in molecular simulation. It has already shown to be useful in the simulation of a range of systems, including improved conformational sampling in peptides,[1,2] proteins,[3] and polymers,[4,5] studies of phase transitions in water clusters,[6] simulations of lattice models,[7,8] and in biased Monte Carlo (MC) for crystal structure determination.[9] A useful review of parallel tempering methods, with particular reference to many molecular simulation problems, has been produced recently by Earl and Deem.[10]

The key to the success of parallel tempering methods is a good overlap of the configurational energy between successive replica systems. Without this, many attempted relica swaps will be rejected. For many molecular applications, a useful guide is that approximately 20−25% of moves should be accepted. This figure can be arrived at by using an iterative scheme for the optimal allocation of temperatures[11] or by tuning temperature intervals to maximize the mean squared displacement of a system.[12] However, as system size grows, the ratio of the width of the configurational energy distribution over the number of particles, N, effectively becomes more sharply peaked. This is because the width of the energy distribution increases as $\sqrt{N}$, but the average energy increases as N^9. This means that as the system size increases, the number of replicas must grow as $\sqrt{N}$ to keep comparable sampling over similar temperature ranges.

While replica exchange favors a small system size, parallel molecular dynamics (MD) simulation favors large systems. In a typical MD simulation, as the number of sites increases, the CPU time required for computing the pair interactions increases more quickly than the communication time required between processors. However, while parallelization is often very efficient when $N > 50\,000$, for small systems of a few thousand particles, communication costs can easily win out. This is a particularly serious limitation in replicated data algorithms,[13,14] which typically require a global sum (all reduce) operation to sum and distribute forces over all processing cores on every time step. However, even for more efficient domain decomposition strategies,[13,14] where communication is limited to a minimum and global sum operations are avoided, parallelization for a few thousand particles quickly leads to the algorithm becoming communications bound. Moreover, in a normal domain decomposition for a few thousand particles, the number of parallel subdomains physically possible is severely restricted by the size of the simulation box. While parallelization is not necesssary for some small systems, increasingly there is a desire to carry out molecular simulation to observe events that normally occur rarely (barrier crossing, crystallization, structural transformations, and transitions). Coupled with the relative cheapness and high availability of arrays of com-

* Author for correspondence. Tel.: +44 191 334 2144. Fax: +44 191 386 1127. E-mail: mark.wilson@durham.ac.uk.

10.1021/ct800255r CCC: $40.75
Published on Web 09/19/2008

modity processors, it would be extremely useful to be able to apply large parallel compute arrays to study relatively small systems.

The current work presents a simple simulation approach, which combines parallel domain decomposition with parallel tempering methods into a single efficient code, **gbmoldd**, which is suitable for the study of small molecular systems composed of atomistic or coarse-grained potentials. The combination of r parallel tempering replicas and an efficient domain decomposition of a small system over N_p processors cores, means that rN_p commodity processor cores can be concentrated on the simulation of a small chemical system. Typical values of $N_p = 8$ ($2 \times 2 \times 2$ domain decomposition) and $r = 32$, which are often appropriate for a few thousand sites, already provide for the possibility of using 256 processors efficiently. For systems where domain decomposition can be extended to a $4 \times 4 \times 4$ box, it is already possible to exceed the number of processors which are available on all but the most specialist parallel machines.

2. Computational Method

The program **gbmoldd** has been designed as a simulation code to be used primarily for coarse-grained simulations containing mixtures of anisotropic and isotropic coarse-grained sites within molecular systems, e.g. in liquid crystalline and/or macromolecular systems,[15−17] but it can also be used for modeling of atomistic systems based on Lennard-Jones sites. Message passing in **gbmoldd** is implemented with the message passing interface (MPI) communications protocol. The basic processor arrangement for the algorithm is shown in Figure 1. At the start of the molecular dynamics run, N_t identical copies of the program are started on N_t identical CPU cores using a MIMD parallel approach.[18] Here, for a system of r replicas, $N_t = r \times N_p = r \times N_{px} \times N_{py} \times N_{pz}$, and an initial domain decomposition over N_p processor cores is made for x, y, and z spatial directions. In practice, the processor cores are grouped into groups of size N_p (ideally a group of processors should be physical neighbors with the fastest available interconnects available between group members) and the processor map is distributed to each processor in turn. All MPI communication within the processor group of each replica takes place in the context of MPI intracommunicators; the exchange of information between replicas is facilitated with MPI intercommunicators.

An outline of the algorithm used in this work is shown in Figure 2. The main part of the algorithm is dominated by a domain decomposition approach carried out for each replica in parallel. This makes use of a division of each replica domain into subdomains, and each subdomain into cells which are equal to (or slightly larger than) the short-range cutoff. Particles are initially sorted into the subdomains, and a single CPU core is responsible for the coordinates, velocities, orientations, and orientational derivatives of the particles within its subdomain. This has the usual advantage that adjacent processors only require information about particles in their own subdomain and in cells on the boundary with adjacent nodes. In domain decomposition, there are two main choices of how to carry out the core message passing.[14,19] An initial communication of the subdomain

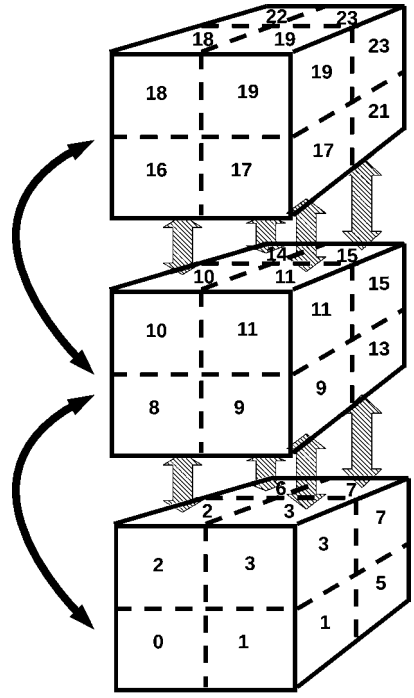

Figure 1. Schematic diagram showing the domain decomposition approach used in this work. MC parallel tempering moves take place between separate parallel systems as indicated by the solid arrows on the left of the diagram. Each separate replica system uses a domain decomposition approach. The shaded arrows between domains correspond to data transfers, which in this case involve a transfer of ensemble averages (rather than coordinates) from each domain.

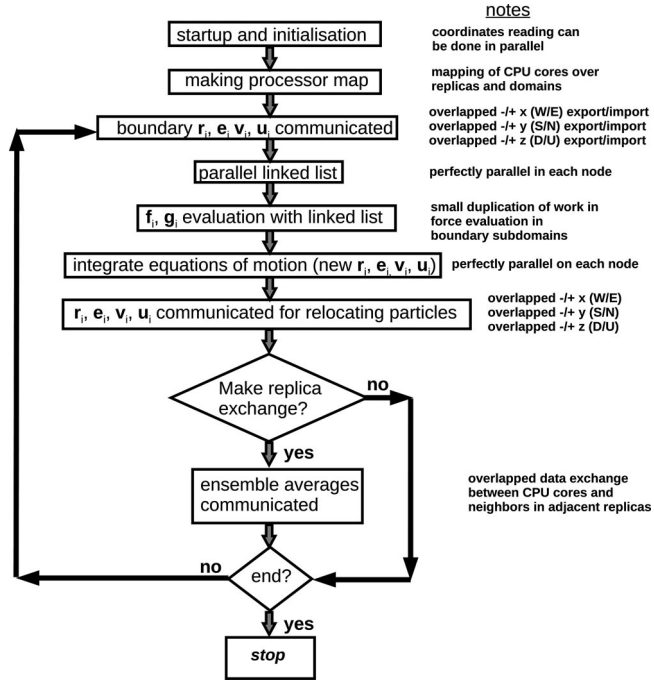

Figure 2. Flowchart diagram showing the structure of the combined parallel tempering−domain decomposition program **gbmoldd**.

coordinates can be carried out, with message passing for the positive and negative directions executed at the same time (which is particularly useful in cases where processors have

more than one network connection). Then if force calculations are carried out separately on each node, there is no need to carry out message passing for the forces (as shown in Figure 2). This leads to some duplication of the force calculation for particles in the boundary subdomains but helps reduce communication costs.[14,19] The alternative involves passing coordinate information for the boundary subdomains, in one direction only, prior to the force calculation, and then passing the force information back in the opposite direction. This approach avoids the duplication of force calculations for boundary particles but involves slightly more message passing. For **gbmoldd**, we adopt the former approach (Figure 2) to reduce communication, because anisotropic particles already involve additional communication costs associated with the transfer of particle orientations as well as positions.

Although different cutoffs can be used for different types of interaction sites, in **gbmoldd**, the subdomains are always set up to correspond to the largest cutoff in the system (usually between two anisotropic particles) to avoid multiple layers of message passing for different types of sites. For molecular systems involving multiple bonded sites, for the force decomposition to be successful, all 1−2, 1−3, and 1−4 based potentials must have a maximum interaction distance, which is less than or equal to the width of a subdomain. This is always the case for atomistic systems but is usually also the case for most coarse-grained systems. Within the simple (nonreallocation) force scheme described above, when atoms taking place in bonded potentials are split across subdomains, the potential is evaluated in each subdomain separately to obtain the correct forces but the energy is only counted once. This tends to be a very minor cost in terms of total CPU time.

The rest of the molecular dynamics algorithm follows a conventional domain decomposition strategy. For anisotropic particles, we use an anisotropic form of the Velocity Verlet algorithm[20,21] (an alternative anisotropic leapfrog algorithm is also available). For anisotropic particles, the simulation keeps track of the position vectors, $\mathbf{r}(t)$, orientations, $\mathbf{e}(t)$, velocities, $\mathbf{v}(t)$, and orientational derivatives, $\mathbf{u}(t)$, and the total kinetic energy is given by

$$K = \sum_{i=1}^{N} \frac{m_i v_i^2}{2} + \sum_{i=1}^{N} \frac{I_i u_i^2}{2}$$

for masses, m_i, and moments of inertia, I_i. The integrator used forces, $\mathbf{f}(t)$, and gorques, $\mathbf{g}(t)$, and integration uses the following steps, which make use of a variable, ξ (with $\dot{\eta} = \xi$), to control the thermostat and are executed in parallel making use of a central difference Stoermer−Verlet integrator[22] (NB: it is also possible to carry out the integration in other ensembles but the best results for replica exchange are obtained for constant *NVT*):

1. Evaluate $\mathbf{f}(t)$, $\mathbf{g}(t)$, and virial.
2. Advance $\mathbf{v}(t)$ and $\mathbf{u}(t)$ by a half-step time-step.
3. Evaluate kinetic energy, K.
4. Advance $\mathbf{r}(t)$, $\mathbf{v}(t)$, $\xi(t)$, and $\eta(t)$ by a time-step (using half-step quantities from above).
5. Reallocation (see below).
6. Advance $\mathbf{v}(t)$ and $\mathbf{u}(t)$ by a second half-step time-step.

At the end of the positions/orientations integration step, particles which leave a subdomain are reallocated to neighboring processors. As usual in domain decomposition, both the initial coordinate pass (prior to the force calculation) and the particle reallocation have to take place by passing in the x, y, and then z directions in turn, with a coordinate sort in between to ensure that nodes connected via edges or vertices (as well as faces) are able to interchange particles.[19]

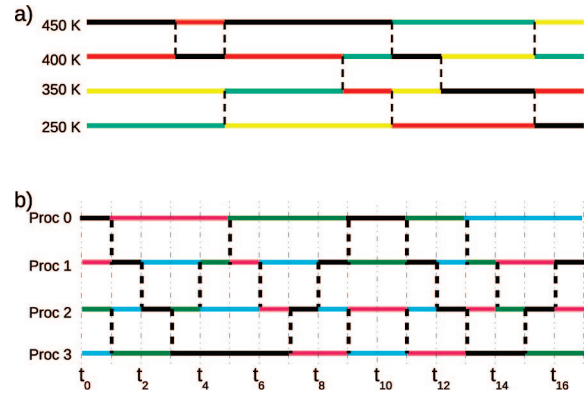

Figure 3. Schematic diagram showing replica exchange approaches. (a) Traditional approach. Coordinates are exchanged between replicas with each processor handling a single temperature with the color coding indicating the "time evolution" of individual coordinate sets. (b) Approach used in this work. Temperatures are exchanged between processors with the color coding indicating the path of temperatures through the processors; ensemble averages must be calculated over individual temperatures by data exchange.

At the end of N_{steps} molecular dynamics steps, the system attempts to undertake a "parallel tempering swap". Here, a swap between two replicas i and j is accepted with the probability

$$\min\{1, \exp[+(\beta_i - \beta_j)(U_i - U_j)]\}$$

where U_i and U_j are the potential energies of each separate replica. In practice, this also involves a scaling of the velocities to ensure that the average kinetic energy obeys equipartition.[23]

As with the communications in domain decompostion, there are two alternative approaches for communications in replica exchange (Figure 3). Coordinates can be swapped between replicas or the ensemble conditions of the replica can be swapped. The former involves a coordinate swap for each accepted move, the latter involves a swap of ensemble information, i.e. in the case of temperature parallel tempering, a swap of the temperatures between replicas takes place. To avoid swapping coordinates, we have implemented the latter. The advantage of doing this is that the communication costs are greatly reduced by avoiding coordinate swaps. However, additional care must be taken in computing thermodynamic averages and other simulation output data. Unlike a normal simulation (and unlike the situation where coordinate swapping takes place), these can not be computed directly by each processor separately and written at the end of the simulation because the average must be made over the individual ensembles. Each replica therefore has to swap any thermodynamic (or other) averages being calculated during the simulation run, at the same time as the temperature is

swapped. To do this efficiently, each subdomain within a replica needs to swap information with its corresponding subdomain in the other replica. This pairwise swap can occur simultaneously across all swapping domains (Figure 1) using a single send and receive operation on each processor involved in a swap. In cases where simulation data (e.g., coordinates and velocities for postprocessing) are written to files, care must also be taken to ensure that the correct coordinate data belonging to a single temperature is used. In practice, this can be carried out most efficiently in a postprocessing step. Each separate CPU simply writes its own coordinate/velocity data independently to a separate file, with each set tagged by simulation step and temperature flag. Postprocessing information, such as the velocity autocorrelation for each temperature, is then calculated by a single program, which sorts the multiple data files combining data from different domains belonging to the same temperature.

It should be noted that it is also possible to swap the potential used for molecular dynamics within this approach (Hamiltonian replica exchange).[24] Doing so allows for potential softening approaches, whereby one can soften selected parts of the Hamiltonian. Here, each parallel copy of the program running on each processor core contains an alternative set of potentials. Message passing simply involves the transfer of an integer flag to swap which potential is used within each replica. As with temperature parallel tempering, the running averages are swapped at the same time as the replicas, and the data within files saved for postprocessing are identified by the time-step and an unique integer tag for each of the potentials.

3. Simulations

Calculations for this work are based on two models: a united atom *n*-tetracosane (C_{24}) system simulated in the constant *NVT* ensemble, at a density of 776.2 kg m^{-3} using using the Trappe force field[25] and a 1 fs time-step and a (L/D = 5) soft repulsive spherocylinder model (corresponding to a cut-and-shifted 12:6 potential for the minimum distance of two line-segments[26,27]). The latter was simulated in reduced units in the *NVT* ensemble, with $D = \sigma = \varepsilon = 1$, using a reduced density of $\rho^* = N\sigma^3/V$, a reduced temperature of $T^* = kT/\varepsilon = 1$, and the methodology described by Earl et al.[26] Parallel tempering Monte Carlo moves were attempted every 200 time-steps. For the purposes of benchmarks, coordinate data was saved every 10 000 time-steps (corresponding to 10 ps for the united atom model), which is typical of that required for many molecular or coarse-grained simulations. The algorithm was tested on two parallel cluster systems. The CLX system at the CINECA supercomputer center in Italy is based around 512 2-way IBM X335 nodes (Xeon Pentium IV at 3.06 GHz) with Myrinet interconnects, while the newer BCX system used the next generation of cluster technology with Opteron Dual Core 2.6 GHz processors and Infiniband (5 Gb s^{-1}) interconnects for each two (dual-core) processor node.

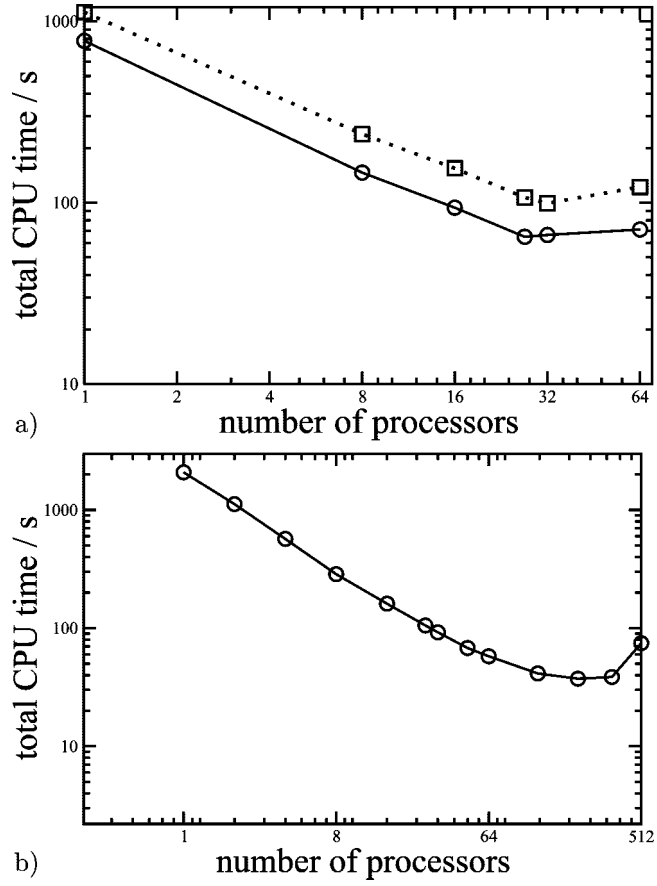

Figure 4. (a) Timings for a system of 216 *n*-tetracosane molecules (5184 united atom sites) on the CINECA CLX (dotted line) and BCX (bold line) clusters as a function of the number of CPU cores. (b) Timings for 64 000 spherocylinder sites on the BCX cluster.

4. Results and Discussion

Figure 4 shows the typical parallel scaling behavior of a domain decomposition program. We show two systems, a small system of 216 *n*-tetracosane molecules (5184 sites) and a larger system of 64 000 soft repulsive spherocylinders. The scaling shown is extremely system size dependent as expected. The maximum speed for the small united atom system is achieved for a 3 × 3 × 3 domain decomposition on the BCX system, and thereafter, addition of parallel cores leads to a degredation in performance. It is interesting to note that changing to the next generation of parallel cluster (BCX instead of CLX) does not help the scaling signifcantly (in fact parallel scaling is slightly worse). This is because for the Lennard-Jones united atom system used, the CPU time required per step is rather small. So although parallel communications are improved in going from Myrinet to Infiniband, this is more than compensated by the increases in processor speed for the newer system. As with many molecular dynamics models of this size, eight cores provides a good balance between speedup, CPU cost, and performance and provide an absolute speedup on the BCX system by a factor of 5.3× compared to a single processor. Clearly with more expensive potentials, or if long-range electrostatic forces, etc. were included, increasing the ratio of computation/communication leads to even better scaling.

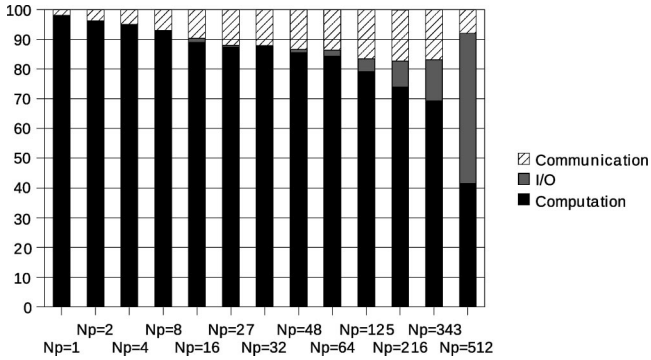

Figure 5. (a) Breakdown of total processing time in terms of computation, I/O, and communication for 64 000 spherocylinder sites on the BCX cluster.

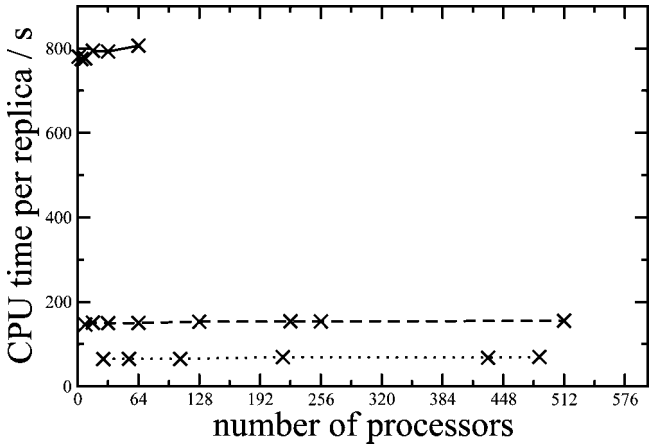

Figure 6. CPU time per replica showing almost perfect scaling for a 1 × 1 × 1 (bold line), 2 × 2 × 2 (dashed line), and 3 × 3 × 3 domain decomposition (dotted line) for 216 *n*-tetracosane molecules.

As expected, an increased computation/comunication ratio is reflected in the better parallel scaling seen for the 64 000 site spherocylinder system (bottom of Figure 4), which shows increases in performance up to 216 processors (a 6 × 6 × 6 domain decomposition). For this system, the relative costs of computation, commmunications, and input/output (I/O) are shown in Figure 5. As the processor number increases, the percentage of time spent in computation drops significantly. Interestingly, at very large numbers of processors, the I/O starts to become a significant cost. Although I/O is carried out in parallel for **gbmoldd**, the problems of many processor cores writing to the same filesystem in parallel is clearly seen. At 125 cores, I/O is becoming a major contribution to the total time required for a run. Alongside communication and I/O costs, the effects of some force duplication for boundary atoms, imperfect load balancing due to density fluctuations, and additional sorting all limit parallel scaling (a full discussion of these features is given in the recent work of Hess et al.).[28]

In contrast to domain decomposition, the parallel tempering communication costs are extremely small. In Figure 6, we show the CPU time per replica for 1 × 1 × 1, 2 × 2 × 2, and 3 × 3 × 3 domain decompositions of the (5184 atom site) *n*-tetracosane system. The time per replica is virtually unchanged as the number of replica systems grow in each

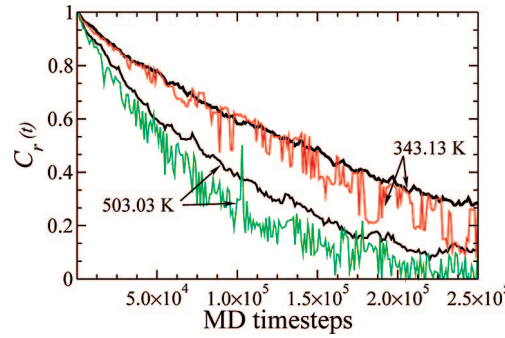

Figure 7. Time correlation function, $C_r(t)$, for the normalized molecular end-to-end vector calculated for C_{24} united atom chains in the liquid phase. Results are shown for two temperatures from conventional molecular dynamics simulations (bold black lines) superimposed on the results from parallel tempering (red and green lines). The latter are taken from a simulation with 32 replicas with each replica started from the same initial coordinates.

case. In the algorithm presented above, this is helped by avoiding coordinate swaps. The swapping of ensemble information (temperatures and averages) can be achieved by overlapping communications between a set of processor cores within a replica and the corresponding cores in the neighboring replica. Even for processor numbers of over 256, there is only a very small increase in CPU time, caused principally by parallel I/O degredation.

The real benefits of parallel tempering combined with domain decomposition become apparent in equilibrating molecular systems with slow relaxation times at a range of temperatures. In Figure 7, we compare the slow decay of the time correlation function of the molecular end-to-end vector in the C_{24} system, defined by

$$C_r(t) = \left\langle \frac{(\mathbf{r}_1(0) - \mathbf{r}_{24}(0))\cdot(\mathbf{r}_1(t) - \mathbf{r}_{24}(t))}{|\mathbf{r}_1(0) - \mathbf{r}_{24}(0)||\mathbf{r}_1(t) - \mathbf{r}_{24}(t)|} \right\rangle$$

The decay of this function is influenced by both internal conformational changes and (to a lesser extent) by molecular rotation. However, for chain molecules, the decay of $C_r(t)$ is rather slow as illustrated in the figure. In fact, for longer (polymer) chains, the decay of $C_r(t)$ becomes particularly slow. This occurs for chain lengths greater than the entanglement length of the polymer. Here, reptation becomes the dominant mechanism for a polymer chain to relax. This provides a major limitation in the use of MD simulation to equilibrate polymer melts. The plots in Figure 7 result from starting each simulation (including each replica) from the same configuration. Here, for the parallel tempering simulations, 32 replicas are used to span the temperature range of 343.13−735.66 K. For both temperatures plotted in Figure 7 (and also all temperatures in the PT simulation), parallel tempering is seen to improve the decay of $C_r(t)$ by facilitating the movement of the system through phase space.

The results of this paper suggest that parallel tempering (replica exchange MD) and domain decomposition combine well together within a single molecular dynamics code. As a consequence, it becomes cost-effective to use a relatively small number of processors in an efficient domain decomposition and to then add a further level of parallelization

through replica exchange to improve sampling of phase space. In this way, large parallel clusters may be applied to a single small system problem. While domain decompostion is limited for a small number of particles (both in efficiency and in the physical limits imposed by the number of domains used), replica exchange is most efficient with relatively small systems, because the distribution of energy states in a smaller system allows for larger pertubations of the temperature or the potential between replicas.

Finally, it is worth noting that the approach adopted in this work is particularly useful for many of today's commodity clusters. Typical commodity systems can have two quad-core processors sharing RAM on a single board. Here, it becomes very cost-effective to use the fast shared memory communications for the communication intensive domain decomposition within a single replica and use slower off-board communications (gigabit ethernet, Myrinet, or Infiniband) for the less communications bound parallel tempering.

5. Conclusions

A combined domain decomposition−parallel tempering algorithm is described and implemented for the coarse-grained molecular simulation program **gbmoldd**. Results are presented for benchmarks on two systems: a C_{24} united atom alkane and a coarse-grained spherocylinder system. It is shown that this combined approach allows for very large number of processors (>256) to be applied efficiently to a relatively small system (∼5000 sites) with minimal communication costs.

Acknowledgment. The authors thank the UK research council, EPSRC, for providing funding through grant GR/S43054/01 and the CINECA supercomputer centre in Italy for providing computer time on its CLX and BCX systems. M.R.W. thanks the HPC Europa++ programme for providing funding for a visit to CINECA and the research group of Prof. C. Zannoni (University of Bologna).

References

(1) Hansmann, U. H. E. *Chem. Phys. Lett.* **1997**, *281* (1−3), 140.

(2) Wu, M. G.; Deem, M. W. *Mol. Phys.* **1999**, *97* (4), 559.

(3) Lin, C. Y.; Hu, C. K.; Hansmann, U. H. E. *Proteins* **2003**, *52* (3), 436.

(4) Bedrov, D.; Smith, G. D. *J. Chem. Phys.* **2001**, *115* (3), 1121.

(5) Yan, Q. L.; de Pablo, J. J. *J. Chem. Phys.* **2000**, *113* (3), 1276.

(6) Nigra, P.; Carignano, M. A.; Kais, S. *J. Chem. Phys.* **2001**, *115* (6), 2621.

(7) Swensen, R. H.; Wang, J. S. *Phys. Rev. Lett.* **1986**, *57* (21), 2607.

(8) Rathore, N.; de Pablo, J. J. *J. Chem. Phys.* **2002**, *116* (16), 7225.

(9) Falcioni, M.; Deem, M. W. *J. Chem. Phys.* **1999**, *110* (3), 1754.

(10) Earl, D. J.; Deem, M. W. *Phys. Chem. Chem. Phys.* **2005**, *7* (23), 3910.

(11) Rathore, N.; Chopra, M.; de Pablo, J. *J. Chem. Phys.* **2005**, *122* (2), 024111.

(12) Kone, A.; Kofke, D. *J. Chem. Phys.* **2005**, *122* (20), 206101.

(13) Smith, W. *Comput. Phys. Commun.* **1991**, *62*, 229.

(14) Wilson, M. R.; Allen, M. P.; Warren, M. A.; Sauron, A.; Smith, W. *J. Comput. Chem.* **1997**, *18* (4), 478.

(15) Hughes, Z. E.; Wilson, M. R.; Stimson, L. M. *Soft Matter* **2005**, *1* (6), 436.

(16) Stimson, L. M.; Wilson, M. R. *J. Chem. Phys.* **2005**, *123*, 034908.

(17) Wilson, M. R.; Ilnytskyi, J. M.; Stimson, L. M. *J. Chem. Phys.* **2003**, *119* (6), 3509.

(18) Hwang, K.; Briggs, F. A. In *Computer Architecture and Parallel Processing*; McGraw-Hill Book Company: New York, 1985; pp 32−35.

(19) Wilson, M. In *Advances in the Computer Simulations of Liquid Crystals*; Pasini, P., Zannoni, C., Eds.; Kluwer Academic Publishers: Norwell, MA, 2000; chapter 13, pp 389−412.

(20) Ilnytskyi, J. M.; Wilson, M. R. *Comput. Phys. Commun.* **2001**, *134*, 23.

(21) Ilnytskyi, J. M.; Wilson, M. R. *Comput. Phys. Commun.* **2002**, *148*, 43.

(22) Cuetos, A.; Ilnytskyi, J. M.; Wilson, M. R. *Mol. Phys.* **2002**, *100* (24), 3839.

(23) Sugita, Y.; Okamoto, Y. *Chem. Phys. Lett.* **1999**, *314* (1−2), 141.

(24) Fukunishi, H.; Watanabe, O.; Takada, S. *J. Chem. Phys.* **2002**, *116* (20), 9058.

(25) Martin, M. G.; Siepmann, J. I. *J. Phys. Chem. B* **1998**, *102* (14), 2569.

(26) Earl, D. J.; Ilnytskyi, J.; Wilson, M. R. *Mol. Phys.* **2001**, *99*, 1719.

(27) Cuetos, A.; Martinez-Haya, B.; Rull, L. F.; Lago, S. *J. Chem. Phys.* **2002**, *117* (6), 2934.

(28) Hess, B.; Kutzner, C.; van der Spoel, D.; Lindahl, E. *J. Chem. Theory Comput.* **2008**, *4* (3), 435.

CT800255R