Large Scale Molecular Dynamics Simulations using the Domain Decomposition Approach

David Brown* **and Bernard Maigret**

Laboratoire de Chimie Théorique - UMR CNRS 7565, Université de Nancy I,
BP 239, 54506 Vandoeuvre-lès-Nancy Cedex, France

This paper reviews the computational aspects of a method for applying domain decomposition to a general purpose molecular dynamics program. The algorithm is suitable for either distributed memory parallel computers or shared memory machines with message passing libraries. The method allows molecules of arbitrary connectivity to be simulated within the domain decomposition approach. The algorithm contains techniques to handle both rigid bond constraints and special CH_2 constraints, as well as two, three- and four-body intramolecular potentials and van der Waals and Coulombic interactions. Benchmark results of such a program (*ddgmq*) are illustrated.

1. Introduction

At the molecular level the detailed structure and dynamics of many systems are very difficult to discern by spectroscopic methods (x-ray, NMR, etc.). As a complement to such conventional approaches, interest has turned in recent years to atomistic computer simulation, usually molecular dynamics (MD), methods [1]. In many cases these simulations have provided detailed insight into phenomena. However, the time and length scales accessible to simulation are restricted by the computing power available and for atomistic simulations this is currently in the nanoscale regime. Thus, many technologically and biochemically interesting systems which extend over distances larger than 10 nm and/or relax on time scales longer than nanoseconds are difficult to simulate adequately using standard MD codes running on single processor computers. It is essential to develop new parallel algorithms capable of handling complex biochemical and technological systems. Solving this problem opens up the prospect of simulating systems of a much larger scale than was ever previously possible and thus the opportunity of advancing our understanding of many processes at a fundamental level.

Early attempts at parallelizing MD simulations were largely limited to atomic systems and to highly specific problems, *e.g.* [12, 19, 28, 29, 41]. During this time, a number of strategies were proposed for mapping MD onto multiple-instruction multiple-data (MIMD) machines based on either particle decomposition, *i.e.* assigning particles to home processors using their indices, or domain decomposition, *i.e.* assigning particles to processors according to their position in space [19, 33, 34, 40, 41, 46, 47]. Although there have been more recent developments involving force decomposition [36, 37] and attempts to combine strategies [11], the problem has remained that the best parallel solution for a particular simulation depends on factors like the average number of particles per processor and the speed of the individual processors. However, for large scale simulations where there are many more particles than processors, there is little doubt that domain decomposition (DD) is the optimum approach.

* SnailMail: LMPC, Bât IUT - Savoie Technolac, 73376 Le Bourget-du-Lac Cedex, France
Email: David.Brown@univ-savoie.fr

2. Molecular dynamics

As an introduction to some of the difficulties involved in applying domain decomposition to a molecular system of arbitrary connectivity some of the common practices carried out in molecular dynamics computer simulation [1, 20, 21] are first reviewed. To keep matters relatively simple only classical systems will be considered; the parallelization of methods which introduce quantum mechanics into the dynamics is not considered here.

In a classical MD simulation a molecule is modelled as a number of point masses, atoms, connected by a network of interconnecting bonds. Molecules interact with one another via the sum of all possible pair interactions between their constituent atoms. The approximation that atoms on different molecules experience a force between them *independent* of all the other atoms present is a relatively crude one but is almost universally made in classical MD; incorporating 3-body, 4-body etc. effects leads quickly to prohibitive computing costs. It is generally argued that the pair potential is an *effective* potential, rather than a true representation of the actual pair interaction between isolated atoms, and as such takes into account the higher order terms through the parametrization.

The main purpose of these so-called non-bonded effective pair potentials is to prevent two atoms from occupying the same region of space (short range repulsive interaction). Attractive terms in the potential hold liquid/solid systems together and influence the preferred packing structure of the molecules to a greater or lesser extent depending on their exact nature. An example of a pair potential widely used in MD simulations is the Lennard-Jones 12-6 potential

$$\Phi_{LJ}(|\mathbf{r}_{ij}|) = 4\varepsilon((\sigma/|\mathbf{r}_{ij}|)^{12}-(\sigma/|\mathbf{r}_{ij}|)^{6}) \qquad (1)$$

where the pair of interacting atoms, {i,j}, have a separation vector denoted by $\mathbf{r}_{ij}$ ($=\mathbf{r}_i-\mathbf{r}_j$), ε is the well-depth of the potential and σ is the distance at which the potential is zero. This is just one of many such potentials which combine a repulsive term, dominant at short range, and a quickly diminishing attractive tail. In practice a truncation, r_c, is normally applied beyond which the force between particles is defined to vanish. Although this is an approximation the restriction of the range within which *explicit* interacting pairs have to be found is necessary in order to render the problem tractable.

With regard to the parallelization, the exact form of the short range non-bonded pair potentials is of no importance; the real problem lies in calculating uniquely the distance between all possible interacting pairs. However, one of the continuing problems with MD simulations concerns the treatment of the charge-charge interactions. Formally these are described by the Coulombic potential

$$\Phi_c(|\mathbf{r}_{ij}|) = q_i q_j / (4\pi\varepsilon_o|\mathbf{r}_{ij}|) \qquad (2)$$

where q_i and q_j are the partial charges on the two atoms and ε_o is the permittivity of the vacuum. The 1/r nature of this potential means the energy diminishes very slowly with range and in the program to be described here, the Ewald lattice summation method [18, 48] has been used. This method effectively splits the Coulombic energy into two sums, one performed in real space, involving explicit calculation of pair distances, and a second in Fourier space, which involves a sum over reciprocal lattice vectors. In the 3D periodic boundary conditions, used in MD to reduce otherwise dominant boundary effect, the Ewald summation nominally fits well into a DD strategy. First, the real space part of the sum involves a potential which quickly diminishes with range and can, thus, be treated like all other short range potentials. Second, the reciprocal space summation, formally a sum over pairs of atoms, can be factorized into a single sum involving atomic positions only, within a sum over reciprocal lattice vectors, **k**. The parallelization of this second part is then trivial in DD with each processor performing the sum over k-vectors for each atom in its domain. The Ewald method does not scale linearly, however, and more about this will be said later.

Atoms within the same molecule may also interact through pair potentials in the same way as described above. This depends if the interactions are deemed to be 'non-bonded' or 'bonded'. Normally a certain number of 'bonded' interactions maintain the local geometry within molecules.

Atom pairs forming a chemical bond can be kept close to a certain distance apart by a continuous potential. Similarly other potentials can be used to force preferred structure upon, for example, bond angles, torsion angles and the out of plane motion of trivalent atoms. The exact form of these potentials is again not particularly important. However, for the parallelization by decomposition in domains the most important point is that a certain number of atom positions are required in order to calculate the potential and the forces. For example, a bond angle, θ, is defined by the positions of three contiguous atoms {i, j, k} (i bonded to j and j bonded to k and i≠k),

$$\cos\theta = (\mathbf{r}_{ij} \bullet \mathbf{r}_{kj})/|\mathbf{r}_{ij}||\mathbf{r}_{kj}| \qquad (3)$$

Similarly, four contiguous atoms {i, j, k, l} are required to define the torsion (or dihedral) angle, τ, between the planes of the two groups {i, j, k} and {j, k, l}

$$\cos\tau = -\frac{(\mathbf{r}_{ij} \times \mathbf{r}_{jk}) \bullet (\mathbf{r}_{jk} \times \mathbf{r}_{kl})}{|\mathbf{r}_{ij} \times \mathbf{r}_{jk}|\,|\mathbf{r}_{jk} \times \mathbf{r}_{kl}|} \qquad (4)$$

It is inevitable that a decomposition in domains will result in atoms which define bonds, bond angles, torsions etc. being placed on different processors. The parallelization of the calculation of these 'bonded' interactions requires, thus, the unique identification of the group of atoms, hereinafter referred to as n-plets, in question by just one processor. An added minor complication is that atoms involved in these intramolecular n-plets are often considered as not being subject to the non-bonded potentials; their effect having being incorporated in the intramolecular potential. The parallel algorithm has then to be able to distinguish between these 'bonded' and 'non-bonded' pairs.

Given a set of initial coordinates and velocities for the atoms, the boundary conditions and a description of the interactions between particles, MD progresses by calculating the total force on each atom and then integrating the equations of motion forward by a short time interval, Δt, using finite difference methods. Done repeatedly, this process builds up a trajectory for each atom in the system from which various thermodynamic and dynamic properties can be calculated. The equations of motion can be of the Newtonian form or be more complicated in a number of ways, e.g. in order to force the system to have a certain average temperature and pressure [1]. The integration step in the above cases does not pose any problems for domain decomposition as each particle's position is propagated forward independently once the forces are known. The actual algorithm used to do the time integration is not directly important with respect to the parallelization.

The use of continuous potentials to impose a certain geometry on molecules does pose some indirect problems. Realistic force constants for bond stretching modes are very high and lead to high frequency motions. These restrict the maximum time step that can be used to integrate the equations of motion. Although multiple time steps algorithms [49, 51, 52] are one solution to this problem they don't in themselves solve a more serious problem which concerns the poor coupling between the high-frequency low-amplitude modes and the rest of the modes in the system. In completely classical MD simulation this presents a real problem as it can lead to non-equipartition of energy and hence results which have nothing to do with the sampling conditions required. A standard solution to these problems is to use rigid constraints.

The simplest form of constraint is that of a rigid bond between atoms. However, other more complicated forms [22, 43] can involve more than two atoms. A particular example is the special constraints required to freeze all the degrees of freedom of hydrogen molecules in a CH_2 group [22]. For a X-CH_2-Y group, holding the mid-point of the H-H vector on the bisector of the X-C-Y bending angle and maintaining the H-H vector perpendicular to the vector joining the X and Y atoms removes all the degrees of freedom of the hydrogens. Such a constraint involves the positions of five atoms. Although there are different methods available to incorporate a solution of the constraints problem into the equations of motion, we will restrict the discussion in this paper to probably the most common one based on the iterative procedure 'SHAKE' [44]. SHAKE solves the constraints problem by first allowing particle positions to be propagated forward in the absence of the forces of constraint to provisional positions at the next time step. Final positions are then obtained at the next step by applying all the constraints in turn and iterating this process until all constraints are satisfied

to within some specified tolerance. In scalar codes the most recent positions are always used in order to speed up convergence; an atom may be involved in several constraints. This recursive operation impedes vectorization so either coordinates at the previous iteration are used, and increased vectorization traded against an increased number of iterations, or methods to avoid recursion are utilised [32].

Incorporating constraints into the dynamics ameliorates the problem of short time steps and largely solves the poor coupling problem and, despite the extra complexity and CPU costs of solving the constraints problem, leads to net savings in scalar or vector codes. With respect to domain decomposition, SHAKE introduces an extra problem in that, at the integration step, the propagation of a particle's position forward in time is no longer a process independent of all other particles in the system. Inevitably atoms involved in constraints will straddle domain boundaries, thus implying communication at each iteration of the SHAKE procedure.

3. Parallel Molecular Dynamics

3.1 Background

In the DD strategy, each processor element (PE) is responsible for particles residing in a particular sub-volume of the MD box and each processor accumulates the total forces on the particles in its domain and integrates their equations of motion. Although DD is relatively straightforward to implement for atomic systems in which only pair interactions exist (and there are now numerous examples [2, 8, 23, 27, 29, 34, 35, 42, 50]), the connectivity of molecular systems introduces a number of new difficulties [7, 17]. These can be summarized as follows:-

(1) Multiparticle potentials and specialized constraints require that the group of n atoms involved in the calculation be simultaneously known by one PE which is assigned to carry out the necessary computations.

(2) Particles subject to rigid constraints may lie in different domains. The procedure SHAKE, generally used to impose rigid constraints, implies a communication between processors at each of several iterations.

(3) Not all pairs of atoms interact via non-bonded pair potentials.

As a result of the above, most attempts to parallelize general purpose MD codes have relied on simpler to implement particle decomposition techniques [4, 15, 25, 26, 30, 31, 45] which are unsuitable for large-scale simulations as each processor requires a knowledge of all atomic positions. Some improvement is possible using force decomposition techniques [37] but, as yet, there are only a few examples in the literature of DD MD programs being used for molecular systems [7, 13, 14, 17, 38].

In the following section, some general remarks are made about our implementation of domain decomposition. We then discuss the extension to molecular systems of arbitrary connectivity. Further details can be found in a recent article [10] and in the on-line manual for the *gmq* and *ddgmq* programs [6].

3.2 General considerations

In the DD scheme used, the atoms are initially assigned to the $N_{p_x} \times N_{p_y} \times N_{p_z} = N_p$ processors on the basis of the x, y and z coordinates, *i.e.* 3-D decomposition of space. The exact shape of the MD box does not have to be orthorhombic and assignments are actually carried out using the set of scaled coordinates $\mathbf{s}_i = \mathbf{h}^{-1}\mathbf{r}_i$ where the matrix $\mathbf{h}=\{\mathbf{a}, \mathbf{b}, \mathbf{c}\}$ is defined from the basis vectors of the MD box. The scaled space is cubic and facilitates easy assignment of particles to processors. The scaled space also facilitates the assignment of atoms to the $N_{c_x} \times N_{c_y} \times N_{c_z} = N_c$ link-cells within each domain. The link-cell strategy [1] is useful both as a way to speed up the creation of a neighbour table [1] and as a means of determining which atoms lie close to domain boundaries and, thus, have to be communicated to neighbouring domains.

At this point, each PE is only aware of particles in its own domain. As atoms are assigned to processors, it is inevitable that some molecules will be distributed among two or more of them. Decompositions based on molecular centres-of-mass can avoid this problem [16] but they are not an option in the general case where molecules span many domains.

Once the assignments to domains and link-cells have been made, the first step is a search for all interacting pairs of atoms. This requires the communication of those coordinates required by other domains to complete their 'working' environment. In order to minimize communications at this stage, the conventional link-cell search pattern is modified [8]. A schematic representation of the form of the modification in 2-D is shown in Fig. 1; the extension to 3-D follows straightforwardly.

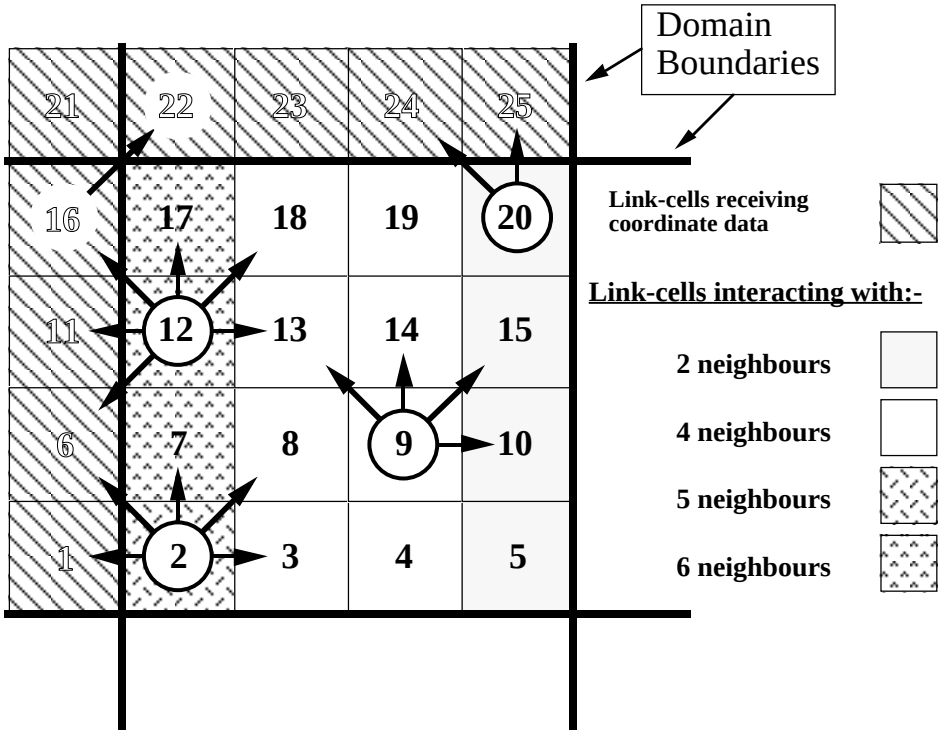

Figure 1 *A schematic 2-D representation of the link-cell neighbour search pattern used in the domain decomposition program. The thick black lines mark the boundaries between regions which different processors are responsible for. In this example all the 16 link-cells that are resident on the central processor (numbered in **bold** type) are searched using the pattern shown. In addition all the particles in the received cell 16 are interacted with those in 22.*

A link-cell search for interacting pairs normally treats each link-cell in the same way. Particles within a link-cell are interacted amongst themselves and then with the contents of half the 26 (8 in 2-D) nearest neighbour link-cells [1]. If this search method was to be applied in a domain decomposition code, it would require direct or indirect communications with 17 (5 in 2-D) of the nearest neighbour processors. By modifying the search pattern of link-cells according to their position within a domain, direct and indirect communications are only required with 7 (3 in 2-D) of the nearest neighbour domains and can be accomplished efficiently with just three (two in 2-D) communication steps. The flow of information during these three communication steps [7] is demonstrated in Fig. 2.

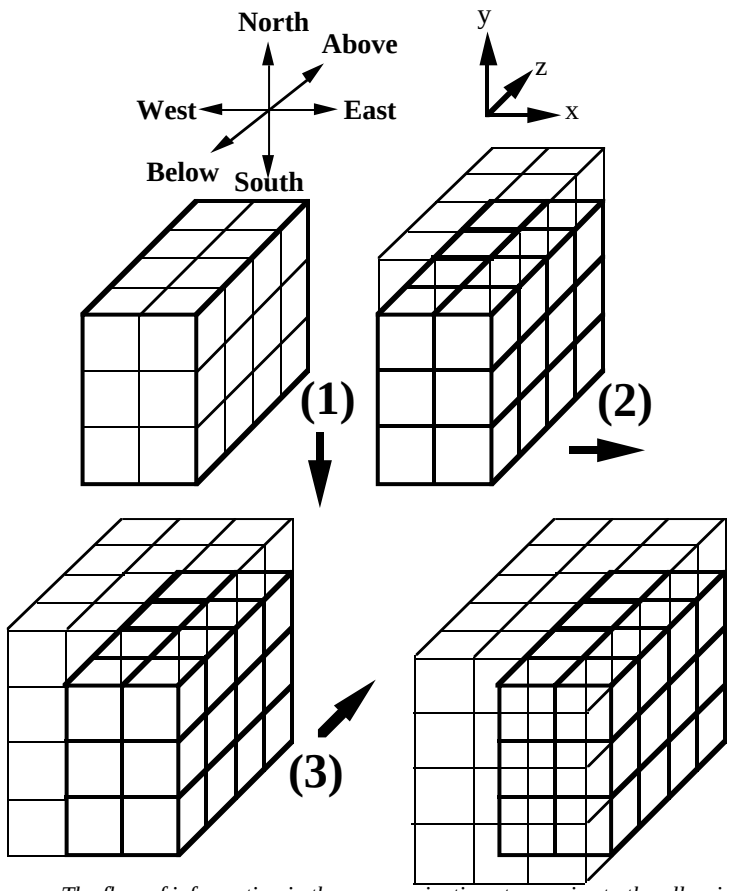

Figure 2 *The flow of information in the communication stage prior to the all-pair search. The figure shows an initial set of 2×3×4 link-cells in a domain (top left). At the first communication step all PEs pass to their southern neighbour all the coordinates of sites in the link-cells on their southern periphery. This results in each PE now having access to coordinates in an extra layer of link-cells (top right). Two further communication stages follow, to the east and to the above neighbours, in an analogous manner to give the final working environment (lower right).*

As can be seen from Fig. 2, a subject domain's working environment of link-cells is built up by communications in each of the three perpendicular directions, *i.e.* with respect to the aforementioned scaled space used to assign particles initially to PEs. The nomenclature used is that the neighbouring PE in the -y ("southward") direction is called South, that in the +x ("eastward") direction is called East and that in the +z "upward" direction is called Above. In the first stage of communication, each PE simultaneously sends information for all particles found on the southern layer of link-cells to its southern neighbour. At the same time, each PE receives some information from its northern neighbour. This increases the working environment of a PE by one layer of link-cells. In the second and third stages similar exchanges occur in the other two directions, as shown in Fig. 2.

Following this initial communication stage, each processor has access to enough of the coordinate information to perform its share of the all-interacting-pairs search. This initial pair search not only

allows the interaction of all non-bonded pairs but also the formation of neighbour tables of non-bonded pairs. These neighbour tables are used in subsequent steps until they become redundant. In the worst possible case, this is when any site has diffused a distance greater than or equal to half the shell width (a parameter used in the construction of the neighbour table [7]) since the last rebuild. Only at this stage is it also necessary for sites to get reassigned to different PEs, should they have diffused out of their previous home PEs, so as to ensure that sites appear in their correct link-cells for the next rebuild. Subsequent steps are handled by the neighbour tables and other lists formed so it is irrelevant if a site wanders out of the region that its home PE is nominally responsible for.

3.3 Domain decomposition for molecular systems

The main problem encountered in applying domain decomposition to molecular systems is the unique identification of the n-plet groups required for multi-particle potentials and constraints. In their work on a domain decomposition algorithm for a molecular system, Esselink and Hilbers [17] show that, *as long as the distance between any two atoms in an n-plet is no greater than that of the truncation distance of the non-bonded pair potential, it is certain that at least one processor will have all the n atoms in its working environment*. As a result, no extra communication costs need to be incurred over those of the search for all-interacting-pairs. The same result was also found for linear chain systems [7], despite the fact that a different communication environment was used.

Given the above guiding principle, the problem thus reduces to one of *identifying* that processor which will be temporarily responsible for a particular n-plet and *signalling* to it that this is the case. Extensive details of the solution to this problem are given in [10]; in summary it involves the following novel features:-

(i) In addition to its coordinates and velocities, each atom carries around with it details of its atom type, global atomic index, global molecular index and information concerning the neighbours with which it is has 'bonded' interactions.

(ii) All bends, torsions and groups requiring special constraints, *e.g.* those for CH_2 groups as in [22], are assigned to a particular atom and are also carried by this atom whichever processor it moves to.

(iii) At the first step, and other steps where the neighbour lists *etc*. have to be rebuilt, a table is required indicating the local index and processor identity of all atoms known to each PE, *i.e.* both those a processor is directly responsible for and all those it receives during the communication stages prior to the search for all interacting pairs.

(iv) Using the tables formed in (iii), any two atoms found during the pair search to be involved in a 'bonded' interaction make further lists which contain each others indices and processor identities in all the PEs they are known in. This information is communicated back to each atom's home PE at the end of the pair search. By doing this, each atom knows for each of its 'bonded' neighbours all PEs in which they both appear in and their local indices therein. It is then a simple matter of finding for each n-plet the processor which is aware of all n atoms and signalling to it that it is responsible for that n-plet. This procedure allows each PE to build lists of all the n-plets it will handle for this and following steps until the lists are deemed redundant.

Although this procedure increases some of the communication costs, much of the information is invariant for the period that each processor works with the same set of atoms, *i.e.* between neighbour table rebuilds, and so the effect of the extra overhead is ameliorated.

Previously, we discussed a parallel implementation of SHAKE appropriate for rigid bond constraints [7] and this approach has only to be slightly adapted to accommodate the special CH_2 group constraints. At each SHAKE iteration, the rigid bond constraints are imposed first and all displacements to atoms occurring in other PEs are sent back to the home PEs. The resulting positions, updated for the imposition of the rigid bond constraints, are then used as input to an iteration of the CH_2 special constraints. As for the rigid bonds, this involves an outward communication of current positions followed by a return of any displacements to atoms which occur in other PEs. Using this approach, the algorithm converges within a reasonable number of iterations.

3.4 The general purpose parallel MD code *ddgmq*

Using the techniques summarized above, an existing general purpose scalar code, *gmq*, has been successfully parallelized. The working code, *ddgmq*, has been implemented on Silicon Graphics Power Challenge and Origin 2000, IBM SP2, Cray T3E and Sun Enterprise 10000 machines with communications between processors being handled by calls to the MPI message passing library. The program contains a parallel SHAKE algorithm to handle both rigid bond constraints and special CH_2 constraints. Simulations are performed using 3-D periodic boundary conditions within a primary MD box defined by three arbitrary basis vectors, *i.e.* even triclinic cells can be simulated. Techniques are implemented for allowing control of the temperature and the pressure tensor in the system. Full details of the functionality of the *gmq* and *ddgmq* codes and their mode of operation can be found in the manual [6].

The code is currently being used to perform ~1 ns simulations involving up to ~100000 atoms with full account taken of the electrostatic interactions [5]. The new parallelized general MD code can perform such computations in reasonable times (2 or 3 weeks). Such simulations were unfeasible to do within a reasonable time scale using scalar codes.

4. Benchmark

The benchmark system is a small peptide called echistatin, which contains 49 amino acids (713 atoms), immersed in a 60^3 Å^3 cubic (periodic) cell filled with 6903 water molecules and one chloride ion, to maintain charge neutrality, i.e. a total of 21423 atoms in total at a density of 997.97 kg m^{-3}. The force-field used to specify the interactions between atoms was based on CVFF. The values used throughout for the Ewald summation were α=0.29 Å^{-1}, r_c=8.5 Å and K_{max}=14 (see [6]). A real space truncation of 8.5 Å allows the system to be decomposed into a maximum of 3×3×3=27 domains each of ~20 Å side length and each containing the minimum number of link-cells allowed by the code, *i.e.* 2 per dimension each with a length greater than r_c plus the shell-width (1.5 Å) used in the creation of the neighbour table. Rigid bond constraints were satisfied to a *relative* tolerance, 10^{-5}. The benchmark itself consisted of performing 100 integration steps of length 1 fs on a configuration previously equilibrated at a temperature of ~298 K; further details can be found in [10]. The mean CPU times per step and relative speed-ups as a function of the number of processors are given in table 1.

	SGi Origin 2000		**IBM SP2**		**CRAY T3E**	
No. of PEs. (N_P)	CPU time per step /s (τ_P)	Speed-Up τ_1/τ_P	CPU time per step /s (τ_P)	Speed-Up $2*\tau_2/\tau_P$	CPU time per step /s (τ_P)	Speed-Up $12*\tau_{12}/\tau_P$
1	24.20	1	-	-	-	-
2	12.19	2.0	14.64	2	-	-
4	4.98	4.9	7.66	3.8	-	-
8	2.51	9.6	4.27	6.9	-	-
12	1.80	13.4	3.46	8.5	4.36	12
18	1.32	18.3	2.60	11.3	2.87	18.2
27	1.02	23.8	2.48	11.8	1.75	29.9

Table 1. The results for the echistatin in water benchmark for the general purpose parallel MD program *ddgmq* running on three different, and heavily utilised, machines:-
SGi Origin 2000: INRIA, Nancy (64 PEs, MIPS R10000, 195 MHz, 8 GByte RAM)
IBM SP2: CNUSC, Montpellier (127 PEs, P2SC, 256 Mbyte RAM per PE)
CRAY T3E: IDRIS, Orsay (256 PEs, DEC Alpha EV5, 300 MHz, 128 Mbyte RAM per PE)

The benchmark is too large to run on one processor of the IBM SP2 or even 8 on the CRAY T3E, a problem related to the distributed memory of these machines compared to the physically distributed/logically shared memory of the SGi Origin 2000, so speed-ups for these machines are

based on the time for the minimum number of processors usable. As all the timings were obtained with the machines in multi-user mode and subject to varying loads not too many conclusions can be drawn. The speed-ups on the SGi and Cray machines hold up well but the code performs less well on the SP2.

5. Discussion

5.1 Ewald vs P^3M

It is well known that the Ewald method doesn't scale linearly with system size and many attempts have been made to overcome this using other techniques and current opinion [39] favours particle-particle particle-mesh (P^3M) type approaches [24]. For large systems on scalar machines there is no doubt such approaches have a big advantage, however, within DD this is not quite so clear. It has to be remembered that the real space pair search has to be carried out in any case for the short range potentials. The best that can be achieved then is to reduce the CPU time for the reciprocal space sum, or the particle-mesh part of P^3M, down to an insignificant fraction of the total. Traditional FFT approaches to solving this problem using P^3M are not easily adapted to DD. Alternative iterative routes [3] in principal get over the mapping onto DD but don't deliver the same accuracy as Ewald at any less cost. In any case, for the size of systems that are feasible on the present generation of machines the cost of the reciprocal space part of the Ewald sum is about the same as for the real space sum so at best a factor of two speed-up would be obtained. Given this and the fact that P^3M approaches are, unlike Ewald, *intrinsically* approximate methods, often requiring the optimisation of many tuning parameters, we prefer to continue using our current methods to evaluate Coulombic sums.

5.2 Constraints vs continuous potentials

For systems where Coulombic interactions have to be taken into account, such as the benchmark system above, the overall cost of the parallel SHAKE algorithm is < 2%, i.e. the calculation is dominated by the cost of the non-bonded pair interactions. In other systems where Coulombic potentials are absent and the range of the non-bonded potential is very short [9] the cost rises to ~20%. So in either case the CPU cost of using rigid constraints is easily justified by the increase in time step and the removal of equipartition problems.

5.3 Load balancing

In simulations of pure liquids there is usually a homogeneity in the density which means that there is a natural load balancing; each processor receives roughly equal numbers of particles. In multiphase or interfacial systems there is likely to be a variation in the number density of sites within the simulation box and perhaps also a variation of the complexity of molecules also; leading to imbalances in the numbers of pairs, bends, torsions etc. in the lists for each PE. In this situation there will be benefits to be gained from balancing the load on the processors.

5.4 Increasing the number of processors

One of the restrictions of the current implementation of DD is that the truncation range of the interaction potential determines the smallest domain into which the system can be decomposed. This is necessary at present to ensure that only local communications with neighbouring domains are required to carry out the all-pair search. This places a restriction on the maximum number of processors that can be used. To subdivide the system into smaller domains whilst maintaining the same interaction truncation distance implies communications between non-neighbouring PEs. A method which accomplishes this indirectly by local communications alone has recently been described [35] and this may be a way to increase the efficacity of DD.

Acknowledgments

We acknowledge the support of the Charles Hermite Centre in Nancy, the IDRIS centre in Orsay and the CNUSC centre in Montpellier for the provision of computer time on the various machines and for the kind help of their technical staff.

References

[1] Allen, M.P. and D.J. Tildesley, *Computer Simulation of Liquids*, Clarendon Press, Oxford, 1987.

[2] Beazley, D.M. and P.S. Lomdahl, "Message-passing multi-cell molecular dynamics on the Connection Machine 5", *Parallel Computing*, **20**, 173-195, (1994).

[3] Beckers, J.V.L., C.P. Lowe and S.W. de Leeuw, "An iterative PPPM method for simulating Coulombic systems on distributed memory parallel computers", *Molecular Simulation*, **20**, 369-383, (1998).

[4] Berendsen, H.J.C., D. van der Spoel and R. van Drunen, "Gromacs - a Message-Passing Parallel Molecular-Dynamics Implementation", *Computer Physics Communications*, **91**, 43-56, (1995).

[5] Brown, D., *A Summary of Research Activities*, Habilitation à Diriger des Recherches Thesis, Henri Poincaré, Nancy I, 1998.

[6] Brown, D., *The gmq User Manual Version 2*, 1998:
available at http://www.lctn.u-nancy.fr/Chercheurs/David.Brown

[7] Brown, D., J.H.R. Clarke, M. Okuda and T. Yamazaki, "A Domain Decomposition Parallel-Processing Algorithm For Molecular- Dynamics Simulations Of Polymers", *Computer Physics Communications*, **83**, 1-13, (1994); *ibid* **86**, 312 (1995).

[8] Brown, D., J.H.R. Clarke, M. Okuda and T. Yamazaki, "A Domain Decomposition Parallelization Strategy For Molecular- Dynamics Simulations On Distributed Memory Machines", *Computer Physics Communications*, **74**, 67-80, (1993).

[9] Brown, D., J.H.R. Clarke, M. Okuda and T. Yamazaki, "A large scale molecular dynamics study of chain configurations in the n=100 alkane liquid", *J. Chem. Phys.*, **104**, 2078-2082, (1996).

[10] Brown, D., H. Minoux and B. Maigret, "A domain decomposition parallel processing algorithm for molecular dynamics simulations of systems of arbitrary connectivity", *Comp. Phys. Comm.*, **103**, 170-186, (1997).

[11] Bruge, F., "A Mixed Geometric-Systolic Approach to Parallel Molecular-Dynamics Simulations", *Computer Physics Communications*, **90**, 59-65, (1995).

[12] Bruge, F. and S.L. Fornili, "Concurrent molecular dynamics simulation of spinodal phase transition on transputer arrays", *Comp. Phys. Comm.*, **60**, 31-38, (1990).

[13] Chynoweth, S., U. Klomp and L. Scales, "Simulation of organic liquids using pseudo-pairwise interatomic forces on a toroidal transputer array", *Comp. Phys. Comm.*, **62**, 297-306, (1991).

[14] Clark, T.W., R. van Hanxleden, J.A. McCammon and L.R. Scott. *Parallelizing Molecular Dynamics using Spatial Decomposition.* in *Proc. Scalable High Performance Computing Conference-94.* 1994. Knoxville, TN: IEEE Computer Society Press, Los Alamitos, CA.

[15] Debolt, S.E. and P.A. Kollman, "Ambercube Md, Parallelization Of Ambers Molecular-Dynamics Module For Distributed-Memory Hypercube Computers", *Journal Of Computational Chemistry*, **14**, 312-329, (1993).

[16] Eisenhauer, G. and K. Schwan, "Design and Analysis Of a Parallel Molecular-Dynamics Application", *Journal Of Parallel and Distributed Computing*, **35**, 76-90, (1996).

[17] Esselink, K. and P.A.J. Hilbers, "Efficient Parallel Implementation Of Molecular-Dynamics On a Toroidal Network .2. Multiparticle Potentials", *Journal Of Computational Physics*, **106**, 108-114, (1993).

[18] Ewald, P.P., *Ann. Phys.*, **64**, 253, (1921).

[19] Fincham, D., "Parallel computers and molecular simulation", *Molecular Simulation*, **1**, 1-45, (1987).

[20] Frenkel, D. and B. Smit, *Understanding molecular simulation: from algorithms to applications*, Academic Press Inc., San Diego, 1996.

[21] Haile, J.M., *Molecular Dynamics Simulation: Elementary Methods*, John Wiley & Sons Ltd., New York, 1992.

[22] Hammonds, K.D. and J.-P. Ryckaert, "On the convergence of the SHAKE algorithm", *Comp. Phys. Comm.*, **62**, 336-351, (1991).

[23] Hedman, F. and A. Laaksonen, "An Approach to Data-Parallel Molecular-Dynamics For Liquids", *International Journal Of Modern Physics C Physics and Computers*, **4**, 41-48, (1993).

[24] Hockney, R.W. and J.W. Eastwood, *Computer Simulation using Particles*, IOP, Bristol, 1987, 540.
[25] Hwang, Y.S., R. Das and J.H. Saltz, "Parallelizing Molecular-Dynamics Programs For Distributed-Memory Machines", *Ieee Computational Science & Engineering*, **2**, 18-29, (1995).
[26] Janak, J.F. and P.C. Pattnaik, "Protein Calculations On Parallel Processors .2. Parallel Algorithm For the Forces and Molecular-Dynamics", *Journal Of Computational Chemistry*, **13**, 1098-1102, (1992).
[27] Kalia, R.K., A. Nakano, D.L. Greenwell, P. Vashishta and S.W. de Leeuw, "Parallel Algorithms For Molecular-Dynamics Simulations On Distributed-Memory Mimd Machines", *Supercomputer*, **10**, 11-25, (1993).
[28] Liem, S.Y., D. Brown and J.H.R. Clarke, "An investigation of the homogeneous shear non-equilibrium molecular dynamics method", *Phys. Rev. A.*, **45**, 3706-3713, (1992).
[29] Liem, S.Y., D. Brown and J.H.R. Clarke, "Molecular dynamics simulations on distributed memory machines", *Comp. Phys. Comm.*, **67**, 261-267, (1991).
[30] Lin, S.L., B.M. Pettitt and G.N. Phillips, "Molecular-Dynamics With Charmm On a Parallel Computer", *Faseb Journal*, **6**, A 464-a 464, (1992).
[31] Mattson, T.G. and G. Ravishanker, "Portable Molecular-Dynamics Software For Parallel Computing", *Acs Symposium Series*, **592**, 133-150, (1995).
[32] Müller-Plathe, F. and D. Brown, "Multi-colour algorithms in molecular simulatons: vectorization and parallelisation of internal forces and constraints", *Comp. Phys. Comm.*, **64**, 7-14, (1991).
[33] Petersen, H.G. and J.W. Perram, "Molecular dynamics on Transputer arrays. I. Algorithm design, programming issues, timing experiments and scaling projections", *Molecular Physics*, **67**, 849-860, (1989).
[34] Pinches, M.R.S., D.J. Tildesley and W. Smith, "Large scale molecular dynamics on parallel computers using the link-cell algorithm", *Molecular Simulation*, **6**, 51-87, (1991).
[35] Plimpton, S., "Fast Parallel Algorithms For Short-Range Molecular-Dynamics", *Journal Of Computational Physics*, **117**, 1-19, (1995).
[36] Plimpton, S. and B. Hendrickson, "A new parallel method for molecular dynamics simulations of macromolecular systems", *J. Comp. Chem.*, **17**, 326-337, (1996).
[37] Plimpton, S. and B. Hendrickson, "Parallel Molecular-Dynamics Simulations Of Organic Materials", *International Journal Of Modern Physics C Physics and Computers*, **5**, 295-298, (1994).
[38] Plimpton, S., R. Pollock and M. Stevens. *Particle-Mesh Ewald and rRESPA for Parallel Molecular Dynamics Simulations*. in *SIAM Conference on Parallel Processing for Scientific Computing*. 1997.
[39] Pollock, E.L. and J. Glosli, "Comments on the P3M, FMM, and the Ewald method for large periodic Coulombic systems", *Comp. Phys. Comms.*, **95**, 93-110, (1996).
[40] Raine, A.R.C., D. Fincham and W. Smith, "Systolic loop methods for molecular dynamics simulation using multiple Transputers", *Comp. Phys. Comm.*, **55**, 13, (1989).
[41] Rapaport, D.C., "Microscale hydrodynamics; discrete-particle simulation of evolving flow patterns", *Phys. Rev. A.*, **36**, 3288-3299, (1987).
[42] Rapaport, D.C., "Multimillion Particle Molecular-Dynamics .3. Design Considerations For Data-Parallel Processing", *Computer Physics Communications*, **76**, 301-317, (1993).
[43] Ryckaert, J.-P., "Special geometric constraints in the molecular dynamics of chain molecules", *Molecular Physics*, **55**, 549-556, (1985).
[44] Ryckaert, J.-P., G. Ciccotti and H.J.C. Berendsen, "Numerical integration of the cartesian equations of motion of a system with constraints: molecular dynamics of liquid alkanes", *J. comput. Phys.*, **23**, 327-41, (1977).
[45] Sato, H., Y. Tanaka and T. Yao, "Molecular-Dynamics Simulation On an Ap1000 Distributed Memory Parallel Computer", *Fujitsu Scientific & Technical Journal*, **28**, 98-106, (1992).
[46] Smith, W., "Molecular-Dynamics On Distributed Memory (Mimd) Parallel Computers", *Theoretica Chimica Acta*, **84**, 385-398, (1993).
[47] Smith, W., "Molecular dynamics on hypercube parallel computers", *Comp. Phys. Comm.*, **62**, 229-248, (1991).

[48] Smith, W., "A Replicated Data Molecular-Dynamics Strategy For the Parallel Ewald Sum", *Computer Physics Communications*, **67**, 392-406, (1992).
[49] Streett, W.B., D.J. Tildesley and G. Saville, "Multiple time step methods in molecular dynamics", *Mol. Phys.*, **35**, 639-648, (1978).
[50] Tamayo, P., J.P. Mesirov and B.M. Boghosian. *Parallel approaches to short range molecular dynamics simulations*. in *Proceedings of Supercomputing '91*, 1991, Albuquerque, New Mexico: IEEE Computer Society Press.
[51] Tuckerman, M.E., G.J. Martyna and B.J. Berne, "Molecular dynamics algorithm for condensed systems with multiple time scales", *J. Chem. Phys.*, **93**, 1287-1291, (1990).
[52] Zhou, R.H. and B.J. Berne, "A New Molecular-Dynamics Method Combining the Reference System Propagator Algorithm With a Fast Multipole Method For Simulating Proteins and Other Complex-Systems", *Journal Of Chemical Physics*, **103**, 9444-9459, (1995).