Theor Chim Acta (1993) 84: 399–411

Theoretica Chimica Acta

© Springer-Verlag 1993

Simulation of condensed phases using the Distributed Array Processor

Michael P. Allen

H. H. Wills Physics Laboratory, Royal Fort, Tyndall Avenue, Bristol, BS8 1TL, UK

Received October 1, 1991/Accepted November 27, 1991

Summary. The use of a massively parallel computer for simulations of condensed matter systems at Bristol University is reviewed, with a discussion of the factors influencing the choice of algorithms. Emphasis is placed on the importance of adopting simple, easily-modifiable algorithms, based where possible on geometrical domain decomposition. Several examples of scientific applications are given.

Key words: Condensed phases – Distributed Array Processor – Parallel computers

1. Introduction

1.1. Parallel computers and scientific research

This is a short review of work carried out in theoretical physics at Bristol University, using the Distributed Array Processor (DAP). The condensed matter theory group is interested in the statistical mechanics of inhomogenous systems and phase transitions, including liquid/vapour wetting phenomena, magnetic multilayers, high-T_c superconductors, liquid crystals, and surface-adsorbed molecules.

The DAP was provided by the Science and Engineering Research Council (SERC) under its Computational Science Initiative, the main aim of which is to place computing equipment in individual research laboratories, to support scientific research for which local computer power is essential. This complements SERC's provision of central supercomputer time, and the general computing infrastructure available at most U.K. universities and research establishments. The DAP is a fine-grained, massively-parallel, single-instruction-multiple-data (SIMD) machine.

In a recent review of the scientific achievements of the Computational Science Initiative [1], it is clear that many elements of the U.K. scientific community have a strong commitment to the development of methods for parallel computer architectures. In the short term, it is obviously easier to generate scientific results on conventional scalar and vector mini-supercomputers, but in the long term the investment of effort in parallel algorithm development may be crucial.

In this paper, I shall emphasize that the characteristics of the DAP have allowed us to use simple algorithms for the simulation of atomic and spin models in statistical mechanics; many of these algorithms are well known on scalar and vector machines. Simplicity is important when, as in our case, programs are continually being developed and modified in response to fresh scientific challenges. It is also important to us that new students, research assistants and visitors are able to learn quickly how to program the machine to work efficiently and without errors.

The layout of the paper is as follows. In the rest of this section I shall give a short summary of the simulation techniques of interest, and a description of the DAP. Then, I shall describe a variety of methods used to simulate both lattice-spin and atomic systems, mostly based on the approach of geometrical parallelism (domain decomposition). Some references to the scientific work carried out in our group will be given. Finally, I shall draw some general conclusions regarding the applicability of fine-grained SIMD machines in this field.

1.2. Simulation methods

Classical condensed-matter systems may be modelled in a variety of ways. These may be split into categories roughly as follows, in increasing order of realism: discrete or continuous spins on a lattice, examples being the well-known Ising and Heisenberg models; hard particles such as hard spheroids and spherocylinders, free to translate and rotate; soft particles, such as the Lennard–Jones and Gay–Berne [2] potentials; and finally 'realistic' interatomic potential models, possibly with internal flexibility and distributed charges to represent the electron density.

To date, most of our DAP work has concentrated on spin systems (for which the architecture is ideal) and on simple atomic systems, representative of the last two categories above. While some hard-particle simulations may be vectorized and parallelized quite efficiently, most of our work in this area [3] has not been carried out on the DAP. This is because the interesting system sizes are still relatively small, so highly parallel architectures are disfavoured. As the interest shifts to larger systems, this situation will change. Here I shall concentrate on spin and atomic systems, in a regime where large system sizes are essential, so the 'scaling' problem (i.e. small system but large computer) does not arise.

The traditional simulation methods are Monte Carlo and molecular dynamics, although hybrids are possible as we see elsewhere in these proceedings [4]. Monte Carlo involves the selection of random attempted moves, which are then accepted or rejected according to some stochastic prescription, to generate states sampled from a desired statistical ensemble. This may be applied to any of the systems mentioned above. The essential requirement is that it is possible to evaluate interaction energies, usually between pairs of atoms or spins, efficiently. (Here we restrict our interest to pairwise interactions.) Molecular dynamics is essentially the step-by-step solution of the deterministic evolution equations. It may be applied to all the above systems except for discrete-state spins (although even here the simulation of cellular automata is in the same spirit). The requirement is that, at each time step, it is possible to evaluate pair energies and forces efficiently.

As will be seen below, the common situation is that interactions are of short range compared with the overall size of the system (but see Sect. 4.3). We seek a way of avoiding the consideration of out-of-range interactions. The ideal solution to this problem on a parallel computer is to map this geometrical situation directly onto the structure of the machine. Good local connectivity on a topologically three-dimensional grid will allow efficient evaluation of short range interactions in a physically three-dimensional system. The interaction range will dictate the distance over which data must be transferred. A further requirement is that the number of particles must be comparable with, or exceed, the number of processors: then each processor can be responsible for a single particle or a region of space containing a group of particles. Otherwise the scaling problem appears, and an efficient way must be found of distributing the interactions within a given region of space amongst several processors. The simpler, large-system, situation applies in our case.

2. DAP architecture and programs

2.1. Structure

The DAP is a massive fine-grained parallel computer, with the processors arranged in a 32×32 two-dimensional array (a 64×64 version is also available). Each processor has its own store, there are fast nearest-neighbour connections, and row and column data highways for broadcasting and fetching operations. The processors are bitwise rather than floating point chips, and the machine has a floating-point performance of the order of 10 Mflops; however operations on short integers, and especially logical variables, are very fast indeed. An 8-bit coprocessor array is now available, boosting the floating point performance to ≈ 60–75 Mflops; one might reasonably expect to achieve 40 Mflops on real applications. All the work described here was performed on the original machine without the coprocessors.

2.2. Language

The DAP is programmed in a parallel extension of Fortran. Parallel data objects, two-dimensional matrices and one-dimensional vectors, are defined and manipulated by single high-level instructions. Originally these variables were constrained to have dimension 32, but this restriction is relaxed in the latest version of the language. Nonetheless, two array indices are mapped, in general, onto the two dimensions of the processor array; the simplest parallel constructs involve an implied double DO loop over these indices, much as one can imagine an implied single DO loop on a vector machine. Logical objects are used as masks, to screen out the results of parallel operations, like 'bit vectors' on some vector machines.

Global summations, one- and two-dimensional shifts with or without cyclic boundary conditions, and broadcast operations, are all provided as efficient intrinsic functions.

3. Algorithms for lattice systems

Since the DAP architecture is a regular grid, it is well suited to simulation of systems having a permanent lattice structure. Here I describe various applications and techniques studied in Bristol.

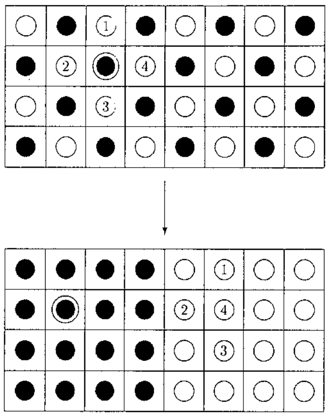

Fig. 1. Multispin coding. We show the original (*upper*) and transformed (*lower*) lattice, using a *black/white checkerboard* labelling. With nearest-neighbour interactions each sublattice may be updated in parallel, holding the other one fixed. The nearest neighbours (1–4) of a typical black spin (*circled*) are shown. Here and throughout we illustrate the methods using a two-dimensional 4 × 4-processor machine as an example

3.1. Multi-spin coding

The term 'multi-spin coding' refers to the encapsulation of several logical variables or bits into a single word of storage, which is then processed in one pass by hardware designed to execute integer and floating-point operations efficiently. This is an early example of SIMD parallel processing on a single processor. It has been with us for at least 20 years [5] and is well explained in the standard references [6]. It is implemented in a natural way on the DAP, as the simple illustration in Fig. 1 shows. A standard checkerboard black/white sublattice structure is imposed on a square lattice having spins at each site. A parallel data transform is applied to segregate the black and white spins. Provided that interactions are restricted to nearest neighbours, it is permissible to update all the black spins at once, simultaneously and independently, by the usual Monte Carlo rules. Typical neighbours of a given spin are shown in the figure. Then the same procedure is applied to the white spins. The configuration can be kept in its transformed representation for almost the whole simulation; the reverse transform need only be applied occasionally, for example when a picture of the configuration is required.

In common with many other groups, we have applied this approach to simulations on a variety of lattices, in both two and three dimensions. Some examples follow.

3.1.1. Wetting phenomena. A fluid in a pore or capillary [7] may be modelled as a lattice gas (a logical variable at each site denoting the presence or absence of fluid) within attractive walls. Nearest-neighbour coupling terms, and a longer-ranged wall-fluid potential, represent the essential physics. The wall acts to shift the first-order liquid-vapour transition, turning it into a capillary condensation line. The bulk critical point is shifted, and becomes a capillary critical point. In addition, a transition occurs at the surface, between a thin and a thick film of adsorbed fluid. These 'prewetting' transitions are first order, and extend from the bulk phase transition line (where the two lines meet at the 'wetting' temperature) to terminate at a prewetting critical point. In an extensive series of simulations on the DAP [8] the prewetting line and the prewetting critical point for this model have been located. In addition, it has been shown that the prewetting critical point has two-dimensional Ising-like exponents. In other words, despite the fact that the adsorbed film is truly a three-dimensional entity, correlation lengths can only diverge in the two directions parallel to the wall, so, sufficiently close to the critical point, the critical fluctuations have two-dimensional character. This result was widely anticipated, but had never before been demonstrated explicitly.

3.1.2. Adsorbed molecular monolayers. When gases are adsorbed on a solid surface, for example N_2 on graphite, they can form molecular monolayers. The orientational ordering in such a system is of interest: for example N_2 forms a striped herringbone arrangement. A simple model, based on a triangular lattice with 3-state Potts-like spins, with nearest-neighbour coupling constants dependent on the direction of the site-site vector, has been devised to model this system [9]. Monte Carlo simulations, covering a large range of the phase diagram including the herringbone, ferromagnetic, and disordered phases, have been carried out on the DAP [10]. Comparison with theoretical predictions of the phase boundaries revealed that the ferromagnetic $\rightarrow$ disordered phase transition is quite well understood, but the herringbone $\rightarrow$ disordered phase transition is not adequately described. Fluctuations seem to destabilize the herringbone phase to a much greater extent than is predicted by theory.

3.1.3. Alloys. Similar methods have been applied to the simulation of magnetic and compositional order in nickel-rich Ni_cFe_{1-c} alloys [11] using a model with Ising-like magnetic and compositional freedom at each site. First a homogeneous alloy was investigated, and a fit to experimental data gave estimates of the Ni–Ni, Ni–Fe and Fe–Fe magnetic exchange interaction strengths, the latter turning out to be antiferromagnetic. The experimental behaviour of the system was adequately reproduced by the model, and several interesting discrepancies between mean field theory and simulation were noted. Then the same model was used to investigate a modulated alloy in which thin layers (between three and fifteen atomic layers thick) of iron and nickel are alternated to form a superlattice. Such systems may have considerable technological importance and our interest in them is prompted by experimental work being carried out at Bristol.

3.1.4. Liquid crystal films. The power of the DAP has enabled us to investigate the properties of thin films of liquid crystals, using a simple continuous-spin lattice model [12], in unprecedented detail. The shift in the transition temperature due to finite film width, and the orientational adsorption profiles, have been determined and compared with theory [13]. Simulations of this same model in bulk, with and without periodic external fields, have been used to determine accurately, for the first time, the Frank elastic constant, K [14]. We have shown that the reduced Frank constant $C = K/\overline{P_2}^2$ ($\overline{P_2}$ being the nematic order parameter) *increases* with increasing temperature, contrary to mean-field predictions. We also simulated directly the Freedericksz transition for this system, applying competing bulk and surface fields, showing that an elastic theory of the effect is qualitatively correct, but that the nature of the transition makes it an inaccurate method of determining K in a simulation. Currently we are investigating a version of the model with applied surface fields, which is predicted to exhibit a number of bulk and surface orientational transitions with variation of coupling strength and/or temperature. The computational effort necessary to determine the entire phase diagram by conventional methods is prohibitive. We are employing energy histogram methods and long runs ($\geqslant 10^6$ attempted moves per particle) with system sizes up to 2×10^4 spins; this would be impossible without a dedicated machine.

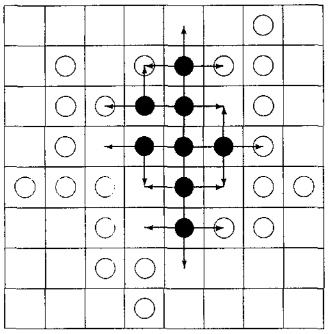

Fig. 2. Cluster updating on the DAP. In the 'ants-in-a-labyrinth' method, a population of 'ants' (*filled circles*), grown initially from a single site, extends its frontier until the entire cluster (*open circles*) is filled. Trial expansion steps, all conducted in parallel, are shown as *arrows*; each one landing on a new cluster site, generates a new 'ant', ready for the next step

3.2. Cluster updating

Because of the persistence of long-wavelength fluctuations near a critical point, there is a need to make large-scale moves in order to sample configuration space efficiently. The cluster-updating approach [15, 16] proceeds as follows. Contiguous clusters of identical spins are identified, and a nearest neighbour bond network is established using a stochastic prescription for the creation of bonds. This defines a set of subclusters. An attempt is made to flip all the spins in each sub-cluster simultaneously. A percolation theory treatment shows that, at the critical point, each sub-cluster can be treated independently. Away from the critical point, effective cluster-cluster interactions are introduced.

Such an algorithm can be efficiently implemented on the DAP, because efficient parallel algorithms exist to identify clusters: for example, the so-called 'ants-in-a-labyrinth' method [17, 18] illustrated in Fig. 2. Also, cluster-cluster interactions may be efficiently computed using single shift operations. This method has been tested on the DAP for the lattice-gas adsorption system discussed in the previous section.

3.3. Molecular dynamics and hybrid Monte Carlo

Molecular dynamics of a lattice spin system is easily implemented on the DAP, since it is a parallel step-by-step advancement of the configuration in accordance with the coupled differential equations obtained from the laws of motion. Similarly, the combination of molecular dynamics and Monte Carlo known as 'hybrid Monte Carlo' [19] is equally suitable for the DAP. This technique is described in more detail by Heerman [4], but briefly each step consists of carrying out a short molecular dynamics run, followed by a global acceptance or rejection. The method has been applied to a fully frustrated XY-model, relevant to models of high-T_c superconductivity [20].

3.4. Mass-tensor dynamics

Another way of accelerating simulations is to adopt the molecular dynamics approach and attempt to choose the particle masses, or moments of inertia, so as to equalize the timescales of all the fundamental modes of the system. In this way the problem of dealing with slow global evolution while having to use a short timestep to cope with the fast modes, is avoided. A general approach of this kind [21] is to write Hamilton's equations in the form:

$$\mathscr{H}(\{q_i, p_i\}) = \tfrac{1}{2}\sum_{ij} p_i (M^{-1})_{ij} p_j + \mathscr{V}(\{q_i\})$$

$$\dot{p}_i = -\partial\mathscr{V}/\partial q_i$$

$$\dot{q}_i = \sum_j (M^{-1})_{ij} p_j$$

where the q_i are coordinates and the p_i are conjugate momenta. Now the mass has become a 'mass tensor' $\boldsymbol{M}$, coupling different degrees of freedom together. However, this complicated form of the kinetic energy does not affect the ensemble averages of configurational properties, just the dynamics. This approach has been used occasionally since its invention [22], but only for a few systems is the best choice of $\boldsymbol{M}$ obvious. The method has been implemented on the DAP for a model of a fluid membrane [23] in which lattice variables q_i represent the height of the surface, and the potential part of the hamiltonian is a discretization of the bending energy $\mathscr{V} = \sum_{ij} q_i V_{ij} q_j$, where $V_{ij} = (\nabla^2)^2_{ij}$ (∇^2 is the lattice Laplacian). For this system it can be shown that $M_{ij} = V_{ij}$ is a good choice. Indeed, in its simplest form, this model is exactly soluble, and (with this choice of $\boldsymbol{M}$), all the oscillations have equal frequencies. The real interest is in simulating the membrane in a confining potential, for which an exact solution is not available.

This method is quite suitable for the DAP involving solution of a large system of coupled linear equations, but for maximum efficiency in the general case a sparse matrix solver would be best (both $\boldsymbol{V}$ and $\boldsymbol{M}$ are sparse). Such a routine is not yet available on the DAP, to our knowledge. For the model described above, a fast solution on the DAP is possible via Fourier transformation [23].

4. Algorithms for atomic systems

In the following sections we consider molecular dynamics algorithms for a variety of physical situations, as implemented on the DAP. Several general reviews of molecular dynamics algorithms on parallel machines are available elsewhere [24–26].

4.1. *Atomic crystals – short-range forces*

For atomic crystals, in which no diffusion occurs, we have essentially a lattice problem. The same considerations apply as in the previous section, except that interactions are usually not restricted to nearest neighbours, but instead extend out to some finite range (typically a few lattice spacings). In the following we write out some pseudocode, based on a machine with ideal architecture for the problem: a two- or three-dimensional lattice (cubic for simplicity) with fast local connectivity. The code can be adapted later to fit the actual machine available (for example by replacing an implied loop over one index by an explicit one). Each processor holds the position, velocity and force components for one atom.

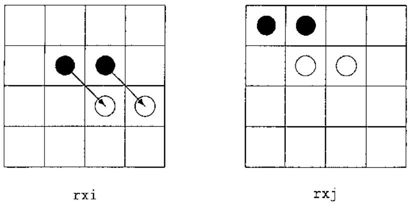

Fig. 3. The positions in *rxj* are obtained by shifting those in *rxi* by the prescribed amount *dx*, *dy*, *dz*, and implementing cyclic boundary conditions. A single parallel subtraction *rxi*−*rxj* gives all the relative coordinates for use in the force routine. We show two representative interactions computed this way

The heart of the program is the calculation of the force on each atom. This is accomplished by an outer set of loops:

```
loop over $dx, dy, dx$
    call force $(dx, dy, dz)$
end loop.
```

The loop terminates when all displacements within the potential cutoff have been treated. The region spanned by *dx*, *dy*, *dz* need not be cubic: the range of the loops can be tailored to approximately fit a spherical cutoff. The force routine calculates, for each atom in parallel, its interaction with another atom displaced by *dx*, *dy*, *dz* lattice spacings in the three coordinate directions (see Fig. 3).

```
rxi = rx
ryi = ry
rzi = rz
rxj = rx( +dx, +dy, +dz)
ryj = ry( +dx, +dy, +dz)
rzj = rz( +dx, +dy, +dz)
rxij = rxi - rxj
ryij = ryi - ryj
rzij = rzi - rzj
rijsq = rxij ** 2 + ryij ** 2 + rzij ** 2
pairs = rijsq .lt. rcutsq
uij(pairs) = ... pair potential energy
fxij(pairs) = ... \
fyij(pairs) = ... |pair force vector
fzij(pairs) = ... /
fxi = fxij
fyi = fyij
fzi = fzij
fxj = -fxij( -dx, -dy, -dz)
fyj = -fyij( -dx, -dy, -dz)
```

$$
\begin{aligned}
&fzj = -fzij(-dx,\ -dy,\ -dz)\\
&ui = uij/2\\
&uj = uij(-dx,\ -dy,\ -dz)/2\\
&fx = fx + fxi + fxj\\
&fy = fy + fyi + fyj\\
&fz = fz + fzi + fzj\\
&u = u + \mathrm{sum}(ui) + \mathrm{sum}(uj).
\end{aligned}
$$

All the variables except for dx, dy, dz, and u are three-dimensional objects, and, apart from the final global sum, all the operations are fully parallel. The *pairs* mask is used to filter out those interactions beyond the cutoff. It is simple to translate this pseudo-code into a form suitable for the DAP: the main change is that one index (let's say iz, spanning the z coordinate) must be introduced and looped over explicitly. Cyclic boundary conditions are assumed to be built in to the shifting operations: $rx(+dx,\ +dy,\ +dz)$ is short for a cyclically shifted array. If it is necessary to use 'end-off' shifts rather than cyclic ones, then duplicate layers of atomic positions must be used around the basic simulation box, to mimic the effects of periodic boundaries (see [25, 26]).

4.2. *Atomic fluids – short-range forces*

For fluids, the simplest approach is to map the problem onto a regular lattice and then proceed as in the previous section. Since fluids are disordered, the mapping will be imperfect in some way; since atoms in a fluid diffuse, the mapping will have to be performed afresh or updated at intervals.

This approach has been around for many years [27] and is known as the 'link-cell' method; on a scalar machine a list is used to point to the labels of atoms within each lattice cell. On an ideal parallel machine of three-dimensional topology, subsidiary lists are not needed. There are two main variants of the method. In one [28], each cell may be defined to hold exactly one particle and particles are assigned to cells by sorting on their coordinates. In the other [25–27], each cell is mapped to a volume of space and each particle within this volume is assigned to the cell. In general a cell will be capable of holding several particles, but a variant of the method employs a fine mesh such that the corresponding volumes are able to hold at most one particle.

4.2.1. Monotonic grid method. In the method due to Boris [28] a grid is set up such that, as one traverses the storage array in each coordinate direction, the appropriate coordinate changes monotonically:

$$
\begin{aligned}
rx(ix,\ iy,\ iz) &\leqslant rx(ix+1,\ iy,\ iz)\\
ry(ix,\ iy,\ iz) &\leqslant ry(ix,\ iy+1,\ iz)\\
rz(ix,\ iy,\ iz) &\leqslant rz(ix,\ iy,\ iz+1).
\end{aligned}
$$

The idea is that the arrangement in store (illustrated in Fig. 4) roughly, but not precisely, corresponds to the spatial arrangement of atoms. Evaluation of interactions involves shifts out to some cutoff, with a safety margin to allow for this imprecision. The main loop and the force routine are exactly the same as in

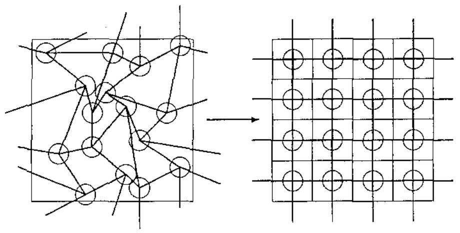
Fig. 4. The monotonic logical grid used in the Boris method

the previous section. Initially the assignment to cells is performed by a global sort on a key formed in some way from the x, y and z coordinates. The updating of the assignment is performed at regular intervals, and involves comparisons and swaps of neighbouring coordinates in an iterative, but efficient way. Example pseudo-code might be as follows:

```
... permute $y$ planes 1<->2 3<->4 ...
$swap = (ry - ry(0, +1, 0))$ .gt. 0
$swap = swap$ .and. $odd$
$swap = swap$ .or. $swap(0, -1, 0)$
$rx(swap) = \mathrm{merge}(rx(0, +1, 0), rx(0, -1, 0), odd)$
$ry(swap) = \mathrm{merge}(ry(0, +1, 0), ry(0, -1, 0), odd)$
$rz(swap) = \mathrm{merge}(rz(0, +1, 0), rz(0, -1, 0), odd)$
... permute $y$ planes 2<->3 4<->5 ...
$swap = (ry - ry(0, +1, 0))$ .gt. 0
$swap = swap$ .and. $even$
$swap = swap$ .or. $swap(0, +1, 0)$
$rx(swap) = \mathrm{merge}(rx(0, +1, 0), rx(0, -1, 0), even)$
$ry(swap) = \mathrm{merge}(ry(0, +1, 0), ry(0, -1, 0), even)$
$rz(swap) = \mathrm{merge}(rz(0, +1, 0), rz(0, -1, 0), even)$
```

Here we are looking at the y coordinates; similar code is applied to the x and z directions, and then the whole procedure iterated until no further swaps are needed. The variables *odd* and *even* are logical masks, containing *true* values for, respectively, odd and even values of the *iy* index, and *false* values elsewhere; *swap* identifies adjacent pairs that need swapping. As before, $rx(0, +1, 0)$ represents a matrix cyclically shifted by one position in the y direction. This code has been tested on the DAP; it works quite well, although there is a tendency for the grid to become highly distorted when large density fluctuations occur. The force routine is completely parallelized, but there is some wasted effort in computing out-of-range interactions, because of the need for a safety margin in the grid shift cutoff.

4.2.2. Cell method. In the cell method [25–27] each cell contains a (variable) number of atoms. The first atom in each cell belongs to 'layer 1', the second to 'layer 2' etc. (the term 'layer' does not have any geometrical meaning here).

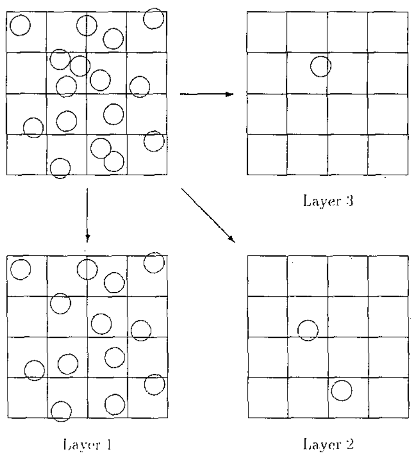

Fig. 5. The layer decomposition used in the Rapaport method

These are conveniently stored by adding an extra index to the coordinate arrays: *rx*(*ix*, *iy*, *iz*, *il*) etc. The assignment is illustrated in Fig. 5. The main loop again involves comparing entire matrices of coordinates with shifted copies, and looping over the shifts out to a cutoff dictated by the potential. Now there are additional loops over the 'layer' indices *il* and *jl* in shifted and unshifted data sets.

```
loop over $il, jl, dx, dy, dx$
    call force $(il, jl, dx, dy, dz)$
end loop
```

The force routine is very similar to that given before, with the addition of the indices *il*, *jl* to pick the appropriate layer from coordinates *rx*(*ix*, *iy*, *iz*, *il*) etc.

Again, updating the assignment is efficient, involving local comparisons of coordinates and local reassignment of storage locations. The force routine is completely parallelized, but there is some wasted effort in examining 'interactions' involving empty storage locations. In the 'small cell' variant, the layer indices are omitted altogether, and the cells contain either one or zero particles; the same comments on efficiency apply, and the choice of cell size is partly a matter of trial and error. A modification of this algorithm, for molecules adsorbed in multilayers on a solid surface, is being developed on the DAP, in collaboration with R.M. Lynden-Bell (Cambridge).

4.3. Atomic fluids – long-range forces

When all pair interactions must be computed, the advantages of a geometrical decomposition disappear, and a simple systolic loop is sufficient. This may be employed in the well-known Brode-Ahlrichs fashion [29], originally developed for vector machines. One particle is allocated to each processor: for N particles, the $\frac{1}{2}N(N-1)$ interactions are treated in $\frac{1}{2}(N-1)$ operations each treating N interactions in parallel. Between one operation and the next, a cyclic one-dimensional shift is carried out, as illustrated in Fig. 6. The method works best when

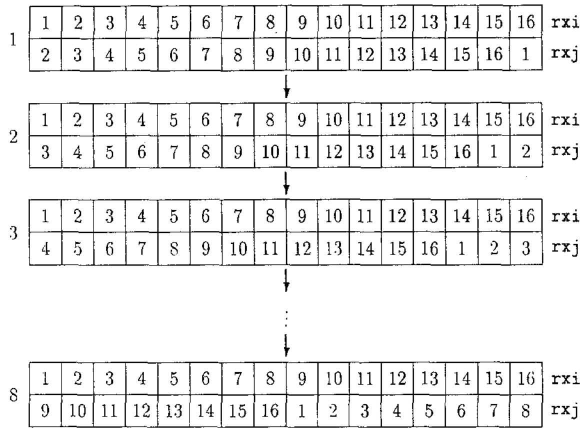
1
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 rxi
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 1 rxj
2
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 rxi
3 4 5 6 7 8 9 10 11 12 13 14 15 16 1 2 rxj
3
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 rxi
4 5 6 7 8 9 10 11 12 13 14 15 16 1 2 3 rxj
8
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 rxi
9 10 11 12 13 14 15 16 1 2 3 4 5 6 7 8 rxj

Fig. 6. The Brode-Ahlrichs algorithm for a 16-processor machine, proceeding through 1, 2, 3, . . . , 8 steps. Note the duplication in the last step

N is odd; for N even (as shown in the figure) the last operation computes each distinct interaction twice. This must be handled using a logical mask, or by dividing the results of this step by two.

One useful variant for potentials with a cutoff is to sort the coordinates in one direction, and cut short the shifting procedure as soon as the interactions have moved out of range. This is basically the Boris method again, but applied to one dimension only; it has proved useful for systems in which the simulation box is much longer in one direction than in the others.

5. Conclusions

In this paper, I have tried to review the simplest techniques used to simulate condensed phases on the DAP at Bristol. A common feature of these techniques is that they are not new: they are often derived from well-known scalar and vector algorithms, and indeed our massively parallel, fine-grained, SIMD machine has much in common with a vector processor. The most useful techniques make good use of geometrical parallelism, and for three-dimensional problems a machine with good three-dimensional connectivity would be ideal. However, a two-dimensional machine is satisfactory, for a wide range of situations. The single feature of this work for which higher connectivity might be useful is that occasionally we wish to carry out a global sum (to calculate the total energy, for instance); however the DAP provides efficient ways of doing this, and in any case the ratio of processor power to communications bandwidth is sufficiently low that communications is rarely a limiting factor in our work.

To be sure, I have not touched on some of the most tricky problems facing computer simulators on parallel architectures. For very inhomogeneous systems (for example, exhibiting two-phase coexistence, self-assembling systems like micelles, or rough surfaces) the regular geometrical decompositions described above will become quite inefficient. It would be worthwhile to devise a fast adaptive scheme, allowing variable cell size and shape, and to implement this on a parallel computer, to handle such highly inhomogeneous systems. Another point is that we have not faced the 'scaling problem' because the systems we wish to study on the DAP are quite large. We must remember that we are in the business of choosing the best tools to do the job at hand, not trying to fit the task to the machine. For the problems I have described above, the DAP is very suitable. We also simulate smaller systems, for quite long periods of time; I have not described this work, as it is not so suitable for the DAP. We are fortunate that chip technology has advanced sufficiently rapidly that single-processor machines and some vector processors are still adequate in many cases for this work. Nonetheless, as we become more ambitious, and start to reach the limits of single-processor machines in this field, we will have to face this challenge.

Acknowledgements. Thanks are due to my colleagues at Bristol, Dave Nicolaides, Bob Evans, Balazs Gyorffy, and many others, for their support and advice, and to David Fincham and Bill Smith (Daresbury Laboratory) for helpful discussions. The DAP was provided by the Science and Engineering Research Council.

References

1. Allen MP, Guest MF (1991) Scientific achievements of the computational science initiative. SERC Report (June 1991)
2. Gay JG, Berne BJ (1981) J Chem Phys 74:3316
3. Allen MP, Frenkel D, Talbot J (1989) Comput Phys Rep 9:301
4. Heerman D (these proceedings)
5. Friedberg R, Cameron JE (1970) J Chem Phys 52:6049
6. Binder K (1986) Monte Carlo methods in statistical physics. Topics in Current Physics 7, 2nd ed; (1987) Applications of the Monte Carlo method in statistical physics. Topics in Current Physics 36, 2nd ed
7. For an introduction to this subject see Charvolin J, Joanny JF, Zinn-Justin J (eds) (1989) Liquids at interfaces. Les Houches, Session XLVIII. 1988 Elsevier, Amsterdam
8. Nicolaides DB, Evans R (1989) Phys Rev B 39:9336; ibid (1989) Phys Rev Lett 63:778
9. Sluckin TJ (1988) J Phys A: Math Gen 21:1415
10. Allen MP, Armitstead K (1989) J Phys A: Math Gen 22:3011
11. Taylor MB, Gyorffy BL, Walden CJ (1991) J Phys Cond Mat 3:1575
12. Allen MP (1989) Molecular Simulation 4:61
13. Telo da Gama MM, Tarazona P, Allen MP, Evans R (1990) Mol Phys 71:801
14. Cleaver DJ, Allen MP (1991) Phys Rev A 43:1918
15. Swendsen RH, Wang JS (1987) Phys Rev Lett 58:86
16. Niedermayer F (1988) Phys Rev Lett 61:2026
17. Dewar R, Harris CK (1987) J Phys A: Math Gen 20:985
18. Allen MP, O'Shea SF (1987) Molecular Simulation 1:47
19. Duane S, Kennedy AD, Pendleton BJ, Roweth D (1987) Phys Lett 195B:216
20. Nicolaides DB (1991) J Phys A24:L231
21. Bennett CH (1975) J Comp Phys 19:267
22. Batrouni G, Katz G, Kronfeld A, Lepage G, Svetitsky B, Wilson K (1985) Phys Rev D, 32:2736
23. Nicolaides DB (private communication)
24. Fincham D (1987) Molecular Simulation 1:1; Fincham D (1990) in: Catlow CRA, Parker SC, Alled MP (eds) Computer modelling of fluids polymers and solids. NATO ASI Series 293:269, Kluwer
25. Rapaport DC (1988) Computer Phys Rep 9:1; Rapaport DC (1990) in: Catlow CRA, Parker SC, Allen MP (eds) Computer modelling of fluids polymers and solids. NATO ASI Series 293:249, Kluwer; (1991) Comput Phys Commun 62:198; ibid 62:217
26. Smith W (1991) Comput Phys Commun 62:229
27. Quentrec B, Brot C (1975) J Comput Phys 13:430
28. Boris J (1986) J Comput Phys 66:1
29. Brode S, Ahlrichs R (1986) Comput Phys Commun 42:51