Molecular dynamics simulations on transputer networks

Abhijit T. Paithankar, B. L. Tembe and Dilip M. Ranade

In this article, the details of parallelization of the classical molecular dynamics simulations of molecular fluids on transputer networks are presented. These simulations for a system containing 64 water molecules are performed on several computers and the timings per molecular dynamics step on these computers as well as on a 4-transputer system are compared. It is shown that parallelization provides a useful alternative for performing molecular dynamics simulations.

MOLECULAR dynamics (MD) simulations are being used extensively in studying the thermodynamic and transport properties of classical fluids. The MD method can be readily used to study systems that involve complex and realistic interparticle interactions[1–3]. The method requires a large amount of data processing, especially for systems with a large number of polyatomic molecules. The calculations of the required averages implies that several hundreds of thousands of the members of an appropriate ensemble have to be generated iteratively. Thus, the simulations assume supercomputing proportions. The symmetry and uniformity of computations of forces on molecules in liquids and solutions make these computations ideally suited for parallel processing.

A major trend in computer architecture over the past decade has been the move from uniprocessor systems to systems with multiple processing elements. Since the cost of small processors has been decreasing and the cost of high-performance processors has continued to be high, it is likely that a system consisting of many small, cheap, low-performance processors is more cost-effective than a system with a single high-performance processor. Thus, parallel processing is becoming an easy and cost-effective solution to the requirements of high-performance computing. In the present article, we report the MD simulations performed on a multitransputer network. A system of 64 water molecules is used as a test case. The simulations are also extended to aqueous solutions. In the last few years, considerable efforts have been put in vectorizing and parallelizing the algorithms for molecular dynamics and Monte Carlo simulations[4, 5]. The results of these simulations not only provide an 'exact' solution to the problem of the motion of a many-particle system, but also provide useful test data against which the results of analytical theories can be tested.

The organization of this article is as follows. To begin with, we describe the methodology of molecular dynamics and the system used in our simulations. Parallelization of the MD algorithm for a system of transputers is discussed next, followed by a comparison of the MD performance on different computer systems.

Methodology and the model system

In the molecular dynamics method, a system containing a collection of N molecules is considered. Given the initial positions and velocities, the successive configurations and velocities are calculated as a function of time by solving the equations of motion for all the particles in the system[1–3, 6a–6c]. The volume, composition and the total energy of the system are fixed in the usual MD simulations. In the MD method, the equations of motion for an N-particle system are solved numerically by using a suitable algorithm. The total potential energy and the force on each particle are calculated by choosing a suitable form for the intermolecular pair potential. From the calculated forces on each of these particles, the positions and the velocities of the particles at successive time intervals are calculated.

One of the simplest and commonest algorithms for solving the classical equations of motion is the Verlet algorithm. This algorithm is used in the present simulations. In this algorithm, the new positions of the particles are obtained from a knowledge of the current and the previous positions and the total force on the particles in the current positions. In the case of the rigid models of polyatomic molecules, the bond lengths and the bond angles need to be kept fixed during the simulation. Hence, the Verlet algorithm is modified to accommodate the constrained dynamics[6a, 6b].

Since the MD method deals with a finite system of particles ($50 \leq N \leq 1000$) which is too small compared to the bulk systems studied in the laboratory, a suitable boundary condition is necessary if the properties calculated are to resemble the properties of the bulk systems. In the use of periodic boundary conditions (PBC), the N particles of the system are contained in a

The authors are in the Department of Chemistry, Indian Institute of Technology, Powai, Bombay 400 076, India

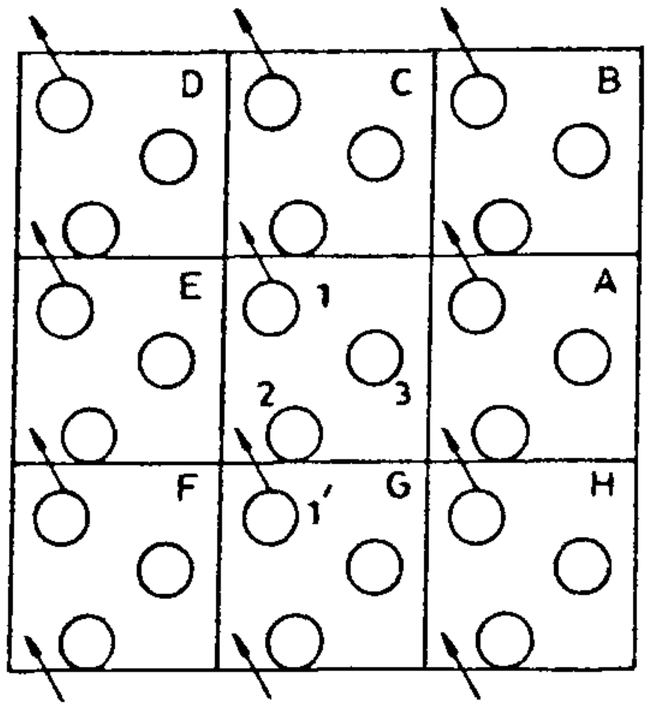

Figure 1. Periodic boundary conditions in two dimensions. The central cell is surrounded by identical cells on all sides. If the molecule 1 from the central cell moves to the cell C, the corresponding molecule 1′ enters the central cell from the cell G.

central box, but this box is supposed to be surrounded by an infinite array of identical boxes in all the three directions. Each box contains N particles with configurations identical to those of the particles in the central box. The way PBCs are imposed is shown in Figure 1. One actually follows the dynamics of the N particles in one unit cell (which is generally a cube). When a move carries a particle (labelled 1 in Figure 1) out through one face of the central unit cell, the move also carries the corresponding particle (labelled 1′ in Figure 1) into the central unit cell through the opposite face. Thus, the number of particles and, hence, the density and the composition of the fluid are conserved. Each particle thus finds itself immersed in an infinite medium. In calculating the interparticle interactions, the minimum image method is usually employed, in which only those particles whose interparticle distance is less than half the box length are considered for the computation of interaction forces and energies. If the difference in the x, y or z coordinates of two particles in the unit cell is greater than half the box length, the image of the particle in the adjacent cell whose corresponding coordinates are less than half the box length is considered.

The model of the liquid (water) used in the present MD simulations is referred to as the simple point charge (SPC) model of water[7]. Each H_2O molecule is a rigid triatomic containing three interaction sites. The oxygen atom of each molecule carries a charge of $-0.82e$ and both the hydrogens possess a charge of $0.41e$, where e is the magnitude of electronic charge. The O–H bond length is 1.0 Å and the H–O–H bond angle is 109.47°. The interactions between the particles are of two types: the electrostatic (Coulombic) interaction and the Lennard–Jones (LJ) interaction. The LJ forces act only between the oxygen atoms, while the electrostatic forces between all intermolecular pairs of charges are taken into account. The LJ potential is given by

$$U(r) = \left\{\frac{A}{r^{12}} - \frac{B}{r^6}\right\}, \tag{1}$$

where $A = 629{,}400$ kcal · Å^{12}/mol, $B = 625.5$ kcal · Å^6/mol and r is the inter-oxygen distance in angstroms.

The calculation of the new positions from the forces using the Verlet algorithm does not conserve the interatomic distances and the bond angles in the molecules. To maintain the correct distances within the molecules, the SHAKE algorithm is used. In this algorithm, the bond lengths are iteratively corrected (while conserving the overall momentum) until the errors in bond lengths are within a specified tolerance limit.

The SHAKE correction formula is

$$\delta \boldsymbol{r}_c\,(i) = (R^2 - R_0^2)/[2(\boldsymbol{R} \cdot \boldsymbol{R}) * m_1/\mu] * \boldsymbol{R}', \tag{2}$$

$$\boldsymbol{R}_{\text{new}}\,(i) = \boldsymbol{R}\,(i) + \delta \boldsymbol{r}_c\,(i), \tag{3}$$

where $\delta \boldsymbol{r}_c\,(i)$ is the correction applied to an atom i, $\boldsymbol{R}\,(i)$ is the position to be corrected, $\boldsymbol{R}_0$ is the constraint distance, R is the old (latest uncorrected) distance, m_1 is the mass of atom 1, μ is the reduced mass ($m_1 m_2/(m_1 + m_2)$) and the bold centre dot denotes the scalar (dot) product. A corresponding correction is applied to atom 2 of mass m_2. The velocities are calculated as follows:

$$\boldsymbol{V}(i) = [\boldsymbol{R}_{\text{new}}\,(i) - \boldsymbol{R}_{\text{cur}}\,(i)]\,/\mathrm{d}t. \tag{4}$$

Using these velocities, the kinetic energy of the particles is calculated. Using the algorithm, an MD trajectory of a suitable length (e.g. 10^5 to 10^6 time steps) is generated.

Parallel implementation of the MD simulation

The similarity in the computation of the forces on all the molecules makes the MD method ideal for a simulation using parallel processing. Indeed, one of the reasons why the MD method has gained importance is the ease and efficiency of its implementation on modern supercomputers, many of which have parallel (multiprocessor) architectures. Since the computations for all molecules are the same, by equally distributing the system over many processors, one can ensure a balanced processing on each processor, thus reducing the idle time and also achieving a near-one ratio of speed-up per processor.

The INMOS transputer family is a range of system components each of which combines processing, memory and interconnect in a single VLSI chip. The transputers can be used as building blocks for concurrent processing systems. Parallel systems have

also been designed using other processors such as INTEL's i860s.

A process is an instance of a program that is executing on a processor. A parallel program may consist of several processes executing concurrently on the processors of the system. Each process can communicate with one or more of the other processors by exchanging data over interconnecting links. The processing elements together with the links form the parallel processing system. Topology refers to the structure of such a system. For instance, processors may be interconnected to form a ring, a mesh, a tree or some other interconnection scheme. The communication requirements of a particular parallelization scheme determine the topology of the system. While the present MD simulations use the ring topology, instances of applications using other schemes are: matrix multiplication (two-dimensional mesh topology) and coding/sorting (tree topology). Some real-time systems use maximally interconnected structures, although this is limited by the number of hardware links available at each node. The transputer has four nodes. In an MD simulation, each particle interacts with every other particle in the system. When the collection of molecules is partitioned over several transputers, the coordinates of each molecule have to be communicated to every other transputer for the calculation of forces. In a ring topology this transfer is transparent and allows different transputers to share the processing load equally. In a tree topology the bridging transputers have to do additional processing, which not only increases the idle time but also makes the algorithm quite complex. Hence, the ring topology has been a common choice for doing MD simulations.

Ring topology refers to transputers linked in a closed ring of clockwise and anticlockwise channels for interprocessor communication. The entire set of molecules is distributed equally among the N processes running in parallel. Each process calculates the forces on its set of molecules and, after collecting the information from all the other processors, determines the new positions and velocities for each simulation step.

For calculating the total force on each atom, the corresponding process needs to access the coordinates of the atoms on all the other processes. Since shared memory between parallel processes is not possible, the coordinates of all the atoms on each process are communicated to all the other processes. Similarly, the resulting values need to be added after each cycle. Hence, the partial sums in each process are also communicated to the root processor, which also communicates with the host system for input/output. Figure 2 represents the way the four processors are linked together using ring topology. The clockwise channels are used for communicating the coordinates,

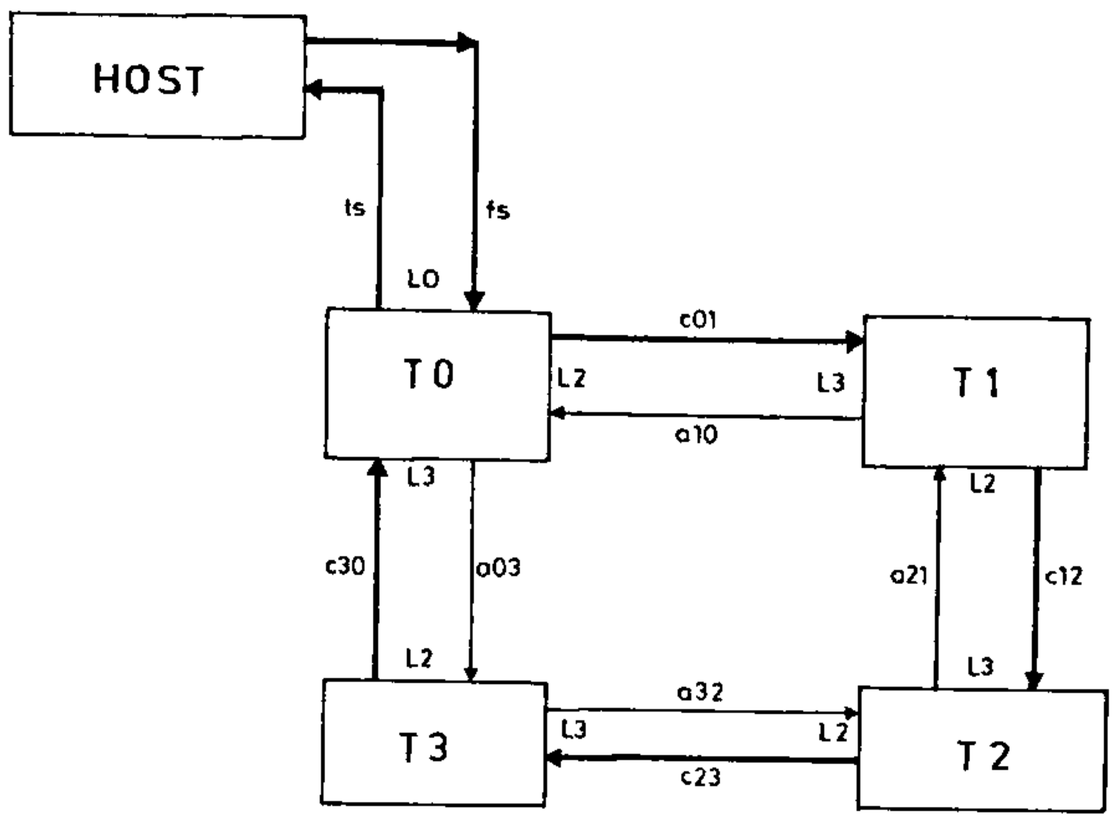

Figure 2. Link diagram showing the ring topology of the transputer network T0, T1, T2 and T3. The transputer T0 communicates with the host computer for input/output. The labels cij and aji are the clockwise and the anticlockwise communication channels between the transputers i and j The hardware links are shown as L2 and L3

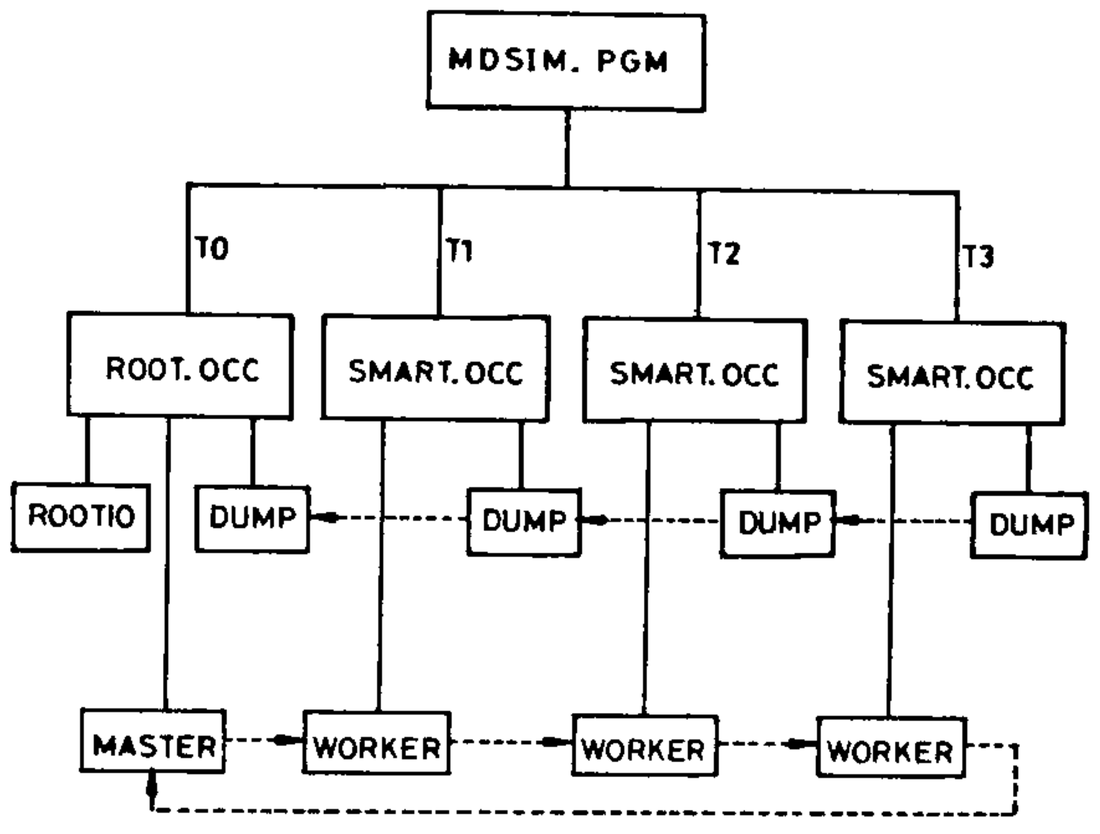

Figure 3. The processes running on different transputers and the communications between them The program MDSIM PGM configures the processes ROOT (ROOTIO, MASTER and DUMP) and SMART (WORKER and DUMP) on the four transputers.

while the anticlockwise channels are used for communicating the configurations and other averaged quantities for storage and further analysis.

There is one monitoring process on the root transputer which handles the host input/output and begins every simulation step by sending a signal to the computing processes. This is the ROOTIO process. The computing processes are the MASTER process on the root processor, which heads the ring, and the WORKER processes on the other processors in the ring. The MASTER and WORKER processes perform all the calculations. The clockwise channels are used for communication of coordinates, while the anticlockwise

channels are used for relaying the results by the DUMP process, which runs in parallel with the computing routines. The communication scheme between the transputers is shown in Figure 2.

A brief description of each routine is presented below. The placement of the various processes and the communications between them are illustrated in Figure 3.

(a) **ROOTIO.** The routine ROOTIO resides on the root processor. This communicates with the host as well as the MASTER and DUMP of the root processor, which run in parallel with ROOTIO. It carries out the following functions:

(i) It reads in the initial configuration and the other relevant parameters, and distributes the coordinates over the four processors.
(ii) It initiates each simulation cycle by sending the 'Calculate' signal to the MASTER, which then propagates it over the ring.
(iii) It accepts the summed up values sent by DUMP, processes them and outputs the results. (These include the potential energy, the kinetic energy, the temperature, etc.)

(b) **MASTER.** This is the computing routine on the ROOT processor. It communicates with the corresponding computing routines (i.e. WORKERs) on the adjacent processors, and with the ROOTIO and the DUMP routines on the local (ROOT) processor. It carries out the following functions:

(i) It receives and distributes over the ring the sets of coordinates read in at the beginning of the simulation.
(ii) On receiving the 'Calculate' signal from the ROOTIO at the beginning of every cycle, it sends it ahead to the next processor in the ring and computes the local pairs of interactions. It then forms an image (copy) of the local coordinates and sends it to the next WORKER, while receiving the image from the previous WORKER. The computations between pairs of atoms on different processors are performed using this image. The image is then sent on to the next processor while another image is received from the previous processor. This cycle is repeated until the images of all the processors in the ring are processed, and the image of the local set has returned.
(iii) The forces accumulated by the image are then added to the corresponding forces accumulated locally to get the total force on each atom. This is done because the pairs processed by the images and those processed locally are disjoint sets. This way of partitioning ensures a balanced amount of processing over the image cycles on the four processors, thereby maximizing the efficiency.
(iv) After the forces are merged, the new coordinates are calculated using the VERLET algorithm. The intramolecular constraints are applied using the SHAKE algorithm. The computations are performed using the periodic boundary conditions.

(c) **DUMP.** This routine communicates with the WORKER/MASTER and the DUMPs of the adjacent processors. The DUMP routine on the root processor communicates with ROOTIO. The DUMP routine uses the anticlockwise channels to communicate the dump data to the previous processor, so that it finally reaches the ROOTIO.

(d) **WORKER.** The tasks performed by the WORKER and the MASTER are almost identical. The calculations of the forces and the new coordinates are identical in the MASTER and all the WORKER processes. The main difference is that the MASTER process communicates with the ROOTIO process, which is resident only on T0 (or the root) transputer, and also with two adjacent WORKERs, namely WORKERs 1 and 3. WORKER 1 (on T1) communicates with the MASTER and WORKER 2. WORKER 2 (on T2) communicates with WORKERs' 1 and 3. WORKER 3 (on T3) communicates with WORKER 2 and the MASTER. The MASTER also distributes the initial configuration over all the processors in the ring.

Since all pairs of particles need to be considered in computing the forces on molecules, the positions of all the particles in the system must be communicated to all the processors. In the ring topology used in the present work, a set of particles residing on one processor can visit all the P processors in P transfers along the ring. Also, all the particles have to visit all the processors in as much time if the communication is performed in parallel. The program uses the method of image transfers to communicate the coordinates to the other processors. For the molecules on a given processor (referred to as the *local molecules*), all the other molecules are split into two equal disjoint parts. The interactions of the molecules of one of these sets is calculated by the source processor when the images visit it, while the interactions of the other disjoint set are calculated when the image of this local set visits the other corresponding processors. This ensures that an equal amount of computing is performed after every image rotation. In Table 1, the scheme used to achieve this is given. The communication scheme is also depicted in Figure 4. The communication of coordinates could require a large amount of time because of the necessary process synchronization. However, if the number of molecules per processor (N/P, where P is the number of processors) is large, then this communication

Table 1. The scheme employed for image transfer and the processing carried out in each computational cycle

Cycle no	Images on transputers				Interactions computed on			
	T0	T1	T2	T3	T0	T1	T2	T3
1	S1	S2	S3	S4	S1–S1	S2–S2	S3–S3	S4–S4
2	S4	S1	S2	S3	$S1_A$–S4	S2–$S1_B$	S3–$S2_B$	S4–$S3_B$
3	S3	S4	S1	S2	$S1_A$–S3	$S2_A$–S4	S3–$S1_B$	S4–$S2_B$
4	S2	S3	S4	S1	$S1_A$–S2	$S2_A$–S3	$S3_A$–S4	S4–$S1_B$
5	S1	S2	S3	S4	S1–S1	S2–S2	S3–S3	S4–S4

In this table, S1, S2, S3 and S4 refer to the four sets of molecules (each containing $N/4$ molecules) placed initially on the transputers T0, T1, T2 and T3, respectively. After each cycle, the images residing on each transputer/processor are rotated clockwise on the four transputers. In cycle 1, the image of S1 resides on T0, while in cycle 2, the image of S4 resides on T0 In cycle 1, the interactions between all the molecules on each transputer are computed and these interactions are represented by Sn–Sn, where $n = 1, 2, 3, 4$ In cycle 2, the first half of the molecules of S1 (i e. $S1_A$) interact with all the molecules of the images of S4 on transputer T0. The first half of the molecules is denoted by the subscript A, while the second half is denoted by the subscript B. On the transputer T2, in the second cycle, the second half of S2 interacts with all of S3. In each cycle, the same amount of computations are performed on each transputer After four cycles, the interaction between each molecule with every other molecule has been calculated The fifth cycle is a repetition of the first, the ith step is a repetition of the $(i-4)$th, albeit with new values of the coordinates of the molecules

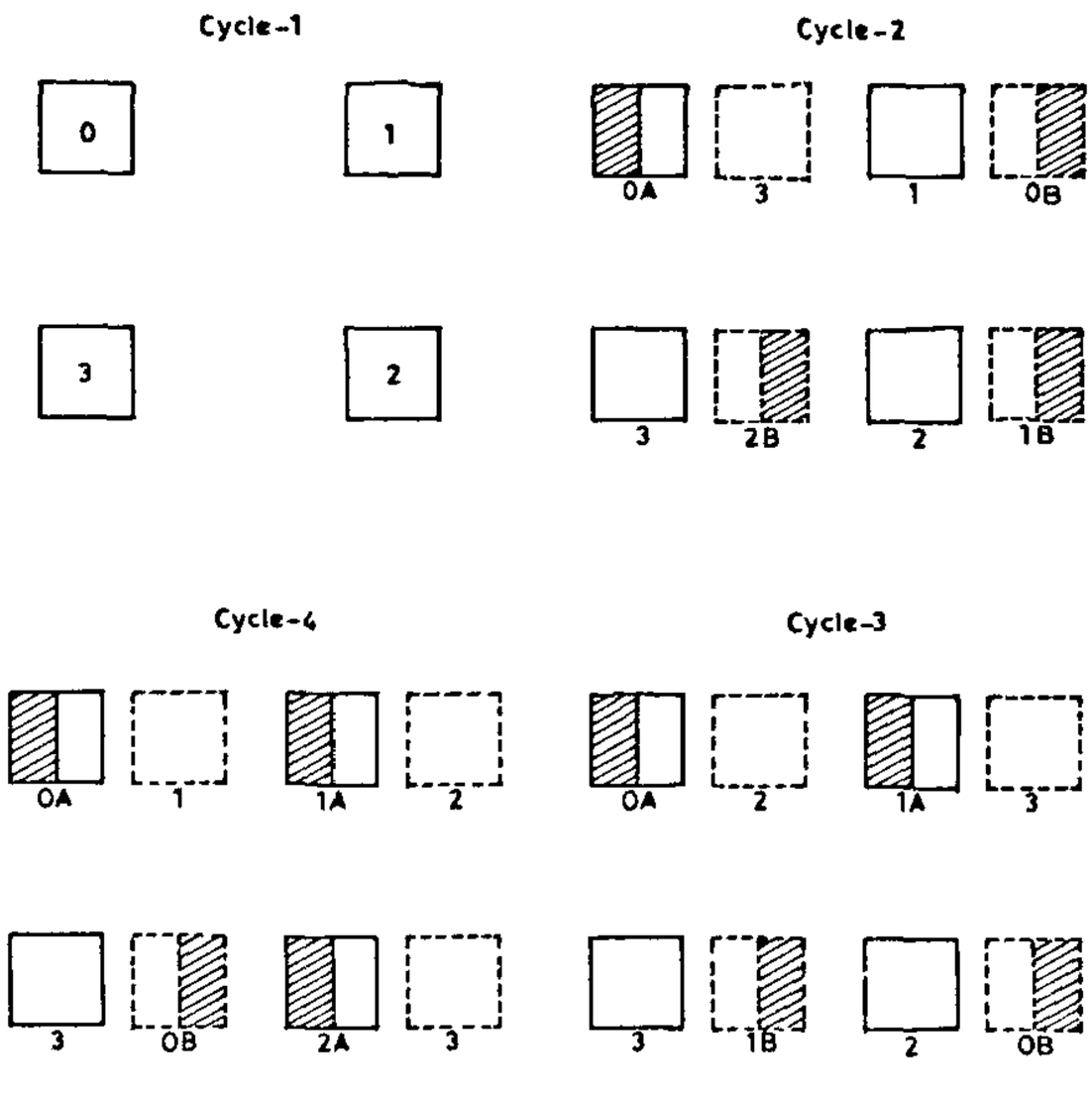

Figure 4. The communication scheme used in the four cycles that constitutes one MD step. The boxes drawn with thick lines refer to the molecules resident on the individual transputers and the boxes drawn with dashed lines refer to the visiting images In cycle 1, the interactions among all the particles on the respective transputers T0, T1, T2 and T3 are calculated The shaded regions refer to half the number of particles on the processors In cycle 2, the first half (labelled A) of the molecules on T0 interact with all the images of the molecules of T3, all the molecules of T1 interact with the second half (labelled B) of the images of the molecules of T0, and so on

Table 2. Time required on four different systems for a single cycle of simulation

System	Time taken	Mflops extracted
CDC CYBER 180/840	0 8 s (CPU sec)	0 87
Magnum MultiRisc	0 315 s (CPU sec)	2 24
386 PC	8 0 s	0 087
4-Transputer Superfour Board	0 406 s	1 74

time is small compared to the time taken for calculating the pair interactions. Hence, the communication overhead does not amount to a significant proportion of the processing time. Therefore, it is advantageous to use a multiprocessor system for a large enough N/P so that a good speed-up is ensured.

Performance of the parallel algorithm

The program written in OCCAM 2 (ref. 8–14) was executed for several sample configurations and the results were compared with those from the sequential Fortran program written for the same MD simulation[15] and executed on some other computer systems. OCCAM 2 was used here since it is a language optimally suited for use on transputers. As can be seen from Table 2, the performance compares favourably with the mainframe systems. The simulation for a binary system containing two ions and 64 water molecules requires only 1.5% additional time. Writing the code in other languages and extending it to other molecules is straightforward, and the speed-up achieved will not depend very much on the language used for writing the program.

Conclusions

A major application of classical molecular dynamics of molecular liquids is the calculation of structure factors (from the pair correlation functions) and equilibrium thermodynamic properties such as energy, free energy and pressures, and, finally, an equation of state valid over a wide enough range of temperature and density. Properties of fluids and fluid mixtures under abnormal conditions have also been estimated by using molecular dynamics. Another major application of molecular dynamics includes calculation of the various time correlation functions. Using these functions, relaxation times, and transport properties such as diffusion constant, ionic mobilities, viscosities and so on can be estimated. The algorithms for molecular dynamics can be parallelized quite readily and a ring topology of processors appears to be most suited for an effective calculation of the forces on each molecule of the fluid. Several other applications have also shown the advantages of using parallelization in computational chemistry[17,18].

It is observed in the present work that a near-linear speed-up results for each processor that is added. Results from other sources[5] indicate that beyond about 20 processors, the communication overhead assumes large proportions, making further parallelization not very efficient. A recent work[19] shows that incorporating an i860 processor along with transputer networks can further help in increasing the computation speed, thereby enabling us to approach a speed of over 100 Mflops, which is currently available on most of the supercomputers today. The methodology adopted in the present study can be readily extended to processors other than transputers.

1 Allen, M. P. and Tildesley, D J, *Computer Simulations of Liquids*, Oxford University Press, 1987.
2. McCammon, J. A. and Harvey, S. C., *Dynamics of Proteins and Nucleic Acids*, Cambridge University Press, 1987.
3 Karplus, M and Petsko, G. A., *Nature*, 1990, **347**, 631–639.
4 Teleman, O. and Jonsson, B., *J Comput Chem*, 1986, **7**, 58–66, Janak, J. F. and Patnaik, P. C, *J. Comput Chem*, 1992, **13**, 1098–1108
5 Pritchard. D. J and Scott, C. J. (eds), *Applications of Transputers 2, Proc. 2nd Intern Conf on Applications of Transputers*, IOS Press, Amsterdam, 1990. See the following three articles in particular 'Monte Carlo simulations of biomolecular systems using transputer arrays' by Jones, D. M and Goodfellow, J M.; 'Transputer molecular dynamics with electrostatic forces' by Miller, S, Fincham, D, Jackson, R A and Mitchell, P. J.; and 'Molecular dynamics simulation of proteins on an array of transputers' by Raine, A. R
6 *a* Verlet, L., *Phys Rev*, 1967, **159**, 98–105, *b* Ryckaert, J P., Ciccotti, G and Berendsen, H. J C, *J. Comput. Phys*, 1977, **23**, 327–341; *c*. Berendsen, H. J C., Postma, J P M, van Gunsteren, W. F. and Hermans, J, *Intermolecular Forces* (ed Pullman, B), Reidel, Dordrecht, 1981.
7. Jorgensen, W L, Chandrasekhar, J, Madura, D, Impey, R W and Klein, M. L., *J Chem Phys*, 1983, 79, 926–935.
8 Pountain, D. and May, D, *A Tutorial Introduction to Occam Programming*, Hollen St. Press, 1987.
9 The Occam Reference Manual, INMOS Limited
10. The Occam 2 Toolset User's Manual, INMOS Limited
11 Carlyle, A, *Transputers and Occam*, Prentice-Hall, 1987
12. Jones, G, *Programming in Occam 2*, Prentice-Hall, 1988
13 Fox, G. C. *et al*, *Solving Problems on Concurrent Processors*, Chap. 9, Prentice-Hall, 1988, vol. 1
14. Matrorana, V. Migliore, M and Fornili, S L., in *Parallel Programming of Transputer-Based Machines*, Proceeding #7, Occam User Group (ed Muntean, T), IOS Press, Amsterdam, 1988, pp 128–134
15. Vijaykumar, P. and Tembe, B L., *J Chem Phys*, 1992, **97**, 4356–4367.
16 DST Workshop on Parallel Processing, 5–9 March 1993, conducted in the Department of Chemistry, University of Poona, India.
17. Limaye, A C. and Gadre, S. R, *J. Chem. Phys*, 1994, **100**, 1303.
18 Bhusari, A D, Bhate, V. V and Pal, S, *Curr Sci*, 1992, **62**, 293
19 Shirsat, R., Limaye, A. and Gadre, S R., *J. Comput Chem*, 1993, **14**, 445.

ACKNOWLEDGEMENTS. Paithankar and Ranade would like to thank CDAC for financial and computational support We would like to thank Gr Capt. M. M Sharma, Prof S. R Gadre and V Avaghade for their encouragement We would also like to thank the members of the Application Group of CDAC and Prof. S S S P. Rao and K. V. Ramani of the Computer Science Department of I I T. Bombay for their kind cooperation during the progress of this work.

Received 17 January 1994, revised accepted 8 July 1994

RESEARCH ARTICLES

A hypothesis for the origin of peninsular seismicity

K. N. Khattri

Wadia Institute of Himalayan Geology, Dehradun 248 001, India

The seismic activity of the peninsular India has been examined. The seismicity is considerably feeble compared to that of the plate collision boundary (i.e. the Himalayas) towards the northern parts of India. However, several linear seismic zones have been identified. One of these runs along the NW direction and divides the peninsula into two large blocks. Most of the seismic zones are trending along one of the two directions, NW or NE. These directions are consistent with the conjugate system of faults expected to develop in an approximately NS compressive-strain regime. The collision of the Indian plate provides just such a strain environment. Due to the resistance to subduction of the Indian plate, a part of the strain due to plate convergence is released in the peninsula by tectonic escape of the peninsular blocks along the NW-trending faults towards either the NW or the SE direction, giving rise to many of the seismic zones.

THE feebleness of seismicity of the peninsular India in contrast to the high seismicity of the Himalayas had induced a false paradigm of stability and safety from seismic hazard of the former. Lately, the 1967 Koyna and the 1993 Killari (Maharashtra) earthquakes have shattered this paradigm and nudged the earth scientists