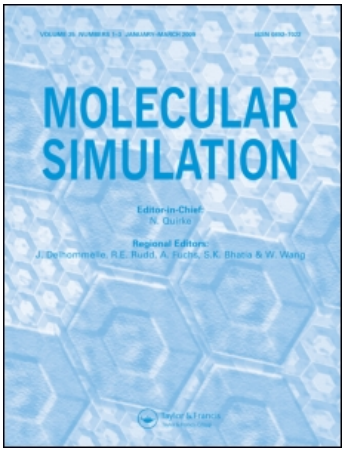

Molecular Simulation
Publication details, including instructions for authors and subscription information:
http://www.informaworld.com/smpp/title~content=t713644482

Computer Simulation of Liquid Methanol I. Molecular Dynamics on a Supernode Transputer Array

J. Casulleras[a]; E. Guardia[a]
[a] Departament de Fisica i Enginyeria Nuclear, Universitat Politècnica de Catalunya, Barcelona, Spain

To cite this Article Casulleras, J. and Guardia, E.(1991) 'Computer Simulation of Liquid Methanol I. Molecular Dynamics on a Supernode Transputer Array', Molecular Simulation, 7: 3, 155 — 169
To link to this Article: DOI: 10.1080/08927029108022150
URL: http://dx.doi.org/10.1080/08927029108022150

Molecular Simulation, 1991, Vol. 7, pp. 155–169
Reprints available directly from the publisher
Photocopying permitted by license only

COMPUTER SIMULATION OF LIQUID METHANOL I. MOLECULAR DYNAMICS ON A SUPERNODE TRANSPUTER ARRAY

J. CASULLERAS and E. GUARDIA

Departament de Física i Enginyeria Nuclear, Universitat Politècnica de Catalunya, Pau Gargallo 5, E-08028 Barcelona, Spain

(*Received October 1990, accepted November 1990*)

A parallel Molecular Dynamics program for liquid methanol has been implemented on a supernode containing sixteen T800 Transputers. The long range forces are treated by the Ewald summation method. A very high efficiency has been achieved for system sizes ranging from 125 to 512 molecules by exploiting both the possibilities of these parallel processors and the complexity of the simulated system.

KEY WORDS: Parallel computers, molecular dynamics, methanol, Ewald summation.

INTRODUCTION

Computer simulations of statistical mechanical systems constitute a broad field of research in which it is of clear interest to exploit new parallel programming techniques and computers [1–3]. Parallel computers offer the possibility of performing simulations of hitherto impossible size/time scale. At another level, they allow a small research group to easily handle simulations that would otherwise require the use of a remote supercomputer. Transputer-based networks appear to be specially suitable because of their great versatility and low cost. In the last few years, parallel programs to perform lattice [4], molecular dynamics [5,6] and molecular design simulations [7] have been implemented on transputer arrays.

In this paper we present a parallel algorithm for molecular dynamics (MD) simulations of polar liquids, handling the long ranged forces by means of Ewald summation. The parallel MD program was written in Occam 2 [8], debugged and run using the INMOS IMSD700D Transputer Development System (TDS) for IBM-PC/AT/XT [9]. The calculations were carried out on a PARSYS Supernode made up of sixteen T800 transputers [10], each one having a 64-bit floating point unit, an interface to a 4 Mbyte dynamic memory and four 20 Mbit/sec bidirectional links for the communications between them and with the IBM-PC/AT host computer. The simulated system was liquid methanol at normal room temperature conditions. Four runs were performed, with the number of molecules in the simulation box equal to 125, 216, 343, and 512, respectively, allowing us to analyse the efficiency of our parallel algorithm as a function of the size of the system. The influence of the system size on the calculated structural, orientational and dynamic properties is presented in a coming paper [11]. We are presently performing very long simulations (more than 500 ps) to study dielectric relaxation in liquid methanol and the results will soon be reported elsewhere.

DETAILS OF THE MOLECULAR DYNAMICS SIMULATIONS

The liquid methanol MD simulations were carried out at approximate room temperature conditions (T = 298 K and ρ = 0.01478 molecules /Å^3). The system was made up of N molecules in a cubic box with periodic boundary conditions. Four simulations were made with N = 125, 216, 343, and 512, respectively. The OPLS potential due to Jorgensen [12] was used for the methanol-methanol interactions. In this model, the rigid methanol monomer is represented by three interaction sites, the methyl group centered on carbon (C) and the oxygen (O) and hydroxyl hydrogen (H_0) atoms. The bond lengths and angles are r(C–O) = 1.430 Å, r(O–H_0) = 0.945 Å and $<COH_0$ = 108.5 deg. The charge assignments yield a dipole moment μ = 2.22 D. The interaction energy between molecules a and b (U_{ab}) is given by

$$U_{ab} = \sum_{i}^{\text{on } a} \sum_{j}^{\text{on } b} \left[\frac{q_i^a q_j^b}{r_{ij}} + \frac{A_{ij}}{r_{ij}^{12}} - \frac{C_{ij}}{r_{ij}^{6}} \right] \tag{1}$$

Combining rules are used such that $A_{ij} = (A_{ii}A_{jj})^{1/2}$ and $C_{ij} = (C_{ii}C_{jj})^{1/2}$. The A and C parameters may be expressed in terms of the Lennard Jones σ's and ε's as $A_{ii} = 4\,\varepsilon_i\sigma_i^{12}$ and $C_{ii} = 4\,\varepsilon_i\sigma_i^{6}$. The site charges and Lennard-Jones parameters are listed in Table 1.

To perform the simulations we employed the leap-frog Verlet integration algorithm proposed by Berendsen *et al.* [13] with a time-step Δt = 0.0025 ps and a velocity scaling time constant τ = 0.05 ps. The SHAKE procedure [14] was used to keep the interatomic distances fixed. The long range coulombic interactions were treated by the Ewald summation method. With this assumption, the total energy (U) of the N molecules in the cubic box is given by [15]

$$\begin{aligned} U = {} & \frac{1}{2} \sum_{a,b=1}^{N} \sum_{i,j=1}^{3} \left[q_i^a q_j^b \frac{erfc\,(\alpha\, r_{ij})}{r_{ij}} + \frac{A_{ij}}{r_{ij}^{12}} - \frac{C_{ij}}{r_{ij}^{6}} \right] \\ & - \frac{1}{2} \sum_{a=1}^{N} \sum_{\substack{i,j=1 \\ (i \neq j)}}^{3} q_i^a q_j^a \frac{erf\,(\alpha r_{ij})}{r_{ij}} \\ & + \frac{1}{2\pi L} \sum_{\mathbf{h} \neq 0} \frac{\exp\left(- \frac{\pi^2 |\mathbf{h}|^2}{\alpha^2 L^2} \right)}{|\mathbf{h}|^2} \left\{ \left[\sum_{a=1}^{N} \sum_{i=1}^{3} q_i^a \cos(2\pi/L\, \mathbf{h} \cdot \mathbf{r}_i) \right]^2 \right. \\ & \left. + \left[\sum_{a=1}^{N} \sum_{i=1}^{3} q_i^a \sin(2\pi/\mathrm{L}\, \mathbf{h} \cdot \mathbf{r}_i) \right]^2 \right\} \\ & - \frac{\alpha}{\sqrt{\pi}} \sum_{a=1}^{N} \sum_{i=1}^{3} (q_i^a)^2 + \frac{2\pi\, N\, \mu^2}{3\, L^3} \end{aligned} \tag{2}$$

We took $\alpha = 5.5/L$. The real space summation was truncated by means of a spherical

Table 1 Intermolecular potential parameters.

atom	*q (e)*	*ε (k_B)*	*σ (Å)*
C	+0.265	104.17	3.775
O	−0.700	85.55	3.071
H_0	+0.435	0.	0.

molecular based cut-off with a radius equal to $L/2$ and the summation in reciprocal space was restricted to lattice vectors $\mathbf{h}$ having $|\mathbf{h}| \leqslant |\mathbf{h}|_{max} = 5$ [11].

PARALLEL ALGORITHM DESIGN AND IMPLEMENTATION

General Considerations

A typical MD calculation contains, for each update of the system, the following "elementary" steps:

i) Evaluation of the forces acting on each particle.

ii) Calculation of the new positions and velocities by integrating the equations of motion.

iii) Calculation of physical properties, such as radial distribution functions or time correlation functions.

All these tasks have to be distributed among several processors. Ideally, the task distribution should be organized so that the time lost by any processor waiting for the results of another processor is minimized. Moreover, the distribution of tasks should be done in such a way that all the processors take the same time to perform their task (i.e., achieving an optimum load balance), since the total time to perform an entire cycle is determined by the longest of the individual times. Therefore, for large numbers of processors it may be acceptable that one of them takes less time than the rest, but the opposite situation (i.e., that one processor takes significantly more time to complete its task than the mean) must definitely be avoided.

The parallel MD program described in this work has been implemented on a computer consisting of sixteen T800 Transputers, each one of them being a "complete computer" in itself, which can be connected to any four of the remaining transputers. The topology we used is represented in Figure 1.

Prior to the description of our parallel algorithm, we wish to point out that we have not tried to reduce to the minimum the number of total communications between processors (which is not in itself a goal), but to keep them at a level that do not delay the computing tasks. As will be shown, this has been possible due to the capability of the transputers to simultaneously communicate between themselves while performing an independent calculation, as well as to the exploitation of the complexity of the simulated system.

Calculation of the forces

From equation (2) it follows that the total force $\mathbf{F}_i$ acting on a given particle can be written as the sum of two contributions,

$$\mathbf{F}_i = \mathbf{F}_i^{rel} + \mathbf{F}_i^{abs} \tag{3}$$

where $\mathbf{F}_i^{rel}$, the contribution from the real space summation, depends on the relative positions between particles, and $\mathbf{F}_i^{abs}$, the reciprocal space contribution, can be suitably expressed in terms of the absolute positions of all the particles in the system:

$$\mathbf{F}_i^{abs} = \frac{2\,q_i^a}{L^2} \sum_{\mathbf{h}\neq 0} \frac{\mathbf{h}}{|\mathbf{h}|^2} \exp\left(-\frac{\pi^2|\mathbf{h}|^2}{\alpha^2 L^2}\right) \{c(\mathbf{h}) \sin(2\pi/L\mathbf{h}\cdot\mathbf{r}_i) - s(\mathbf{h}) \cos(2\pi/L\,\mathbf{h}\cdot\mathbf{r}_i)\} \tag{4}$$

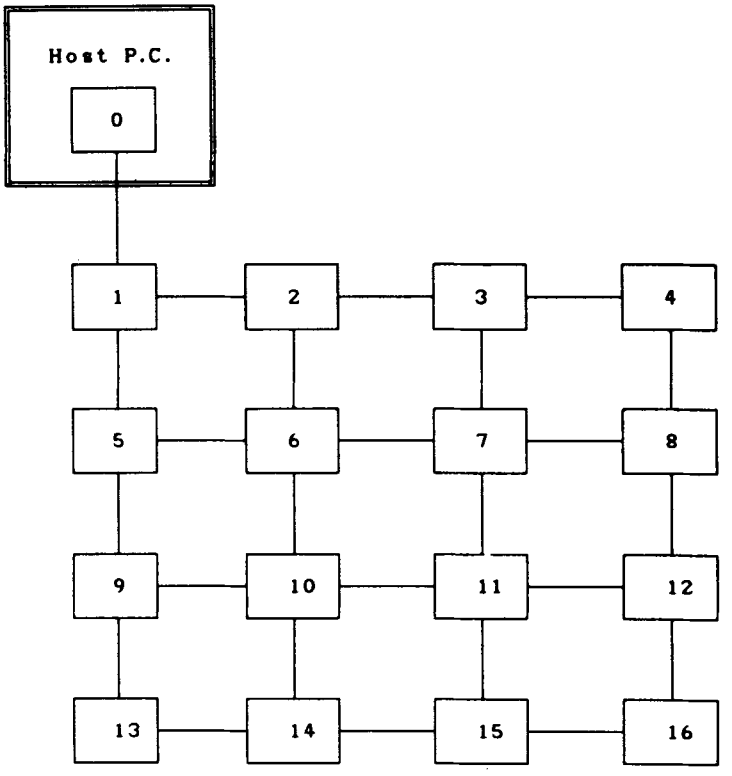

Figure 1 Topology of the transputer network. The lines —— represent communication links between the processors.

where

$$c(\mathbf{h}) = \sum_{a=1}^{N} \sum_{i=1}^{3} q_i^a \cos(2\pi/L\ \mathbf{h} \cdot \mathbf{r}_i) \tag{5}$$

and

$$s(\mathbf{h}) = \sum_{a=1}^{N} \sum_{i=1}^{3} q_i^a \sin(2\pi/L\ \mathbf{h} \cdot \mathbf{r}_i) \tag{6}$$

We have used different methods of parallelization in the calculation of $\mathbf{F}_i^{\text{rel}}$ and $\mathbf{F}_i^{\text{abs}}$, according to their different nature: the interaction distribution is suitable for $\mathbf{F}_i^{\text{rel}}$, while the particle distribution is more appropriate for $\mathbf{F}_i^{\text{abs}}$.

In our strategy each processor knows the position of all particles at the beginning of each time step. That allows to perform the calculation of the forces $\mathbf{F}_i^{\text{rel}}$ in a concurrent way by assigning 1/16 of the total number $N(N\text{-}1)/2$ of intermolecular interactions to each processor, which will do its job without having to ask anybody else for additional information. At the end of this calculation each processor P has a list of the forces. $\mathbf{F}_{i,P}^{\text{rel}}$ corresponding to its assigned pair interactions. All this lists

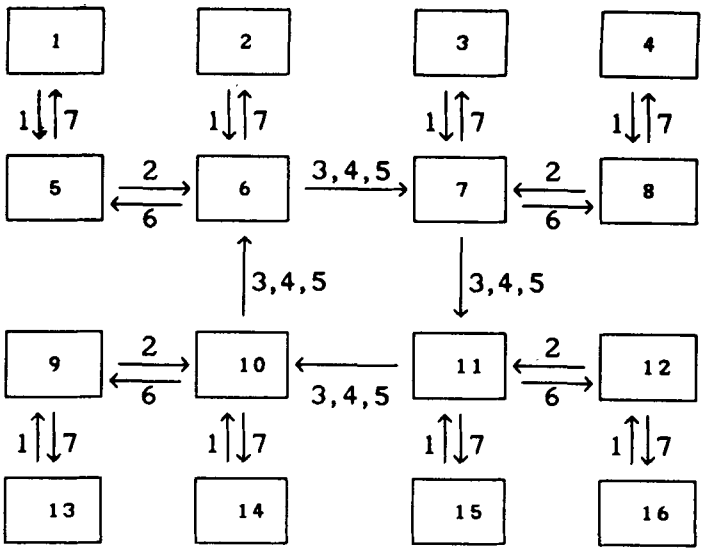

Figure 2 Routing of the $F_{i,P}^{\text{rel}}$, $c_P(\mathbf{h})$ and $s_P(\mathbf{h})$ values when performing the sums of equations (7), (8), and (9), respectively.

have to be added up in order to get the total forces $\mathbf{F}_i^{\text{rel}}$, i.e.,

$$\mathbf{F}_i^{\text{rel}} = \sum_{P=1}^{16} \mathbf{F}_{i,P}^{\text{rel}} \tag{7}$$

We have used a cascade method to perform this sum and to send it back to all the processors. In Figure 2 is shown the routing of the $F_{i,P}^{\text{rel}}$ values. They are first concentrated in the central four transputers (steps 1 and 2). In the following steps 3, 4, and 5 the total sum is performed on these central transputers. In the final steps 6 and 7, the F_i^{rel}'s are sent back to the remaining transputers. The whole process, which requires quite a bit of communication, can be performed simultaneously to the calculation of the reciprocal space forces $\mathbf{F}_i^{\text{abs}}$.

The evaluation of the $\mathbf{F}_i^{\text{abs}}$ has been divided in two steps, which have been parallelized by distributing the N molecules among the 16 processors. At the first step, each processor P determines the quantities $c_P(\mathbf{h})$ and $s_P(\mathbf{h})$, which involve a sum of the form (5) and (6), respectively, but only over the molecules assigned to it. These quantities may be added up by means of the same cascade method used for $\mathbf{F}_{i,P}^{\text{rel}}$ to obtain the (all-particle) common terms $c(\mathbf{h})$ and $s(\mathbf{h})$, i.e.,

$$c(\mathbf{h}) = \sum_{P=1}^{16} c_P(\mathbf{h}) \tag{8}$$

$$s(\mathbf{h}) = \sum_{P=1}^{16} s_P(\mathbf{h}) \tag{9}$$

At the second step, each processor calculates the $\mathbf{F}_i^{\text{abs}}$ corresponding to its assigned particles by using equation (4).

Calculation of the positions and velocities

As previously mentioned, the integration algorithm proposed by Berendsen *et al.* [13] has been used. This is a leap-frog Verlet algorithm that can be summarized as follows: given the positions $\mathbf{r}_i(t)$ and velocities $\mathbf{v}_i(t\text{-}\Delta t/2)$ for all particles, and the total kinetic energy $K(t\text{-}\Delta t/2)$,

1. Compute new velocities

$$\mathbf{v}_i(t + \Delta t/2) = \lambda\,(\mathbf{v}_i(t - \Delta t/2) + \mathbf{a}_i(t)\Delta t) \tag{10}$$

The scaling factor λ is given by

$$\lambda = \left\{1 + \frac{\Delta t}{\tau}\left(\frac{T^{\text{ref}}}{T(t - \Delta t/2)} - 1\right)\right\}^{1/2} \tag{11}$$

where T^{ref} is a reference temperature and τ is an adjustable parameter.

2. Compute new (unconstrained) positions

$$\mathbf{r}_i(t + \Delta t) = \mathbf{r}_i(t) + \mathbf{v}_i(t + \Delta t/2)\,\Delta t \tag{12}$$

3. Apply constraints to coordinates by means of the SHAKE procedure [14]
4. Compute constrained velocities

$$\mathbf{v}_i(t + \Delta t/2) = (\mathbf{r}_i(t + \Delta t) - \mathbf{r}_i(t))/\Delta t \tag{13}$$

All these steps have been straighforwardly parallelized using the same particle distribution as for the calculation of $\mathbf{F}_i^{\text{abs}}$.

Determination of physical properties

In order to have an easily modifiable algorithm, the determination of physical properties is usually assigned to the host processor. In our case, however, the available memory of the host transputer (2 Mbytes to which one has to subtract a substantial part devoted to the compiler, the configurer, ...) did not allow this approach for realistic values of N. As an alternative, since the total task of steps i) and ii) is (in general) not divisible by 16, we assigned to the first transputer a smaller part of these tasks than to the others, such that the additional task of step iii) could be entirely assigned to it.

Summary of the Algorithm

The algorithm is divided in the following seven tasks, which are schematically represented in Figure 3:

1. Each processor determines the $c_p(\mathbf{h})$ and $s_p(\mathbf{h})$ corresponding to its assigned molecules. The first processor, though, which has less molecules assigned to it, starts by determining all the properties of the system we are interested in.
2. The values of $c_p(\mathbf{h})$ and $s_p(\mathbf{h})$ are sent and added up. This process is executed concurrently with step 3.
3. Each processor performs the calculation of the pair interactions $\mathbf{F}_{i,P}^{\text{rel}}$ assigned to it. No communication is necessary as they all know the positions of all

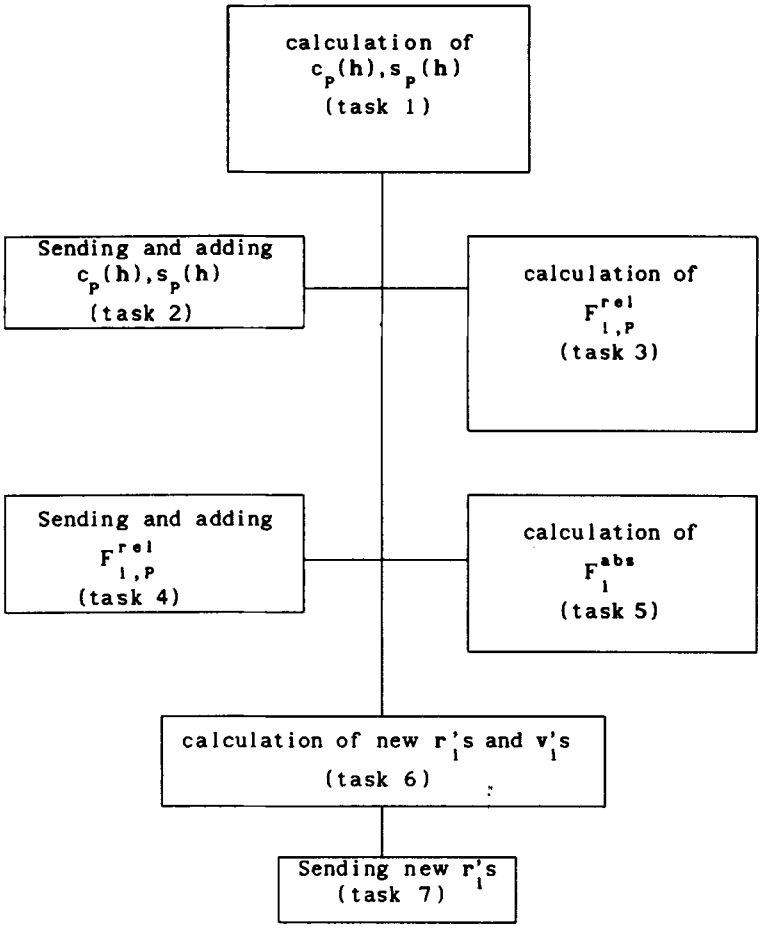

Figure 3 Schematic representation of the parallel algorithm described in this work. The vertical lines represent sequential executions and the horizontal lines represent executions done in parallel. The size of the boxes indicates the relative duration of each task.

particles. At the end of this process, as step 2 has already been terminated, steps 4 and 5 may start immediately.

4. The values of the forces $\mathbf{F}^{rel}_{i,P}$ acting on each particle are sent and added up. Again, this process will be executed at the same time as step 5.
5. Each processor calculates the $\mathbf{F}^{abs}_i$ forces corresponding to its assigned particles. Once this has been done and every processor has received (step 4) the pair interaction contribution $\mathbf{F}^{rel}_i$ for its assigned molecules, step 6 is ready to start.
6. Each processor calculates the new position $\mathbf{r}_i$, velocity $\mathbf{v}_i$ and kinetic energy of its assigned particles.
7. These new positions and kinetic energy circulate following a topological ring

through the network, so that after one round every processor knows the position of all particles as well as the total kinetic energy.

DISCUSSION OF THE EFFICIENCY

In order to give a precise idea of how all these tasks are running in the transputer array, we give in Table 2 (for the case $N = 343$) the duration in TDS time units (1 second $\approx$ 16000 TDS units) of the seven tasks in which a complete cycle has been decomposed. In this illustrative example we have made every processor run sequentially its seven tasks, in order to know how long it takes for the communication tasks to be executed.

In first place, one can remark that the calculation tasks 1, 5 and 6, respectively, take almost exactly the same time in all processors (but the first, as expected). However, the calculation of pair interactions (task 3) takes slightly different times, due to statistical fluctuations in the number of interacting pairs introduced by the spherical cut-off at $L/2$. In this context it is important to note that if any correlation exits between the molecule labels (which are related to the pairs that every processor will choose) and their position coordinates, such a correlation will cause different execution times in the different processors (in our first attempts, a couple of processors took an excess of 30% over the mean execution time). As such correlations tend to disappear very slowly, it is worthwhile to randomly numerate the molecules of the initial configuration. Even so, a further improvement of our algorithm should be the elimination of the statistical fluctuations, as their long duration makes it possible to dynamically adjust the list of pair assignments. That would lead to a saving of 2% in the execution time of task 3, which is the longest one. Although this is not a big effect here, it may deserve some attention for larger processor arrays, as the importance of the fluctuations would increase as the pair number per processor diminishes.

Let's now turn to the communication tasks 2, 4 and 7. The first one (concerning the

Table 2 Execution times in TDS units of the tasks assigned to each processor for $N = 343$ and $|\mathbf{h}|_{max} = 5$. All communication and calculation tasks run sequentially.

Processor	*Task 1*	*Task 2*	*Task 3*	*Task 4*	*Task 5*	*Task 6*	*Task 7*
1	20361	1606	91742	5540	6104	1328	483
2	32003	1606	92572	5539	10264	2268	482
3	31999	1612	92363	5525	10264	2262	483
4	32000	1606	93648	5539	10263	2223	483
5	31998	1606	94078	5539	10264	2251	482
6	32003	1606	92340	5539	10264	2252	482
7	32004	1612	92770	5525	10264	2241	483
8	32006	1606	91789	5539	10263	2251	483
9	31999	1607	91095	5554	10263	2234	483
10	32003	1614	91451	5568	10264	2241	483
11	32000	1614	92915	5568	10264	2246	483
12	32004	1615	92459	5569	10263	2230	484
13	32000	1600	90558	5554	10264	2240	483
14	32003	1614	91642	5568	10264	2212	483
15	32003	1614	92560	5568	10264	2251	483
16	32000	1615	92947	5569	10263	2252	484

communication of the values of $c_P(\mathbf{h})$ and $s_P(\mathbf{h})$) may run at the same time as the next calculation (task 3) is executed. Since this calculation is longer than the communication (see Table 2), by the time that the pair forces are evaluated each processor may immediately start with the following calculation, task 5, together with the transmission of the pair forces (task 4). Again, this communication will terminate before the computation (task 5), and thus by the end of the determination of the reciprocal forces each processor may begin to update the velocities and positions of its assigned particles (task 6). Once this is done, the positions of the particles of each processor must be known by all the others (task 7). Therefore, in practice, every transputer will execute without any dead time all the calculations corresponding to a time-step of the simulation, and they will have to wait only for the communication of the updated positions, which takes less than 0.5% of the total time. As a consequence, we are very close to the ideal situation where all the processors have equal loads (but the first, which plays a special role) and no delays are originated from the information traffic. It is remarkable that, in spite of the amount of communications required, the biggest (although tiny!) departure from this ideal picture does not come from this volume of communications, but from the fluctuations in the number of interacting pairs, which in our approach could be easily fixed since all the processors know the positions of all particles.

On the other hand, in a real execution (i.e., with the communication tasks 2 and 4 being performed in parallel with the calculation tasks 3 and 5, respectively), the execution times given in Table 3 are obtained. The comparison with Table 2 indicates that almost all the time spent in the communication tasks 2 and 4 is saved. Thus, calculations and communications are indeed performed simultaneously.

The relative weight of the different tasks depends on the size of the physical system. In Figure 4 is displayed the duration of each task as a function of N. One can note that, as expected, the Ewald summation (tasks 1 and 5) and the integration of the motion (task 6) grow as N, while the evaluation of the pair forces (task 3) behaves as N^2. On the other hand, one can see that almost all the communication time of tasks

Table 3 Execution times in TDS units of the tasks assigned to each processor for $N = 343$ and $|\mathbf{h}|_{max} = 5$. Communication tasks 2 and 4 are done in parallel with the calculation tasks 3 and 5 respectively.

Processor	*Task 1*	*Tasks 2 & 3*	*Tasks 4 & 5*	*Task 6*	*Task 7*
1	20361	91982	7189	1328	483
2	32002	92829	10306	2268	482
3	32090	92600	10306	2262	483
4	32000	93898	10318	2223	483
5	31998	94384	10612	2251	483
6	32004	92853	11265	2251	482
7	32004	93295	11273	2240	483
8	32006	92081	10622	2251	483
9	32000	91392	10617	2234	483
10	32004	91964	11271	2240	483
11	32000	93438	11287	2246	483
12	32003	92752	10628	2229	483
13	32000	90795	10312	2241	484
14	32002	91880	10313	2213	484
15	32003	92805	10317	2251	483
16	32001	93189	10318	2252	484

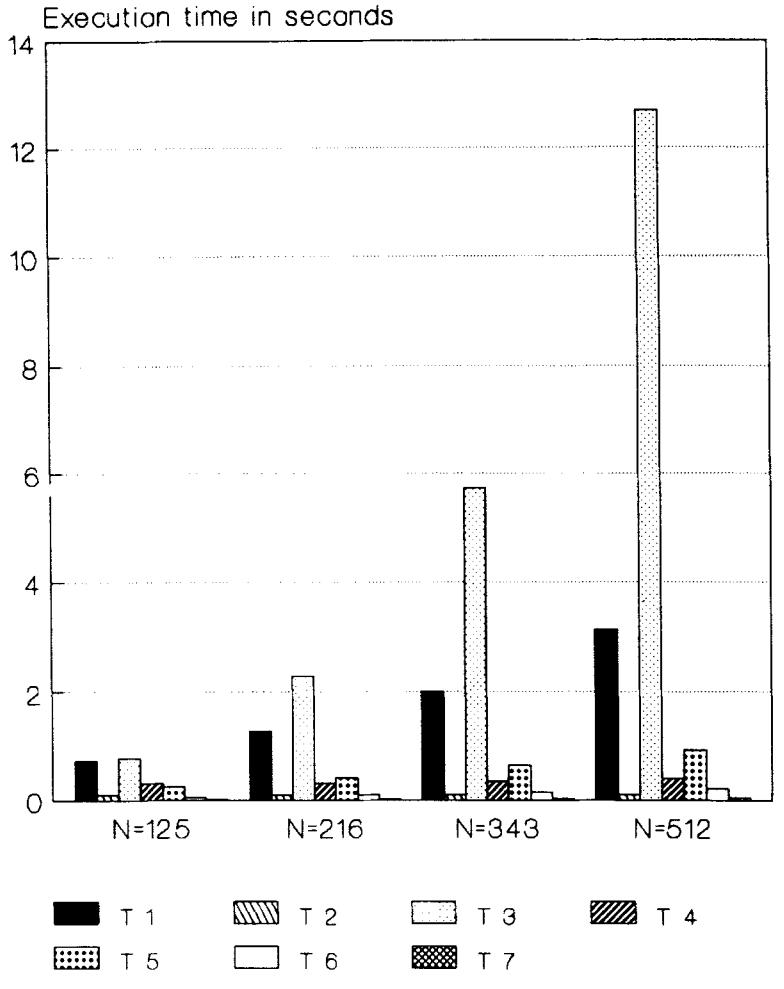

Figure 4 Execution times of the different tasks of the MD program described in this work, for $|\mathbf{h}|_{max} = 5$, as a function of N.

2 and 4 may be absorbed in the subsequent tasks 3 and 5. The final delay originated by the communication of the new particle positions (task 7) is hardly observable.

Finally, we turn to the global efficiency attained in our MD simulations. It can be calculated as

$$Eff \equiv \frac{T_1/T_n}{n} \tag{14}$$

where T_1 and T_n are the execution times in 1 and n = 16 processors, respectively. The values obtained from Tables 2 and 3 (and the corresponding ones for other values of N and $|\mathbf{h}|_{max}$) coincide up to 0.1% with the measured values when it has been possible to run the program on one single transputer (i.e., when its 4 Mbyte memory was enough to contain the code and data). The efficiency for the different cases is given

Table 4 parallel algorithm efficiency in % for different values of N and $|\mathbf{h}|_{max}$.

N / $\|\mathbf{h}\|_{max}$	*125*	*216*	*343*	*512*
4	82.8	91.9	93.8	97.1
5	90.6	95.4	96.6	97.4
6	95.0	95.7	97.2	97.8

in Table 4, where one can distinguish two different regions: for those values of N and $|\mathbf{h}|_{max}$ that fully admit the "absorption" of task 4 in task 5 the efficiency is at least 95%. Note that the physical limit we are interested in corresponds precisely to this region. For the other values of N and $|\mathbf{h}|_{max}$ (i.e. $(N, |\mathbf{h}|_{max}) = (125,4), (125,5)$ and $(216,4)$) the efficiency is lower. In summary, we have obtained very satisfactory efficiencies, especially in the cases of large N and $|\mathbf{h}|_{max}$, i.e. for such systems where the use of and need for parallelization is the greatest.

CONCLUDING REMARKS

Two main comments may be made from our work. First of all, we have confirmed the capacity of transputer arrays for performing realistic MD calculations. In fact, the full simulations we have performed [11], which would have required 23 months on a VAX 8600, took 40 days to complete on the Supernode.

The second remark is about the high efficiencies that have been attained, although we have not at all exhausted the possibilities of further parallelization offered by the system to be simulated. For example, we imposed – for simplicity – the condition that the communication between processors of all the results of a task would be postponed until the completion of the whole task. Of course, that restriction could be relaxed if necessary. However, our initial assumption that just playing with the different aspects of the model (Ewald summations, pair forces, . . .) could lead to satisfactory efficiencies has been proved right. In our opinion, this has been a specific example of a general behaviour of simulations: the more complex and realistic (and computing demanding) the model is, the more possibilities of parallelization it will offer. It seems, thus, that transputer arrays have indeed a great potential for statistical mechanics simulations.

Acknowledgments

We are indebted to A. Fernández for his help in solving the technical problems encountered in the development of this work. The financial support by CICYT, Project TIC-0299/1989, and by Parallel Computing Action, Project EXPRIT II (PCA-4146) is also acknowledged.

APPENDIX

We provide in this Appendix a listing of a piece of the Occam 2 code used to implement the parallel algorithm described in this work. In this illustrative example, two tasks are distributed amongst the processors. One of these tasks is parallelized by using the

particle distribution. In the other case the interaction distribution is used. The listing shows how both methods of parallelization are implemented. The "program" runs simultaneously on each of the 16 transputers. The special role of the first transputer is specifically treated.

```
SEQ

  ------------------------------------------------------------
  ------------------------------------------------------------
  ---  decide which particles are assigned to each processor ----
  ------------------------------------------------------------
  ------------------------------------------------------------
  ---  at the end of this section this information will be
  ---  contained in n.parts.each [j] , npi [j] , npf [j] ,
  ---  which are the number of molecules, and the initial an
  ---  final molecules, respectively, assigned to processor j.
  -------------- np = number of molecules  -------------------
  ---------------  i = processor identifier -------------------
  n1 := np / 16
  nexcess := np - ( 16 * n1 )


  IF
    (nexcess = 0)
      SEQ j = 1 FOR 16
        n.parts.each[j] := n1
    TRUE
      SEQ
        nalt := 15 - nexcess
        IF
          (nalt > n1 )
            nalt := n1
          TRUE
            SKIP
      n.parts.each[1] := n1 - nalt
      j.last := 16 - ( nexcess + nalt )
      SEQ j = 2 FOR j.last - 1
        n.parts.each[j]:= n1
      SEQ j = j.last + 1 FOR 16 - j.last
        n.parts.each[j]:= n1 + 1


n1 := 0
SEQ j = 1 FOR 16
  SEQ
    npi[j] := n1
    n1 := n1 + n.parts.each[j]
    npf[j] := n1 - 1


------------------------------------------------------------
------------------------------------------------------------
--------   decide which pairs are assigned to each processor   -------
------------------------------------------------------------
------------------------------------------------------------
```

```
--- at the end of this section this information will
--- be contained in  n.pairs.each [j] , npairi [j] , npairf [j]
--- npi1 [j] , npi2 [j] , npf1 [j] , npf2 [j] , which are the number
--- of pairs, the initial and final pair, and the 2 molecules involved
--- in the initial and final pair, respectively, assigned
--- to processor j

npairs  := (   np * ( np - 1(INT) )   ) / 2(INT)

n1 := npairs / 16
nexcess := npairs - ( 16 * n1 )

IF
  (nexcess = 0)
    SEQ j = 1 FOR 16
      n.pairs.each[j] := n1
  TRUE
    SEQ
      nalt := 15 - nexcess
      IF
        (nalt > n1 )
          nalt := n1
        TRUE
          SKIP
      n.pairs.each[1] := n1 - nalt
      j.last := 16 - ( nexcess + nalt )
      SEQ j = 2 FOR j.last - 1
        n.pairs.each[j]:= n1
      SEQ j = j.last + 1 FOR 16 - j.last
        n.pairs.each[j]:= n1 + 1

n1 := 1
SEQ j = 1 FOR 16
  SEQ
    npairi[j] := n1
    n1 := n1 + n.pairs.each[j]
    npairf[j] := n1 - 1

n1:= 0
j := 1
SEQ ind1 = 0 FOR ( np - 1(INT) )
  SEQ ind2 = ( ind1 + 1(INT) ) FOR ( (np - ind1) - 1 )
    SEQ
      n1 := n1 + 1(INT)
      IF
        (n1 = npairi[j])
          SEQ
            npi1[j] := ind1
            npi2[j] := ind2
        (n1 = npairf[j])
          SEQ
            npf1[j] := ind1
            npf2[j] := ind2
            j       := j + 1
        TRUE
          SKIP
```

```
-------------------------------------------------------------------
-------------------------------------------------------------------
----------        performing the calculation        ---------------
-------------------------------------------------------------------
-------------------------------------------------------------------

SEQ iter = itac FOR niter   --   ( niter = number of time steps )
  SEQ

    ------------------------------------------------------------
    ------------------------------------------------------------
    ------                particle distribution:            ------
    ------     computation of the contribution coming       ------
    ------   from the molecules assigned to processor i     ------
    ------------------------------------------------------------
    ------------------------------------------------------------

    SEQ n.m = npi[i] FOR n.parts.each[i]
      SEQ
        ...  Compute the contribution coming from molecule  n.m


    ------------------------------------------------------------
    ------------------------------------------------------------
    -------             interaction distribution:          --------
    -------      computation of the pair interactions      --------
    -------           assigned to processor i              --------
    ------------------------------------------------------------
    ------------------------------------------------------------


    np1:= npi1[i]
    np2:= npi2[i]
    WHILE ( (np1 < npf1[i]) OR ( (np1 = npf1[i]) AND (np2 <= npf2[i]) ) )
      SEQ
        ...  compute the interaction between molecules np1 and np2
        np2 := np2 + 1
        IF
          (np2 = np)
            SEQ
              np1 := np1 + 1
              np2 := np1 + 1
          TRUE
            SKIP
```

References

[1] D. Fincham, "Parallel computers and molecular simulation", *Mol. Sim.* **1**, 1 (1987).

[2] E. Clementi, G. Corongiu and J.H. Detrich, "Parallelism in computations in quantum and statistical mechanics", *Comp. Phys. Comm.*, **37**, 287 (1985).

[3] D.C. Rapaport, "Large-scale molecular dynamics simulation using vector and parallel computers", *Comp. Phys. Rep.*, **9**, 1 (1989).

[4] C.R. Askew, D.B. Carpenter, J.T. Chalker, A.J.G. Hey, M. Moore, D.A. Nicole and D.J. Pritchard, "Monte Carlo simulation on transputer arrays", *Parallel Comput.*, **6**, 247 (1988).

[5] F. Burgé, V. Martorana and S.L. Fornili, "Concurrent molecular dynamics simulation of ST2 water on a transputer array", *Mol. Sim.*, **1**, 303 (1988).
[6] H.G. Petersen and J.W. Perram, "Molecular dynamics on transputer arrays I. Algorithm design, programming issues, timing experiments and scaling projections", *Mol. Phys.*, **67**, 849 (1989).
[7] D.N.J. White, J.N. Ruddock and P.R. Edgington, "Molecular design with transparallel supercomputers", *Mol. Sim.*, **3**, 71 (1989).
[8] K.C. Bowler, R.D. Kenway, G.S. Pawley and D. Roweth, *An introduction to Occam 2 Programming*, Chartwell-Bratt, Lund (1987).
[9] *Transputer Development System*, INMOS, Prentice-Hall, Bristol (1988).
[10] M. Homewwod, D. May, D. Shepherd and R. Shepherd, *The IMS Transputer, IEE Micro*, **7**, 1 (1987).
[11] J. Casulleras and E. Guàrdia, "Computer simulation of liquid methanol II. System size effects" to be submitted to *Mol. Sim.*
[12] W.L. Jorgensen, "Optimized intermolecular potential functions for liquid alcohols", *J. Phys. Chem.*, **90**, 1276 (1986).
[13] H.J.C. Berendsen, J.P.M. Postma, W.F. Van Gunsteren, A. DiNola, and J.R. Haak, "Molecular dynamics with coupling to an external bath", *J. Chem. Phys.*, **81**, 3684 (1984).
[14] J.P. Ryckaert, "The method of constraints in molecular dynamics. General aspects and application to chain molecules", in *Molecular-Dynamics Simulation of Statistical-Mechanical Systems*, G. Ciccotti and W.G. Hoover, eds, North-Holland, Amsterdam, 1986.
[15] M.P. Allen and D.J. Tildesley, *Computer Simulation of Liquids*, Clarendon Press, Oxford, 1987, pp. 155–162.