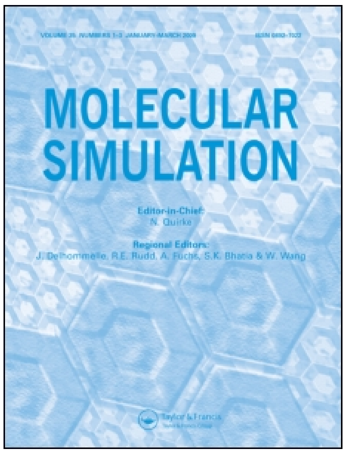

**Molecular Simulation**
Publication details, including instructions for authors and subscription information:
http://www.informaworld.com/smpp/title~content=t713644482

# Molecular Dynamics Simulations on Shared-Memory Multiple Processor Computers

Makoto Yoneya[a]; Tomohiko Ouchi[ab]
[a] Hitachi Research Laboratory, Hitachi, Ltd., Ibaraki, Japan [b] Device Development Center, Hitachi, Ltd., Tokyo, Japan

**To cite this Article** Yoneya, Makoto and Ouchi, Tomohiko(1995) 'Molecular Dynamics Simulations on Shared-Memory Multiple Processor Computers', Molecular Simulation, 15: 5, 273 — 280
**To link to this Article: DOI:** 10.1080/08927029508022341
**URL:** http://dx.doi.org/10.1080/08927029508022341

*Molecular Simulation*, 1995, Vol. 15, pp. 273–280
Reprints available directly from the publisher
Photocopying permitted by license only

# MOLECULAR DYNAMICS SIMULATIONS ON SHARED-MEMORY MULTIPLE PROCESSOR COMPUTERS

MAKOTO YONEYA and TOMOHIKO OUCHI*

*Hitachi Research Laboratory, Hitachi, Ltd.*
*7-1-1 Omika, Hitachi, Ibaraki 319-12, Japan*

*(Received May 1995, accepted July 1995)*

A parallel algorithm for molecular dynamics simulations of polyatomic molecular systems is presented. The algorithm is optimized for shared-memory multiple processor computers by assigning each molecule to be a parallelization unit which is usually taken for each atom or interaction pair. A method to distribute all interaction pairs to each molecule in almost equal numbers is described. The algorithm is tested by simulation of a liquid crystal molecule system on four-and eight-processor workstations. Overall speedups of 3.6 and 5.7 are obtained on four-and eight- processors, respectively. A non-iterative algorithm to constrain bond-length degrees of freedom is also tested with regard to its efficiency for parallel computation. A performance superior to that of the iterative SHAKE algorithm was obtained.

KEY WORDS: Molecular dynamics simulation, parallel computers, SHAKE.

## 1. INTRODUCTION

Recently, enhanced computing power has allowed molecular dynamics (MD) simulations to be applied to increasingly more complex systems. But for further industrial applications, more computing power is still desired. Consequently, development of a new algorithm which speeds up MD simulations is very important and parallelization on multiple processor computers is one way to achieve the speedup. In particular, shared-memory multiple processor computers are similar to conventional unirocessor machines in their architecture and it is easy to migrate from sequential programming compared to the case for distributed-memory multiple processor computers.

In this article, we present a parallel MD algorithm to run shared-memory multiple processor computers which is optimized for polyatomic molecular systems (e.g. molecular liquids/crystals and liquid crystals). Mertz *et al.* [1] discuss a similar subject, but they develop the corresponding algorithm for mainly macromolecular systems (i.e. proteins). In the case of polyatomic molecular systems, we can utilize each molecule (i.e. a group of atoms) as a parallelization unit. This makes the parallelization unit larger than when each atom or interaction pair serves as the unit. The size of the paralleliz-

* Current address: Device Development Center, Hitachi, Ltd., 2326 Imai, Oume, Tokyo 198, Japan.

ation unit (in other words, the parallelization grain size) is very important for parallelization efficiency. Because the balance between the hardware numbers of processors and the software size of the parallelization unit is related to the balance between the amounts of communications and actual calculations in the whole computation. We can obtain high parallelization efficiency by minimizing the rate of the communications amount to the calculations amount.

In parallelization of the MD simulations, the simplest choice of the parallelization unit would be each atom or interaction pair. For shared-memory multiple processor computers, these units cause an imbalance between the numbers of hardware processors (currently, a few dozen) and the numbers of software parallelization units ($10^3$–$10^5$ atoms or many more interaction pairs). Hence, a larger size of the parallelization unit is preferable for shared-memory multiple processor computers and taking each molecule as a unit is one way. An alternative way would be simply use of a stride with a number of atoms as the parallelization unit. But in the former case, we can confine the calculation-related intramolecular freedom to within the parallelization unit by taking the unit for each molecule. This also contributes to minimization of the amount of communications in parallel calculations and thus the former one (unit of molecule) is superior to the latter one.

In the following, we first describe the actual parallel algorithm, and then present the results of some test runs using multiple processor workstations. We also test a non-iterative constraint algorithm to compare its efficiency in the parallel calculation to the popular iterative algorithm (i.e. SHAKE [3]).

## 2. PARALLEL ALGORITHM

In general, most of the computational time in MD simulations is spent in calculation of intermolecular interactions (non-bonded Lennard-Jones and electrostatic interaction) and neighbor list generation which can speed up the interaction calculation [1]. In contrast, the intramolecular degrees of freedom (bond stretching, bond angle bending, dihedral torsioning etc.) calculation makes a much less contribution to the total computational time than the former and is easy to parallelize with the unit of the molecule. Hence, we first focus on parallelization of the calculation of the intermolecular interactions.

### 2.1 *Intermolecular Interaction Calculation*

Before we describe the main subject (i.e. the parallel algorithm), we should mention how we calculate the intermolecular interactions. The intermolecular interactions of polyatomic molecular systems can be calculated by an all atom-atom pair calculation without considering the existence of molecular structures. But from some viewpoints, it is better to utilize the molecular information. For example, we can use the information to search for close atom pairs because the molecular position is closely related to the positions of the atoms which make up the molecule. The electrostatic interaction between the neutral summed up charge molecule is a much shorter range than that of a simple charged pair interaction. This allows a shorter cut-off range to be used which results in less computational time. For these reasons, we calculate the intermolecular interactions with the molecular-molecular pair base. This means that if a molecular pair is calculated, all the atom-atom pairs between these two polyatomic molecules are calculated, but if a molecular pair is not calculated, none of the atom pairs between them are calculated. This kind of group pair based interaction calculation is also implemented in some popular MD programs (for example the MD program GROMOS [2] uses the "charge group" concept).

However, the disadvantage is that an elongated molecule such as a mesogen can have a center of mass which lies just outside the cut-off sphere of a central molecule and yet two atoms in these molecules can approach closely. For this situation, we actually utilize a hybrid neighbor list. Here, hybrid means that the neighbor list contains not only molecular close-pair information, but also atomic pair one. The cut-off sphere of the molecular center of mass is utilized for the first level selection in close pair search, and if the pairs are within this sphere, the closest atomic pair (in this molecular pair) is identified and stored as part of neighbor list information with molecular pair information. Then, another (smaller) cut-off sphere of this closest atomic pair is utilized for deciding whether this molecular pair is actually calculated or not. Thus, by using this hybrid list, we can avoid the problem with the elongated molecule still using a molecular based pair list.

Then, our problem is how to parallelize the intermolecular interaction calculation based on the molecular-molecular pairs. If we consider $N$ molecules in the system, there are $N(N-1)/2$ molecular pairs in principle. To calculate the intermolecular interactions with the molecular base parallelization unit, we have to distribute these $N(N-1)/2$ molecular pairs to each of the $N$ molecules. When we do this, we have to equalize the number of distributed pairs because this corresponds to the load balance between the parallelization units. If these loads are unbalanced, parallel efficiency is reduced by the synchronization wait.

We solve this problem by using a new distributed pair list. We call this pair list an "odd even" distributed pair-list (OEDPL) since it utilizes information of index odd-evenness in its distribution process. The distribution procedure can be described by using the example of Figure 1 as follows. First, Figure 1a shows the interaction pair matrix whose row and column index $i, j$ stands for the molecule indices to form the $i$-$j$ interaction pair. The interaction pairs are shown by lower half matrix elements without trace elements. The pairs can be sequentially numbered as in Figure 1a and they can be tabulated as a linear list (Fig. 1b). In Figure 1b, $n$ is the pair index, and $i$ and $j$ are the corresponding pair molecular indices. Our OEDPL is generated from this global pair list (Fig. 1b) by distributing pairs depending on the odd-evenness of the pair index $n$. For example, the pair whose pair index $n=1$ is distributed to $i$ (in this case $i=2$) according to the oddness of the pair index $n$. By contrast, the pair $n=2$ is distributed to $j$ (in this case $j=1$) according to the evenness of the pair index. By following this rule, all pairs can be distributed (as in Fig. 1c) equally (the total molecule number $N$ is odd) or differing only by one ($N$ is even) in their total distributed pair number. In Figure 1c, the molecule with index 1 for example, has molecular i-j pair of 1-3, 1-4, 1-7 and so on and the molecule with index 2 has the pair of 2-1, 2-5, 2-6 and so on. Clearly, the correspondence between odd-evenness of pair index $n$ and whether $i$ or $j$ the pair will be distributed can be reversed.

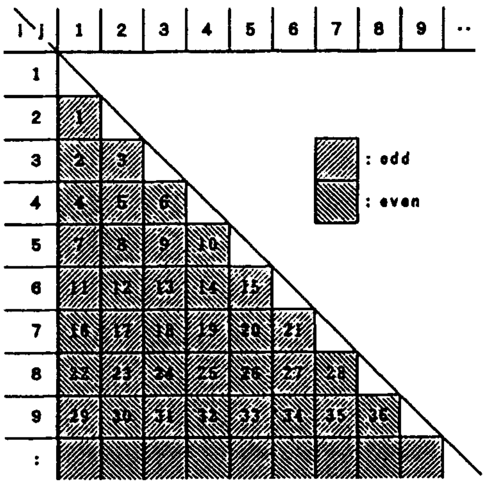

Figure 1-a.

| n | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | ·· |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| I | 2 | 3 | 3 | 4 | 4 | 4 | 5 | 5 | 5 | 5 | 6 | 6 | ·· |
| J | 1 | 1 | 2 | 1 | 2 | 3 | 1 | 2 | 3 | 4 | 1 | 2 | ·· |

Figure 1-b.

| I | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | ·· |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $J_1$ | 3 | 1 | 2 | 2 | 1 | 1 | 2 | 2 | 1 | 1 | 2 | 2 | ·· |
| $J_2$ | 4 | 5 | 4 | 5 | 3 | 3 | 4 | 4 | 3 | 3 | 4 | 4 | ·· |
| $J_3$ | 7 | 6 | 7 | 6 | 7 | 5 | 6 | 6 | 5 | 5 | 6 | 6 | ·· |
| : | : | : | : | : | : | : | : | : | : | : | : | : | : |

Figure 1-c.

**Figure 1** Generation procedure of OEDPL.

By using this procedure, the total $N(N-1)/2$ interaction pairs are equally or almost equally distributed to each of the $N$ molecules as the OEDPL. Then, the distributed molecular neighbor list is generated based on this OEDPL in parallel and the actual interaction calculation is done based on this distributed neighbor list, also in parallel. The reduction of pair numbers from the basic OEDPL to the distributed neighbor list can be considered uniform between each molecule, because there is no systematic relation between molecular indices and geometrical positions of molecules. As a result, the total computational effort in the calculation of intermolecular interactions also can be equalized by our algorithm which uses the OEDPL.

## 2.2 *Intramolecular Interaction Calculation*

Next, we consider the parallel calculation which is related to the intramolecular degrees of freedom. As we noted at the beginning of this section, parallelization of the intramolecular interaction calculation in each molecule is straightforward, i.e. just parallelize with the unit of molecule. But there is one problem related to the calculation of constraint forces. To constrain some intramolecular degrees of freedom (for example bond length), utilization of the SHAKE algorithm [3] is popular in MD simulations of polyatomic molecular systems. Since SHAKE is an iterative method, there is a possibility for load imbalance between molecules due to the difference in iteration numbers to achieve the convergence criterion. Mertz *et al.* [1] reported reduction of parallelization efficiency due to this load imbalance with SHAKE. To solve this problem, utilization of a non-iterative constraint algorithm [4] is a natural solution. In this study, we tried the non-iterative algorithm NIMM which was developed by our group [5]. As a result, all the calculations related to intramolecular degrees of freedom are efficiently parallelized with the unit of each molecule. These calculations can be done without communications between other parallelization units (i.e. molecules) in our method.

Figure 2 shows the program structure in the program analysis diagram (PAD) [6] representation. The hatched region stands for the portion which is calculated in parallel.

## 3. TEST SIMULATION RESULTS

We made test simulations of the liquid crystal molecule 4-*n*-pentyl-4'-cyanobiphenyl (5CB) to check the efficiency of the parallel algorithm described above. The test simulations were done on a Stardent TITAN 3000 (four processors) and a Silicon Graphics IRIS4D/280 (eight processors).

The 5CB model we used is the same one described by Picken *et al.* [7]. Force field parameters were taken from the same reference (they were basically GROMOS force field parameters [2]) and all bond-lengths were constrained to fixed values by the NIMM algorithm. The calculations were done with a 64-molecule system (1216 total atoms) under periodic boundary conditions with cut-off length of 1.4 nm and a liquid state temperature (around 320 K). The initial configuration was made by (over $10^4$ steps) MD runs from the $\alpha$-F.C.C. structure. The computational time was measured for a 100-step simulation with a 2.0 fs time step. In the calculations, total energy was conserved with sufficient accuracy.

Figure 3 shows the results of the relationships between the number of processors and calculation steps per second (Figs. 3a and 3b correspond to results with the TITAN and IRIS, respectively). The dashed line represents the ideal linear speedup. The speedup

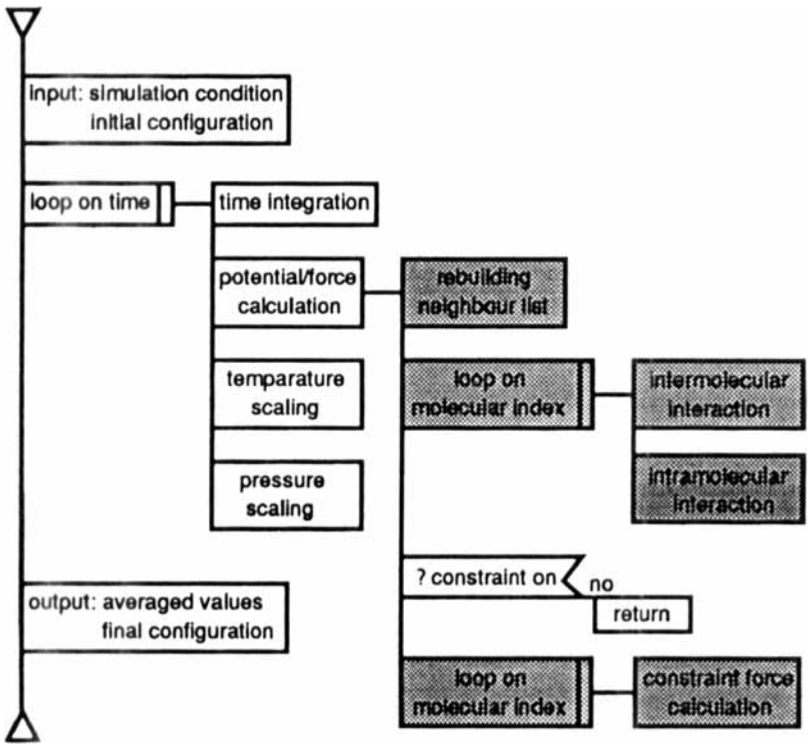

**Figure 2** Program Structure in PAD.

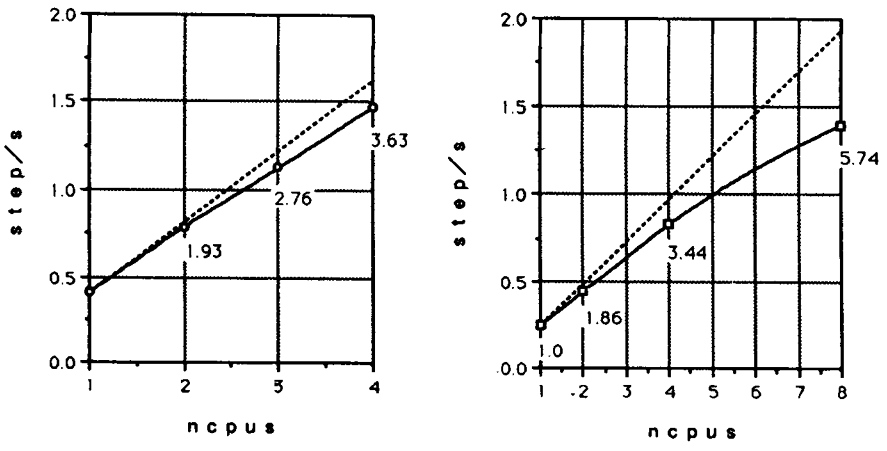

Figure 3 - a. Figure 3 - b.

**Figure 3** Relationship between number of processors (ncpus) and computational speeds.

obtained was almost linear up to four processors (3.6 on the four-processor TITAN) and slightly reduced above that (5.7 on the eight-processor IRIS). These speedups were obtained from the total computational time including file inputs and outputs etc. We utilized a similar technique as in Mertz *et al.* [1] to eliminate synchronization in updating energy and force variables by using processor private copies of these variables. Hence, the same order of speedup (in the same IRIS) with a parallel computation of completely independent calculations (e.g. the parallel ray-tracing calculations [9]) are obtained in our method.

Next, we analyzed the results for speedup of the constraint force calculation by the NIMM algorithm. Figure 4 shows results corresponding to those of Figure 3a, but which have been seperated into speedups of non-constraint and constraint force calculations. In this figure, the speedup of the constraint force calculation (by NIMM) was almost the same as for the non-constraint force. This was unlike the results of Mertz *et al.* [1] who reported a clear difference in speedup between the constraint force calculation (by SHAKE) and the non-constraint force calculation. In the test simulations above, NIMM was 5.6 times faster than the SHAKE method with the four-processor IRIS in the constraint part calculations and it was also 8.7 times faster in simulations of 216 molecules of SPC water [8].

## 4. DISCUSSION

We described the parallel algorithm which used each molecule as a parallelization unit for MD simulations of polyatomic molecular systems. For macromolecular systems,

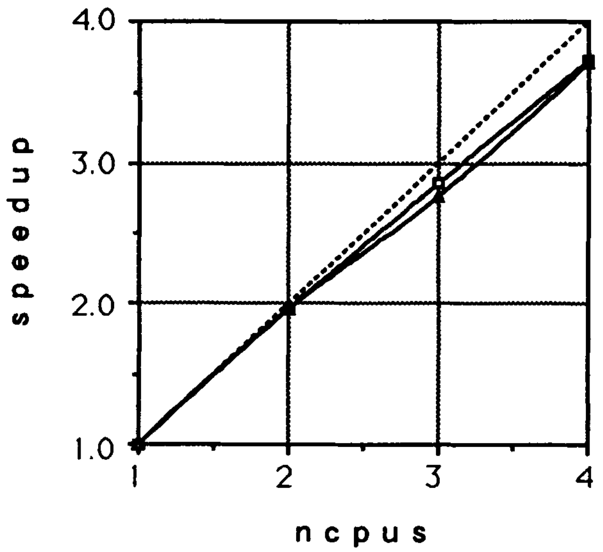

Figure 4 Relationship between number of processors (ncpus) and computational speeds decomposed to non-constraint parts (□) and constraint parts (Δ).

our method can be extended by changing the unit from molecule to residue based charge groups in the macromolecule. In the contraint force calculation by NIMM, the core matrix solving part itself should be parallelized (for macromolecular simulations).

We also showed our method provides reasonable load balancing for simple polyatomic molecular systems (e.g. liquid crystal, SPC water) in a far simple manner than other extensive load-balancing algorithms [10]. But this may not be true for systems where the molecules are non-uniformly spatially distributed (see the last paragraph of section 2.1).

As is shown in Figure 2, we only parallelize the portion which is related to the potential and force calculations. The parallel performance showed that this limitation was reasonable for shared-memory parallel computers with a few processors. If the number of processors increases, the remaining parts of MD computation should also be parallelized as in massively parallel machines to gain high parallel performance. All in all we consider that our method works well in wide variety of applications with multiple workstations which are now becoming popular.

## *Acknowledgements*

We would like to thank the Mechanical Engineering Laboratory of Hitachi Ltd. for offering computational time on their IRIS workstation. We are indebted to the referees for the readings of the manuscript.

## *References*

[1] J. E. Mertz, D. J. Tobias, C. L. Brooks, III and U. C. Singh, "Vector and Parallel Algorithms for the Molecular Dynamics Simulation of Macromolecules on Shared-Memory Computers", *J. Comp. Chem.*, **12**, 1270 (1991).

[2] W. F. van Gunsteren, and H. J. C. Berendsen, "*GROMOS manual*", BIOMOS BV (1987).

[3] J. P. Ryckaert, G. Ciccotti and H. J. C. Berendsen, "Numerical Integration of the Cartesian Equations of Motion of a System with Constraints: Molecular Dynamics of *n*-Alkanes", *J. Comp. Phys.*, **23**, 327 (1977).

[4] S. Miyamoto and P. A. Kollman, "SETTLE: An Analytical Version of the SHAKE and RATTLE Algorithm for Rigid Water Models", *J. Comp. Chem.*, **13**, 952 (1992).

[5] M. Yoneya, H. J. C. Berendsen and K. Hirasawa, "A Non-Iterative Matrix Method for Constraint Molecular Dynamics Simulations", *Mol. Simul.*, **13**, 395 (1994).

[6] Y. Futamura, T. Kawai, M. Tsutsumi and H. Horikoshi, "Development of Computer Programs by Problem Analysis Diagram (PAD)", Proc. of 5th ICSE, IEEE Computer Society, New York (1981).

[7] S. J. Picken, W. F. van Gunsteren, P. TH. van Duijnen and W. H. de Jew, "A Molecular Dynamics Study of the Nematic Phase of 4-*n*-pentyl-4'-cyanobiphenyl", *Liq. Cryst.*, **6**, 357 (1989).

[8] H. J. C. Berendsen, J. P. M. Postma, W. F. van Gunsteren and J. Hermans, in "Intermolecular Forces. Interaction Models for Water in Relation to Protein Hydration", ed B. Pullman (Reidel, Dordrecht) 331 (1981).

[9] T. Sakai, private communications.

[10] J. E. Boillat, F. Brug and P. G. Kropf, "A Dynamic Load-Balancing Algorithm for Molecular Dynamics Simulation on Multi-Processor Systems", *J. of Comp. Phys.*, **96**, 1 (1991).