Efficiency of Dynamic Load Balancing Based on Permanent Cells for Parallel Molecular Dynamics Simulation

Ryoko Hayashi and Susumu Horiguchi
School of Information Science,
Japan Advanced Institute of Science and Technology,
Tatsunokuchi, Ishikawa 923-1292. Japan
TEL: +81-761-51-1268, FAX: +81-761-51-1149
ryoko@jaist.ac.jp hori@jaist.ac.jp

Abstract

This paper addresses a dynamic load balancing method of domain decomposition for 3-dimensional Molecular Dynamics on parallel computers. In order to reduce interprocessor communication overhead, we are introducing a concept of permanent cells to the dynamic load balancing method. Molecular Dynamics simulations on a parallel computer T3E prove that the proposed method using load balancing much improves the execution time. Furthermore, we analyze theoretical effective ranges of the dynamic load balancing method, and compare them with experimental effective ranges obtained by parallel Molecular Dynamics simulations. As the result, the theoretical upper bounds predict experimental effective ranges and are also valid on commercial parallel computers.

1 Introduction

The Molecular Dynamics (MD) method has been hailed as one of the important research strategies in physics, chemistry, and material computer design[1]. Simulations on conventional computers are limited to cases of 10^3 to 10^6 particles, because of the huge execution time and the limited memory space. Parallel computers, on the other hand, are capable of much larger scale and more detailed MD simulations. However, the communication overhead in parallel computers sometimes prevents high performance parallel computing. Tamayo *et al.*[2] applied a Domain Decomposition Method (DDM) to short-range MD simulations and achieved high performance in parallel simulations. Beazley *et al.*[3] reviewed the parallelization methods of MD simulations. They mentioned DDM as one of the efficient parallelization methods of MD simulations.

Although DDM is an excellent parallelization method, computational load unbalance much reduces parallel performance as the concentration of molecules increases. Brugé and Fornili[4] proposed a load balancing method for one dimensional DDM which partitions 2-dimensional simulation space along one axis into domains and assigns a domain to a processing element (PE). Their load balancing method changes a domain boundary along one axis of simulation space continuously. Kohring[5] also proposed one dimensional DDM by discretely moving boundary. These methods are, however, effective only for 2-dimensional MD simulations and are not extended to 3-dimensional MD simulations easily, since it is difficult to keep neighboring relationships among PEs regular and the communication overhead among PEs is extremely large.

For the 3-dimensional short-range MD simulations, we proposed a dynamic load balancing method (DLB) by introducing a concept of *permanent cells* to minimize interprocessor communication overhead [6][7][8]. DLB was proved to achieve computational load balancing among PEs by 3-dimensional parallel MD simulations on CM-5[6][7]. However, permanent cells restrict redistribution of computational load. This paper describes theoretical upper bounds of load balancing for MD simulation on parallel computers with different architectures. The experimental DLB effective ranges are obtained by parallel 3-dimensional MD simulations on T3E[9].

This paper is organized as follows: Section 2 describes DLB based on permanent cells in DDM. Section 3 shows implementation of DDM and DLB on a parallel computer T3E and the parallel performances are discussed in detail. In Section 4, theoretical effective ranges of DLB are analyzed by estimating the upper bounds of the particle concentration ratio. Section 4 also shows that the theoretical upper bounds can predict the effective range of load balancing for parallel MD simulations by comparing the theoretical upper bounds with experimental effective ranges. Section 5 is the conclusion.

2 Parallel Molecular Dynamics Simulation

2.1 Molecular Dynamics simulation

MD computes the interaction between particles using equations of motion. The new position and the velocity of each particle are computed by the numerical integration of the classical equations of motion at every time step. The forces between particles are obtained by the summation of interactions between two particles.

The interaction between two particles is obtained by the Lennard-Jones potential as follows,

$$V(r_{ij}) = 4\epsilon \left\{ \left(\frac{\sigma}{r_{ij}}\right)^{12} - \left(\frac{\sigma}{r_{ij}}\right)^{6} \right\}, \tag{1}$$

where r_{ij} is the distance between the i-th particle and the j-th particle. ϵ and σ are parameters given for each substance. The Lennard-Jones potential is one of the short-range potentials, because an increase of r_{ij} makes the interaction very small. This type of potential generally uses a "cut-off distance" r_c. MD simulations compute all forces between N particles, and require $N(N-1)/2$ interactions. Using cut-off distance r_c, the complexity of interaction calculation is improved to $O(N)$ by DDM.

2.2 Domain decomposition method

DDM allocates domains of the simulation space on PEs. Figure 1 illustrates DDM in 2-dimensional simulations. Figure 1 shows an example of DDM in a 2-dimensional simulation space. A space containing particles is divided into small squares referred to as "cells" as shown in Figure 1. Each row of cells is allocated to each PE as a "domain". In this case, PEs are virtually inter-connected as a ring. If the size of cells is greater than r_c, essential forces between the particles are obtained by computing interactions of particles within 8 neighboring cells for the 2-dimensional space. Since interactions between particles are restricted to neighboring cells, the computational complexity of DDM is $O(N)$. DDM greatly reduces computation time and interprocessor communication overhead.

For 3-dimensional MD simulation, DDM cells are cubic, thus essential forces between the particles are obtained by computing interactions of particles within their own cell and 26 neighboring cells. DDM in 3-dimensional simulation space has several domain shapes closely related to inter-processor communication overhead. Figure 2 shows three domain shapes in 3-dimensional MD simulations. We discussed relationships of these domain shapes and communication overhead[8], and we pointed out that square pillar

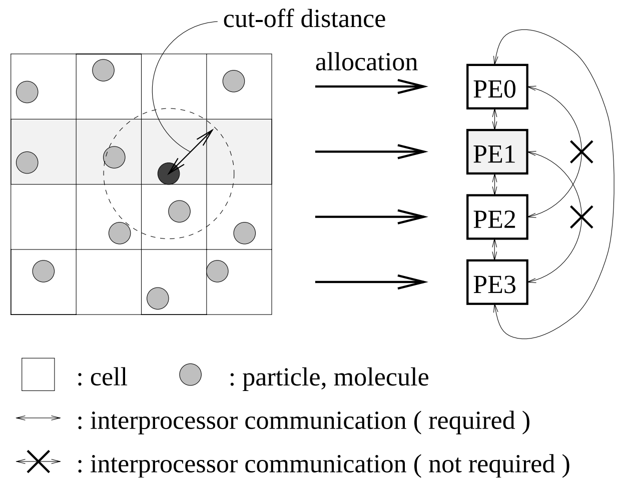


Figure 1. An illustrated example of Domain Decomposition Method in 2-dimensional MD simulation.

domain as shown in Figure 2 (b) is the best domain shape for mid-size MD simulations on mid-size parallel computers. With square pillar domain, PEs are virtually connected as a 2-dimensional torus, and PEs and domains have 8 neighbor relationships virtually.

Square pillar domain is suitable for DLB as the first step, because neighboring relationship between domains is simple. Cube domain in Figure 2 (c) is suitable for large-scale MD simulations on massively parallel computers. However, the number of neighboring PEs with cube domain is large and DLB becomes more difficult. From this reason, we discuss DLB using square pillar domain.

2.3 Dynamic load balancing method

DLB realizes uniform load allocation on PEs by moving cells among neighboring PEs, that is, *cell redistribution*. However, cell redistribution may break the 8 neighbor relationships among the PEs. The PE neighboring relationships must be preserved because an irregular communication pattern increases the communication overhead. Therefore we introduce a concept of permanent cells in order to keep the PE neighboring relationships and to reduce the communication overhead among PEs.

To keep the 8 neighbor relationships among PEs, each domain should avoid contact with other domains except for its neighboring 8 domains. DLB classifies cells in each domain into two types: movable cells and permanent cells. Figure 3 shows a 2-dimensional cross-section of the square pillar domain and the case of 81 cells' simulation space distributed to 9 PEs. The permanent cells consist of a row and a column of cells. They are not redistributed among the

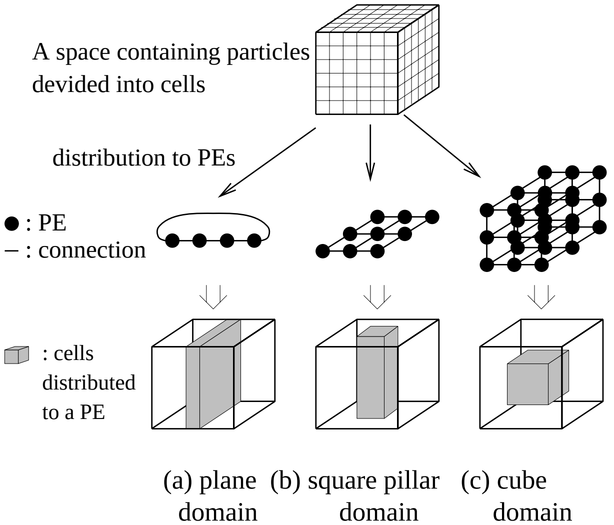


Figure 2. Three domain shapes in 3-dimensional MD simulations.

neighboring PEs. They make a wall between a PE and PEs that should not communicate with the PE. Thus the permanent cells maintain the 8 neighbor relationships.

To explain DLB procedure, we consider the case of 9 PEs and 81 cells as shown in Figure 3. $PE(i,j)$ denotes the address of a PE. All the cells have their own addresses as well. Let $C_{i,j}$ be the cell contained in a domain allocated on $PE(i,j)$. A permanent cell is expressed as $C^{p}_{i,j}$, and a movable cell is expressed as $C^{m}_{i,j}$. In Figure 3, the PE has 4 movable cells and 5 permanent cells. The cell redistribution is executed as follows:

1. $PE(i,j)$ sends its execution time of the last time step to neighboring PEs.

2. $PE(i,j)$ finds the fastest PE (PE_{fast}) among itself and the neighboring 8 PEs.

3. $PE(i,j)$ decides a cell C_{send} to be sent to PE_{fast} as follows,
   Case 1: $PE_{fast} = PE(i-1,j-1), PE(i-1,j)$, or $PE(i,j-1)$
   If $PE(i,j)$ has movable cells, $C_{send} = C^{m}_{i,j}$. If $PE(i,j)$ has no movable cells, $C_{send} = 0$.
   Case 2: $PE_{fast} = PE(i-1,j+1)$ or $PE(i+1,j-1)$
   For these PEs, $PE(i,j)$ has no cells which can be sent, so $C_{send} = 0$.
   Case 3: $PE_{fast} = PE(i,j+1), PE(i+1,j)$, or $PE(i+1,j+1)$
   If $PE(i,j)$ has received any cells from PE_{fast}, then $PE(i,j)$ returns one of these cells. For the case of $PE_{fast} = PE(i+1,j)$, then $C_{send} = C^{m}_{i+1,j}$. If $PE(i,j)$ has no $C^{m}_{i+1,j}$, then $C_{send} = 0$.

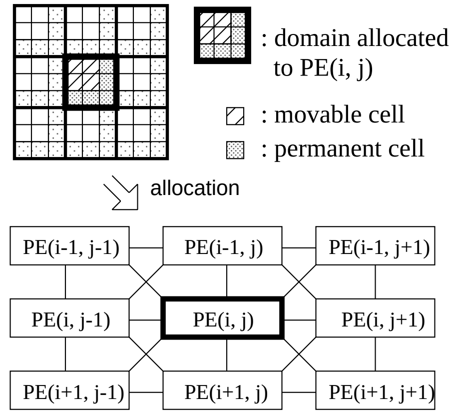


Figure 3. An example of allocation of cells to processing elements.

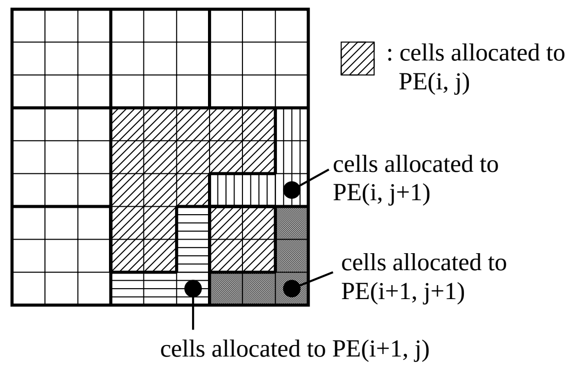


Figure 4. Cell redistribution example for the case $PE(i,j)$ has the maximum number of cells (A PE has 3×3 cells initially).

4. $PE(i,j)$ sends addresses of PE_{fast} and C_{send} to neighboring PEs.

These operations are carried out for load balancing among neighboring PEs to allocate cells to PEs whose computational load is lighter than other PEs.

The overhead of DLB is small so that MD simulations are able to execute DLB operations every time step. If one PE receives too much cells from its neighboring PEs, the PE can return cells to the neighboring PEs during several time steps.

Figure 4 shows an extreme example of DLB. Let $PE(i,j)$ be the fastest PE among 9 PEs. $PE(i,j)$ is able to receive cells from 3 PEs as shown in Figure 4. After the cell redistribution, $PE(i,j)$ has up to 2.3 times the number of cells allocated initially. The permanent cells keep 8 neighbor structure, but they limit computational load redistribution of DLB because they restrict direction and number of cells for cell redistribution. Here we call it *DLB limit*.

3 Implementation result

3.1 Parallel Computer T3E

DLB and DDM are implemented on a parallel computer T3E [9] to study parallel performances. T3E has 128 PEs, and each PE has a DECchip 21164 (300MHz, 600MFLOPS, 1200 MIPS). DECchip 21164 contains 8KB of data cache, 8KB of instruction cache, and 96KB of secondary cache. The PEs are connected with 3-dimensional torus interconnection and communication performance is 2.8GB/second per PE. T3E supports MPI(Message-Passing Interface) and FORTRAN90, and we use them for implementation. We implement DDM on T3E using a SPMD type algorithm. Execution time in this paper is measured by MPI's function: MPI_Wtime.

3.2 Physical condition

Parallel MD simulations were executed for the case with the reduced temperature $T^*_{ref} = 0.722$ and the reduced density $\rho^* = 0.256$. The Lennard-Jones parameters are chosen for the Argon value[1]. Since the reduced temperature is equivalent to the temperature under the Argon's boiling point, these physical conditions simulate supercooled gas phase. A cut-off distance r_c between 2.5 and 3.5 reduced distance is chosen for the Lennard-Jones potential in general, here 2.5 is used. The system keeps the number of particles N, the volume V, and the energy E constant. The time step Δt is 0.064. The temperature is scaled to T^*_{ref} every 50 time steps. The boundary condition of the simulation space is the periodic boundary condition. Under these physical conditions, particles keep concentrating.

Here we represent the numerical methods in our DLB program and DDM program. The programs use the same size cells. The size of the cells is equal to r_c, or a little larger than r_c. For the numerical integration of the equations of motion, we implemented the velocity form of the Verlet algorithm[1]. In order to obtain neighbor molecules for each molecule, the programs compute distances between two molecules with every combination of molecules within each cell and its neighboring 26 cells. Our programs recompute and replace the relationships between cells and molecules every time step.

3.3 Execution time

Here we use the following values to explain execution conditions of MD simulations.

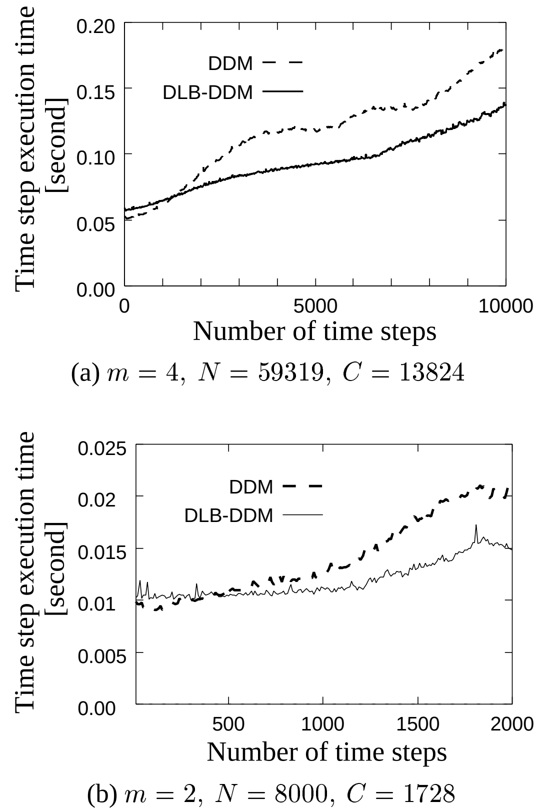


Figure 5. Execution time as a function of time step of domain decomposition method and DLB domain decomposition method on 36 PEs of T3E $(T^*_{ref} = 0.722,\ \rho^* = 0.256)$**.**

N**:** Number of particles.

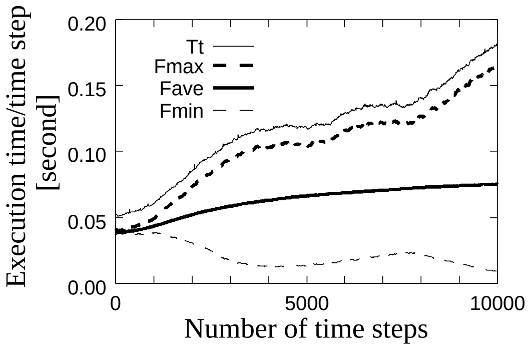
C**:** Number of cells.

P**:** Number of PEs.

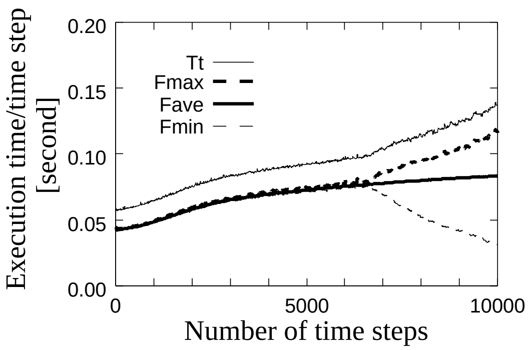
m**:** The size of square pillar domain's cross-section. m is given as $m = C^{1/3}/P^{1/2}$, and the square pillar domain has $m^2C^{1/3}$ cells.

To discuss parallel simulation performances of DLB and DDM, we explain two cases: $m = 2$ and $m = 4$. In the $m = 2$ case, $1/4$ of a domain is movable cells. On the other hand, in the $m = 4$ case, $9/16$ of a domain is movable cells. Because the ratio of movable cells in the $m = 4$ case is larger than in the $m = 2$ case, DLB has larger load balancing capability in the $m = 4$ case than in the $m = 2$ case.

Figure 5 shows the execution time per time step of MD simulations as a function of the number of time steps. Figure 5(a) is the result of an $m = 4$ simulation, and Figure 5(b) is that of an $m = 2$ simulation. The solid lines



(a) DDM execution time



(b) DLB-DDM execution time

Figure 6. Execution time and force calculation time of domain decomposition method and DLB domain decomposition method on 36 PEs of T3E $(m = 4,\ N = 59319,\ C = 13824,\ T^*_{ref} = 0.722,\ \rho^* = 0.256)$**.**

(DLB-DDM) in the figures show execution time of DDM with DLB and broken lines (DDM) shows execution time of DDM without DLB. In both cases, DLB-DDM almost maintains the same execution time during thousands of time steps in the simulation, but the execution time of DDM rapidly increase with time steps. In $m = 4$ case, efficiency of DLB is clearer than the $m = 2$ case. Because the load balancing capability in the $m = 4$ case is larger than the capability in the $m = 2$ case.

Next, we investigate the force computing time among the execution time of the simulation. Figure 6 shows the detail of execution time with $m = 4$. Figure 6(a) is the detail of execution time of DDM, and Figure 6(b) is the detail of DLB-DDM. The four lines in Figure 6 represent execution time as follows:

Tt Execution time of each time step.

Fmax The maximum values of force calculation time among PEs.

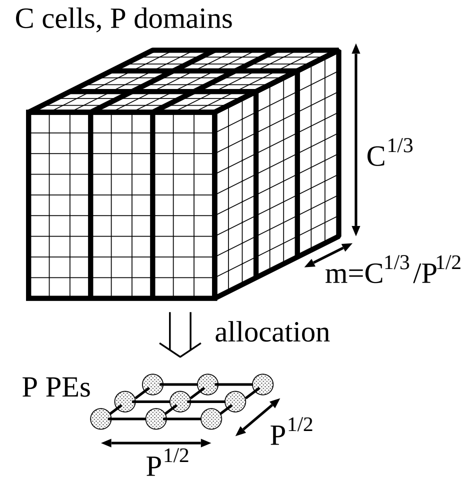


Figure 7. Square pillar domain for 3-dimensional Molecular Dynamics simulations and the relationships of parameters C**,** P**,** m $(C = 216,\ \ P = 9,\ \ m = 2)$**.**

Fave The average values of force calculation time among PEs.

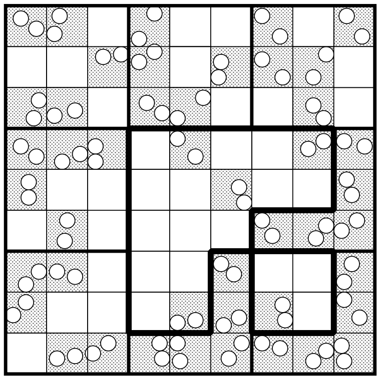
Fmin The minimum values of force calculation time among PEs.

The execution time *Tt* depends on the maximum time of force computing *Fmax* in both, because of the synchronization among PEs. Differences between *Fmax* and *Fmin* of DDM rapidly increases with the time step in Figure 6(a). On the other hand, DLB-DDM maintains the small difference between *Fmax* and *Fmin* during thousands of time steps in the simulation. This means that DLB effectively achieves the uniform load allocation. After the 7000 time step in Figure 6(b), the difference between *Fmax* and *Fmin* of DLB-DDM increases. This means particles' concentration becomes too much, and the concentration is beyond DLB limit after the 7000time step.

4 Effective ranges of DLB

4.1 Theoretical upper bounds of particle concentration ratio

DLB with the permanent cells much improves parallel simulation performances for most cases. However, DLB has limits of load balancing capability, thus the parallel simulation performance of DLB depends on the simulation conditions. Next, we discuss the theoretical boundary of effective ranges of DLB by introducing a particle concentration ratio.



: cell (contains no particle)

: cell (contains 2 particles)

: largest allocated area

: domain

: particle

Figure 8. Relationships of parameters n, C, C_0, C', C'_0 for the analysis $(N = 90,\ C = 81,\ P = 9,\ m = 3,\ C_0 = 36,\ C' = 21,\ C'_0 = 16)$**.**

First, we show the basic parameters for theoretical performance estimation of DLB. Figure 7 shows a square pillar domain decomposition for MD simulation. Let C be the number of cells, P be the number of PEs, and m be the size of square pillar domain as shown in Figure 7. m is given by $m = C^{1/3}/P^{1/2} > 1$.

Next, we introduce several parameters in order to explain the concentration of computational load. Figure 8 shows a 2-dimensional example where a PE has maximum number of cells (maximum domain). The maximum domain contains C' cells. C' is the summation of the number of cells in a square pillar domain and all the movable cells in the 3 neighboring PEs. Since $C' = m^2 + 3(m-1)^2$ in the 2-dimensional case, $C' = [m^2 + 3(m-1)^2]C^{1/3}$ in the 3-dimensional case. Let C_0 be the number of cells in the whole simulation space which contain no particles, and C'_0 be the number of cells containing no particle in the maximum domain. We explain particle concentration ratio in the whole simulation space by C_0/C. Suppose the case $C - C_0$ cells contains an average number of particles $N/(C - C_0)$. In Figure 8, $N = 90$, $C = 81$, $C_0 = 36$, thus $N/(C - C_0) = 2$ and each cell containing particles has 2 particles. Let n be the concentration factor defined by $n = (C'_0/C')/(C_0/C) \geq 1$. In Figure 8, $n = (16/21)/(36/81) \simeq 1.7$.

We introduced particle concentration ratio C_0/C and other parameters to explain computational load allocation. C_0/C might be smaller than the upper bounds function $f(m,n)$, when DLB achieves the uniform computational load allocation. Let's estimate $f(m,n)$. Average number of particle per PE is N/P. In the maximum domain, the ratio of the number of cells containing particles $C' - C'_0$ to the number of cells C' is given by,

$$\frac{C' - C'_0}{C'} = 1 - n \cdot \frac{C_0}{C}. \qquad (2)$$

If the number of particles in the maximum domain is larger than the average number of particles per PE, DLB can allocate computational loads to PE uniformly. Then the following inequality is given for uniform load balancing,

$$[m^2 + 3(m-1)^2]C^{1/3}\left(1 - \frac{nC_0}{C}\right)\frac{N}{C - C_0} \geq \frac{N}{P}. \qquad (3)$$

The above inequality is transformed into the following inequalities;

$$[m^2 + 3(m-1)^2]C^{1/3}P\left(1 - \frac{nC_0}{C}\right) \geq C - C_0, \qquad (4)$$

$$\begin{aligned} & [m^2 + 3(m-1)^2]C^{1/3}P - C \\ \geq \quad & \{n[m^2 + 3(m-1)^2]\frac{P}{C^{2/3}} - 1\}C_0. \end{aligned} \qquad (5)$$

Multiplying both sides of the above inequality by m^2/C and replacing $C^{1/3}/P^{1/2}$ by m, the following inequality is obtained,

$$\begin{aligned} & [m^2 + 3(m-1)^2] - m^2 \\ \geq \quad & \{n[m^2 + 3(m-1)^2] - m^2\}\frac{C_0}{C}. \end{aligned} \qquad (6)$$

Using $n[m^2 + 3(m-1)^2] - m^2 > 0$, the above inequality becomes as follows,

$$\frac{3(m-1)^2}{n[m^2 + 3(m-1)^2] - m^2} \geq \frac{C_0}{C}. \qquad (7)$$

Then, $f(m,n)$ is given by,

$$f(m,n) \quad = \quad \frac{3(m-1)^2}{m^2(n-1) + 3n(m-1)^2} \geq \frac{C_0}{C}. \qquad (8)$$

In the cases of $m = 2$, 3 and 4, we have

$$f(2,n) = \frac{3}{7n-4}, \qquad (9)$$

$$f(3,n) = \frac{4}{7n-3}, \qquad (10)$$

$$f(4,n) = \frac{27}{43n-16}. \qquad (11)$$

Then, the following relationship is obtained,

$$f(2,n) \leq f(3,n) \leq f(4,n) \quad \text{for } n \geq 1. \qquad (12)$$

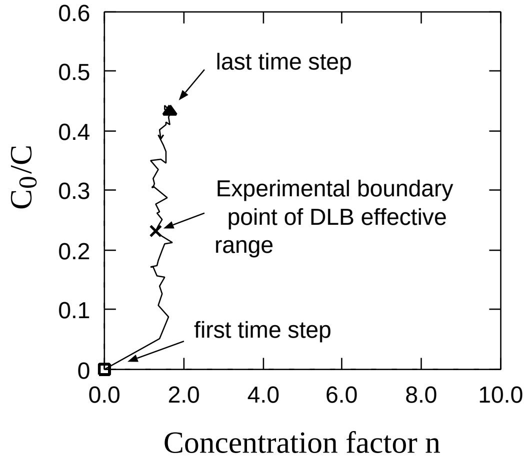


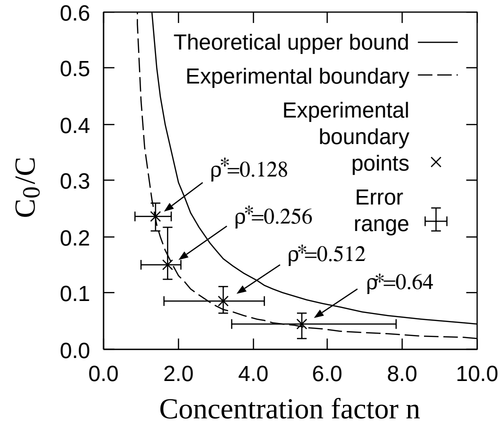
Figure 9. An example of a trajectory with an MD simulation in $n - C_0/C$ space.

4.2 Experimental effective ranges of DLB

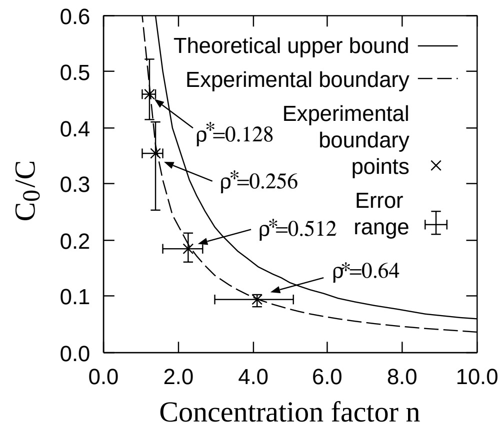
To discuss theoretical estimation for effective ranges of DLB, experimental effective ranges are obtained by parallel MD simulation with the proposed DLB. Suppose we use a few actual m values and $n - C_0/C$ space. The theoretical upper bounds function $f(m, n) = C_0/C$ is a line in $n - C_0/C$ space with an actual m value. An MD simulation draws a trajectory in $n - C_0/C$ space by computing n and C_0/C values every time step. Figure 9 is an example of the trajectory in an MD simulation. In Figure 9, the trajectory begins from $(0, 0)$ and goes up, because of low temperature and low density, which are the physical conditions in this paper. We can decide an experimental boundary point in a trajectory of an MD simulation by finding a time step at which the difference between the maximum and the minimum of force computing time begins to increase.

Next, we decide experimental boundary points for effective ranges of DLB on T3E. Each experimental boundary point is the average value of ten executions of parallel MD simulation. The ten data contain of five initial data, and an MD simulation with each initial data is executed twice. Since parallel MD simulations do not guarantee that one of the PEs has the maximum domain every time, n is estimated by using the average C_0'/C' of two PEs: one PE has the maximum number of cells, and the other PE has the maximum number of cells that containing no particle.

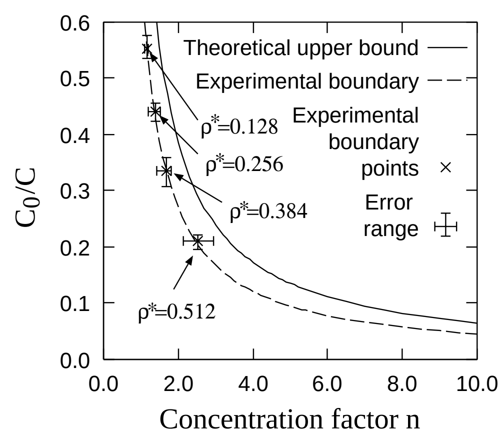
Figure 10 shows the theoretical upper bounds, experimental boundary points and experimental boundaries as a function of the concentration factor n. Each experimental boundary point has an error range. The four experimental boundary points correspond to four values of reduced density ρ^*. By applying the least-squares method to the ex-



(a) $m = 2$



(b) $m = 3$



(c) $m = 4$

Figure 10. Theoretical upper bounds of C_0/C and experimental boundary points in MD simulations on 36PEs of T3E as a function of n.

Table 1. The ratio of the experimental boundaries (E) to the theoretical upper bounds (T) of DLB.

m	Ratio E/T		
	16PEs	36PEs	64PEs
2	0.45	0.44	0.52
3	0.60	0.62	0.62
4	0.70	0.69	0.72

perimental boundary points, we calculated the experimental boundary. Figure 10(a), (b), (c) show the $m = 2$ case, the $m = 3$ case, and the $m = 4$ case, respectively.

In all of the cases, the experimental boundary points are always less than the theoretical upper bounds $f(m, n)$. The experimental boundary points fit the experimental boundaries very well in these figures. Comparing Figure 10(a) with 10(c), it seems that experimental boundary is closer to theoretical upper bound in the $m = 4$ case than in the $m = 2$ case. From the relationships of the inequality (12), the theoretical upper bound of the $m = 4$ case is larger than that of the $m = 2$ case. Therefore experimental load balancing capability in the $m = 4$ case seems larger than the capability in the $m = 2$ case.

We next compare the theoretical upper bounds with the experimental boundaries. Table 1 shows the ratio of the experimental boundaries (E) to the theoretical upper bounds $f(m, n)$ (T): E/T with $m = 2$, $m = 3$, and $m = 4$. The ratios E/T are obtained from parallel MD simulations on 16 PEs, 36 PEs, and 64 PEs. In Table 1, three E/T values with the same m are almost equal for different number of PEs, thus E/T do not highly depend on the number of PEs. Since the E/T is increasing as m becomes large, the experimental boundary is closer to the theoretical upper bounds with larger m. It is seen that the experimental boundary of DLB is larger than half of the theoretical upper bounds for most cases.

5 Conclusion

The proposed DLB based on a concept of permanent cells maintains regularity of communication pattern among neighboring PEs. Parallel MD simulations with DLB evaluated on the parallel computer T3E shows that DLB achieves a sufficient computational load allocation to PEs and reduces the execution time of parallel simulations. The theoretical upper bounds of the proposed DLB have been analyzed by introducing particle concentration ratio. Furthermore, the theoretical upper bounds have been compared with the experimental boundary obtained by parallel MD simulations on T3E. As the result, theoretical upper bounds are able to predict actual effective range of DLB.

Acknowledgment

A part of this research was supported by a Grant-in-Aid for scientific research of the Ministry of Education, Science and Culture of Japan.

References

[1] D. W. Heermann. *Computer Simulation Methods in Theoretical Physics*. Springer-Verlag Tokyo, 1990. 2nd edition.

[2] P. Tamayo, J. P. Mesirov and B. M. Boghosian. Parallel Approaches to Short Range Molecular Dynamics Simulations. In *Proceedings of Supercomputing '91*, pages 462–470, 1991.

[3] D. M. Beazley, P. S. Lomdahl, N. Grønbech-Jensen, R. Giles, and P. Tamayo. *Parallel Algorithms for Short-range Molecular Dynamics*, volume 3 of *Annual Reviews in Computational Physics*, pages 119–175. World Scientific, 1995.

[4] F. Brugé, S. L. Fornili. Concurrent molecular dynamics simulation of spinodal phase transition on transputer arrays. *Computer Physics Communications*, 60:31–38, 1990.

[5] G. A. Kohring. Dynamic load balancing for parallelized particle simulations on MIMD computers. *Parallel Computing*, 21:683–693, 1995.

[6] R. Hayashi and S. Horiguchi. Parallelized Simulation of Molecular Dynamics by Domain Decomposition Strategy. In *Proc. of 1st World Congress on Systems Simulation*, pages 353–358, 1997.

[7] R. Hayashi and S. Horiguchi. A Parallel Molecular Dynamics Simulation by Dynamic Load Balancing Based on Permanent Cells (in Japanese). *Transactions of Information Processing Society of Japan*, 40(5):2152–2162, 1999.

[8] R. Hayashi and S. Horiguchi. Domain Decomposition Scheme for Parallel Molecular Dynamics Simulation. In *Proceedings of HPC Asia '97*, pages 595–600, 1997.

[9] Cray Research, Inc. *CRAY T3E Fortran Optimization Guide*, 1996.