PAPER Special Issue on Parallel or Distributed Supercomputing

Parallelized Simulation of Complicated Polymer Structures and Its Efficiency

Kazuhito SHIDA[†], **Kaoru OHNO**[††], *Nonmembers*, **Masayuki KIMURA**[†], *and* **Yoshiyuki KAWAZOE**[††], *Members*

SUMMARY A large scale simulation for polymer chains in good solvent is performed. The implementation technique for efficient parallel execution, optimization, and load-balancing are discussed on this practical application. Finally, a simple performance model is proposed.

key words: *parallel supercomputing, computational physics, Monte-Carlo simulation, dynamic load balancing*

1. Introduction

Needless to say, polymers are new material of great importance in a variety of practical applications. Moreover, polymers dissolved in solvents is a matter of special interest for statistical physics because of its complexity. In addition, chemists are daily creating special kinds of polymers which have much more complicated topologies. For example, star shaped polymers (Fig. 1) which consist of a number of chains sharing one end, have interesting features and those are hardly understandable by analytical theories. The polymer solutions, in which many polymer chains are entangled among solutions, is another important examples. These materials introduce interesting theoretical problems in physics, and computer simulation provides one of the powerful methods to analyze them.

Since this type simulation requires vast amount of computation, to exploit parallelism efficiently is quite important for carrying out large scale simulations. In the present paper, such simulation works we have performed to estimate various physical properties of polymers with complex topology is introduced, and we will discuss their efficiency and model their performance. We believe that the performance analysis in such an actual application is quite important. A Connection Machine CM-5 with 64 nodes by Thinking Machines Co. is employed to carry out the actual processing.

2. Model

The word polymer means chemical compound which consists of repetition of the same small molecular structures (called segments). For example, poly-ethylene is described in chemical formula as $(CH_2)-(CH_2)-.....-(CH_2)$. Since every (CH_2) has several choices of relative position to the adjacent molecule, long chain polymer has an enormous number of possible conformation. In their solution, what has the largest effect on their physical property is this vast degree of freedom because the expectation value of a physical value f is given by a formula including summations taken over all possible conformations. The formula is

$$< f > = \frac{\sum_x f(x) e^{-E(x)/k_B T}}{\sum_x e^{-E(x)/k_B T}}, \quad (1)$$

where k_B is the Boltzmann constant (1.38066×10^{23} Joule/Kelvin), T the temperature of the system. The conformations are denoted as x. The function f can be any function of the system and complexity of the problem has a large dependence on the form of f. A simple example of f is the gyration radius of the polymer as the measure of the extension of the polymer chains,

$$< r >_g = \sum (r_{center} - r_i)^2 \quad (2)$$

where r_{center} is the position of the center of gravity, and r_i the position of the each component molecule, While in the calculation of the hydrodymanic radius of polymer, 3(Number of the segments) + 1 linear algebraic equations must be solved for each samples.

In the present simulation study, the polymer chains are discretized into jagged lines on a cubic lattice as shown in Fig. 2. In spite of coarseness of this approximation, this model represents various conformations of

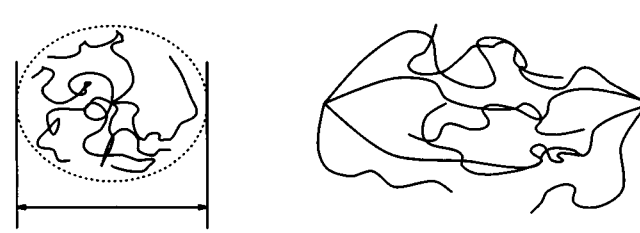
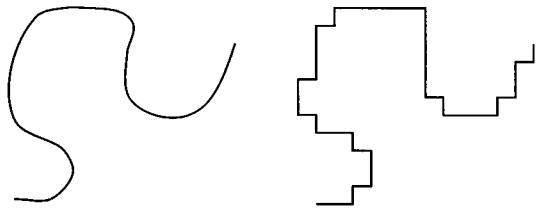


Fig. 1 Star shaped polymers. They behave like particles in solution (left). If two stars come nearby, complicated entanglement effect occur (right).

Manuscript received September 5, 1996.

[†]The authors are with Japan Advanced Institute of Science and Technology, Hokuriku, Ishikawa-ken, 901-21 Japan.

[††]The authors are with Institute for Materials Research, Tohoku University, Sendai-shi, 980-77 Japan.



(A) (B)

Fig. 2 (A) Flexible chain. (B) Lattice representation.

flexible and continuous chains quite well, especially for long chains.

Since a lattice model is employed, each conformation must be described as a sequence of vectors,

$$S_L = < r_0, r_1, r_2 \ldots r_L >, \tag{3}$$

for single chain with L segments.

In the present work all of the $E(x)$ are assumed to be same and the evaluation seems tractable. However, all conformation must satisfy following conditions

$$\forall i \forall j \ (i < j \cap \mid i - j \mid = 1 \rightarrow \mid r_i - r_j \mid = 1) \tag{4}$$

(Connectivity of the chain)

$$\forall i \forall j \ (i \neq j \rightarrow r_i \neq r_j) \tag{5}$$

(Excluded volume of the chain segments)

to inhibit unphysical disconnections and collisions between segments. These conditions make the analytical estimation of the summation quite difficult and the complete enumeration of possible conformation is impossible due to the vast number of them (Roughly 5^L in three dimension).

The procedure called Monte Carlo (MC) simulation which constructs sample conformations by random number generator is useful for the present purpose.

However, another problem arises if one apply the MC technique to the polymer chain system in primitive manner. For example, consider the case of a polymer solution which consists of 20 polymer chains each of them having 20 segments and is confined in $10 \times 10 \times 10$ cubic cell. To construct possible conformations, 20 chains with random conformations must be placed at random places in the cell. At the point of installation of the final chain, about two fifth of the lattice sites are already occupied by precedent chains. According to such observation, success possibility of entire process is less than $(\frac{5}{6} \times \frac{1000-400}{1000})^{20} = 10^{-6}$. Hence this process is quite simple but hardly succeeds. By this, the computational complexity is severely increased.

Recently, a new kind of MC simulation named enrichment algorithm for lattice polymers has been proposed by Ohno and Binder[1]. This method significantly reduces computational complexity, to the level which can be finished on supercomputers in reasonable durations.

3. Implementation

The CM5 parallel supercomputer system is physically a combination of a large number of SPARC RISC MPU chips with vector arithmetic units loosely coupled by FAT-Tree network, and provides both MIMD and SIMD programming styles. Thus we decided to employ node level C language and used the CMMD message passing library, to carry out the entire simulation in MIMD mode of CM5. Since in this usage of CM5, automatic vectorization is not available, the vector arithmetic unit in each node is utilized directly by CDPEAC macro library. Extensive attention was paid to keep high vector efficiency in the collision checking part of the code, because in this part over 90 percent of the computing time is spent. For example, if distance of some chains at initial condition is grater than two times of the maximum length of the chain, check of the collision between them is automatically skipped. Even if vector length is decreased by this method, it is automatically recovered by changing way of strip-mining. In other words, we used CM5 as loosely coupled 64 small vector processors.

The present algorithm is described basically as follows for the case of a polymer solution of f chains:

(1) Prepare initial n samples of $L = 1$.

$$S_1 = < r_{0,0}, r_{1,0}, r_{2,0} \ldots r_{f,0} > \tag{6}$$

All of $r_{0,0}, r_{1,0}, r_{2,0} \ldots$ must be chosen randomly. Otherwise we lose possibility of obtaining some conformations of the solution.

(2) Make α *candidates* of sample of $l = 2$

$$S_2 = < r_{0,0}, r_{0,1}, r_{1,0}, \ldots r_{f,0}, r_{f,1} > \tag{7}$$

for each of $l = 1$ samples. The value of α is set so that enough number of samples are available in Phase 4. By choosing the displacements between $r_{i,0}$ and $r_{j,0}$ by random number generator as one of unit vectors, the connectivity condition of the chains is automatically satisfied.

(3) Check all candidates under the excluded volume condition; If any newly added segments occupy the same position as other segments in a sample candidate, the candidate is eliminated.

We found a new algorithm for the checking.

$$\begin{aligned} &\textit{While } i < L \ \{ \\ &\textit{If} \mid R_i - R_{new} \mid = 0 \\ &\qquad \textit{Then} \ \{ \text{ Return with FALSE} \} \\ &\qquad \textit{Else} \ \{ \ i \mathrel{+}= \mid R_i - R_{new} \mid \} \\ &\qquad \} \ \text{Return with TRUE} \end{aligned} \tag{8}$$

When all R_i are connected, this is a very fast and efficient algorithm to guarantee that no R_i is the same as R_{new}. However, in actual code we employed usual

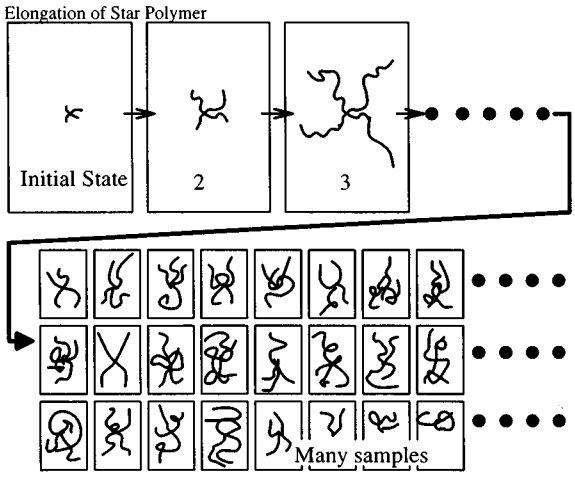


Fig. 3 Schematic explanation of entire simulation. In the real simulation, these sample configurations are distributed among processors.

"one-by-one" algorithm which is more suitable for vector arithmetic units in the nodes of CM5.
(4) Evaluate $f(r_{0,0}, r_{0,1}, r_{1,0}, r_{1,1}, \ldots r_{f,0}, r_{f,1})$ for each of sample conformations and calculate the average of the results. It is written in the files as physical results.
(5) Repeat phases 3 and 4 until the arm length reaches the arbitrary goal.

Figure 3 shows schematic explanation of the entire simulation. It is obvious from above-mentioned algorithm that enrichment algorithm is suitable for parallel processing. The reason is that all processings except for taking average in Phase 4 is independent. we can postulate that computational time of each processor for each simulation step is linear to the number of sample conformations they have. Hence the number of the configuration samples in each processor will be used later as a measure of load balancing.

4. Results

We performed parallelized simulations for star polymers under various conditions and the physical results have been given elsewhere [2], [3]. Here we will discuss only efficiency of the simulation as a parallel code.

As a typical condition for the efficiency evaluation, conformations of star polymer with 6-arms which are 50 monomers long are generated without any dynamic load balancing. The number of samples to be generated is

$$64 \times \begin{cases} 100L & \text{when } L < 10, \\ 900 + 10L & \text{when } L \geqq 10. \end{cases} \tag{9}$$

When distribution of samples is never changed during simulation, The progress of number of sample conformations in each processor, behaves as presented in Fig. 4. Since load of processors diverged as arms are elongated, this simulation is obviously inefficient, although the distribution of sample has no effect on the statistics of the physical data. Therefore, dynamic load balancing has turned out to be necessary for this kind of simulation.

A dynamic load balancing is added to the code by moving samples among processors after each monomer elongation phase has done. This means frequent interprocessor communication is also needed. To carry out this task by fast two-body message-passing, all processing nodes are sorted in the order of number of samples they have, and processor pairs are made simply like 1st and 64th, 2nd and 63rd, and so on, and excess configurations are transported between each pair. This load balancing scheme is very easy to be implemented, but apparently sometimes does not give sufficient balance by one trial, because the average load of the 1st and 64th is usually different from average load of all processors. To overcome this problem, the same procedure is repeated after each monomer elongation if it is necessary.

This scheme is tested with the same condition as above. In this case, repeating the procedure is not needed to achieve a sufficiently nice load balance (Fig. 5). The number of samples in each processor is converged within a few percent of their target value.

To evaluate how communication overhead caused by automatic load balancing affects the performance, and how well the presented automatic load balancing scheme works for simulations for conditions with larger complexity, time for communication for the load balancing scheme is measured in four complicated initial conditions for various numbers of iteration of the procedure. The initial conditions are named Disc, Graft, Hollow, and Threestars according to their physical shapes. All of them mimic the situation interesting in terms of polymer physics and we are planning some of them to be our future research subjects. Moreover, since they will require far more computation than present work, efficient parallelization of them will be more important. The result is presented in Table 1. The leftmost column is the number of iteration of load balancing procedure. In each case, the first column is entire processing time, the second column percentage of communication time only for transferring samples to adjust load balance. Other kind of communication, e.g. gathering physical data, global synchronization, and initialization of simulation (broadcasting object code to processors), are not included because to measure these communication time correctly is difficult in the current CM5 operating system. Generally, iterative load balancing with 4 to 5 times is sufficient to obtain enough performance. Although the time for communication slightly increases as iteration increases, they remain acceptable.

As a conventonal measure to show how effectively parallelization is done. Achieved speedups as functions of the number of processors is presented in Fig. 6.

To discuss this result, it must be taken into account

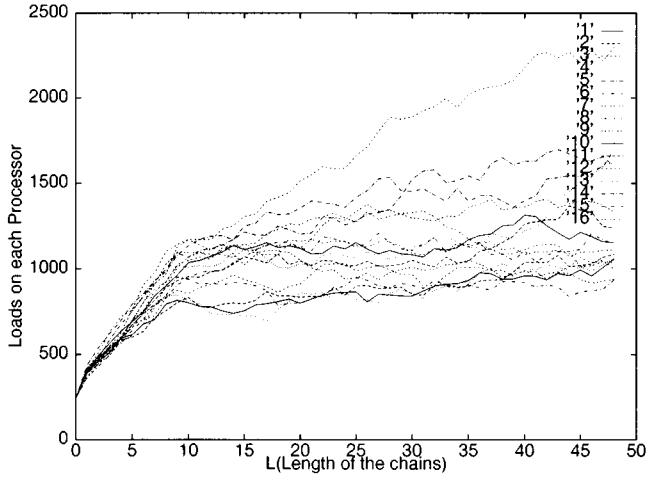


Fig. 4 Progress of load in each processors without any dynamic balancing. Results for processors 17 to 64 are omitted for simplicity without loss of important information.

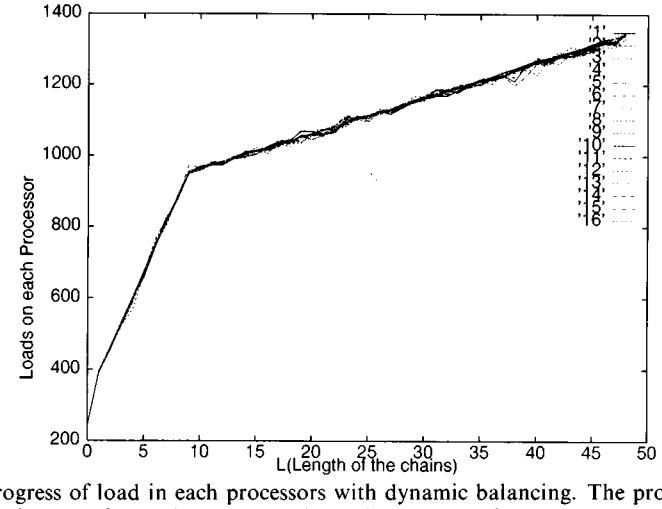


Fig. 5 Progress of load in each processors with dynamic balancing. The procedure presented is performed after each monomer elongation. Results for processors 1 to 16 among 64 are shown.

Table 1 Entire processing time, communication time and its percentage, for various number of repetitions of auto balancing scheme (at the leftmost column) after each elongation.

	Disc		Graft	
1	95.13 Sec.	3.28%	119.93 Sec.	3.16%
2	80.60 Sec.	4.26%	95.04 Sec.	3.85%
3	76.63 Sec.	4.78%	90.52 Sec.	4.68%
4	75.97 Sec.	5.64%	89.14 Sec.	5.43%
5	75.47 Sec.	5.80%	88.34 Sec.	5.70%
6	75.71 Sec.	6.36%	87.63 Sec.	5.72%
	Hollow		Threestars	
1	66.08 Sec.	3.03%	92.99 Sec.	3.10%
2	56.20 Sec.	4.43%	83.31 Sec.	4.55%
3	53.02 Sec.	4.84%	80.43 Sec.	5.38%
4	52.24 Sec.	5.60%	78.85 Sec.	5.60%
5	51.72 Sec.	5.85%	78.60 Sec.	6.26%
6	51.41 Sec.	5.82%	78.37 Sec.	6.38%

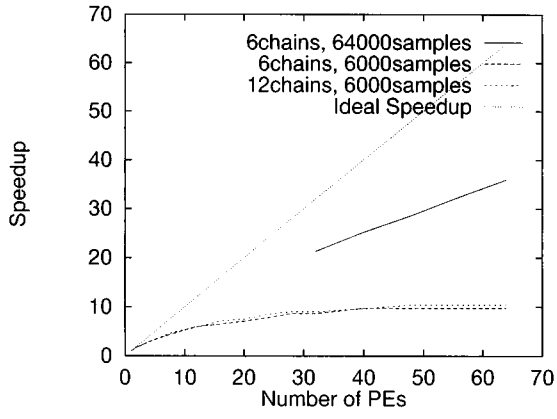


Fig. 6 Achieved speedup as a function of the number of processors for various conditions.

that the enrichment algorithm needs a large amount of memory because positions of every monomer in every sample conformations must be stored. This makes full size and practical simulation on one processing node of CM5 impossible. By this reason, speedup of large scale simulation was estimated by speedup of smaller scale simulation and elapsed time of both scale. The performance deviation from the ideal speedup in Fig. 6 for the case of full simulation (64,000 samples) can be due to such a way of estimation. Another possible reasons are the remaining imbalance of samples and several inherent parallel overheads come from CM5 environment.

Despite of that, it seems that an acceptable speedup and efficiency is obtained in parallel processing by 64 processors for the full simulation. On the other hand, in the reduced version, efficiency is significantly degraded and speedup is saturated at about 50 processors. Since precise physical result needs as many samples as possible, performance saturation at such a small number of processors is not a serious problem. In other words, the condition

$$\frac{\partial(\text{Time for parallel part})}{\partial(\text{Number of the processors})} < 0, \tag{10}$$

which is required by the Gustafson's Law [5] for high parallel efficiency, can be satisfied for any number of processors by making the size of the probrem sufficiently large.

5. Efficiency Analysis

In the previous chapters, our discussion is based on a prospect that the procedure consists of many independent tasks like our simulation is suitable for simple task-wise parallelization and must exhibit quite high efficiency by such way. In this last chapter we will discuss the point by creating a model of the procedure and estimating the elapsed time for them.

Single step of Phase3 and 4 in the description in Chapter 3 is simplified to a model procedure:

(A) Do N_{task} times of certain processing. They will succeed only in probability p. If succeeds each will produce a sample data set for the next phase.
(B) Evaluate certain function f for each of produced data set.

Between Phase A and B, no data is moved among processors, in other words an owner computing rule on task is employed in the model. It is assumed that elapsed time of Phase A and B do not depend on the samples and they are denoted as T_A and T_B, respectively.

The size of the problem must be defined by N_{task}. Because what we need is the evaluation of the function on samples as many as possible for precise Monte Carlo evaluation. When we need $N_{required}$ times sample evaluation, N_{task} must be $N_{required} \times (1/p)$.

Since there is no way to say which task will succeed in Phase A, the execution time of the model can be defined only as an expectation value. The expectation value of the time taken by a serial computer for entire process is

$$< T_{serial} > = T_A N_{task} + p \times T_B N_{task}. \tag{11}$$

Since T_A and T_B work as merely prefactors in this analysis, and in our formulation general tendency does not depend on the values of them, we assume that $T_A = 0$, $T_B = 1$. This extreme situation corresponds to the case of f with quite large complexity like the hydrodynamic radius mentioned in Chapter 2.

For a parallel computer with N_{pe} processors, elapsed time $T_{parallel}$ is calculated as follows, noting that $T_{parallel}$ is decided by the slowest processor. A processor finishes its Phase B in nT_B second at probability

$$C\left(\frac{N_{task}}{N_{pe}}, n\right) p^n (1-p)^{\frac{N_{task}}{N_{pe}} - n}. \tag{12}$$

By this, the probability of all processor finish their Phase B within nT_B and at least one processor finish it exactly in nT_B is given by

$$P(n) - P(n-1), \tag{13}$$

where $P(n)$ is

$$\left\{ \sum_{i=0}^{n} C\left(\frac{N_{task}}{N_{pe}}, i\right) p^i (1-p)^{\frac{N_{task}}{N_{pe}} - i} \right\}^{N_{pe}}. \tag{14}$$

The probability distribution of the parallel efficiency is obtained by this. We will use arbitrary values (0.5 and 0.1) of p as examples instead of that from actual simulations, because p is a complicated function of physical conditions. Small values of p are related to dense polymer solutions and we believe the hypothetical value $p = 0.1$ is typical or even too large in such cases. The result is given in Fig. 7 for several conditions. Smaller value of p gives rise to smaller number of tasks in Phase B on average, and larger variance of them compared to their average in this model. This model study

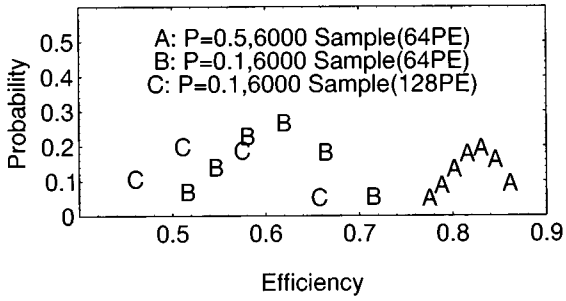


Fig. 7 Probability distribution of the efficiency for several conditions.

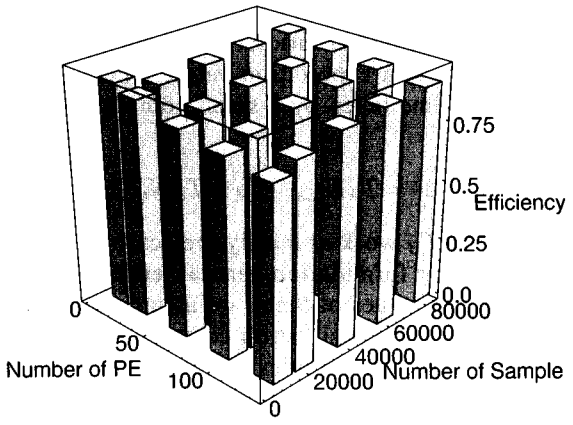


Fig. 8 The expectation value of the parallel efficiency as a function of problem size and the number of processors.

clearly shows that the efficiency can be quite small if p is small, despite of large inherent parallelism of the problem. The degradation of the efficiency is obviously comes from the fluctuations of the loads among processors, thus it become worse as p or N_{task} decreases. The result is partly consistent with the actual speedups presented in Fig. 6.

Finally, the value of parallel efficiency

$$\frac{< T_{serial} > /N_{pe}}{< T_{parallel} >} \tag{15}$$

at $p = 0.5$ is calculated as a function of problem size and N_{pe} and the result is given in Fig. 8.

This data will be useful in the new framework recently proposed for evaluating efficiency and scalability of parallel algorithm [7]. We may obtain their "isoefficiency-function" which is a size of a problem enough to keep the parallel efficiency constant as a function of the number of processors. The p-dependence of the isoefficiency-function of this algorithm will be quite interesting matter to discuss. However, we will keep this for our future theme and here confine ourselves to just mention the efficiency can be small, nevertheless the scalability of enrichment algorithm seems to be very nice.

6. Conclusion

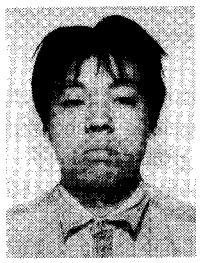
Numerical simulations of complicated polymer structures have been carried out. Result of parallel processing shows that in this application, what most affects the parallel efficiency is not the communication time, but the fluctuation of the randomly changing load. To improve the load balancing, a simple scheme based on the average load of each processor was introduced. A satisfactory efficiency was achieved by this scheme. We introduced a simple performance model for our code and analyze efficiency of the model. The model study made it clear that parallel efficiency depends on the fluctuation of the load and scalability of the algorithm is quite nice.

Acknowledgments

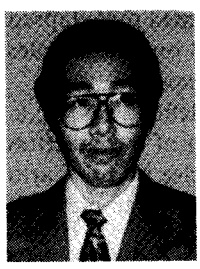
The authors would like to thank the Information Science Center at JAIST and the Information Science Group at IMR for their continuous support of computer facilities.

References

[1] K. Ohno and K. Binder, "Monte Carlo simulation of many-arm star polymers in two-dimensional good solvents in the bulk and at a surface," Journal of Statistical Physics, vol.64, pp.781–806, 1991.
[2] K. Ohno, K. Shida, M. Kimura, and Y. Kawazoe, "Monte Carlo study of the second virial coefficient of star polymers in a good solvent," Macromolecules, vol.29, pp.2269–2274, 1996.
[3] K. Shida, K. Ohno, M. Kimura, and Y. Kawazoe, "Large scale Monte Carlo simulations of center-adsorbed star polymers," Journal of Chemical Physics, vol.105, no.19, pp.8929–8937, 1996.
[4] S.K. Park and K.W. Miller, "Rundom number generators: Good ones are hard to find," Communications of the ACM, vol.31, pp.1192–1201, 1988.
[5] J.L. Gustafson, G.R. Montry, and R.E.Benner, "Development of parallel methods for 1,024-processor hypercube," SIAM J.Scientific and Statistical Computing, vol.9, no.4, 1988.
[6] D.E. Knuth, "The Art of Computer Programming," vol.2, Addison-Wesley, New York, 1981.
[7] A.Y. Grama, A. Gupta, and V. Kumar, "Isoefficiency function: A scalability metric for parallel and algorithms and architectures," IEEE Parallel & Distributed Technology, pp.12–21, Aug. 1993.

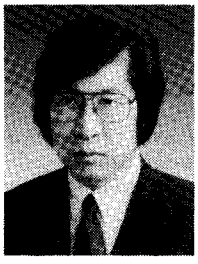


Kazuhito Shida received the BSc from Department of physics, Tohoku University, Sendai, Japan in 1991. He entered School of Information Science, Japan Advanced Institute of Science and Technology in 1992, and he began his research on parallel computing. He is now writing his doctoral thesis on the thema including the content of this paper.



Kaoru Ohno received the DSc from Department of physics, Tohoku University, Sendai, Japan in 1984. He has been a Research Associate of Physics Department, Tohoku University, from 1986 to 1990. He is now an Associate Professor in Institute of Materials Research, Tohoku University.

Masayuki Kimura received Dr.Eng. degree in electrical engineering, from Tohoku University, Sendai, Japan in 1959. He has been a Professor in the Fuculty of Engineering, Tohoku University, from 1970 to 1990. He is now a Professor in School of Information Science, Japan Advanced Institute of Science and Technology.



Yoshiyuki Kawazoe received the DSc from Department of physics, Tohoku University, Sendai, Japan in 1975. He has been a Associate Professor in Education Center of Information Processing, Tohoku University, from 1981 to 1990. Since 1990, he is a Professor in Institute of Materials Research, Tohoku University. He is interested in large scale computational physics and its parallelization.