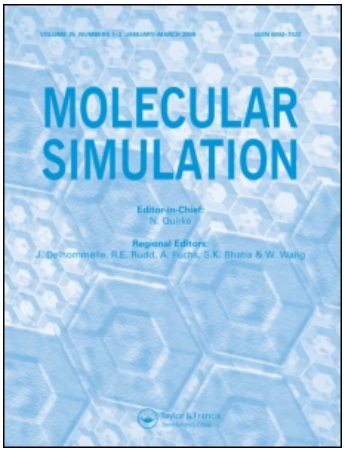

Molecular Simulation
Publication details, including instructions for authors and subscription information:
http://www.informaworld.com/smpp/title~content=t713644482

Efficiency of Parallel CBMC Simulations

Thijs J. H. Vlugt[a]
[a] Department of Chemical Engineering, University of Amsterdam, Amsterdam, The Netherlands

To cite this Article Vlugt, Thijs J. H.(1999) 'Efficiency of Parallel CBMC Simulations', Molecular Simulation, 23: 1, 63 — 78
To link to this Article: DOI: 10.1080/08927029908022112
URL: http://dx.doi.org/10.1080/08927029908022112

Molecular Simulation, 1999, Vol. 23, pp. 63–78
Reprints available directly from the publisher
Photocopying permitted by license only

EFFICIENCY OF PARALLEL CBMC SIMULATIONS

THIJS J. H. VLUGT*

Department of Chemical Engineering, University of Amsterdam, Nieuwe Achtergracht 166, 1018 WV Amsterdam, The Netherlands

(Received February 1999; accepted April 1999)

Configurational Bias Monte Carlo (CBMC) simulations are often used to compute thermodynamic properties of flexible chain molecules. These calculations are still computationally expensive. In this article we will present a parallel CBMC algorithm to reduce the total simulation time. We have tested our algorithm on several parallel computers and we obtained a good scaling for 16 processors.

Keywords: Configurational Bias Monte Carlo; parallel computing; molecular simulation

1. INTRODUCTION

Configurational Bias Monte Carlo (CBMC) [1–4] is an advanced Monte Carlo (MC) technique for simulating flexible chain-molecules. CBMC simulations have been used for the calculation of vapor-liquid equilibria of alkanes [5–9], for the simulation of adsorption of alkanes in porous structures [10–14] and for the simulation of linear and cyclic peptides [15]. In CBMC, a flexible chain molecule is grown segment by segment. For the growth of each segment, several trial orientations are generated of which one is chosen according to its Boltzmann factor. This introduces a bias,

*Fax: +31 20 525 5604, e-mail: tampert@its.chem.uva.nl

which can be removed exactly in the acceptance/rejection rule. It has been shown that CBMC is many orders of magnitude more efficient than an un-biased random growth of a chain-molecule.

The main limitation of CBMC is that for high densities and long chains such as polymers the acceptance probability of a MC trial-move drops. Therefor, CBMC simulations can be computationally quite expensive. Whereas previous implementations of CBMC algorithms have been limited to sequential machines, here we propose a parallel CBMC algorithm. This algorithm is a combination of two existing CBMC algorithms:

1. The Dual Cut-Off CBMC algorithm (DC-CBMC) by Vlugt *et al.* [16]. This algorithm uses an additional bias in the selection of trial segments. It can be shown that hard-core interactions dominate the selection of trial segments. Therefor, one can use a second potential cut-off radius (r_{cut^*}) for the selection of trial segments. The second cut-off radius can be chosen quite small, for example for a Lennard-Jones (LJ) fluid one can use $r_{cut^*} = \sigma$. The use of a second cut-off radius introduces an additional bias in the CBMC algorithm, which can be removed exactly in the acceptance/rejection rule.
2. The parallel CBMC algorithm by Esselink *et al.* [17]. In this algorithm, multiple chains are grown in each CBMC trial move to increase the acceptance probability. Because the growth of multiple chains is independent, this task can be easily distributed among processors. Important to note is that all simulations in the article of Esselink *et al.*, were performed on a single workstation and not on a parallel computer, although the title of that article suggests otherwise. Therefor it remains to be shown whether this algorithm will have a good parallel performance in practice.

In this article, we will not only show that on the basis of these two algorithms we can develop an efficient parallel code, but also that a combination of these algorithms is more efficient than one would predict on the basis of each algorithm independently.

At this point we would like to address the question why one should develop a parallel MC code since the most efficient way of doing MC after all is to use a different sequence of random numbers for each processor and average the results. We do not argue this may be the most efficient way, but not always the most convenient way. This is for example the case when a long equilibration of the system is needed.

2. PARALLEL CBMC

2.1. Introduction

In general, there are several criteria to design a successful parallel algorithm:

1. There should be a good load-balance, every processor should be doing roughly the same amount of work.
2. The amount of communication between processors should be minimized.

There are several reasons why the conventional CBMC algorithm [4] cannot be parallelized efficiently:

1. The growth of a chain molecule is a sequential process by nature.
2. For the selection of a new chain segment from several trial segments, Vlugt *et al.* [16] have shown that a very short potential cut-off radius can be used. This means that this calculation is computationally cheap and therefor it is not possible to parallelize this task efficiently on a large number of processors. Also, such a parallelization would imply that communication between processors is necessary for the growth of every segment of the chain.

To avoid these problems, Esselink *et al.* [17] have developed a CBMC algorithm in which multiple (independent) chains are grown instead of only one chain in the conventional CBMC algorithm. Out of these chains, one is selected according to its Rosenbluth weight and the other chains are thrown away. The effect of the use of multiple chain growth is that it is more likely that one chain is grown in a favorable position, resulting in a larger fraction of accepted trial-moves. This will be most effective when the fraction of accepted trial-moves is low. When the fraction of accepted trial-moves is high, the use of a larger number of trial chains (g) will have a small effect only. This can be demonstrated very easily for the growth of hard-sphere chains. When the probability of generating a chain with an overlap is equal to x and the number of chains that is grown in parallel is equal to g, the relative efficiency η_R (fraction of accepted trial moves per grown chain divided by the fraction of accepted trial-moves for $g = 1$) will be equal to

$$\eta_R(g, x) = \frac{1 - x^g}{(1 - x) * g} \tag{1}$$

This function is plotted in Figure 1. The relative efficiency will always decrease with increasing g. This decrease is less dramatic when the

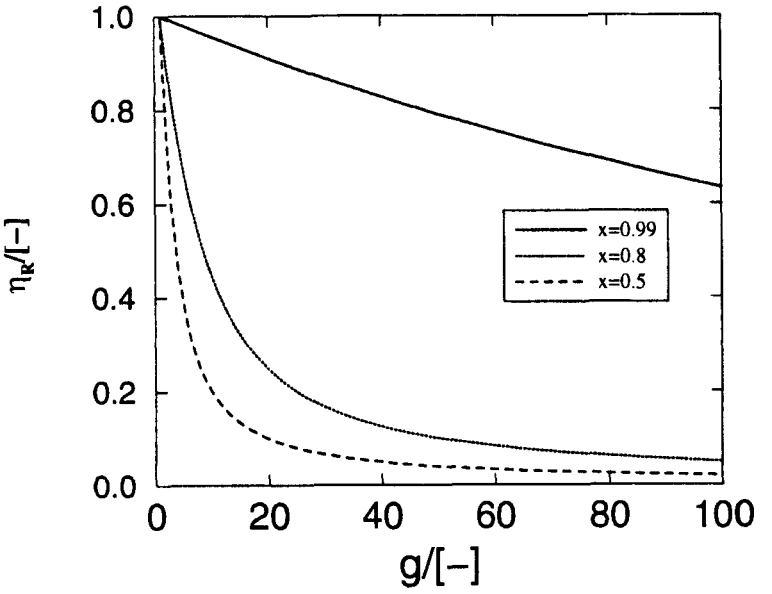

FIGURE 1 Relative efficiency (η_R) as a function of x and g for a hard-sphere chain according to Eq. (1).

probability of generating a chain with an overlap is high. This means that CBMC algorithms with a moderately high acceptance rate like recoil growth [18] will have a poor parallel performance.

2.2. Algorithm

The total external energy $\mathcal{U}$ can be split into three parts

$$\mathcal{U} = \overline{\overline{\mathcal{U}}} + \delta_1 + \delta_2 \tag{2}$$

For the growth of a chain, $\overline{\overline{\mathcal{U}}}$ is used only. δ_1 and δ_2 are correction terms. The meaning of δ_1 and δ_2 will be explained later. Note that this division is completely arbitrary.

We will use the symbol n for a new configuration and o for an old configuration. Repeatedly, the following steps are executed:

1. Among Q processors, g new (n) chains of length l are divided and grown using the standard CBMC algorithm. Note that to avoid load imbalance, mod $(g, Q) = 0$ must hold. For the selection of trial segments, $\overline{\overline{\mathcal{U}}}$ is used

only. This results in a Rosenbluth factor $\overline{\overline{W}}$ for each chain

$$\overline{\overline{W}} = \frac{[\sum_{j=1}^{j=f} \exp[-\beta \overline{\overline{U}}(j,1)]][\prod_{i=2}^{i=l} \sum_{j=1}^{j=k} \exp[-\beta \overline{\overline{U}}(j,i)]]}{f k^{l-1}} \tag{3}$$

In this equation, the number of trial-segments is equal to f for the first bead and k for the other beads and $\beta = 1/kT$.

2. For each chain that has been grown, δ_1 is calculated. This is a correction factor for the Rosenbluth weight $\overline{\overline{W}}$ of each chain:

$$\overline{W} = \overline{\overline{W}} \exp[-\beta \delta_1] \tag{4}$$

3. Out of the g chains, one chain (i) is selected according to

$$p_i = \frac{\overline{W}_i}{\sum_{j=1}^{j=g} \overline{W}_j} \tag{5}$$

4. For the selected chain, δ_2 is calculated, resulting in

$$Z(n) = \frac{\exp[-\beta \delta_2] \sum_{j=1}^{j=g} \overline{W}_j}{g} \tag{6}$$

5. For the old (o) configuration, a similar procedure is used to calculate $Z(o)$. Note that the first chain of the g chains is actually the old chain that is retraced using standard CBMC retracing. The remaining $g-1$ chains are new chains.
6. This trial-move is accepted with a probability

$$\text{acc}(o \to n) = \min\left(1, \frac{Z(n)}{Z(o)}\right) \tag{7}$$

In Appendix A, it is shown that this algorithm obeys detailed balance.

The excess chemical potential of a system is conventionally calculated using Widom's test particle method [4, 19, 20]. For the parallel CBMC algorithm described here it is straightforward to show that

$$\mu_{ex} = -\frac{\ln\langle Z^+ \rangle}{\beta} \tag{8}$$

in which the symbol + is used to denote an additional test chain.

2.3. Discussion

There are several limiting cases of this algorithm that are worth mentioning:

1. $\overline{\overline{U}} = \delta_1 = 0$ and $f = k = g = 1$. We obtain the standard Metropolis acceptance/rejection rule for a completely unbiased MC trial-move

$$\text{acc}(o \to n) = \min(1, \exp[-\beta(\delta_2(n) - \delta_2(o))]) \tag{9}$$

   Equation (8) will reduces to

$$\mu_{ex} = -\frac{\ln\langle \exp[-\beta U^+] \rangle}{\beta} \tag{10}$$

   which is the usual Widom test particle method [4, 19, 20]. The symbol U^+ represents the energy of a test particle.

2. $\delta_1 = \delta_2 = 0$ and $g = 1$. We obtain the standard CBMC acceptance/rejection rule

$$\text{acc}(o \to n) = \min\left(1, \frac{\overline{\overline{W}}(n)}{\overline{\overline{W}}(o)}\right) \tag{11}$$

   Equation (8) will reduce to

$$\mu_{ex} = -\frac{\ln\langle \overline{\overline{W}}^+ \rangle}{\beta} \tag{12}$$

   which is identical to the equation derived in [4]. The symbol $\overline{\overline{W}}^+$ represents the Rosenbluth factor of a test chain that is grown using CBMC.

3. $\delta_2 = 0$ and $g = 1$. We obtain the DC-CBMC acceptance/rejection rule [16].

4. $\delta_1 = \delta_2 = 0$ and $g \neq 1$. We obtain the acceptance/rejection rule for the parallel CBMC algorithm that was proposed by Esselink *et al.* [17]. Note that the algorithm of Esselink *et al.*, is slightly different because for the old configuration, only one chain is grown. The Rosenbluth weights of the remaining $g - 1$ chains of the old configuration are equal to the Rosenbluth weights of the $g - 1$ chains of the new configuration that are not chosen. This scheme also obeys detailed balance. The advantage of our scheme is that it is directly portable to various ensembles like the Grand-Canonical or Gibbs ensemble.

It is interesting to discuss the properties of δ_1 and δ_2 when $g \neq 1$.

1. δ_1 is a correction term that is calculated for each chain individually, which means g times. As chains are divided among processors, the calculation of δ_1 for a single chain is not performed in parallel. When $\delta_2 = 0$, the acceptance probability for a trial move will become one when $g \to \infty$.
2. The other correction term, δ_2, is calculated only for the selected chain, which means only once. This means that the calculation of δ_2 can be divided among processors. If the amount of CPU time required to calculate δ_2 is large and this calculation can not be parallelized, load-imbalance occurs. When $\delta_1 = 0$ and $\delta_2 \neq 0$, the acceptance probability will not be equal to one in the limit of $g \to \infty$.

We thus can conclude that putting the energy term into δ_1 instead of δ_2 increases both the acceptance probability and the CPU time. As the efficiency (η) of a MC algorithm is usually defined by the number of accepted trial-moves divided by the CPU time, the algorithm can be optimized by an intelligent division of the energy between δ_1 and δ_2. There are three parts of the total external energy which are usually put into δ_1 or δ_2:

1. LJ tail corrections. These are equal for all chains and also computationally inexpensive so they can be put safely into δ_2.
2. Fourier part of an Ewald summation [4, 20]. An Ewald summation is usually computationally expensive but it can be parallelized very easily. This term is usually not very different for all chains, so we recommend to put it into δ_2.
3. Intermolecular interactions. These can be usually split into a short-range part and a long-range part [16, 21, 22].

$$\mathcal{U} = \overline{\overline{\mathcal{U}}}(r < r_{\text{cut}^*}) + \delta_i(r_{\text{cut}^*} \leq r < r_{\text{cut}}) \tag{13}$$

in which $i = 1$ or $i = 2$. This division is quite similar to XI-RESPA/XO-RESPA explicit time-reversible integration algorithms derived by Martyna and co-workers [23]. We therefor will use the notation IN-DC-CBMC for $i = 1$ (long range interactions are calculated for each chain) and OUT-DC-CBMC for $i = 2$ (long range interaction are calculated for the selected chain only). Note that in RESPA algorithms a switching function is usually used to smoothen the transition from short-range interactions to long-range interactions. We found that for DC-CBMC the use of a switching function does not influence the performance of the algorithm.

An important quantity is the amount of CPU time for the growth of one chain divided by the total amount of CPU time that is spend in the calculation of $\delta_{1,2}$. When this ratio is low, one can grow multiple chains at almost no computational cost (because most time is spend on calculating $\delta_{1,2}$). This means that for increasing g the efficiency of the algorithm will increase instead of decrease (Fig. 1). This is for example the case for OUT-DC-CBMC with the use of a small value for $r_{cut'}$. We therefor predict that OUT-DC-CBMC will be more efficient than IN-DC-CBMC.

When the calculation of δ_1 and δ_2 is computationally expensive and not easy to calculate in parallel, IN-DC-CBMC will fail because to many correction terms have to be calculated and OUT-DC-CBMC will fail because the calculation of δ_2 can not be parallelized. In Appendix B, we present an algorithm that may solve this problem.

3. RESULTS AND DISCUSSION

To test the efficiency of the algorithm, simulations were performed for the regrowth of one *n*-hexane (C_6) molecule in the zeolite Silicalite. The zeolite was modeled as a rigid crystal. The number of units cells was chosen $2 \times 2 \times 4$, resulting in a system size of 40.044 Å × 39.798 Å × 53.532 Å. For hexane, a united atom model was used. All alkane–alkane and alkane–zeolite intermolecular interactions are described with a LJ potential with a cut-off radius of 13.8 Å. In the Dual Cut-Off scheme, all particles including the zeolite particles were placed on a 3D grid with a grid-size at least equal to the second cut-off radius. This grid has to be recalculated only after an accepted trial-move. Intramolecular interactions include a bond-stretching potential for two bonded atoms, a bond-bending potential for three successive atoms, a torsion potential for four successive atoms and a LJ potential for atoms that are separated by more than three bonds. The number of trial orientations was chosen to be 15 for the first bead and 10 to the remaining beads. These large numbers ensure that the chain growth will not be terminated due to an overlap of all beads, which may result in load imbalance. In addition to regrow trial moves, a very small amount of particle displacement and particle rotation trial moves was used. Further details of the forcefield and the system can be found in Refs. [13, 14, 16].

Our simulation code was written in FORTRAN77 and parallelized using MPI [24]. The following machines were used to test our parallel CBMC algorithm:

1. Sun Enterprice 3000 machine (emerald.epcc.ed.ac.uk) with 4 × 250 MHz UltraSPARC CPU's and 1 Gbyte shared memory. MPICH-1.1.1 was used as communication library.

2. IBM RS/6000 SP (isis.sp.sara.nl) with 76 nodes and 512 MB distributed memory per node. MPI-F was used as a communication library.
3. Cluster (16 nodes) of PC's (http://molsim.chem.uva.nl/cluster) with Intel PentiumII 350 MHz processors and 128 MB per node running Linux. LAM 6.1 was used as a communication library, the switches [-O-c2c-nger] were used for fast communication. The PC's were connected using a 100 Mbit Ethernet network and a 3COM switch.

In all our simulations, we have defined the efficiency of a simulation (η) as the number of accepted trial-moves divided by the CPU time. It is possible to make a direct comparison between different machines/simulations using η only. The relative efficiency η_R is defined as the efficiency divided by the efficiency when the number of grown chains is equal to the number of processors, *i.e.*, $g = Q$. The definition of relative efficiency given in Section 2 is consistent with this definition.

3.1. Second Cut-off Radius

In Figure 2, we have plotted the efficiency as a function of the second cut-off radius $r_{cut'}$ for $g = Q = 1$. For $r_{cut'} = 0$, completely unbiased chains are grown, resulting in a very low efficiency. When $r_{cut'} = r_{cut} = 13.8$ Å, we

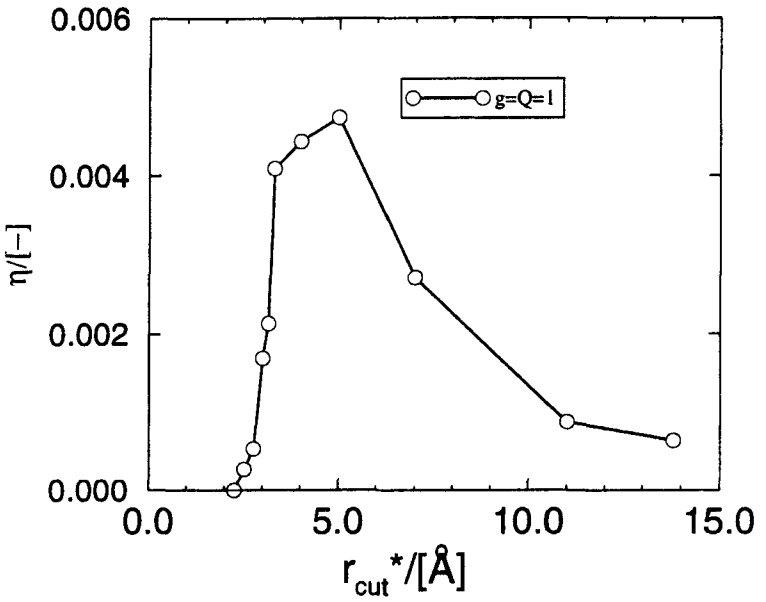

FIGURE 2 Efficiency (η) as a function of the second cut-off radius ($r_{cut'}$) for $g = Q = 1$. Simulations were performed on machine 2.

obtain the original CBMC algorithm. This is not very efficient because for the selection of a trial segment, a lot of long-range energy terms have to be calculated. As in the DC-CBMC scheme the long-range energy has to be calculated only for the selected configuration and not for every trial configuration, DC-CBMC is much more efficient. For a more in-depth discussion the reader is referred to [16].

3.2. Scaling of g

In Figure 3, we have plotted η as a function of g for various values of $r_{cut'}$ for simulations using 4 physical processors. All simulations in this subsection were performed on machine 1. It turns out that the OUT-DC-CBMC scheme is much more efficient than the IN-DC-CBMC scheme. As expected, decreasing the second cut-off radius $r_{cut'}$ increases η. Furthermore, there is a striking difference between IN-DC-CBMC and OUT-DC-CBMC. For IN-DC-CBMC, the efficiency always decreases with an increasing g, resulting in a relative efficiency η_R that is smaller than 1. This is in agreement with Eq. (1). However, for OUT-DC-CBMC and $r_{cut'} \approx 4.0\,\text{Å} - 4.5\,\text{Å}$, there is a maximum in the efficiency, resulting in a relative efficiency that is larger than 1. The reason for this is that here much more CPU time is spend in the calculation of the long-range part of the LJ potential (δ_2) for the selected chain than in the growth of a chain molecule. This means that growing extra chains does not increase the total CPU time too much, but it does increase the fraction of accepted trial moves resulting in an overall efficiency increase. The reason that the growth of a chain molecule is

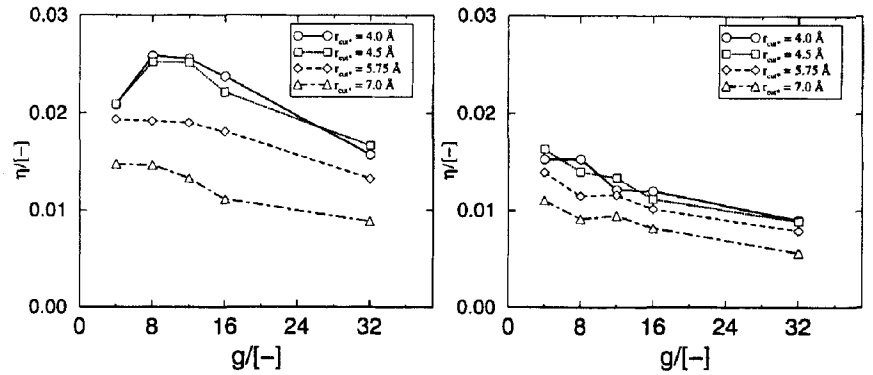

FIGURE 3 Efficiency (η) as a function of the number of chains (g) and the second cut-off radius ($r_{cut'}$). The number of processors was equal to 4. All simulations were performed on machine 1. Left: OUT-DC-CBMC; Right: IN-DC-CBMC.

computationally inexpensive is because of the small second cut-off radius, which explains why this effect only takes place at small $r_{cut'}$. When g becomes too large, the fraction of accepted trial moves hardly increases resulting in a decrease in efficiency. As the optimal efficiency will be at $g = 8$, running the simulation on more than 8 processors will result in an efficiency decrease.

Because of an efficiency increase instead of decrease for OUT-DC-CBMC with increasing g at a small $r_{cut'}$, we can conclude that a combination of multiple chain growth and the use of a second cut-off radius is more efficient that one would expect for these method individually.

3.3. Scaling of Q

To investigate the scaling of the algorithm with the number of processors, we have performed the same simulation on $Q = 1 \cdots 4$ processors while $g = 12$ and $r_{cut'} = 4.5$ Å on machine 1 and $g = 32$ on machines 2 and 3 ($Q = 1 \cdots 32$), see Figures 4 and 5. The reason to choose $g = 12$ on machine 1 is that no

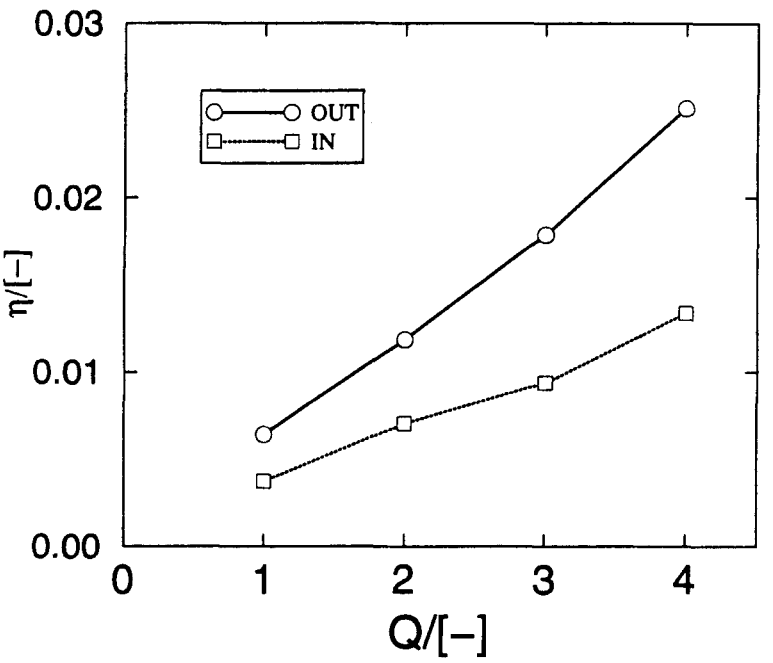

FIGURE 4 Efficiency (η) as a function of the number of processors (Q) for $r_{cut'} = 4.5$ Å and $g = 12$. Simulations were performed on machine 1.

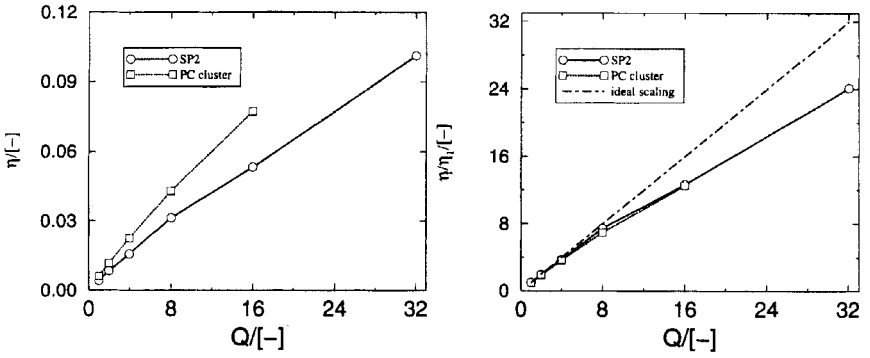

FIGURE 5 Left: Efficiency (η) as a function of the number of processors (Q) for $r_{cut} = 4.5$ Å and $g = 32$. Simulations were performed on machines 2 (SP2) and 3 (PC cluster). The OUT-DC-CBMC algorithm was used. Right: Same, but the efficiency is divided by the efficiency of a simulation on one processor.

load-imbalance will occur for $Q = 1 \cdots 4$. The use of a large number of chains ($g = 32$) for tests on machines 2 and 3 is to test the performance of the algorithm on a large number of processors. Both IN-DC-CBMC and OUT-DC-CBMC have a linear scaling for $Q \leq 4$ on machine 1. On machines 2 and 3, the algorithm scales ideal for $Q \leq 8$. For $Q = 16$ and $Q = 32$, there is a deviation of ideal scaling due to communication overhead.

4. CONCLUDING REMARKS

In summary, we have presented and tested an efficient parallel CBMC algorithm that performs well for 16 processors.

Acknowledgments

Financial support provided from NWO-CW. A large part of the computer resources were generously provided by EPCC (Edinburgh Parallel Computing Centre) and SARA (Stichting Academisch Rekencentrum Amsterdam). The author would like to thank Berend Smit, Peter Bladon, Nigel Wilding, Jean-Christophe Desplat and Jochem Wichers Hoeth for useful discussions.

APPENDIX A

In this Appendix, we will prove that our parallel CBMC algorithm obeys detailed balance. We start with our super-detailed balance [4] equation:

$$\sum_{b_n,b_o} \mathrm{acc}(o \to n) N(o) p_{\mathrm{gen}}(o \to n) = \sum_{b_o,b_n} \mathrm{acc}(n \to o) N(n) p_{\mathrm{gen}}(n \to o) \quad (14)$$

in which $N(i)$ is the probability of finding the system in state i,

$$N(i) \propto \exp[-\beta \mathcal{U}(i)] \quad (15)$$

acc $(a \to b)$ is the acceptance probability for a trial-move from a to b and p_{gen} $(a \to b)$ is the probability for selecting a trial-move from a to b. Super-detailed balance implies that every term on the l.h.s. of this equation is equal to each term of the r.h.s. of this equation. This leads to

$$\frac{\mathrm{acc}(o \to n)}{\mathrm{acc}(n \to o)} = \frac{\exp[-\beta \mathcal{U}(n)]}{\exp[-\beta \mathcal{U}(o)]} \times \frac{p_{\mathrm{gen}}(o \to n)}{p_{\mathrm{gen}}(n \to o)} \quad (16)$$

For p_{gen} $(o \to n)$ we can write

$$p_{\mathrm{gen}}(o \to n) = \frac{\exp[-\beta \overline{\overline{\mathcal{U}}}(n)]}{\overline{\overline{\mathcal{W}}}(n)} \times \frac{\overline{\overline{\mathcal{W}}}(n) \exp[-\beta \delta_1(n)]}{\sum_{i=1}^{i=g} \overline{\mathcal{W}}_i(n)} \quad (17)$$

Similar for p_{gen} $(n \to o)$

$$p_{\mathrm{gen}}(n \to o) = \frac{\exp[-\beta \overline{\overline{\mathcal{U}}}(o)]}{\overline{\overline{\mathcal{W}}}(o)} \times \frac{\overline{\overline{\mathcal{W}}}(o) \exp[-\beta \delta_1(o)]}{\sum_{i=1}^{i=g} \overline{\mathcal{W}}_i(o)} \quad (18)$$

Combining these equations and using $\mathcal{U} = \overline{\overline{\mathcal{U}}} + \delta_1 + \delta_2$ leads to

$$\frac{\mathrm{acc}(o \to n)}{\mathrm{acc}(n \to o)} = \frac{Z(n)}{Z(o)} \quad (19)$$

It is straightforward to see that Eq. (7) obeys this equation.

APPENDIX B

When the calculation of δ_1 and δ_2 is computationally expensive and not easy to calculate in parallel, IN-DC-CBMC will fail because to many correction terms have to be calculated and OUT-DC-CBMC will fail because

the calculation of δ_2 can not be parallelized. This might be true for the simulation of polarizable molecules [25], in which iterative scheme is used to calculated the polarization energy. In this case, the following algorithm can be used:

1. Among Q processors, g new (n) chains are divided and grown using the standard CBMC algorithm. Also assume that mod $(g, Q) = 0$. For the selection of trial segments, $\underline{\underline{U}}$ is used only. This results in a Rosenbluth factor $\underline{\underline{W}}$ for each chain.
2. On each processor a, one of the g/Q chains that was grown on processor a is chosen with a probability

$$p_i = \frac{\underline{\underline{W}}_i}{\sum_{j=1}^{j=g/Q} \underline{\underline{W}}_j} \tag{20}$$

   This is the main difference with the IN/OUT-DC-CBMC scheme in which one chain is chosen from all chains on all processors.
3. Each processor calculates δ_1 for the chain it has chosen. This leads to

$$\overline{W} = \exp[-\beta\delta_1] \sum_{j=1}^{j=g/Q} \underline{\underline{W}}_j \tag{21}$$

4. One of the processors is chosen with a probability

$$p_i = \frac{\overline{W}_i}{\sum_{j=1}^{j=Q} \overline{W}_j} \tag{22}$$

5. For the selected chain on the selected processor, δ_2 is calculated, resulting in

$$Z(n) = \frac{\exp[-\beta\delta_2] \sum_{j=1}^{j=Q} \overline{W}_j}{g} \tag{23}$$

6. For the old (o) configuration, a similar procedure is used to calculate $Z(o)$. Note that the first chain of the g chains is actually the old chain that is retraced using standard CBMC retracing. The remaining $g-1$ chains are new chains.
7. This trial-move is accepted with a probability

$$\text{acc}(o \to n) = \min\left(1, \frac{Z(n)}{Z(o)}\right) \tag{24}$$

For $Q = 1$ this algorithm will reduce to the OUT-DC-CBMC, while for $Q = g$ the algorithm reduces to the IN-DC-CBMC scheme. Note that in this algorithm the acceptance probability will be an explicit function of the number of processors, while this is not the case for OUT-DC-CBMC and IN-DC-CBMC. The proof that this algorithm obeys detailed balance is not given because it is almost similar to the proof in Appendix A. It is quite straightforward to construct variants of this algorithm, for example when a chain is chosen from chains that were grown on 2 processors. In general, one can construct its own algorithm:

1. Find a smart way to split the total external energy for a particular application.
2. Investigate if and how the calculation of the correction term δ_1 can be parallelized efficiently.
3. Construct a new algorithm by combining this information.
4. Prove that the algorithm obeys detailed balance.

References

[1] Siepmann, J. I. and Frenkel, D. (1992). "Configurational-bias Monte Carlo: A new sampling scheme for flexible chains", *Mol. Phys.*, **75**, 59–70.

[2] Frenkel, D., Mooij, G. C. A. M. and Smit, B. (1992). "Novel scheme to study structural and thermal properties of continuously deformable molecules", *J. Phys. Condens. Matter*, **4**, 3053–3076.

[3] de Pablo, J. J., Laso, M. and Suter, U. W. (1992). "Estimation of the chemical potential of chain molecules by simulation", *J. Chem. Phys.*, **96**, 6157–6162.

[4] Frenkel, D. and Smit, B. (1996). *Understanding Molecular Simulations: from Algorithms to Applications*. Academic Press, San Diego.

[5] Smit, B., Karaborni, S. and Siepmann, J. I. (1995). "Computer simulations of vapour-liquid phase equilibria of *n*-alkanes", *J. Chem. Phys.*, **102**, 2126–2140; Erratum (1998). *J. Chem. Phys.*, **109**, 352.

[6] Siepmann, J. I., Martin, M. G., Mundy, C. J. and Klein, M. L. (1997). "Intermolecular potentials for branched alkanes and the vapour liquid equilibria of *n*-heptane, 2-methyl-hexane, and 3-ethylpentane", *Mol. Phys.*, **90**, 687–693.

[7] Zhuravlev, N. D. and Siepmann, J. I. (1997). "Exploration of the vapour-liquid phase equilibria and critical points of triacontane isomers", *Fluid Phase Equilibria*, **134**, 55–61.

[8] Cui, S. T., Cummings, P. T. and Cochran, H. D. (1997). "Configurational Bias Gibbs ensemble simulation of Vapour-Liquid Equilibria of Linear and Short-Branched Alkanes", *Fluid Phase Equilibria*, **141**, 45–61.

[9] Martin, M. G. and Siepmann, J. I. (1998). "Transferable Potentials for Phase Equilibria (TraPPE): I. United-Atom Description of *n*-Alkanes", *J. Phys. Chem. B*, **102**, 2569–2577.

[10] Smit, B. and Maesen, T. L. M. (1995). "Commensurate "freezing" of alkanes in the channels of a zeolite", *Nature*, **374**, 42–44.

[11] Maginn, E. J., Bell, A. T. and Theodorou, D. N. (1995). "Sorption thermodynamics, siting and conformation of long *n*-alkanes in silicalite as predicted by configurational-bias Monte Carlo integration", *J. Phys. Chem.*, **99**, 2057–2079.

[12] Bandyopadhyay, S. and Yashonath, S. (1997). "Conformational analysis of *n*-butane in zeolite NaCaA: temperature and concentration dependenc", *J. Phys. Chem. B*, **101**, 5675–5683.

[13] Vlugt, T. J. H., Zhu, W., Kapteijn, F., Moulijn, J. A., Smit, B. and Krishna, R. (1998). "Adsorption of linear and branched alkanes in the zeolite silicalite-1", *J. Am. Chem. Soc.*, **120**, 5599–5600.
[14] Vlugt, T. J. H., Krishna, R. and Smit, B. (1998). "Molecular Simulations of Adsorption Isotherms for Linear and Branched Alkanes and their Mixtures in Silicalite", *J. Phys. Chem. B*, **103**, 1102–1118.
[15] Deem, M. W. and Bader, J. S. (1996). "A configurational bias Monte Carlo method for linear and cyclic peptides", *Mol. Phys.*, **87**, 1245–1260.
[16] Vlugt, T. J. H., Martin, M. G., Smit, B., Siepmann, J. I. and Krishna, R. (1998). "Improving the efficiency of the CBMC algorithm", *Mol. Phys.*, **94**, 727–733.
[17] Esselink, K., Loyens, L. D. J. C. and Smit, B. (1995). "Parallel Monte Carlo simulations", *Phys. Rev. E*, **51**, 1560–1568.
[18] Consta, S., Wilding, N. B., Frenkel, D. and Alexandrowicz, Z. (1999). "Recoil growth: an efficient simulation method for multi-polymer systems", *J. Chem. Phys*, **110**, 3220–3228.
[19] Widom, B. (1963). "Some topics in the theory of fluids", *J. Chem. Phys.*, **39**, 2802–2812.
[20] Allen, M. P. and Tildesley, D. J., *Computer Simulation of Liquids*. Clarendon Press, Oxford, 1987.
[21] Martin, M. G. and Siepmann, J. I. (1997). "Predicting Multicomponent Phase Equilibria and Free Energies of Transfer for Alkanes by Molecular Simulation", *J. Am. Chem. Soc.*, **119**, 8921–8924.
[22] Procacci, P. and Marchi, M. (1996). "Taming the Ewald sum in molecular dynamics simulations of solvated proteins *via* a multiple time step algorithm", *J. Chem. Phys.*, **104**, 3003–3012.
[23] Martyna, G. J., Tuckerman, M., Tobias, D. J. and Klein, M. L. (1996). "Explicit reversible integrators for extended systems dynamics", *Mol. Phys.*, **87**, 1117–1157.
[24] Forum, Message Passing Interface. "MPI: A message-passing interface standard", http://www.mpi-forum.org.
[25] Bernardo, D. N., Ding, Y., Krogh-Jespersen, K. and Levy, R. M. (1994). "An Anisotropic Polarizable Water Model: Incorporation of All-Atom Polarizabilities into Molecular Mechanics Force Fields", *J. Phys. Chem. B*, **98**, 4180–4187.