Available online at www.sciencedirect.com
SCIENCE DIRECT®
Journal of Computational Physics 212 (2006) 305–317
JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

Hybrid asynchronous algorithm for parallel kinetic Monte Carlo simulations of thin film growth

Yunsic Shim, Jacques G. Amar *

Department of Physics and Astronomy, University of Toledo, McMaster Hall, Mailstop 111, 2801 W. Bancroft Street, Toledo, OH 43606, USA

Received 10 February 2005; received in revised form 13 June 2005; accepted 4 July 2005
Available online 15 August 2005

Abstract

We have generalized and implemented the hybrid asynchronous algorithm, originally proposed for parallel simulations of the spin-flip Ising model, in order to carry out parallel kinetic Monte Carlo (KMC) simulations. The parallel performance has been tested using a simple model of thin-film growth in both 1D and 2D. We also briefly describe how the data collection must be modified as compared to the case of the spin-flip Ising model in order to carry out rigorous data collection. Due to the presence of a wide range of rates in the simulations, this algorithm turns out to be very inefficient. The poor parallel performance results from three factors: (1) the high probability of selecting a Metropolis Monte Carlo (MMC) move, (2) the low acceptance probability of boundary moves and (3) the high cost of communications which is required before every MMC move. We also find that the parallel efficiency in two dimensions is lower than in one-dimension due to the higher probability of selecting an MMC attempt, suggesting that this algorithm may not be suitable for KMC simulations of two-dimensional thin-film growth.

PACS: 81.15.Aa; 05.10.−a; 05.10.Ln; 89.20.Ff

Keywords: Parallel kinetic Monte Carlo; Hybrid asynchronous algorithm; Thin-film growth

1. Introduction

Recently there has been considerable interest [1–14] in the development of parallel algorithms for dynamical simulations of stochastic processes. One of the primary motivations is the desire to carry out

* Corresponding author. Tel.: +1 419 530 2259; fax: 1 419 530 2723.
E-mail address: jamar@physics.utoledo.edu (J.G. Amar).

0021-9991/$ - see front matter © 2005 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2005.07.005

simulations over larger length and time scales. For example, recently it has been shown that rigorous parallel Metropolis [15] Monte Carlo (MC) simulations may be carried out using a conservative asynchronous algorithm [1,2,9,11]. In such an algorithm, a 'local time condition' is used such that only processors whose next attempt time is less than their neighbor's next attempt times may proceed. Since in Metropolis Monte Carlo (MMC) the time of an attempt does not depend on the system configuration, this algorithm guarantees a rigorously correct evolution. In a series of interesting papers [9–11,13], Korniss and co-workers have applied this algorithm to spin-flip MMC simulations and investigated the connections between the scaling behavior of such an algorithm and the KPZ (Kardar–Parisi–Zhang) [16] and Edwards-Wilkinson [17] equations. They also showed that the use of additional 'local-time restrictions' in the form of a small-world network [10,11,13] can efficiently enhance the processor synchronization and thus reduce the memory requirements involved with data-taking.

In contrast to Metropolis Monte Carlo simulations, in kinetic Monte Carlo (KMC) simulations the time for an event depends on the system configuration. In particular, the time for an event executed by a processor may be affected by events in neighboring processors. As a result, the local time condition is not sufficient to guarantee an accurate parallel evolution. However, Lubachevsky [2] has developed, and Korniss et al. [3] have implemented a ''hybrid'' algorithm for parallel dynamical Monte Carlo simulations of the spin-flip Ising model. The basic idea is to apply Metropolis Monte Carlo dynamics to events on the boundary of a processor, but to accelerate interior moves by using the n-fold way [18] which is equivalent to kinetic Monte Carlo. While all KMC moves are immediately accepted, all Metropolis attempts must satisfy the 'local time restriction', i.e. wait until the neighboring processor's attempt time is later before being either accepted or rejected. Due to the presence of the Metropolis Monte Carlo region on the processor boundary, the propagation of fast events over several processors is prohibited, thus resulting in a correct time evolution. Since such an algorithm is equivalent to the conservative asynchronous algorithm for parallel MC, it is generally scalable [9,11,19]. In addition, it has been found to be relatively efficient in the context of kinetic Ising model simulations at intermediate temperatures in the metastable regime [3,20,21].

Here, we first discuss how such a ''hybrid'' algorithm may be used to carry out parallel KMC simulations of thin-film growth. In addition, we discuss the implementation and performance of this algorithm in simulations of a simple model of epitaxial growth in one- and two-dimensions. We note that although the hybrid algorithm is asynchronous, due to the fact that MPI-2 is not widely available, most of our results have been obtained by carrying out ''quasi-synchronous'' simulations using MPI-1. However, for comparison we also present the results of purely asynchronous simulations carried out using MPI-2 for the 1D case. In addition, we briefly discuss how the data collection algorithm used in parallel simulations of the spin-flip Ising model must be modified in order to guarantee rigorous data collection in parallel KMC simulations.

Our results may be summarized as follows. We find that, except for 1D simulations in which relatively large system sizes are typically used, for the typical parameters and processor and system sizes in epitaxial growth simulations, the parallel efficiency of the ''hybrid'' algorithm is significantly limited by three main effects: (1) the high probability of selecting a boundary MMC move which is due to the large range of rates in thin-film growth; (2) the low acceptance probability of boundary MMC moves (also due to the large range of rates in thin-film growth); and (3) the requirement of interprocessor communication before each boundary MMC move. As a result, the hybrid algorithm does not appear to offer an efficient simulation method for parallel KMC simulations of epitaxial growth except at relatively low temperatures. However, our results also indicate that if the interprocessor communication speed can be significantly increased, then reasonable parallel efficiencies may be obtained.

The organization of this paper is as follows. In Section 2, we first describe the epitaxial growth model that was used in our simulations. Section 3 contains a detailed description of the algorithm as well as a general description of the mapping that converts KMC to MMC in one- and two-dimensions. In Section 4, we present results for the parallel performance obtained using this algorithm for a simple model of thin-film growth. Finally, Section 5 contains a brief summary and discussion.

2. Growth model and spatial decomposition

In order to test the efficiency of the hybrid asynchronous algorithm in parallel kinetic Monte Carlo simulations, we have considered a simple model of epitaxial growth in which monomer deposition and diffusion are included but no island relaxation or detachment is allowed. We note that such a model is appropriate for low-temperature epitaxial growth. In this model, atoms (monomers) are deposited onto lattice sites with a (per site) deposition rate F, diffuse (hop) to nearest-neighbor sites with hopping rate D, and attach irreversibly to other monomers or clusters via a nearest-neighbor bond (critical island size of 1). Thus, the key parameter is the ratio D/F of the monomer hopping or diffusion rate D to the deposition rate F. We note that in typical simulations of epitaxial growth, $D/F \gg 1$ and for this reason KMC simulations are typically computationally demanding. In our simulations, the lattice configuration is represented by an array of heights (i.e., a solid-on-solid condition is assumed) while periodic boundary conditions are used.

In order to carry out parallel simulations of a 2D (1D) system of size $L_x \times L_y$ ($L_x \times 1$), domain decomposition using quasi-1D "strips" is used. As shown in Fig. 1, the system is divided along the x-direction into N_p subdomains of width $N_x = L_x/N_p$ and height L_y where N_p is the number of processors. The sites labelled 1 through N_x correspond to those belonging to processor i, while the sites whose numbers are in parentheses (i.e., sites 0 and $N_x + 1$) are in the "ghost region" of processor i, i.e., those sites belong to neighboring processors but must be updated by processor i through communication with the appropriate neighbor before carrying out any boundary moves. Because the range of interaction in our model is one lattice site, two ghost-sites are required in 1D while $2L_y$ ghost-sites are required in 2D. The arrows in Fig. 1 indicate the possible directions for a monomer to hop on a 2D lattice.

In our simulations the hybrid asynchronous algorithm was implemented using two different methods. Since MPI-1 is easily available and relatively easy to implement, in most of our simulations MPI-1 was used. However, since only two-way communications are allowed in MPI-1, in these simulations a quasi-synchronous method was used in which every processor must communicate with its two-neighboring processors before its next attempt or move regardless of whether or not it is a Metropolis attempt or a KMC move. Since this can lead to a large communication cost when KMC events dominate over MMC attempts, for comparison we also carried out some simulations using a purely asynchronous implementation via MPI-2. In these simulations only processors doing an MMC attempt must 'communicate' by accessing the open memory area of the relevant neighboring processor to get the necessary boundary information.

Our simulations were carried out on the SunFire 6800 cluster at the Ohio Supercomputer Center (OSC) which is a shared memory machine with 44 64-bit 900 MHz UltraSparc III processors. This machine was chosen because both MPI-1 and MPI-2 were available on this machine. We note that previous studies of the efficiency of parallel MMC algorithms [9–11] along with parallel simulations of the spin-flip Ising model using the hybrid algorithm [3] indicate that for the case of asynchronous communication the parallel efficiency is only weakly dependent on the number of processors. Accordingly, since typically only a few processors were available on this machine, most of our simulations were carried out using a relatively small value of N_p, i.e. $N_p = 4$. However, for our 1D simulations carried out using MPI-1 (two-way) communications, the dependence of the parallel efficiency on the number of processors was also studied using the Itanium cluster at OSC.

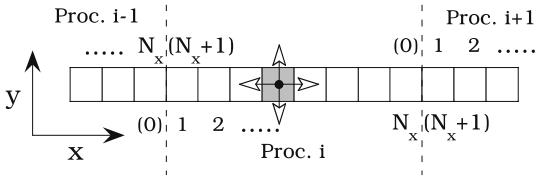

Fig. 1. Schematic diagram of quasi-1D strip decomposition for a 2D system. The sites 0 and $N_x + 1$ are in the "ghost region" of processor i and the arrows indicate the four possible directions for a monomer to hop on a 2D lattice.

Most of our simulations were carried out using a moderate value of D/F, i.e. $D/F = 10^5$. However, in order to determine the dependence of the parallel efficiency on D/F, some simulations were also carried out using a smaller value, i.e. $D/F = 10^3$ corresponding to lower temperature growth. We have also carried out simulations for a variety of different values of N_x in order to determine the dependence of the parallel efficiency on processor size.

As already noted, since the hybrid algorithm is asynchronous, at any given moment in real time the 'virtual time' of each processor may be slightly different. In simulations of the spin-flip Ising model using the hybrid asynchronous algorithm [3], this does not pose a problem for data-taking because each processor may save its configuration whenever its next-event time has just exceeded the desired data-collection time. Accordingly, the corresponding configurations may be re-grouped either during or after the run in order to collect data. However, due to the presence of diffusion in the models studied here, the boundary sites of a given processor (see Fig. 1) may be modified by a neighboring processor whose 'virtual time' is earlier. Thus, when a processor's next-event time exceeds the data-collection time, its boundary sites may not be up-to-date. This error may be corrected by keeping track of the 'last time' before the data-collection time at which each processor modifies its boundary or ghost sites. However, we note that in our MPI-1 simulations the possibility of such errors is relatively low since all processors communicate every cycle. In addition, in our purely asynchronous MPI-2 simulations the time fluctuations between processors is still relatively low due to the large fraction of MMC attempts. As a result, no significant errors were detected and for simplicity we have not included such a correction.

We note that to measure the parallel efficiency all our simulations corresponded to the deposition of 1 monolayer (ML). We also tested the accuracy of our parallel simulations by comparing results for the monomer and island densities with serial simulations. In order to ensure the independence of the random number sequences in each processor, each processor's random number generator was supplied with a different seed which was determined by the processor's identification number, at the beginning of each simulation. We now discuss how the hybrid asynchronous algorithm described in [3] for the case of the spin-flip Ising model may be generalized to apply to more general KMC models.

3. Hybrid asynchronous conservative algorithm

In the hybrid asynchronous algorithm each processor's domain is divided into two separate regions-a boundary or Metropolis Monte Carlo (MMC) region and an interior KMC region (see Figs. 2 and 3). Since the interior KMC moves are not affected by and do not affect the neighboring processors, all KMC moves are immediately accepted. However, since moves in the boundary region are affected by or can affect the neighboring processor, all MMC attempts must wait until the 'local time condition' that the next event time of the corresponding neighboring processor is later is satisfied, before being either rejected or accepted with

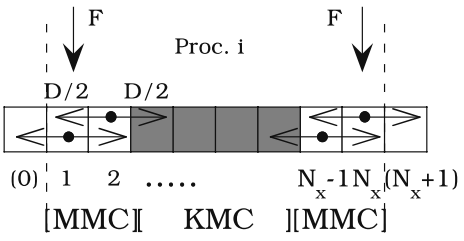

Fig. 2. Schematic diagram showing MMC region and interior KMC region in 1D. The sites 0 and $N_x + 1$ are in the "ghost region" of processor i. The horizontal arrows indicate the two possible directions for a monomer to hop on an 1D lattice and the vertical arrows indicate possible deposition at the sites 1 and N.

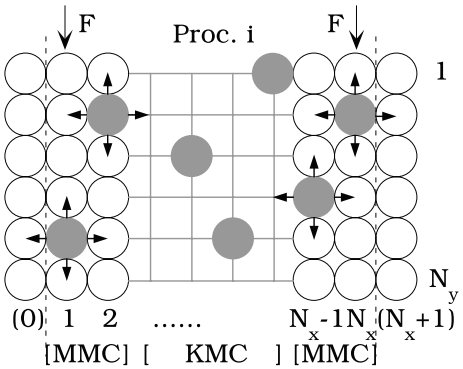

Fig. 3. Schematic diagram showing MMC region and interior KMC region in 2D. The sites 0 and $N_x + 1$ are in the "ghost region" of processor i. Filled circles denote actual particles in each region. Periodic boundary conditions are applied to along the N_y direction.

the appropriate acceptance probability. The choice of either a KMC or an MMC move is determined by the current configuration of each processor's domain.

In order to extend the hybrid asynchronous approach used in parallel Ising spin-flip simulations [3] to the case of parallel KMC simulations, we first carry out a mapping to Metropolis Monte Carlo. In the simplest approach, the KMC event rates R_i may be mapped to Metropolis acceptance probabilities $P^i_{\rm acc}$ by dividing the rate R_i for each possible event by the maximum possible single event rate $R_{\rm max}$ in our model, i.e., $P^i_{\rm acc} = R_i/R_{\rm max}$. We note that $R_{\rm max}$ is a fixed rate which is independent of the configuration and depends *only* on the model. The KMC simulation can then be replaced by a Metropolis simulation in which at each step one of the $N_{\rm MMC}$ possible MMC moves is selected randomly and then accepted with a probability given by $P^i_{\rm acc}$. We note that in the KMC simulation, the time interval between events is given by $\Delta t = -\ln(r)/R^{\rm KMC}_{\rm tot}$, where r is a uniform random number between 0 and 1 and $R^{\rm KMC}_{\rm tot} = \sum R_i$ corresponds to the sum of the rates of all possible events for a given system configuration. Accordingly, the average time before the next event for a given configuration is given by $\langle \Delta t \rangle = 1/R^{\rm KMC}_{\rm tot}$. Since the acceptance probability is one, the average event rate (i.e., the number of events per unit of simulated time) is given by $ER^{\rm KMC} = 1/\langle \Delta t \rangle = R^{\rm KMC}_{\rm tot}$ as expected. Similarly, in the corresponding MMC simulation the time interval between events is given by $\Delta t = -\ln(r)/(N_{\rm MMC} R_{\rm max})$ while a particular Metropolis event is selected with probability $1/N_{\rm MMC}$ and accepted with probability $P_{\rm acc} = R_i/R_{\rm max}$. Accordingly, the average event rate for the Metropolis simulation is the same as for the KMC simulation.

To reduce the rate of rejection due to MMC moves and thus improve the efficiency, the hybrid asynchronous algorithm is used, i.e., Metropolis moves in the interior are replaced by KMC moves. In the simplest possible implementation, the total rate for boundary MMC attempts is given by

$$R_{\rm bdy} = N_{\rm bdy} R_{\rm max}, \tag{1}$$

where $N_{\rm bdy}$ is a fixed number equal to the total number of possible different moves which may occur in the boundary region. As an example, for the 1D model considered here, $N_{\rm bdy} = 10$ since there are 8 possible boundary hops to the left or right and 2 possible deposition moves at the boundaries (see Fig. 2) while $R_{\rm max} = D/2$ corresponding to either a leftward or rightward hop. The probability of selecting a KMC event is then given by,

$$P_{\rm KMC} = R_{\rm KMC}/(R_{\rm KMC} + R_{\rm bdy}), \tag{2}$$

where $R_{\rm KMC}$ is the total rate of KMC moves in the interior region, and the probability of selecting a boundary MMC event is $P_{\rm bdy} = 1 - P_{\rm KMC}$. If a KMC event is chosen, the particular move carried out is selected with a probability proportional to its rate. In contrast, if a boundary event is chosen, the particular MMC

move is selected randomly from all possible boundary moves while the acceptance probability is given by $P^i_{\rm acc} = R_i/R_{\rm max}$ [22]. If a KMC move is selected then that move is performed immediately since it is not affected by neighboring processors. However, if a boundary MMC event is selected, then the move cannot be attempted until the time of the neighboring processor's next move is later. Since the time interval between events is given by

$$\Delta t = -\ln(r)/(R_{\rm KMC} + R_{\rm bdy}), \tag{3}$$

where r is a uniform random number between 0 and 1, one may show that the average event rate ER is the same as for the pure KMC or MMC algorithms.

While such an implementation of the hybrid asynchronous algorithm is more efficient than the pure Metropolis asynchronous algorithm, it has the disadvantage that for parallel KMC simulations in which there are many possible types of events, $N_{\rm bdy}$ will be large, and thus a boundary move will be selected with a high probability even though the acceptance probability will be low. A more efficient approach is to divide the boundary events into different types so that the total rate for boundary events is given by

$$R_{\rm bdy} = \sum_t N_t R_t, \tag{4}$$

where R_t is the rate for a single boundary event of type t and N_t is the number of possible boundary events of that type. We note that for each event type t, N_t is a fixed number which is independent of the configuration and depends *only* on the model and boundary geometry. If a boundary event is chosen, the type t of event is selected with a probability proportional to the total rate $N_t R_t$ for that type while the selected boundary move (chosen randomly from among the N_t moves of that type) is either accepted if it corresponds to an actual move or rejected if it does not, once the 'local time condition' is satisfied. The expressions for the probability of selecting a KMC or boundary event and for the time interval between events are the same as in Eqs. (2) and (3), but using this expression for $R_{\rm bdy}$. The advantage of this method is that even if there are a large number of different types of events, as long as these events have low rates compared to the maximum single event rate $R_{\rm max}$ they will only contribute weakly to the total rate $R_{\rm bdy}$ and thus the MMC rejection rate will be reduced. As an example in our 1D model, if the selected boundary move corresponds to a a hop of a monomer at site N_x-1 to the right, then once the 'local time condition' is satisfied, that move is accepted if there is a monomer present at site N_x-1, and is rejected if there is no monomer at site N_x-1. On the other hand if the selected boundary move corresponds to deposition at site N_x, then that move is immediately accepted. In contrast, using the simpler hybrid method described above, such a deposition move would only be accepted with probability $R_i/R_{\rm max} = 2F/D$.

For the 1D model considered here with $D/F \gg 1$, we have $R_{\rm max} = D/2$ corresponding to either a leftward or rightward hop, $N_D = 8$ since there eight possible boundary diffusion moves with rate $R_D = D/2$, and $N_F = 2$ since there are 2 possible boundary deposition moves with rate $R_F = F$. In this case $R_{\rm bdy} = 4D + 2F$ and $R_{\rm KMC} = (N_x-2)F + n_1 D$ where n_1 is the number of monomers in the interior region of the processor and N_x is the processor size. The probability of selecting a boundary move is then given by,

$$P_{\rm bdy} = \frac{4D + 2F}{4D + N_x F + n_1 D}. \tag{5}$$

Similarly, for the 2D model we have $R_{\rm max} = D/4$ corresponding to either a hop to the east, west, south, or north, $N_D = 16N_y$ since there are $16N_y$ possible boundary diffusion moves with rate $R_D = D/4$, and $N_F = 2N_y$ since there are $2N_y$ possible boundary deposition moves with rate $R_F = F$. In this case $R_{\rm bdy} = (4D + 2\ F)N_y$ and $R_{\rm KMC} = [(N_x-2)F + n_1 D]N_y$ and the probability of selecting a boundary move is again given by Eq. (5).

We note that the results presented here were all carried out using the basic hybrid algorithm Eq. (1) and as expected, no difference was found between the parallel and serial KMC results. Simulations were also

carried out using the more efficient approach (Eq. (4)). However, the parallel efficiency was found to be only slightly higher than obtained using the basic hybrid algorithm.

4. Parallel efficiency

Before presenting our results, we first discuss the factors which determine the parallel efficiency. We define the parallel efficiency PE as equal to the ratio of the execution time t'_{1p} for an ordinary KMC serial simulation of one processor's domain to the parallel execution time t_{N_p} of N_p domains using N_p processors, i.e.

$$PE = \frac{t'_{1p}}{t_{N_p}}. \tag{6}$$

Thus, the total speedup is equal to the PE multiplied by the number of processors N_p. We now discuss the different factors which may affect the PE.

We first note that the parallel execution time t_{N_p} for N_p processors can be written as

$$t_{N_p} = t_p^{MMC} + t_p^{KMC} + t_{com}, \tag{7}$$

where t_p^{MMC} is the average calculation time per processor taken for all MMC attempts, t_p^{KMC} is the average execution time per processor for all KMC events in the interior region, and t_{com} is the average communication time per processor. If we define the ratios $\kappa = \langle t_p^{MMC}\rangle/\langle t_p^{KMC}\rangle$ and $\eta = \langle t_p^{KMC}\rangle/\langle t'_{1p}\rangle$ then the theoretical parallel efficiency may be written as

$$PE = \frac{\langle t'_{1p}\rangle}{\langle t_{N_p}\rangle} \simeq \left[1 + \langle \Delta(\kappa,\eta)\rangle + \frac{t_{com}}{\langle t'_{1p}\rangle}\right]^{-1}, \tag{8}$$

where $\langle \Delta(\kappa,\eta)\rangle = \kappa\eta + \eta - 1$. If we assume that the communication time is negligible compared to $\langle t'_{1p}\rangle$, then the ideal parallel efficiency can be written as

$$PE^{ideal} = [1 + \langle \Delta(\kappa,\eta)\rangle]^{-1}. \tag{9}$$

We note that the ratio η may be conveniently calculated in terms of the numbers of different types of events in a parallel simulation, i.e., $\eta = N_p^{KMC}/N_{1p}^{KMC}$, where N_p^{KMC} is the average number of KMC events per processor in a parallel simulation and N_{1p}^{KMC} is the number of events in a one-processor KMC simulation of a single processor's domain, or equivalently, the total number of actual events per processor in a parallel simulation. In our parallel simulations we have found that this ratio is close to 1 (i.e., almost all real events are KMC events) and as a result $\eta \simeq 1$ and $\langle \Delta(\kappa,\eta)\rangle \simeq \kappa$.

In order to further understand the parallel performance of the algorithm, we have measured a variety of other quantities as defined in Table 1. In particular, in our parallel simulations we have measured the ratio ρ_{MMC} of the total number of MMC attempts N_{MMC} (where each check of the 'local time condition' corresponds to an MMC attempt regardless of whether or not it is satisfied) to the total number N_{tot} of MMC and KMC attempts, as well as the ratio $\rho_{KMC} = 1 - \rho_{MMC}$ corresponding to the ratio of the total number of KMC attempts to N_{tot}. Defining the ratio $\gamma \equiv \langle t_1^{MMC}\rangle/\langle t_1^{KMC}\rangle)$ of the average calculation time, including all attempts and waiting time, to carry out a single MMC event to the average calculation time to carry out a single KMC event, then we may write $\kappa \simeq \gamma\rho_{MMC}/\rho_{KMC}$. Thus assuming $\eta \simeq 1$ as in our simulations, for small values of ρ_{MMC}/ρ_{KMC} corresponding to a small fraction of MMC attempts, the ideal PE will be close to 1 while for large values of this ratio the ideal PE will be small.

We note that the ratio γ is a function of the processor width N_x and D/F and is determined by the balance between the quantities related to the MMC attempts defined in Table 1. For example,

Table 1
Quantities measured for parallel performance

Quantity	Definition
N_{tot}	Total number of MMC and KMC attempts
N_{MMC}	Total number of MMC attempts
N_{KMC}	Total number of KMC events carried out
ρ_{MMC}	$N_{\text{MMC}}/N_{\text{tot}}$
ρ_{KMC}	$N_{\text{KMC}}/N_{\text{tot}}$
ρ_{wait}	$N_{\text{wait}}/N_{\text{MMC}}$
ρ_{empty}	$N_{\text{empty}}/N_{\text{MMC}}$
ρ_{real}	$N_{\text{real}}/N_{\text{MMC}}$

for $D/F = 10^5$ we find $\gamma \simeq 0.33$ for 1D simulations with $N_x = 10^3$ and $\gamma \simeq 0.12$ in 2D with $N_x = 256$, $N_y = 8$. In order to better understand the efficiency of the MMC attempts one may also define the probabilities $\rho_{\text{wait}} = N_{\text{wait}}/N_{\text{MMC}}$, $\rho_{\text{empty}} = N_{\text{empty}}/N_{\text{MMC}}$, $\rho_{\text{real}} = N_{\text{real}}/N_{\text{MMC}}$, where $\rho_{\text{wait}} + \rho_{\text{empty}} + \rho_{\text{real}} = 1$ *and where* N_{wait} corresponds to the number of times the 'local time constraint' was checked and not satisfied, N_{empty} corresponds to the number of times an attempted Metropolis diffusion move which satisfied the local time constraint was rejected since no monomer was present, and N_{real} corresponds to the number of actual Metropolis events. Since the waiting time is the dominant time in the MMC attempts, to a good approximation one may write $\gamma \simeq c\rho_{\text{wait}}$, where c is only weakly dependent on parameters such as D/F and the system size N_x. In particular, we find that $c \simeq 0.6$ in 1D and $c \simeq 0.2$ in 2D.

5. Results

5.1. 1D Model

Fig. 4 shows the measured and ideal parallel efficiencies for this case as a function of the system size N_x for $D/F = 10^5$. As can be seen, for small N_x the ideal PE is close to 0.1 since in this case the KMC rate R_{KMC} is small compared to the MMC rate R_{bdy}. As a result the probability of selecting an MMC attempt (see Eq. (5)) is large. However, with increasing N_x the ideal PE increases, approaching 1 for $N_x > 10^4$. This is consistent with the increase in the ratio $\rho_{\text{KMC}}/\rho_{\text{MMC}}$ of KMC events to MMC attempts with increasing processor size as shown in Fig. 4(a). We note that the ideal parallel efficiency was calculated using Eq. (9) along with directly measured values for η and the approximate form $\kappa \simeq 0.6(\rho_{\text{MMC}}/\rho_{\text{KMC}})\rho_{\text{wait}}$. Also shown in Fig. 4 is the measured PE using MPI-1 for a smaller value of D/F ($D/F = 10^3$). Due to the decreased probability of selecting an MMC attempt, the parallel efficiency is slightly higher for $D/F = 10^3$ than for $D/F = 10^5$.

For large N_x the ideal PE approaches 1. However, due to the cost of communications the measured parallel efficiencies shown in Fig. 4 (b) are significantly lower. In particular, for the MPI-1 simulations, in which communications are required every cycle, the measured PE is significantly lower than the ideal PE due to the cost of communications. For the MPI-2 case the PE is larger (especially for large N_x) since communications are only required for MMC events. However, in this case the PE is still relatively small (approximately 0.2) for $N_x = 10^5$ although it appears to be still increasing with increasing N_x. The relatively slow increase in the measured PE with N_x for large N_x appears to be related to the fact that as N_x increases, the number of MMC attempts also increases despite the decreased probability of selecting an MMC event. In particular, the increased fraction of KMC attempts leads to decreased synchronization between neigh-

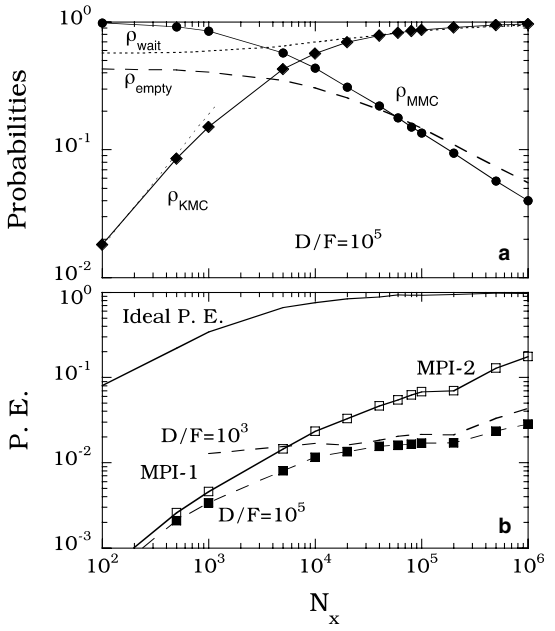

Fig. 4. (a) Event-selection probabilities and (b) parallel efficiency as function of processor size N_x in 1D where $N_p = 4$ and $\theta = 1$ ML. In (b) solid line with open squares corresponds to measured PE using MPI-2 for $D/F = 10^5$.

boring processors. As a result the value of ρ_{wait} increases monotonically with N_x as shown in Fig. 4(a). We note that such an effect might be reduced by using small-world communications to enhance synchronization as described in [11]. However, such an implementation may also add significant communications overhead.

We have also measured the parallel efficiency for the 1D case as a function of D/F ($N_x = 10^5$) as shown in Fig. 5. We note that this system size is typical of that usually used in KMC simulations of 1D epitaxial growth models, since large system sizes are needed to obtain reasonable statistics. Since N_x is large, the ideal PE remains close to 1 even for large D/F. However, as before, due to communication delays the measured values of the parallel efficiency are significantly lower. In addition, the measured PE decreases with increasing D/F since the decrease in the monomer density leads to an increase in the fraction of MMC attempts (ρ_{MMC}) as well as in the fraction (ρ_{empty}) which are 'rejected' (see Fig. 5(a)). Again the parallel efficiency obtained using MPI-2 is significantly larger than that obtained using MPI-1.

Finally, we consider the dependence of the parallel efficiency on the number of processors. As already noted, in previous work using asynchronous communication [3,9–11] the parallel efficiency was found to depend only weakly on the number of processors. Thus we expect that the PE obtained in our MPI-2 simulations should decrease only slightly as the number of processors increases. However, for the MPI-1 simulations this is not necessarily the case. To illustrate this, Fig. 6 shows the N_p-dependence of the parallel efficiency obtained from MPI-1 simulations of the 1D model with $D/F = 10^5$. The processor size $N_x = 10{,}000$ was chosen such that the probability of picking an interior KMC event is approximately equal to the probability of selecting a Metropolis boundary event. As can be seen, the parallel efficiency decreases from $PE \simeq 0.01$ for $N_p = 4$ to $PE \simeq 0.003$ for $N_p = 100$, although it appears to saturate for large N_p. This behavior is similar to that observed previously in simulations using the synchronous sublattice algorithm [6] and is due to local fluctuations in the time horizon which can cause delays when using two-sided communication. While such fluctuations initially increase with N_p they eventually saturate for large N_p since only local fluctuations are involved. Also shown in Fig. 6 is the ideal parallel efficiency which was calculated

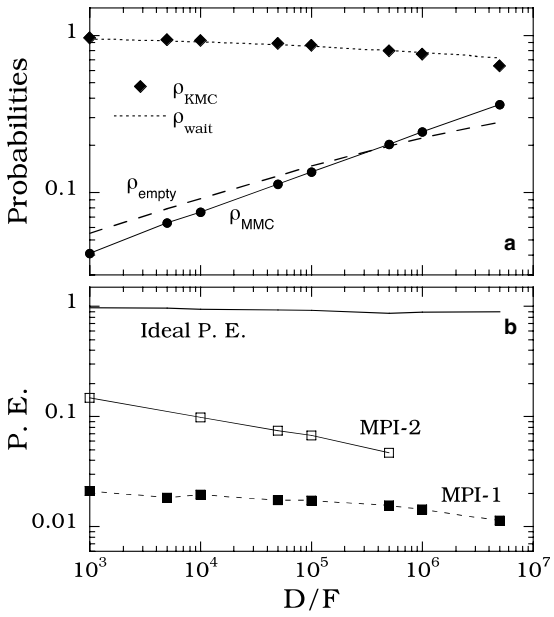

Fig. 5. (a) Probabilities and (b) parallel efficiency as a function of D/F in 1D with $N_p = 4$, $N_x = 10^5$, and $\theta = 1$ ML.

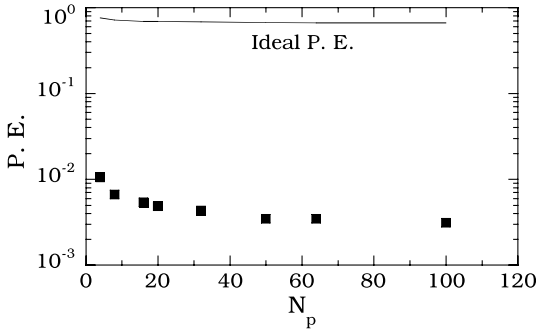

Fig. 6. Dependence of parallel efficiency on N_p for 1D model simulated using MPI-1. Parameters used are $D/F = 10^5$ and $N_x =$ 10,000. Solid line corresponds to ideal PE.

assuming no communication delay. As can be seen the ideal PE is more than two orders of magnitude larger, saturating at a value $PE \simeq 0.7$ for large N_p.

5.2. 2D Model

Fig. 7 shows our MPI-1 results for the PE of the 2D model with $D/F = 10^5$, $N_p = 4$ and fixed system width $N_x = 256$ as a function of system height N_y. We note that the selected values of N_x and D/F correspond to typical values used in KMC simulations of 2D epitaxial growth. As expected the parallel efficiencies and measured probabilities are essentially independent of N_y. However, because of the relatively small value of N_x compared to the 1D case, the probability ρ_{MMC} of selecting an MMC attempt is very close to 1. As a result, the ratio ρ_{MMC}/ρ_{KMC} is large and since $\eta \simeq 1$ the ideal PE is significantly lower than 1. Due to communication delays, the measured parallel efficiencies are even lower (of the order of 10^{-4}–10^{-3}).

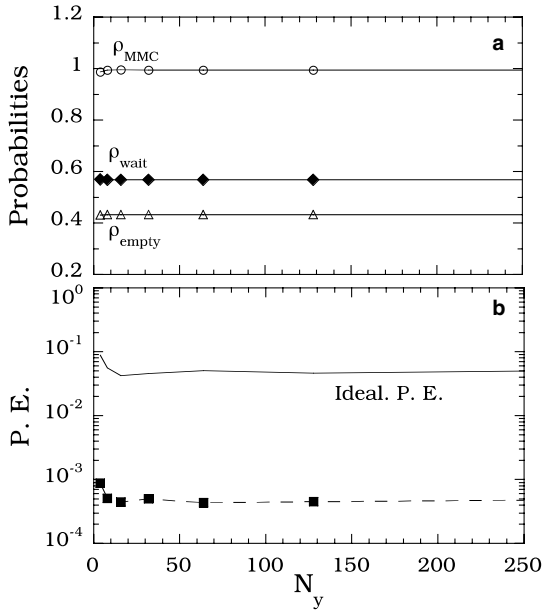

Fig. 7. (a) Probabilities and (b) parallel efficiency in 2D as a function of N_y. Here, $N_x = 256$, $N_p = 4$, $D/F = 10^5$, and $\theta = 1$ ML.

Fig. 8 shows the corresponding results for the parallel efficiency as a function of D/F with fixed processor size $N_x = 256$. Here for convenience we have chosen a small value of N_y since the PE does not depend on N_y. As for the 1D case we find that the measured PE decreases with increasing D/F since this leads to an increased probability of selecting an MMC attempt and an increased probability of rejection. The

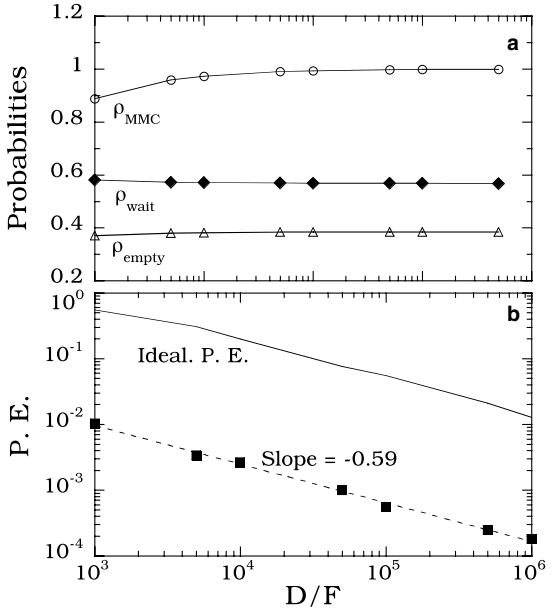

Fig. 8. (a) Probabilities and (b) parallel efficiency in 2D as a function of D/F. Here, $N_x = 256$, $N_y = 8$, $N_p = 4$, and $\theta = 1$ ML.

dependence of the ideal and measured PE's on D/F ($PE \sim (D/F)^{-0.6}$) may also be explained by the well known result for a critical island size of 1 that the monomer density decays as $N_1 \sim (D/F)^{-2/3}$ thus leading to an increase in the MMC 'rejection' rate. We note that while somewhat larger parallel efficiencies are likely to be obtained using MPI-2, we expect that due to the small values of N_x the resulting PE's are still likely to be relatively low. As a result, we did not carry out 2D simulations using MPI-2.

6. Summary and discussion

We have generalized and implemented the hybrid asynchronous algorithm in order to carry out parallel kinetic Monte Carlo (KMC) simulations and tested the parallel performance using a simple model of thin-film growth in both 1D and 2D. In particular, we have described how the hybrid asynchronous algorithm for parallel dynamic Monte Carlo described in [3] for the spin-flip Ising model may be generalized to apply to parallel KMC simulations of thin-film growth. In addition, we have shown that there exist two possible implementations, one corresponding to a "simple" approach in which the probability of selecting a boundary event is proportional to the number of different types of moves in the model, and a somewhat more efficient approach in which the probability depends on the rates of each type of move. We have also briefly described how the data collection must be modified as compared to the case of the spin-flip Ising model [3] in order to carry out rigorous data collection using such an asynchronous algorithm.

In general, for the case of 1D simulations with a typical, large system size, we find that the parallel efficiency is reasonably high, since the probability of selecting an MMC move is relatively low. In addition, we find that the use of MPI-2 (pure asynchronous dynamics) significantly increases the PE due to the fact that communications are only required when an MMC attempt is made. In contrast, in our 2D simulations we find that the parallel efficiency is quite low. In particular, due to the relatively small system sizes used in typical 2D KMC simulations, along with the relatively high range of rates in KMC simulation models, there is a high probability of selecting an MMC move. Due to the 'local time condition' as well as the fact that the probability for a selected MMC move to correspond to an actual event is relatively low, the ideal PE is significantly less than 1. In addition, since every MMC move requires communications, the actual PE is even further reduced. These results suggest that in general this algorithm is not suitable for parallel kinetic Monte Carlo simulations of two-dimensional thin-film growth.

Acknowledgements

This research was supported by the NSF through Grant No. DMR-0219328. We also acknowledge grants of computer time from the Ohio Supercomputer Center (Grant No. PJS0245) and the Pittsburgh Supercomputer center (Grant No. DMR030007JP).

References

[1] K.M. Chandy, J. Misra, IEEE Trans. Software Eng. 5 (1979) 440;
J. Misra, ACM Comput. Surv. 18 (1986) 39.
[2] B.D. Lubachevsky, Complex Syst. 1 (1987) 1099;
B.D. Lubachevsky, J. Comput. Phys. 75 (1988) 103.
[3] G. Korniss, M.A. Novotny, P.A. Rikvold, J. Comput. Phys. 153 (1999) 488.
[4] S.G. Eick, A.G. Greenberg, B.D. Lubachevsky, A. Weiss, ACM Trans. Model. Comput. Simul. 3 (1993) 287.
[5] B.D. Lubachevsky, A. Weiss, in: Proceedings of the 15th Workshop on Parallel and Distributed Simulation (PADS'01) IEEE, 2001.

[6] Y. Shim, J.G. Amar, Phys. Rev. B71 (2005) 115436.
[7] Y. Shim, J.G. Amar, Phys. Rev. B 71 (2005) 125432.
[8] N. Haider, S.A. Khaddaj, M.R. Wilby, D.D. Vvedensky, Comp. Phys. 9 (1995) 85.
[9] G. Korniss, Z. Toroczkai, M.A. Novotny, P.A. Rikvold, Phys. Rev. Lett. 84 (2000) 1351.
[10] B. Kozma, M.B. Hastings, G. Korniss, Phys. Rev. Lett. 92 (2004) 108701.
[11] G. Korniss, M.A. Novotny, H. Guclu, Z. Toroczkai, P.A. Rikvold, Science 299 (2003) 677.
[12] A. Kolakowska, M.A. Novotny, G. Korniss, Phys. Rev. E 67 (2003) 046703.
[13] H. Guclu, G. Korniss, Phys. Rev. E 69 (2004) 065104(R).
[14] B.D. Lubachevsky, V. Privman, S.C. Roy, J. Comput. Phys. 126 (1996) 152.
[15] N.C. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, E. Teller, J. Chem. Phys. 21 (1953) 6.
[16] M. Kardar, G. Parisi, Y.C. Zhang, Phys. Rev. Lett. 56 (1986) 889.
[17] S.F. Edwards, D.R. Willkinson, Proc.R. Soc. London Ser., A 381 (1982) 17.
[18] A.B. Bortz, M.H. Kalos, J.L. Lebowitz, J. Comput. Phys. 17 (1975) 10.
[19] G. Korniss, M.A. Novotny, Z. Toroczkai, P.A. Rikvold, Computer Simulated Studies in Condensed Matter Physics XIII, in: D.P. Landau, S.P. Lewis, H.-B. Schuttler (Eds.), Springer Proceedings in Physics, vol. 86, Springer-Verlag, Heidelberg, 2001.
[20] G. Korniss, C.J. White, P.A. Rikvold, M.A. Novotny, Phys. Rev. E 63 (2001) 016120.
[21] G. Korniss, P.A. Rikvold, M.A. Novotny, Phys. Rev. E 66 (2002) 056127.
[22] We note that if the particular MMC move chosen corresponds to the diffusion of a monomer at a given site say to the right, and there is no monomer at that site, then $R = 0$ and $P_{\mathrm{acc}} = 0$. For large D/F the monomer density is low, and so this may occur quite frequently.