Recent advances in parallel kinetic Monte Carlo: synchronous sublattice algorithm

Y. Shim and Jacques G. Amar

Department of Physics & Astronomy, University of Toledo, Toledo, OH 43606

Abstract. An efficient, semi-rigorous synchronous algorithm for parallel kinetic Monte Carlo simulations is presented. The accuracy and parallel efficiency are studied as a function of processor size, number of processors, and the ratio D/F of monomer hopping rate (D) to deposition rate (F) for two different simple models of epitaxial growth. Since only local communication is required, the algorithm scales, i.e. for a large number of processors the parallel efficiency is independent of the number of processors.

1 Introduction

While kinetic Monte Carlo (KMC) is an extremely efficient method [1–4] to carry out dynamical simulations of stochastic and/or thermally activated processes when the relevant activated atomic-scale processes are known, the standard KMC algorithm is inherently a serial algorithm since only one event can occur at each step and its next event time is determined by the total overall rate for all processes to occur. However, for some problems one needs to simulate larger length and time-scales than can be simulated using a serial algorithm. Thus, it would be desirable to develop efficient parallel kinetic Monte Carlo algorithms so that many processors can be used simultaneously in order to carry out simulations over extended time- and length-scales.

Unlike KMC, the attempt time in Metropolis Monte Carlo (MMC) [5] is independent of system configuration and thus, an asynchronous "conservative" algorithm may be used [6–9]. In such an algorithm all processors whose next attempt time is less than their neighbor's next attempt times are allowed to proceed. Unfortunately such a "conservative" algorithm does not work for kinetic Monte Carlo. Since fast events may "propagate" across processors, the time for an event already executed by a processor may change due to earlier events in nearby processors, thus leading to an incorrect time evolution. One may also consider a hybrid version of the conservative asynchronous algorithm which has been developed by Lubachevsky [7] in the context of Ising simulations. In this algorithm, "n-fold way" [1] simulations are carried out in the interior of each processor, while Metropolis Monte Carlo simulations are carried out at the boundary. Although the presence of the boundary regions leads to a correct time evolution in a processor, because of the possibility of significant rejection of boundary events, the parallel efficiency may be very

low for problems with a wide range of rates for different processes. In addition, due to the asynchronous nature of the algorithm, the parallel efficiency is further reduced by the complexity involved in data taking.

In order to address these problems, we present a simple synchronous sublattice (SL) algorithm for parallel kinetic Monte Carlo simulations. While the SL algorithm is not rigorous, we find that using certain reasonable assumptions on the cycle length and processor size, the results obtained are identical to those obtained in serial simulations and in particular, the algorithm is very efficient. We also find that the parallel efficiency is essentially independent of the number of processors in the large N_p limit, thus leading to linear scaling.

2 Synchronous Sublattice Algorithm

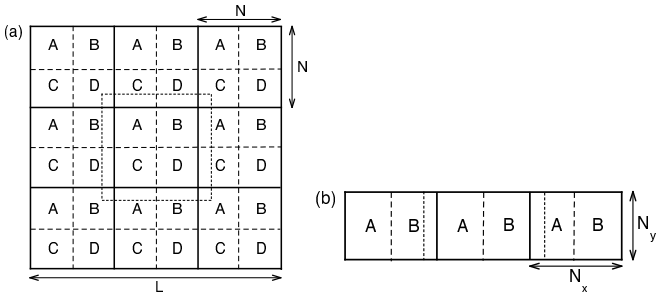

Fig. 1. Two possible methods of spatial and sublattice decomposition. (a) square sublattice decomposition (9 processors) and (b) strip sublattice decomposition (3 processors). Solid lines correspond to processor domains while dashed lines indicate sublattice decomposition. Dotted lines in (a) and (b) indicate "ghost-region" surrounding central processor.

In the synchronous sublattice (SL) algorithm, different parts of the system are assigned via spatial decomposition to different processors. However, in order to avoid conflicts between processors due to the synchronous nature of the algorithm, each processor's domain is further divided into different regions or sublattices (see Fig. 1). At the beginning of a cycle each processor's local time is initialized to zero. One of the sublattices is then randomly selected so that all processors operate on the same sublattice during that cycle. Each processor then simultaneously and independently carries out KMC events in the selected sublattice until the time of the next event exceeds the time interval T. As in the usual serial KMC, each event is carried out with time increment $\Delta t_i = -\ln(r_i)/R_i$ where r_i is a uniform random number between 0 and 1 and R_i is the total KMC event rate. Each processor then communicates any necessary changes (boundary events) with its neighboring processors, updates its event rates and moves on to the next cycle using a

new randomly chosen sublattice. Sublattice selection can be carried out by seeding all processors with the same random number generator so that they all independently select the same sublattice for each cycle. By picking the cycle length T less than or equal to the average time for the fastest possible activated event we do indeed obtain results which are identical to those obtained in serial KMC except for very small sublattice sizes.

3 Results

In order to test the performance and accuracy of our synchronous sublattice algorithm, we have studied two different solid-on-solid (SOS) growth models on a square lattice: a "fractal" growth model and an edge-and-corner diffusion (EC) model with periodic boundary conditions. In the "fractal" model [10], atoms (monomers) are deposited onto a square lattice with deposition rate F, diffuse to nearest-neighbor sites with hopping rate D and attach irreversibly to other monomers or clusters via a nearest-neighbor bond. The EC model is the same as the fractal model except that island relaxation is allowed, i.e. atoms which have formed a single nearest-neighbor bond with an island may diffuse along the edge of the island with diffusion rate $D_e = r_e D$ and around island-corners with rate $D_c = r_c D$.

For the fractal model, the range of interaction corresponds to one nearest-neighbor (lattice) spacing, while for the EC model it corresponds to the next-nearest-neighbor distance. Thus, for these models the width of the "ghost-region" corresponds to one lattice-spacing. If a particle diffuses from the boundary region of a processor into its ghost-region during a cycle, then that particle is no longer free to move during that cycle. We have carried out both serial serial and parallel simulations using the Itanium clusters at the Ohio Supercomputer Center (OSC) as well as on the Alpha cluster at the Pittsburgh Supercomputer Center (PSC).

Fig. 2 shows a comparison of parallel and serial results for the fractal model with $D/F = 10^5$ and a square system of size $L = 256$. The parallel simulations were carried out using a square sublattice decomposition with processor sizes $N = 32, 64$ and 128 corresponding to $N_p = 64, 16$, and 4 respectively, where N_p is the number of processors. In particular, Fig. 2(a) shows the substrate monomer and island densities as a function of coverage θ, while Fig. 2(b) shows the surface width as a function of coverage. As can be seen, there is no difference within statistical error between the serial and the parallel results.

We define the parallel efficiency PE as equal to the ratio of the execution time t'_{1p} for an ordinary serial simulation of one processor's domain to the parallel execution time t_{N_p} of N_p domains using N_p processors,

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

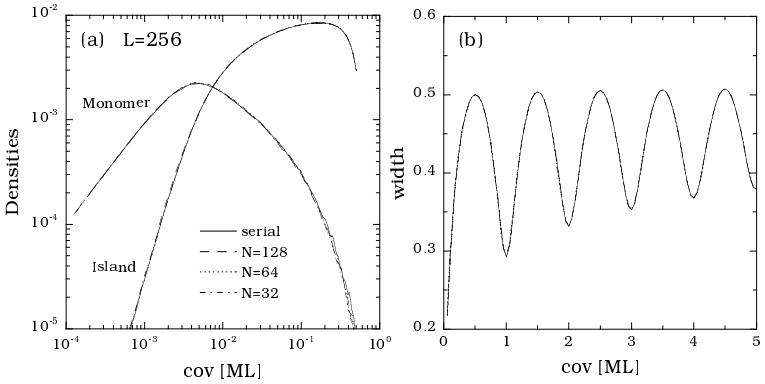

Fig. 2. Comparison between serial and parallel results using synchronous sublattice algorithm with square decomposition ($L = N \times N_p$) for fractal model with $D/F = 10^5$.

Thus, the overall "performance factor" of the parallel simulation (e.g. boost in events/sec over a serial simulation) is given by the parallel efficiency multiplied by the number of processors N_p.

Fig. 3 (a) shows the corresponding results (symbols) for the parallel efficiency as a function of the ratio D/F. Results are shown for parallel KMC simulations with $N_p = 4$ of a square system with system size $L = 1024$. As can be seen, for $D/F \leq 10^6$, the parallel efficiency for the strip geometry is greater than 50% due to the decreased communication overhead. However, with increasing D/F the parallel efficiency decreases significantly since the decrease in the number of events per cycle leads to an increase in the communications overhead as well as in the relative fluctuations. Also shown in Fig. 3 (a) (dashed lines) is the calculated parallel efficiency. As can be seen, there is good agreement between the measured and calculated results (for detail, see Ref.[11]). Fig. 3 (b) shows the update rate per second as a function of N_p. There is a roughly linear speedup of the performance with increasing number of processors N_p. Due to the decreased communication cost, the speed-up using the strip geometry is significantly higher than for the square decomposition.

As we have already shown (see Figs. 2), for sublattice sizes which are not too small, there is perfect agreement between the synchronous sublattice results and the corresponding serial results. However, for very small processor sizes there exists a small "finite-size" effect which leads to results which are slightly different from those obtained using the usual serial KMC algorithm. In particular, as shown in the inset of Fig. 4 (a), there is essentially perfect agreement between the synchronous sublattice results for the fractal model with system size $L = 256$, $D/F = 10^5$, and $N_x = 16 - 256$ and the corresponding serial results. However, for the smallest processor size ($N_x = 8$) there is approximately a 2% difference between the synchronous sublattice

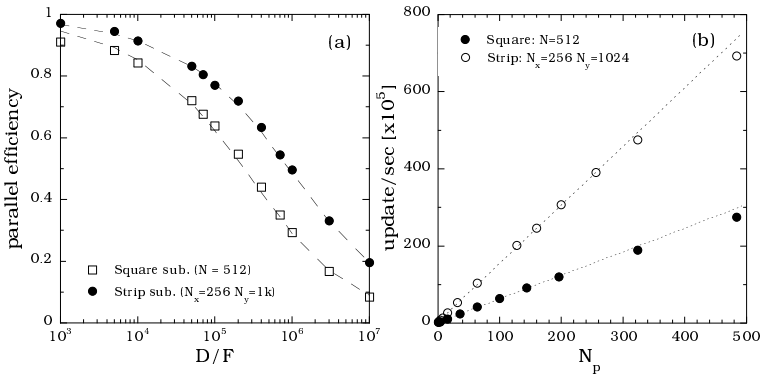

Fig. 3. (a) parallel efficiency for fractal model with $N_p = 4$ ($\theta = 1$ ML) as function of D/F and (b) parallel efficiency (symbols) as function of number of processors N_p for fractal model with $D/F = 10^5$.

results for the peak island density and the corresponding serial results although there are no differences in the monomer density. As shown in Fig. 4 (a), similar results are obtained for the island size distribution N_s.

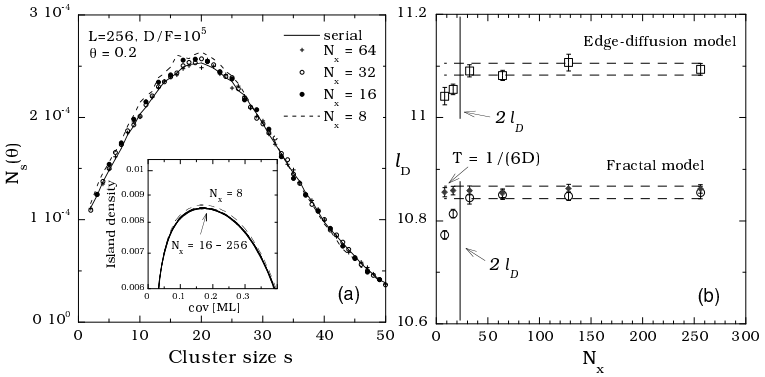

Fig. 4. Finite-size effects in parallel and serial simulations for $L = 256$ with $D/F = 10^5$ and processor sizes $N_x = 8, 16, 32, 64$, and 128 and $N_y = 256$. (a) Island size distribution as a function of cluster size s and the inset shows island density as a function of coverage and (b) diffusion length as a function of N_x for fractal and EC models with $r_e = 0.1$ and $r_c = 0$.

There is one *dynamical* length scale corresponding to the "diffusion length" l_D which plays a particularly important role. The diffusion length may be written in terms of the peak submonolayer island density, i.e. $l_D \sim N_{pk}^{-1/2}$. We find that for a variety of models we studied, there are no finite-size effects for $N_x > 2l_D$ which can be considered as critical processor size. By

measuring the peak island density for $D/F = 10^5$ shown in the inset of Fig. 4, we obtain $l_D \simeq 11$ which implies a critical processor size N_x given by $N_x \simeq 2\, l_D \simeq 22$. This result is in good agreement with the observation of the onset of significant finite-size effects for $N_x < 16$. We also find similar results for the edge-diffusion model with $D/F = 10^5$ and $r_e = 0.1$ and $r_c = 0$. For a smaller cycle-length we expect that the critical processor size N_x corresponding to finite-size effects will be significantly reduced. As shown in Fig. 4 (filled symbols) for the fractal model with cycle length $T = 1/(6D)$, the critical processor size N_x is significantly smaller than the diffusion length l_D. However, for such a reduced cycle length, the parallel efficiency is also significantly reduced.

4 Conclusion

In our algorithm, the maximum cycle length T is given by the inverse of the fastest diffusion rate. For sublattice sizes which are smaller than the diffusion length l_D, weak finite-size effects are observed which lead to deviations from the results obtained using a serial algorithm. However, for sublattice sizes larger than the diffusion length l_D, the results obtained are identical to those obtained in serial simulations. Since in many systems of interest the diffusion length is typically relatively small while significantly larger system sizes are needed to avoid finite system-size effects, the sublattice algorithm should provide a useful, efficient, and accurate method to carry out parallel KMC simulations of these systems. Since only local communication is required, the algorithm scales, i.e. for a large number of processors the parallel efficiency is independent of the number of processors.

References

1. A.B. Bortz, M.H. Kalos, and J.L. Lebowitz, J. Comp. Phys. **17**, 10 (1975).
2. G.H. Gilmer, J. Crystal Growth **35**, 15 (1976).
3. A.F. Voter, Phys. Rev. B **34**, 6819 (1986).
4. J.L. Blue, I. Beichl, and F. Sullivan, Phys. Rev. E **51**, R867 (1995).
5. N.C. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller, J. Chem. Phys. **21**, 6 (1953).
6. K.M. Chandy and J. Misra, IEEE Trans. Software Eng. **5**, 440 (1979); J. Misra, ACM Comput. Surv. **18**, 39 (1986).
7. B.D. Lubachevsky, Complex Systems **1**, 1099 (1987); J. Comput. Phys. **75**, 103 (1988).
8. G. Korniss, Z. Toroczkai, M.A. Novotny, and P.A. Rikvold, Phys. Rev. Lett. **84**, 1351 (2000).
9. G. Korniss, M.A. Novotny, H. Guclu, Z. Toroczkai, and P.A. Rikvold, Science **299**, 677 (2003).
10. J.G. Amar, F. Family, and P.-M. Lam, Phys. Rev. B **50**, 8781 (1994).
11. Y. Shim and J. G. Amar, Phys, Rev. B (accepted) (http://lanl.arXiv.org/abs/cond-mat/0406379).