Simulating extended time and length scales using parallel kinetic Monte Carlo and accelerated dynamics

Jacques G. Amar, University of Toledo

- Kinetic Monte Carlo (KMC) is an extremely efficient method to carry out dynamical simulations when relevant thermally-activated atomic-scale processes are known.

Used to model a variety of dynamical processes from catalysis to thin-film growth

- Temperature-accelerated dynamics (TAD - Sorensen & Voter, 2000) may be used to carry out realistic simulations even when relevant atomic-scale processes are extremely complicated and are not known.

GOAL: to extend both of these techniques in order to carry out realistic simulations over larger system-sizes, longer time scales

**In collaboration with Yunsic Shim*

Supported by NSF DMR-0219328

Parallel Kinetic Monte Carlo

- While standard KMC is extremely efficient it is *inherently* a serial algorithm! **No matter how large the system, at every step only one event can be simulated!**

- In contrast, Nature is inherently parallel!

- We would like to use KMC to carry out simulations of thin-film growth over longer time and length scales

How to "parallelize" the KMC algorithm in order to simulate larger system-sizes, longer time scales?

Temperature Accelerated Dynamics (TAD)

- KMC simulations are limited by requirement that complete catalog of all relevant processes and their rate constants must be specified. However, often all relevant transition mechanisms are not known.

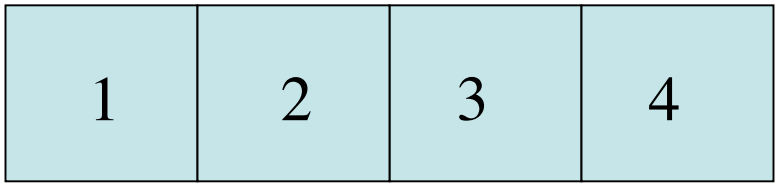
- TAD allows realistic simulations of low temperature processes over timescales of seconds and even hours

- Computational work for TAD scales as N^3 where N = # of atoms, so can only be applied to extremely small systems (a few hundred atoms)

How to "parallelize" the TAD algorithm in order to simulate larger system-sizes?

Parallel KMC - Domain Decomposition

- Domain decomposition is a natural approach since intuitively one expects that widely separated regions may evolve independently “in parallel”

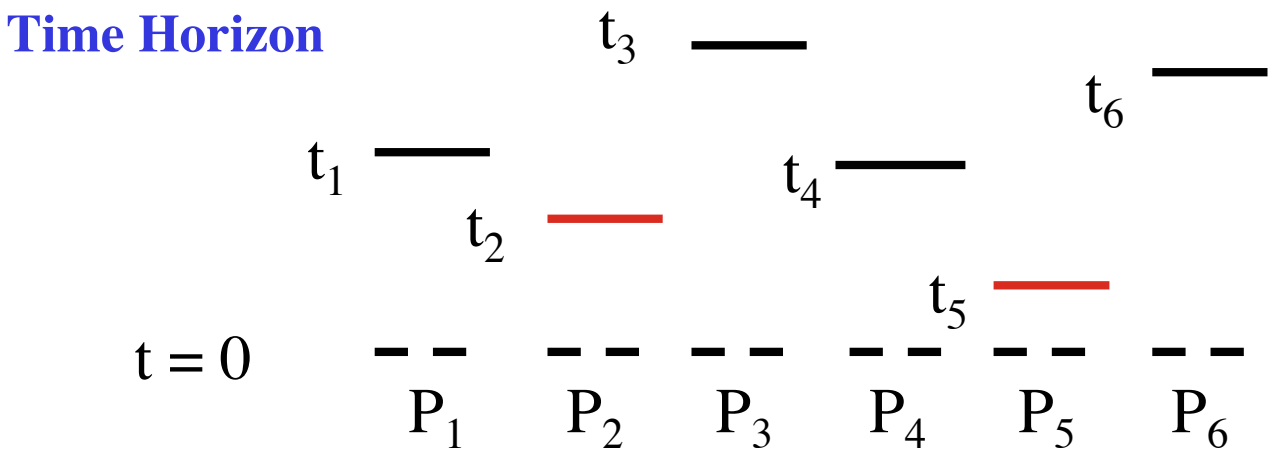


Problems

- In parallel KMC, time evolves at different rates in different regions!
- How to deal with time synchronization?
- How to deal with conflicts between neighboring processors?

Parallel Discrete Event Simulation (PDES)

Conservative Algorithm



Only update processors whose next event times correspond to **local minima** in time horizon (Chang, 1979; Lubachevsky, 1985)

Advantages: works for *Metropolis Monte Carlo* since acceptance probability depends on local configuration but event-times do not.

Disadvantages: does **not** work for *kinetic Monte Carlo* since event-times depend on local configuration. Fast events can "propagate" from processor to processor and lead to **rollbacks**.

Three approaches to parallel KMC

Rigorous Algorithms

- Conservative asynchronous algorithm

  Lubachevsky (1988), Korniss et al (1999), Shim & Amar (2004)

- Synchronous relaxation algorithm

  Lubachevsky & Weiss (2001), Shim & Amar (2004)

Semi-rigorous Algorithm

- Synchronous sublattice algorithm

  Shim & Amar (2004)

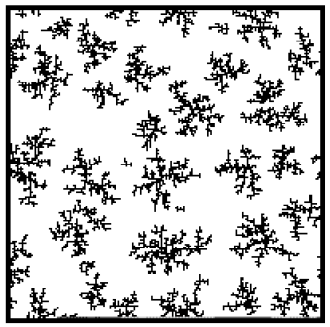
Thin-film growth models studied

"Fractal model"

Deposition rate F per site per unit time

Monomer hopping rate D

Irreversible sticking/attachment (i =1)

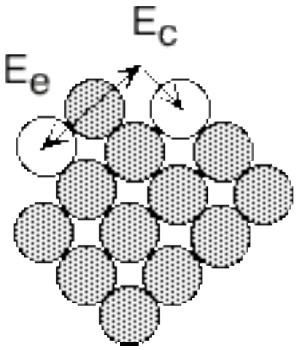


D/F = 10^7

"Edge-diffusion model"

Same as above with edge-diffusion

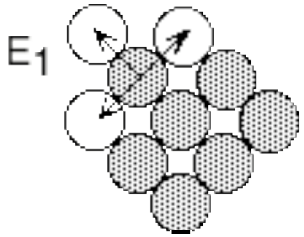
(relaxation) of singly-bonded cluster atoms



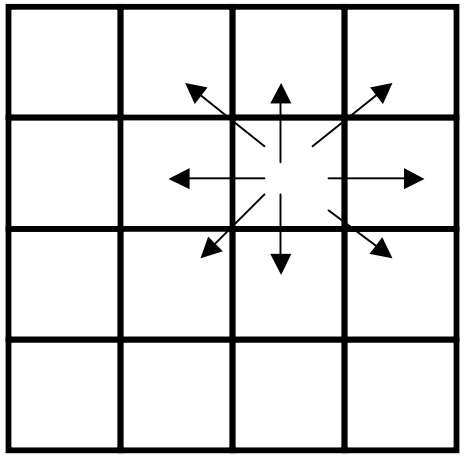
"Reversible attachment model"

Detachment of singly and multiply bonded atoms

(bond-counting model)

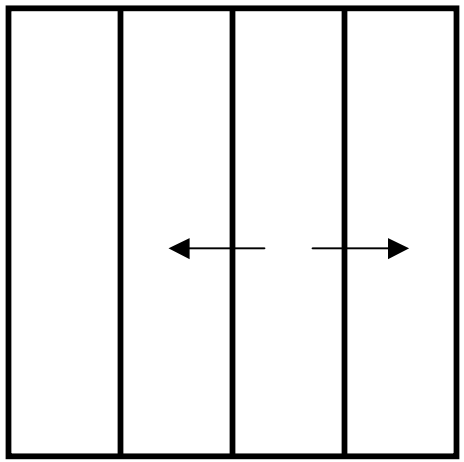


Methods of domain decomposition (2D)



Square decomposition

(8 nbors)



Strip decomposition

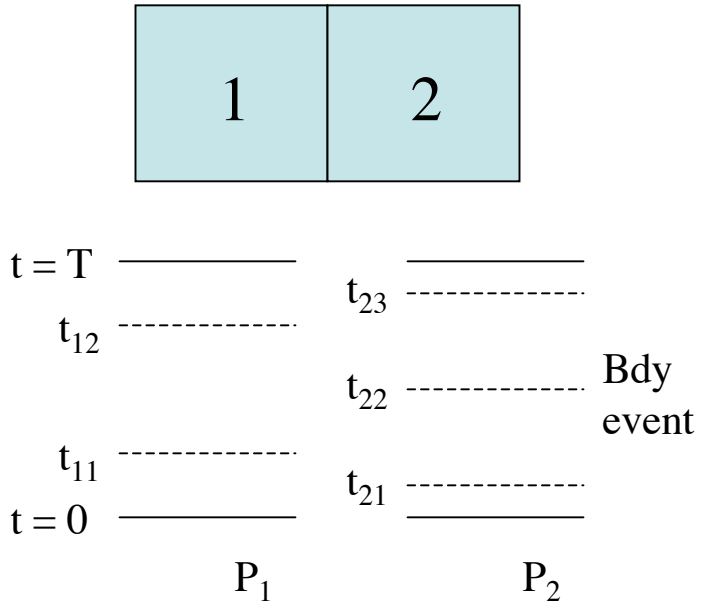
(2 nbors)

Synchronous relaxation (SR) algorithm

(Lubachevsky & Weiss, 2001)

- All processors 'in-synch' at beginning & end of each cycle
- Iterative relaxation - at each iteration processors use boundary info. from previous iteration
- Relaxation complete when current iteration identical to previous iteration *for all processors*

2 processors



One Cycle

Disadvantages:

- ***Complex:*** requires 'keeping list' of all events, random numbers used in each iteration
- **Algorithm does not scale: faster** than CA algorithm but still **slow** due to *global synchronization* and requirement of *multiple iterations per cycle*

Parallel efficiency (PE) of SR algorithm

Average calc. time per cycle T for parallel simulation may be written:

$$t_{av}(N_p) = N_{iter} < n_{max} > (t_{1p}/n_{av}) + t_{com}$$

where: $< n_{max} >/n_{av} \sim T^{-1/2} \log(N_p)^{2/3}$ and $N_{iter} \sim T \log(N_p)^{\alpha}$

$$t_{com} \sim (a + bT) \log(N_p)$$

$$PE = t_{1p}/t_{av} = \frac{1}{N_{iter}(t_{com}/t_{1p} + < n_{max} > /n_{av})}$$

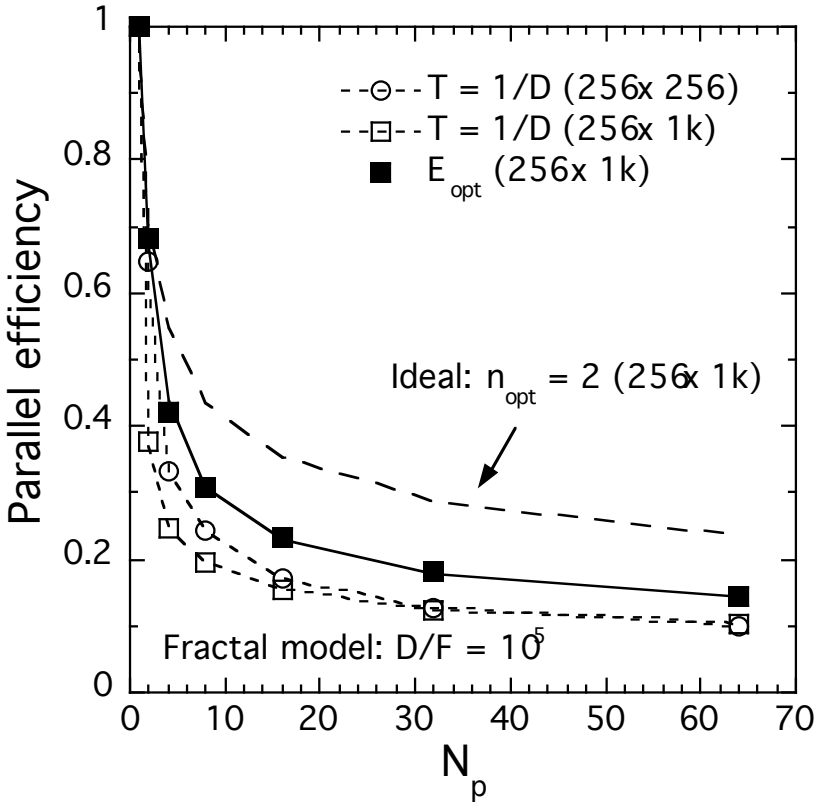
Optimize PE by varying cycle length T (feedback)

In limit of **zero** communication time fluctuations **still** play a role:

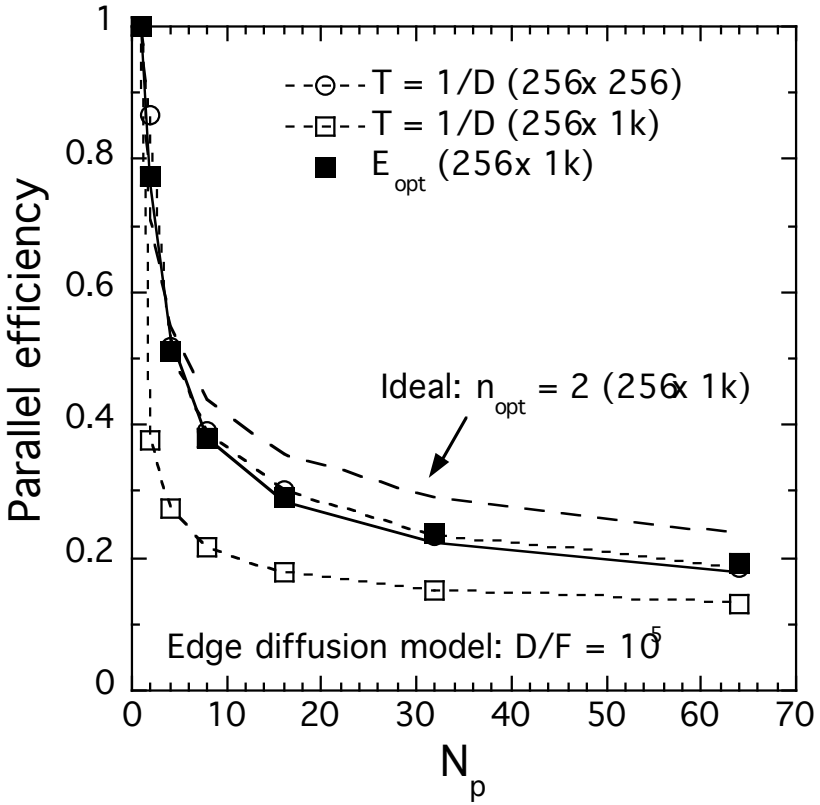
Maximum PE $PE^{max} = (1/N_{iter})(n_{av}/< n_{max} >) \sim 1/\log(N_p)$

Parallel Efficiency of SR algorithm

Fractal model



Edge-diffusion model



---- $PE^{ideal} = 1/[1 + 0.6 \ln(N_p)^{1.1}]$

Synchronous sublattice (SL) algorithm

(Shim & Amar, 2004)

- At beginning of each synchronous cycle one subregion (A,B,C, or D) randomly selected. All processors update sites in selected sublattice only **=> eliminates conflicts between PE's**.

- Sublattice event in each processor selected as in usual KMC. At end of synchronous cycle processors communicate changes to neighboring processors.

Advantages:

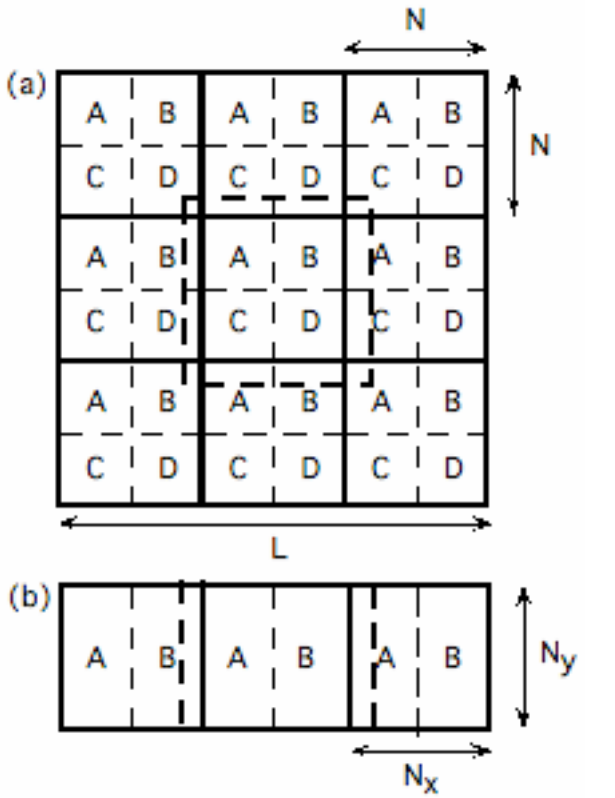
- No global communication required
- Many events per cycle => reduced communication overhead due to latency

Disadvantages:

- *Not rigorous, PE still somewhat reduced due to fluctuations*

2D (square) decomposition

(2 send/receives per cycle)



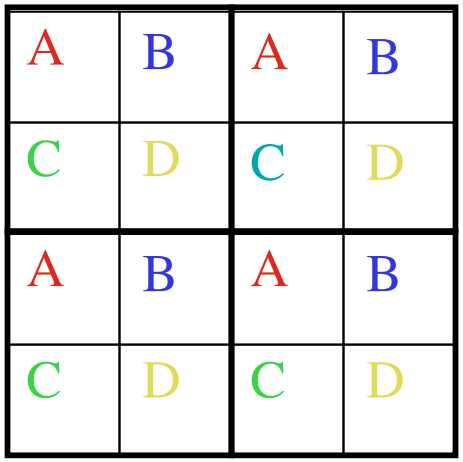
1D (strip) decomposition

(1 send/receive per cycle)

Synchronous sublattice algorithm

(Shim & Amar, 2004)

- Each processor sets its time t = 0 at beginning of cycle, then carries out KMC sublattice events (time increment $\Delta t_i = -\ln(r)/R_i$) until time of next event exceeds time interval T. Processors then communicate changes as necessary to neighboring processors.

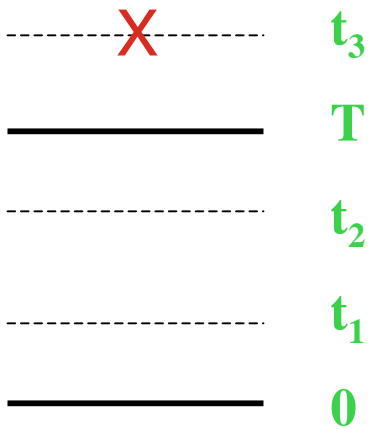


4-processors

- Maximum time interval T determined by maximum ***possible* single**-event rate in KMC simulation.

  For simple model of deposition on a square lattice with deposition rate F per site and monomer hopping rate D, T = 1/D

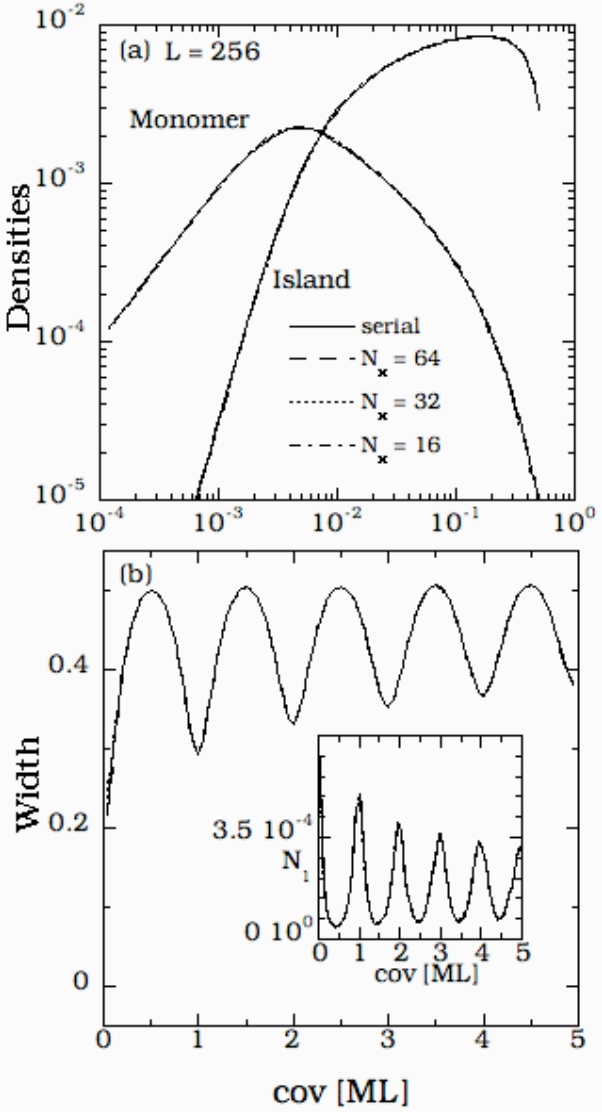
- *Many possible events per cycle!*



2 events

Comparison with serial results

(Fractal model D/F = 10^5, L = 256)



1D strip decomposition

System size 256 x 256

Processor size N_x x 256

$N_p = 4$ $(N_x = 64)$

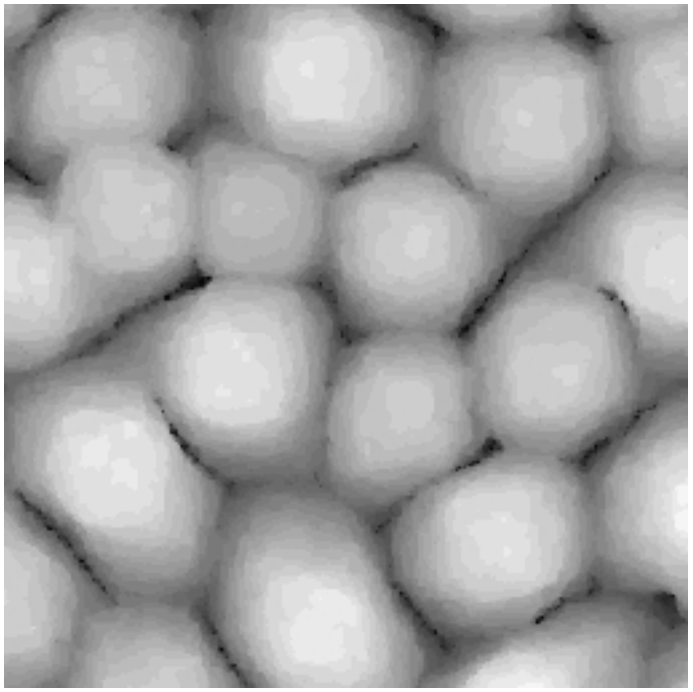
$N_p = 8$ $(N_x = 32)$

$N_p = 16$ $(N_x = 16)$

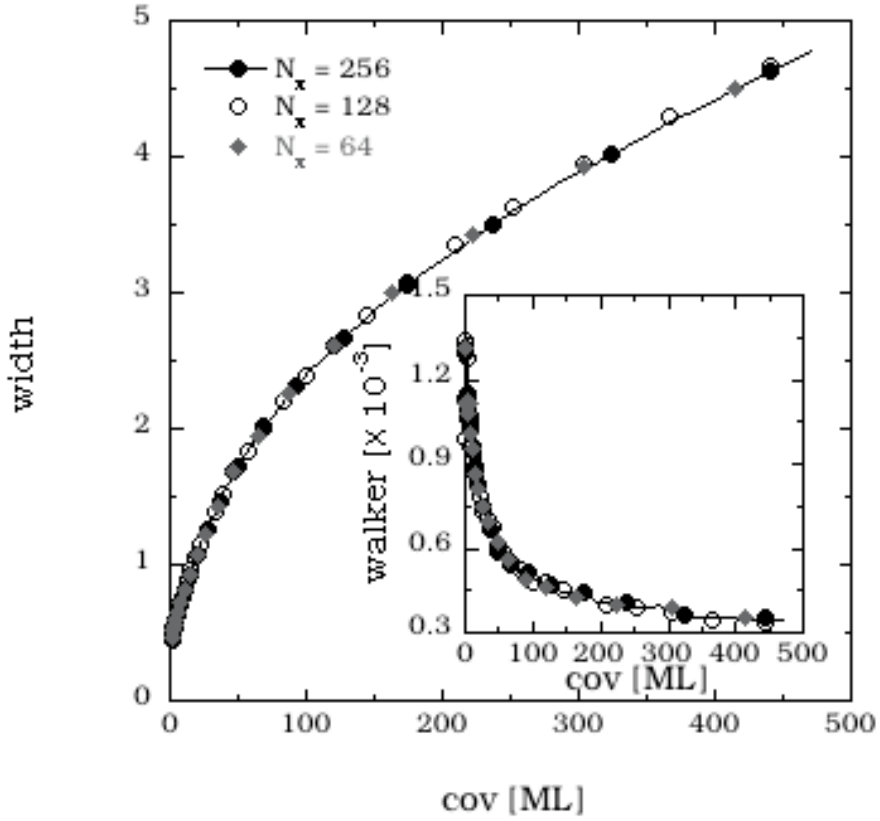
Reversible growth model

T = 300 K, D/F = 10^5, E_1 = 0.1 eV, and E_b = 0.07 eV

N_x = 64 *N_y = 1024 N_p= 16* 128



*512 by 512 portion of **1k by 1k system***

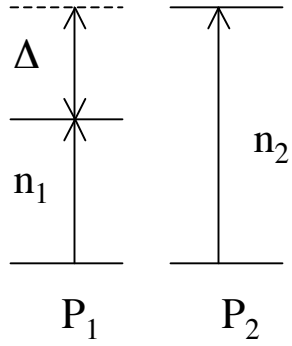


Parallel efficiency (PE) of SL algorithm

Average time per cycle for parallel simulation may be written:

$$t_{av} = t_{1p} + t_{com} + \langle\Delta(\tau)\rangle\,(t_{1p}/n_{av})$$

where $\langle\Delta(\tau)\rangle$ is (average) delay per cycle due to **fluctuations** in number of events in neighboring processors.



Fluctuations

Parallel efficiency ($PE = t_{1p}/t_{av}$) may be written:

$$PE = [\ 1 + (t_{com} / t_{1p}) + \langle\Delta(\tau)\rangle/n_{av}\ ]^{-1}$$

In limit of no communication time fluctuations still play important role:

Ideal PE

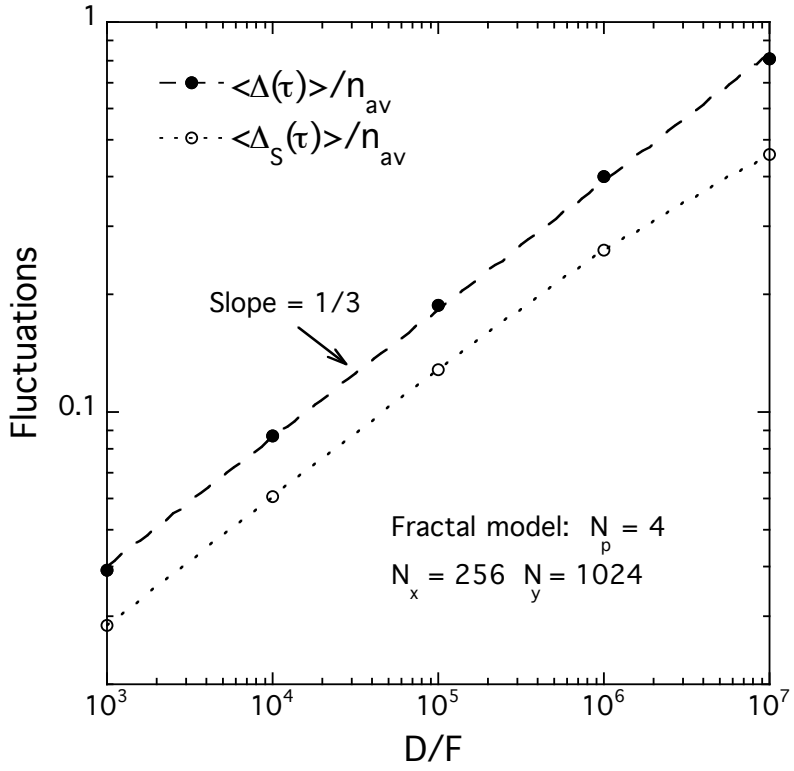
$$PE^{ideal} = [\ 1 + \langle\Delta(\tau)\rangle/n_{av}\ ]^{-1}$$

where $\langle\Delta(\tau)\rangle/n_{av} \sim 1/n_{av}^{1/2}$

Results for $<\Delta(\tau)>/n_{av}$

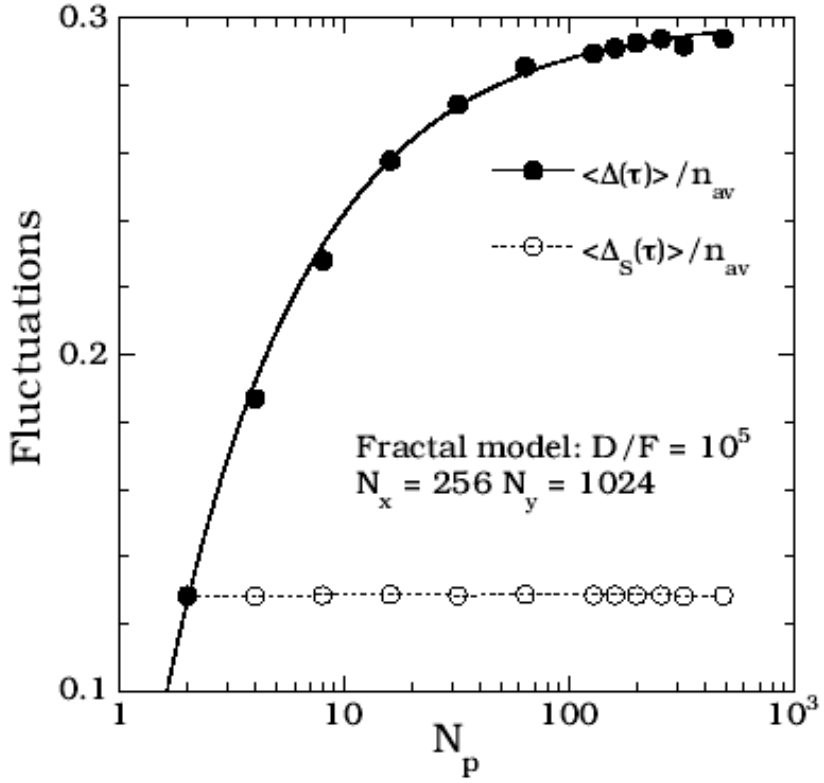
Fractal model

D/F dependence ($N_p = 4$)



$<\Delta(\tau)>/n_{av} \sim (D/F)^{1/3}$

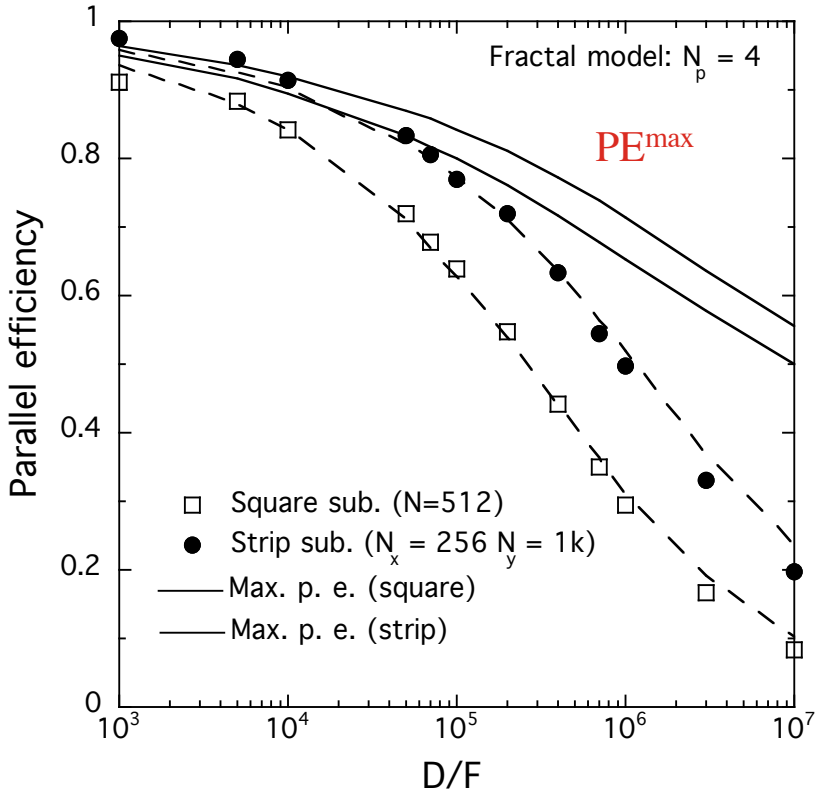
N_p dependence ($D/F = 10^5$)



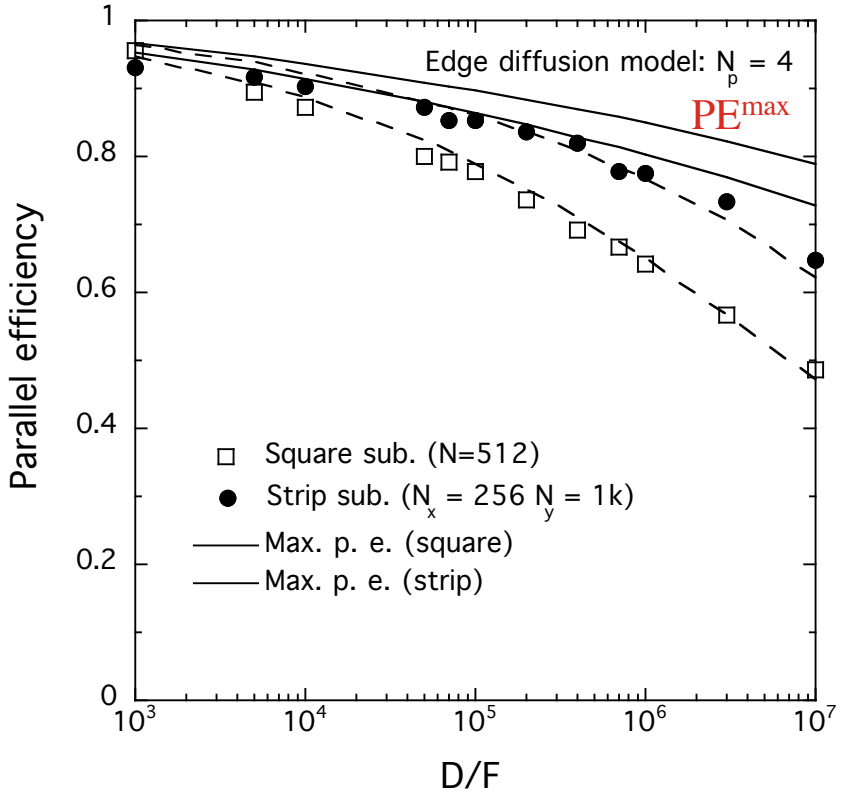
$<\Delta(\tau)>/n_{av}$ saturates for large N_p

Parallel efficiency as function of D/F ($N_p = 4$)

—— $PE^{max} = 1/[1 + 0.2\ (D/F)^{1/3}/(N_x N_y)^{1/2}]$

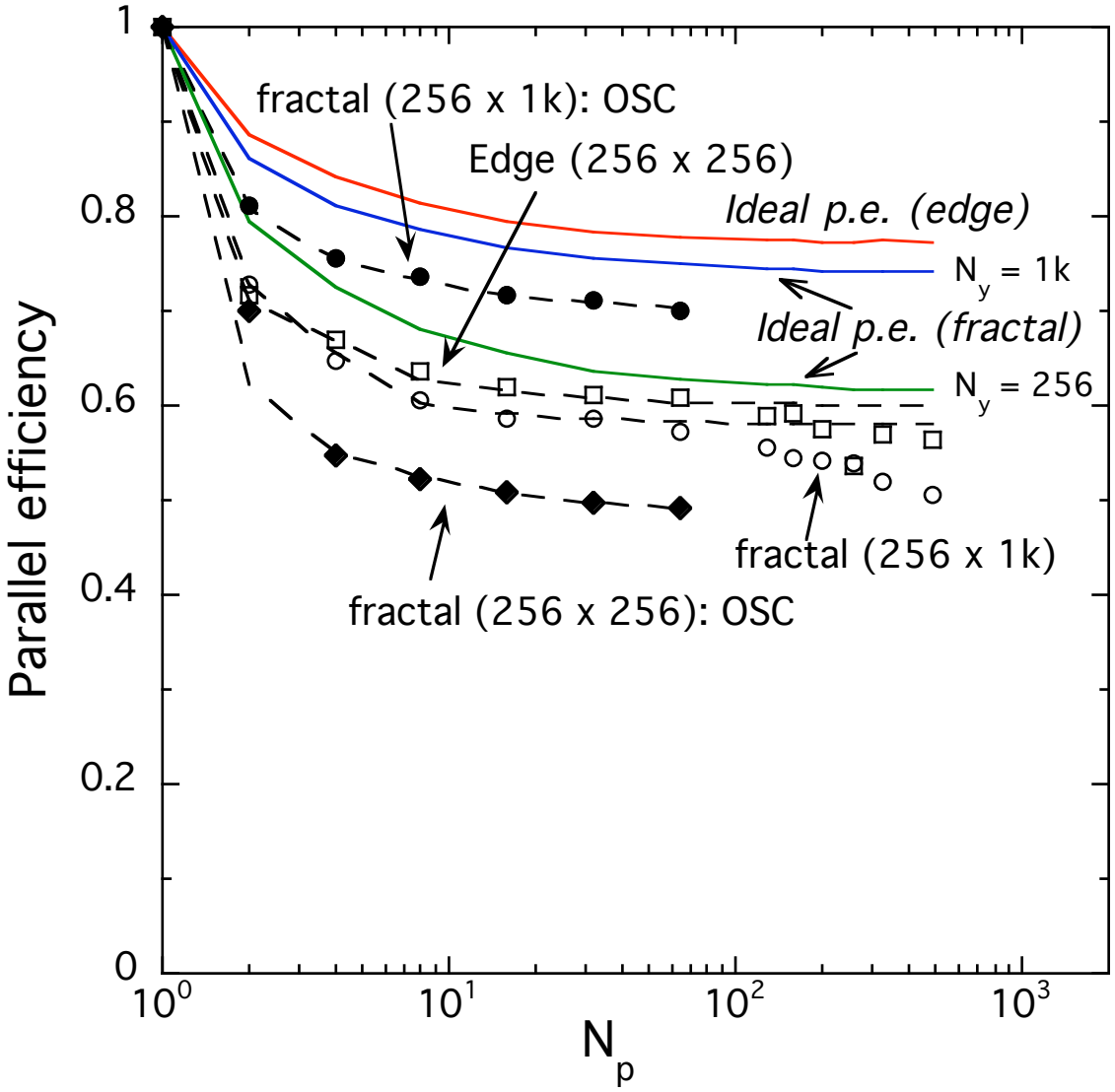


Fractal Model

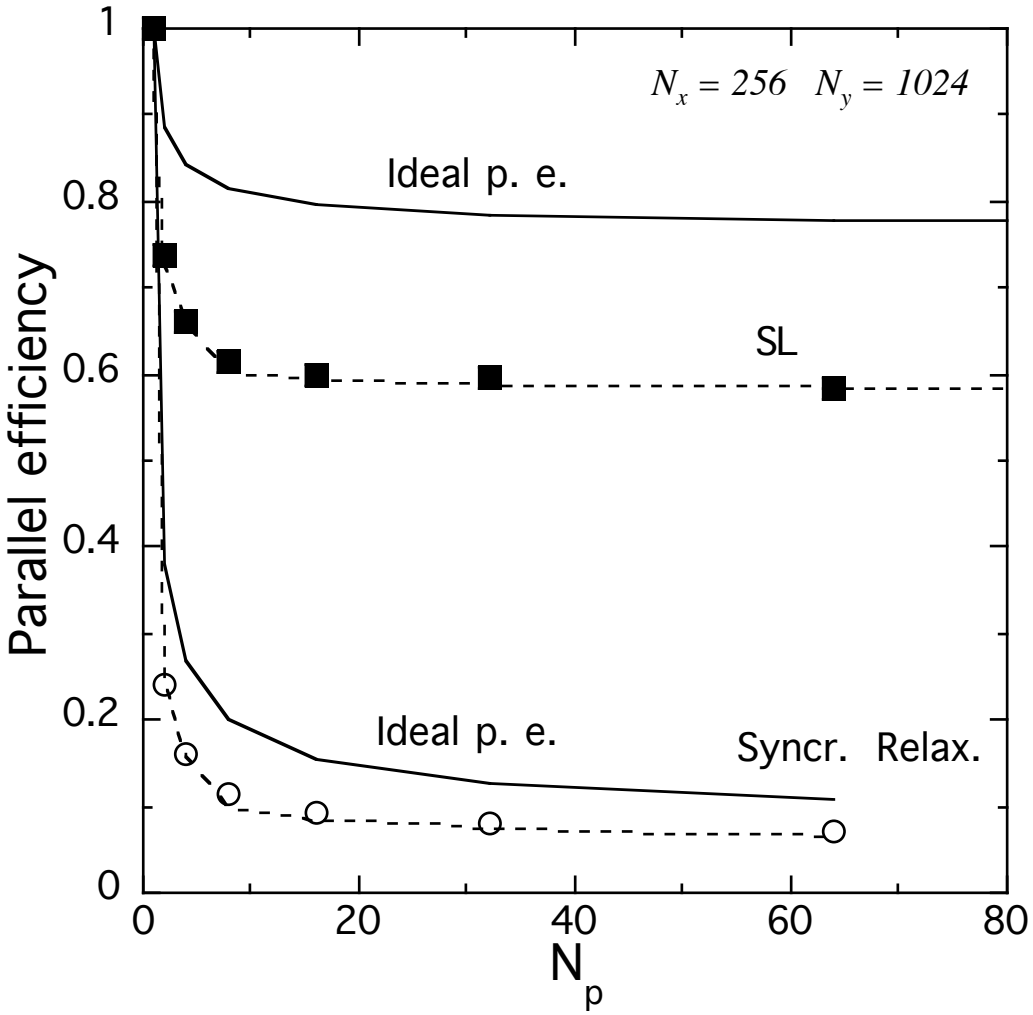


Edge-diffusion Model

Parallel efficiency as function of N_p ($D/F = 10^5$)



Comparison of SR and SL algorithms
Fractal model, D/F =10^5



Summary

- We have studied 3 different algorithms for parallel KMC:

conservative asynchronous (CA), synch. relaxation (SR), synch. sublattice (SL)

- CA algorithm *not* efficient due to rejection of bdy events

- SL algorithm significantly more efficient than SR algorithm

SR algorithm: PE ~ 1/log(N_p)$^\beta$ where $\beta \geq 1$ ⟵ Global synch.

SL algorithm: PE independent of N_p ! ⟵ Local synch.

- For all algorithms, communication time, latency, fluctuations play significant role

- For more complex models, we expect that parallel efficiency of SR and SL algorithms will be significantly increased

Future work

- Extend SL algorithm to simulations with realistic geometry in order to carry out pKMC simulations of Cu epitaxial growth

  => properly include fast processes such as edge-diffusion

- Apply SR and SL algorithms to parallel TAD simulations of Cu/Cu(100) growth at low T (collaboration with Art Voter)

  => Vacancy formation and mound regularization in low temperature metal epitaxial growth

- Develop hybrid algorithm combining SR + SL algorithms

- Develop local SR algorithm

- Implement SL and SR algorithms on shared memory machines