Bull. Nov. Comp. Center, Comp. Science, 27 (2008), 55–62
© 2008 NCC Publisher

Parallel simulation of asynchronous Cellular Automata evolution

K.V. Kalgin

Abstract. For simulating physical and chemical processes on molecular level, asynchronous cellular automata with probabilistic transition rules are widely used being sometimes referred to as Monte-Carlo methods. The simulation requires a huge cellular space and millions of iterative steps for obtaining the CA evolution representing a real scene of the process. This may be attained by allocating the CA evolution program onto a multiprocessor system.

We propose a new parallelization method of asynchronous CA based on its stochastic properties. The efficiency assessment and experimental results are presented.

1. Introduction

Currently there are many CA models of a natural process, which have different state sets, structures of a cellular space, transition functions, and modes of operation.

There are two basic CA modes — synchronous and asynchronous. In some publications, mixed synchronous-asynchronous modes — block-synchronous and ordered asynchronous — are also used [1].

Asynchronous CA (ACA) are used for simulating physical and chemical processes on molecular level, for example, to study oscillatory chemical surface reactions [2], absorbtion, sublimation and diffusion of atoms in epitaxial growth processes[3].

For the ACA simulation of natural processes, much computer power is needed. The basic CA properties — fine-grained parallelism and local interactions — make this model attractive. As a matter of fact, the ACA cannot be so easily parallelized as synchronous CA.

By this moment in the majority of coarse grained parallelization, the ACA are represented as a discrete event model. Within this model there are two methods of parallel simulation [4]: conservative and optimistic. In the general case of a discrete event, the task the conservative simulation method is considered to be inefficient, hence the optimistic simulation is generally used.

For example in [5], the ACA is represented as a discrete event model, and the optimistic method based on Time Warp algorithm is used for the parallel simulation.

This paper is aimed to proposing a coarse grained ACA parallelization method, based on stochastic properties of the ACA, and studying its efficiency. Apart introduction and conclusion, this paper contains four sections: ACA model (Section 2), parallel algorithm (Section 3), efficiency assessment (Section 4), and experiments results of the method testing on Ising CA-model (Section 5).

2. Asynchronous Cellular Automata

Asynchronous Cellular Automata are specified by the following tuple:

$$\mathrm{ACA} = \langle Z^d, A, V, \varphi \rangle.$$

A pair $(x, a) \in Z^d \times A$ is called a *cell*, where $a \in A$ is a *state of the cell* and $x \in Z^d$ is its *coordinates.*

A set of cells $\Omega = \{(x_i, a_i)\} \subset Z^d \times A$ is called a *cellular array* if there does not exist two cells with equal coordinates and $\{x \mid (x, a) \in \Omega\} = Z^d$. Since between the cells in a cellular array and their coordinates there exists a one-to-one correspondence, we will further identify a cell with its coordinates.

The ACA parameters are as follows:

Z^d is a finite subset of discrete d-dimensional space, represented as a set of vectors, which define a *set of cell positions.*

In this paper, we use a 2D rectangular space Z^2:

$$Z^2 = \{(i, j) \mid 1 \le i \le N_x,\ 1 \le j \le N_y\}.$$

A is an *alphabet*, i.e., a set of cell states.

V is a *template*, i.e., a finite set of vectors from Z^d:

$$V = \{v_1, v_2, \ldots, v_{|V|}\}.$$

The template determines the *neighborhood* of a cell x :

$$V(x) = \{x, x + v_1, x + v_2, \ldots, x + v_{|V|}\},$$

the cell x being called a *base cell of the template.*

Further, the von Neumann neighborhood $V = \{(-1, 0), (0, -1), (1, 0), (0, 1)\}$ is used.

φ is a *transition function* with neighboring cell states as arguments and a new state of the cell, as a result: $\varphi : A^{|V|} \to A$. Changing the cell state by a transition function is called *an updating of a cell.*

The ACA simulation is split into *iterations.* An iteration comprises $|Z^2| = N_x N_y$ updating of cells in a random order.

3. Parallel algorithm

For parallelization, a cellular array is partitioned into the disjoint domains D_1, D_2, …, D_p, which are mapped onto parallel processors. A set of *boundary cells* of a domain D_k is denoted by $B_k = \{c \in D_k \mid \exists D_j : j \neq k \;\&\; V(c) \cap D_j \neq \emptyset\}$.

3.1. Stochastic properties of the ACA. Let us consider thebehavior of an ACA implemented on one processor, the cellular array being logically partitioned into domains.

Let $C = \{c_1, \dots, c_{|Z^2|} \mid c_i \in Z^2\}$ be a list of cells, which determines the order of cells updatings in one iteration. Let us denote its different disjoint subsets as follows:

$$\begin{aligned} \mathrm{in}_k &= \{c \in C \mid c \in D_k \setminus B_k\}, \\ \mathrm{bound}_k &= \{c \in C \mid c \in B_k\}, \\ \mathrm{out}_k &= \{c \in C \mid c \in Z^2 \setminus D_k\}. \end{aligned}$$

Let us consider $L_k = \{l_0^k, l_1^k, \dots, l_{|\mathrm{bound}_k|}^k\}$, where $l_0^k = 0$ and l_i^k are numbers of updating steps of cells $c \in B_k$, such that $\{c_{l_i^k} \mid 1 \leq i \leq |\mathrm{bound}_k|\} = \mathrm{bound}_k$.

Let $I_r^k = \{c_i \in C \mid c_i \in \mathrm{in}_k \;\&\; l_{r-1}^k < i < l_r^k\}$ be a list of cells, which are updated inside the domain D_k between two updatings on its boundary.

Then, with C cellular array partitioning, we obtain a unique set of triplets $\Upsilon(C)$:

$$\Upsilon(C) = \{\langle L_k, \mathrm{bound}_k, I_r^k \rangle \mid 1 \leq k \leq p,\ 1 \leq r \leq |\mathrm{bound}_k|\},$$

and the contrary statement is wrong.

Let us consider another updating order C' with the same $\Upsilon(C') = \Upsilon(C)$. Then C and C' may differ only in ordering of cells belonging to different domains' I_r^k. Although, the ordering between cells from each domain and its boundary, and also between the domains boundaries, remain the same. Hence, execution of C and C' with the same initial cellular arrays finalize with the same finale cellular arrays. So, they are called *equivalent*.

So, for execution of the next iteration it is sufficient to form only $\Upsilon(C)$, but not the full order. Since for each iteration the order is generated by a uniform random generator, then $\Upsilon(C)$ is formed according to the following statements:

$$P(l_r^k - l_{r-1}^k = t \mid t > 0) = (1 - \beta_k)^{t-1}\beta_k, \tag{1}$$

where $\beta_k = |B_k|/|Z^2|$, and

$$P(|I_r^k| = t) = C_\Delta^t \alpha_k^t (1 - \alpha_k)^{\Delta - t} \tag{2}$$

is a binomial distribution $\mathrm{Bin}(\Delta, \alpha_k)$, where $\Delta = l_r^k - l_{r-1}^k$, $\alpha_k = |D_k \setminus B_k| / |Z^2 \setminus B_k|$ and C_Δ^t denotes number of t-element subsets of an Δ-element set.

So, the algorithm of $\Upsilon(C)$ is formed as follows:

1. **Generation of $\boldsymbol{l_r^k}$** is performed according to (1), with determination $|\mathrm{bound}_k|$ and $|L_k|$: $|\mathrm{bound}_k| = |L_k| - 1$, $l_{|\mathrm{bound}_k|}^k \leq |Z^2|$, where $l_i^k < l_{|\mathrm{bound}_k|}^k$ holds for each $i < |\mathrm{bound}_k|$.

   Let x be an exponential-distributed random variable with parameter $\lambda = -\log(1-\beta_k)$, then $P(\lfloor x \rfloor = t) = e^{-\lambda(t-1)}(1-e^{-\lambda}) = (1-\beta_k)^{t-1}\beta_k$. Hence,
   $$l_p^k = l_{p-1}^k + \lfloor \mathrm{randE}(-\log(1 - \beta_k)) \rfloor, \tag{3}$$
   where $\mathrm{randE}(\lambda)$ is the number of exponential-distributed pseudo-random numbers obtained by the generator.

2. **Generation of $\boldsymbol{|I_r^k|}$.** Since the considered binomial distribution with a large Δ has a good approximation by the normal distribution $N(\alpha_k\Delta, \alpha_k(1 - \alpha_k)\Delta)$, then:
   $$|I_p^k| = \mathrm{randN}(\alpha_k\Delta, \alpha_k(1 - \alpha_k)\Delta), \tag{4}$$
   where $\mathrm{randN}(\eta, \sigma^2)$ is the number of normal-distributed pseudo-random numbers obtained by the generator.

   For small Δ, a certain number of trials are performed (generation of uniformly-distributed pseudo-random numbers from $[0, 1]$) and calculation of "passed" trials (the number of generated numbers from $[0, \alpha_k]$).

3. **Generation of $\mathbf{bound}_{\boldsymbol{k}}$ and $\boldsymbol{I_p^k}$ elements.** The sizes of the bound_k and I_p^k are already known, but the cells coordinates are not. Such coordinates are generated by a uniformly-distributed pseudo-random numbers generator.

Let us note, that $\Upsilon(C)$ obtained by this algorithm determines a set of equivalent and equal-size lists, whose size is a random variable with an expected value equal to $|Z^2|$. Nonetheless, further the number of updates in a cellular array is assumed to be equal to $|Z^2|$.

3.2. Parallel algorithm:

1. **Generation of $\boldsymbol{\Upsilon(C)}$.** Create $\langle L_k, \mathrm{bound}_k, I_r^k \rangle$ for each domain D_k. After that, processors with neighboring domains D_k and $D_{k'}$ exchange the lists L_k, bound_k and $L_{k'}$, $\mathrm{bound}_{k'}$.

2. **Execution of an iteration.** Cells of each domain D_k are updated in consecutive order according to the generated bound_k and I_p^k.

   If the next cell $c \in \text{bound}_k$ has number l_j^k from L_k, then according to the received $L_{k'}$ and $\text{bound}_{k'}$ the required new states of the neighboring cells $c' \in V(c)$ are received and a new state of the base cell is sent to an appropriate processor:

   (a) for each neighboring domain $D_{k'}$ and for each updating of the cell $c' \in \text{bound}_{k'}$ with number $l_i^{k'} \in L_{k'}$ such that $l_i^{k'} < l_j^k$, $c' \in V(c)$ and the state of the cell c' has not been received yet after the last updating, a new state of the boundary cell c' is received from an appropriate processor.

   (b) the transition function φ is applied to the cell c;

   (c) for each neighboring domain $D_{k'}$, if there exists an updating of the cell $c' \in \text{bound}_{k'}$ with number $l_i^{k'} \in L_{k'}$ such that $l_j^k < l_i^{k'}$ and $c \in V(c')$, then new state of the boundary cell is sent to an appropriate processor.

4. Efficiency assessment

Let us consider the case, when a cellular array is decomposed in the domains of equal size and form. The work time T_p of one iteration on p processors can be assessed in the following way:

$$T_p = |B_k|\,(T_{\text{send}} + T_\gamma) + |D_k|\,(T_\varphi + T_{\text{rand}}),$$

where

T_γ	is the time for generation of two numbers l_p^k and $\lvert I_p^k\rvert$.
T_{rand}	is the time for generation of two uniformly-distributed pseudo-random numbers (cell coordinates).
T_φ	is the time for computing the transition function φ.
T_{send}	is the time for transferring a boundary cell state.

The variables T_γ and T_{rand} remain unchanged with an increase of the processors number, T_{send} changes insignificantly and T_φ changes depending on the size of the domain $|D_k|$ (exactly, it depends on both the domain and the cache size).

The size of the domains is inversely proportional to the processors number $|D_k| = |Z^2|/p$.

The boundary size $|B_k|$ depends on the processors number and the manner of cellular space partitioning. So, with the cellular space partitioning on stripes, the size of the boundary does not change with an increase of the processors number. However, with the cellular space partitioning on rectangles we can obtain the inverse proportion with respect to the square root of the processors number.

If we denote $\tau = T_\varphi + T_{rand}$ and neglect T_γ, then the parallelization efficiency can be assessed as follows:

$$A = \frac{T_1}{p\,T_p} = \frac{\tau|D_k|}{\tau|D_k| + T_{\mathrm{send}}|B_k|}.$$

Therefore for obtaining the efficiency $A \geq \alpha$, we need:

$$\frac{|D_k|}{|B_k|} \geq \frac{\alpha}{1-\alpha}\,\frac{T_{\mathrm{send}}}{\tau}.$$

Assume $T_{\mathrm{send}} \sim 1000\tau$, $|D_k| \sim |B_k|^2$ and $\alpha = 0.75$, then we obtain the following constraint on the domain size: $|D_k| \geq 3000^2$. But in the above assessment the following factors were neglected:

1. Messages can be sent simultaneously with computations to reduce the work time.
2. For the same cellular array size, the time T_φ is different for a different processors number. It also significantly depends on the architecture of a supercomputer node.
3. For computing a transition function at a boundary cell c, we need to wait only for those new states of the cells c', which are needed for computation: $c' \in V(c)$.

In the section below, it is shown that the efficiency about 75 % may be attained on a cluster even when the domain size is less than 3000×3000.

5. Results

We use the Ising model to perform experiments. The Ising model is a popular model of a system of interacting variables in statistical physics. In terms of the ACA, it can be described as follows.

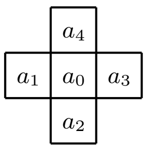

Figure 1

Take sets Z^2 and the Neumann neighborhood V (Figure 1), the alphabet $A = \{-1, +1\}$, where $+1$ for an "up" spin and -1 for a "down" spin.

The total energy of the Ising model is

$$E = \frac{1}{2}\sum_{(x,a)\in\Omega}\ \sum_{(y,b)\in V(x)} ab.$$

The transition function changes the state of the base cell with probability P, which depends on the energy difference ΔE between a new configuration and the previous one ($\Delta E = -2a_0(a_1 + a_2 + a_3 + a_4)$):

$$\varphi(a_0, a_1, a_2, a_3, a_4) = \begin{cases} -a_0, & \text{with probability } P; \\ a_0, & \text{otherwise;} \end{cases}$$

$$P = \begin{cases} 1, & \Delta E < 0; \\ \exp(-\Delta E/kT), & \text{otherwise.} \end{cases}$$

where k is Boltzmann's constant, T is temperature.

The experiments were performed with a cellular array size equal to $|Z^2| = N_x{\cdot}N_y$, where $N_x = 32768$ and $N_y = 2048$, and with partitioning of Z^2 on stripes along Oy ($|B_k| = N_y$).

In Figure 2, the efficiency of the proposed parallel algorithm implementation on supercomputers MVS-15000 (two Power processors on a node) and MVS-100k (two Xeon processors with four cores, each being on node) is shown.

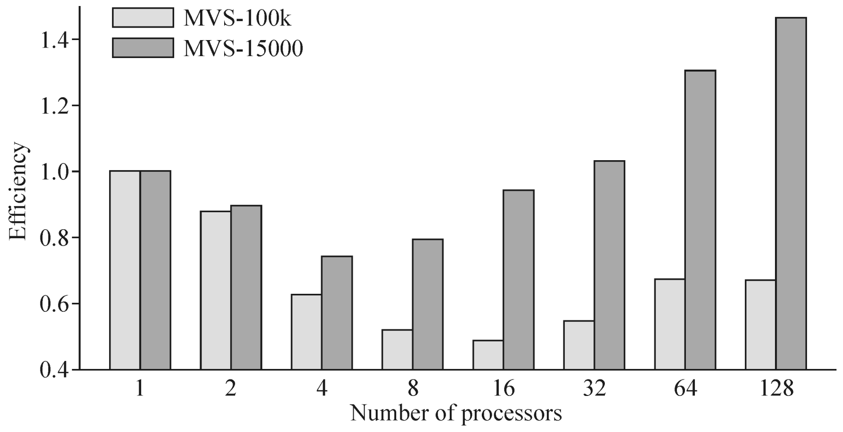

Figure 2

As long as APSIS overheads are too large, comparison of the efficiency was not performed in this paper. The experiments results of parallel implementation of the Ising model with APSIS are presented in [5].

A significant difference in the behavior between the efficiencies obtained on supercomputers is explained by their different architectures:

- on the supercomputer MVS-15000, with an increase of the processors the number portion of cache-misses decreases, so the average time of the cell state reading decreases, therefore T_φ decreases (and reaches the superlinear acceleration);
- because each node of the supercomputer MVS-100k contains eight cores, the average time of the cell state reading is greater with eight working processor, than with one working processor.

6. Conclusion

In this paper, we have presented a new coarse grained algorithm of the ACA parallelization, based on stochastic properties of the ACA, with a good efficiency. The efficiency assessment was made and experimentally tested.

In the future, we are planning to modify the algorithm and to study its efficiency on the ACA with different templates, transition functions and manners of cellular array partitioning.

We are also planning to modify the algorithm for the efficient simulation on the GPU (Graphics Processing Unit) and multi-core systems.

References

[1] Bandman O.L. Coarse-grained parallelization of cellular-automata simulation algorithms // Parallel Computing Technologies.— Heidelberg: Springer, 2007.— P. 370–384.— (LNCS; 4671).

[2] Elokhin V.I., Latkin E.I., Matveev A.V., Gorodetskii V.V. Application of statistical lattice models to the analysis of oscillatory and autowave processes on the reaction of carbon monoxide oxidation over platinum and palladium surfaces // Kinetics and Catalysis.— 2003.— Vol. 44, No. 5.— P. 672–700.

[3] Neizvestny I.G., Shwartz N.L., Yanovitskaya Z.Sh., Zverev A.V. 3D-model of epitaxial growth on porous {111} and {100} Si surfaces // Computer Physics Communications.— 2002.— Vol. 147.— P. 272–275.

[4] Overeinder B.J., Hertzberger L.O., Sloot P.M.A. Parallel discrete event simulation // The Third Workshop Computersystems, Faculty of Electrical Engineering, Eindhoven University, Eindhoven, the Netherlands.— 1991.— P. 19–30.

[5] Overeinder B.J., Sloot P.M.A. Extensions to time warp parallel simulation for spatial decomposed applications / D. Al.-Dabbas, R. Cheng, eds. // Proc. the Fourth United Kingdom Simulation Society Conference (UKSim99).— Cambridge (UK), 1999.— P. 67–73.