Implementation of Large-Scale Cellular Automata Models on Multi-Core Computers and Clusters

Olga Bandman
Institute of Computational Mathematics and Mathematical Geophysics SBRAS
pr. Lavrentieva, 6, Novosibirsk, Russia, 630090
Email: bandman@ssd.sscc.ru

***Abstract*—The purpose of the paper is to develop an approach to allocation a large-scale CA-model onto processing units of a multi-processor system. Two methods are proposed that are based on the principle of maximum similarity between the parallel CA algorithm structure and computer architecture: (1) domain decomposition with implementation on a cluster, if CA model is a sequential CA composition, and (2) allocation of each CA on a processor of a multicore computer with shared memory, if the CA model is a parallel CA composition. Each proposed method is illustrated by an example of parallel implementation of a complex CA-model on multi-core computer or cluster.** [1]

I. Introduction

Cellular Automata (CA) being regarded as a model of spatial dynamics, gradually transit from being an object of study to the stage of being the model for studying natural processes. CA properties, such as nonlinearity of transition functions and irreversibility of evolution, make them particularly useful for investigating the behavior of complex systems, which exhibit self organization and emergency. The number of such investigations increases rapidly, comprising the study of new more complicated phenomena in physics, biology, and chemistry [1]. Nowadays there exists a relatively large bank of well studied CA-models, as well as a developed theory and techniques [2] to construct CA compositions for simulating large scale complex processes. Obviously, the implementation of such a complicated models require high performance computer systems. Wherefrom a new problem arises concerning parallel implementation efficiency. Its solution cannot anymore rely on the inherent fine-grained parallelism of CA, since the allocation of the interacting CAs onto a parallel processor system has to use the coarse grained parallelization methods. So, both levels of parallelism are to be exploited in developing the algorithm for parallel implementation of a CA composition, which is the specific distinction of the problem, increasing the importance of the principle of similarity between the parallel algorithm structure and the computer system architecture. Nowadays, there are two generally used parallel architectures: multi-core computers with shared memory, and clusters of processing nodes with distributed memory. They may be put into correspondence with two forms of parallelism in large scale CA models. Architecture of a multi-core computer with shared memory is similar to that of parallel composition of CAs [2], which is referred to in [3] as a CA-system. Parallel composition implies that each CA processes its own cellular array, using in its transition functions a common pool of variables, which includes cell states of all cellular arrays in the system. The similarity of computer architectures is easily seen: each core computes global transition functions of the corresponding CA, taking variables from shared memory and writing back the resulting next states. A cluster is a set of processing nodes, each having its own memory; frequently, a node is itself a multi-core computer. Cluster is provided of a means for organization any wanted connection structure. Particularly, its structure may match a domain decomposition configuration of a CA cellular array. There are some investigations of large scale CA-model implemented on parallel computer systems. So, in [3] and in [4] the efficiency of the implementation of CA-compositions on multi-core computers with shared memory is explored. In [5] and [6] the composed CA implementation on clusters are used for simulating large-scale tasks.

In this work a general approach to develop parallel algorithms of large scale CA-models implementation on multi-core computers and clusters is proposed. The specialized parallel devices such as Graphical Processing Units and FGPA are not considered here, being regarded as a group of fine-grained architectures, that is beyond the problem.

The paper contains five sections. In Section 2 formal definitions are given. Section 3 is devoted to the peculiarities of constructing a parallel algorithm for implementing a complicated CA superposition on a cluster. In Section 4 it is shown how a CA system may be allocated on a multi-core computer with shared memory. The results are summarized in Conclusion.

II. Formal representation of a CA model

CA is a set of identical computing units, denoted by pairs (u, x), called *cells*, where $u \in A$ is a *cell state* from the alphabet A, $x \in X$ is a name, often being given as a vector $\mathbf{x} = (i)$, or $\mathbf{x} = (i, j)$, or $\mathbf{x} = (i, j, k)$ from a set of coordinates of a finite b-dimensional $(b = 1, 2, 3)$ discrete space with $i = 0, \ldots, I; j = 0 \ldots, J; k = 0 \ldots, K$. The set of cells $\Omega = \{(u_i, x_i) | u \in A, x \in X, x_i \neq x_j\}$ is referred to as a *cellular array*, and a list of states of cells from Ω — as a CA *global state* $\Omega_A = (u_1, u_2, \ldots, u_{|X|})$. A set of names

$$T(\mathbf{x}) = \{\mathbf{x}, \mathbf{x} + \mathbf{a}_1, \ldots, \mathbf{x} + \mathbf{a}_{n-1}\}, \quad (1)$$

[1]The research is supported by Russian Fund of Basic Research (project 11- 01-00567a), by Presidium of Russian Academy of Science (project 15.9 (2012), and Siberian Branch of RAS Interdisciplinary project N 47.

where $\mathbf{a}_j$ is a shift vector, $n = |T(x)|$, is called a *neighborhood*. The cells names from $T(\mathbf{x})$ form a *local configuration*

$$S(\mathbf{x}) = \{(u_0, \mathbf{x}), (u_1, \mathbf{x} + \mathbf{a_1}), \ldots, (u_{n-1}, \mathbf{x} + \mathbf{a_{n-1}})\}. \quad (2)$$

CA functioning is determined by a *local operator* $\Theta(\mathbf{x})$ that may be a composition

$$\Theta(\mathbf{x}) = \Phi(\theta_1(\mathbf{x}), \ldots, \theta_n(\mathbf{x})) \quad (3)$$

of *substitutions*

$$\theta(\mathbf{x}) : S(\mathbf{x}) \to S'(\mathbf{x}, \quad (4)$$

where $|S(\mathbf{x})| \geq |S'(\mathbf{x})|$, the first $m' = |S'(\mathbf{x})|$ cells in $S(\mathbf{x})$ comprising the *base* of $\theta(\mathbf{x})$, and the remaining $m - m'$ cells of $S''(\mathbf{x})$ serving as the *context*. Accordingly, the underlying neighborhoods of $S(\mathbf{x})$, $S'(\mathbf{x})$, and $S''(\mathbf{x})$ are $T(\mathbf{x})$, $T'(\mathbf{x})$ and $T''(\mathbf{x})$, respectively.

A substitution $\theta(\mathbf{x})$ is applicable to a cell $(u, \mathbf{x}) \in \Omega$, if $S(\mathbf{x}) \in \Omega$. Application of $\theta(\mathbf{x})$ results in replacing the states of cells $(u_j, \mathbf{x} + \mathbf{a}_j) \in S'(\mathbf{x})$ by

$$u'_j = f_j(u_1, \ldots, u_n), \quad n = |S(x)|, \quad j = 0, \ldots, |S'(\mathbf{x})|, \quad (5)$$

where $f_j(u_1, \ldots, u_n)$ – is a *transition function*. Context cells states remain unchanged.

Local operator (3) determines the order of its substitutions application to any cell of Ω. The most frequently used local operators are:

- Φ_S – a sequential application of all substitutions, and
- Φ_R – application of one randomly chosen substitution.

Application of $\Theta(\mathbf{x})$ to all $\mathbf{x} \in X$ comprises a global operator $\Theta(X)$. Its application changes $\Omega(t)$ into $\Omega(t+1)$, constituting the t-th iteration of the CA evolution, which is a sequence

$$\Omega(0), \Omega(1), \ldots, \Omega(T).$$

A global operator $\Theta(X)$ is *correct*, i.e. no loss of data can be caused by its execution [7], if the basic underlying neighborhoods of the substitutions, that may be applied simultaneously, do not intersect

$$\begin{array}{ll} T'_k(\mathbf{x}) \cap T'_m(\mathbf{y}) = \emptyset & \forall \mathbf{x}, \mathbf{y} \in X, \\ \forall k, m \in \{1, \ldots, l\}, & l = |\Theta(\mathbf{x}|. \end{array} \quad (6)$$

There are several modes of global transition executing, the main of them are synchronous and asynchronous.

Synchronous mode implies the following sequence of actions:

1) for all $(u, \mathbf{x}) \in \Omega(t)$ the new states u' are computed by (5),
2) in all cells $(u, \mathbf{x}) \in \Omega(t)$ the states $u(\mathbf{x})$ are replaced by $u'(\mathbf{x})$.

An essential constraint to use synchronous mode is the fact that the substitutions (4) of $\Theta(\mathbf{x})$ should have a single cell base, i.e.

$$|S'(\mathbf{x})| = 1 \quad \forall \theta_i(\mathbf{x}) \in \Theta(\mathbf{x}), \quad (7)$$

otherwise they do not satisfy (6).

Asynchronous mode implies the following procedure of local operator application:

1) a cell $(u, x) \in \Omega$ is chosen with probability $p = 1/|x|$,
2) $\Theta(\mathbf{x})$ is applied to a chosen cell and the base cells states $(u, \mathbf{x}) \in S(\mathbf{x})$ are immediately replaced by the corresponding $(u', \mathbf{x}) \in S'(\mathbf{x})$.

By condition, it is accepted that $|X|$ repetitions of 1) and 2) comprise an iteration. Such an agreement is helpful, because it is in accordance with the synchronous mode and with a definition of a step in asynchronous CA, widely used when simulating surface chemical reactions, and called by chemists as kinetic Monte-Carlo method [8].

Ordered asynchronous mode is a modification of the asynchronous mode, when $\Theta(\mathbf{x})$ is applied sequentially to the ordered cell set. Due to the fact, that in asynchronous CA local operator is applied to $\mathbf{x} \in X$ sequentially, the condition (6) is always satisfied. The problem of correct computation of asynchronous CAs arises only when the functioning of the CA is executed in parallel on several processors.

Block-synchronous mode of operation is introduced for providing efficiency of asynchronous CA parallel implementation. It is obtained by transformation of asynchronous mode into a sequence of synchronous computations satisfying correctness condition (6). Let the maximum neighborhood of $\Theta(\mathbf{x})$ be

$$T(\mathbf{x}) = \bigcup_k^n T_k(\mathbf{x}),$$

where $T_k(\mathbf{x})$ is the underlying neighborhood of $\theta_k(x)$. Then the procedure of such a transformation is as follows.

1) X is decomposed into $m = |T|$ nonintersecting subsets, $X = X_0, \ldots, X_m$, in such a way, that according to correctness condition (6), intersection of their cells neighborhoods is empty.
2) At each iteration the substitutions $\theta_k(\mathbf{x})$ are applied in m successive stages. At each l-th stage – to all $\mathbf{x} \in X_l(\mathbf{x})$, synchronously, $l = 0, \ldots, m$.
3) Data exchange between adjacent subdomains of X is performed after each stage.

Besides the above modes any other one is also admissible, if it fits the phenomenon under simulation and satisfies (6). The mode of functioning is an essential parameter of a global operator, i.e. if two CAs differ only by functioning modes, their evolutions may be quite different. Thus, a global operator, defined by its local operator $\Theta(X)$ in (3) should be indexed by the symbol denoting the mode of functioning, i.e. $\rho = \{\alpha, \beta, \sigma, \omega\}$; α, standing for asynchronous mode, β – for block-synchronous, σ – for synchronous, and ω – for ordered asynchronous.

Global composition method is used when a large scale process is to be simulated. It implies the composition of several CAs

$$\aleph = \Psi(\aleph_1, \ldots, \aleph_n), \quad \aleph_k = \langle A_k, X_k, \Theta_{k,\rho}, \Omega_k(0) \rangle.$$

where each $\aleph_k$ processes a cellular array $\Omega_k(t)$ using its own global operator $\Theta_{k,\rho}$, whose transition functions depend in some way on cell states of others $\Omega_l(t)$ of the composition.

Two following global composition methods are the most known:

- Ψ_S – the sequential CA composition, or superposition. Alphabets Ak, $k = 1, \ldots, n$, should be equal or compatible. Compatibility means that there exists a transformation, which makes them proximately equal [2]. Global operators $\Theta_k(X)$, are applied sequentially to cellular arrays, that are based on the same naming set X, but have different Ω_A. An iteration transforms $\Omega(t)$ into $\Omega(t+1)$ by applying $\Theta(X) = \Theta_1(\Theta_2(\ldots(\Theta_n(X))))$ to $\Omega(t)$.
- Ψ_P – the parallel CA composition. Alphabets A_k, $k = 1, \ldots, n$, should be equal or compatible. Each Θ_k processes its own cellular array $\Omega_k(t)$, based on its own X_k. Each global operator $\Theta(X_k)$ uses substitutions $\theta_j(\mathbf{x})$, whose transition functions (5) have basic cells belonging to $\Omega_k(t)$, and context cells, belonging to any $\Omega_l(t)$, $l = 1 \ldots, n$.

Any kind of mixed global composition is also admissible. Moreover, the evolution of the CA depends essentially on initial global state of the CA. So, a CA-model of a certain phenomenon is assumed to be completely defined if the following notions are given:

$$\mathbf{A} = \langle A, X, \Theta_\rho, \Omega(0) \rangle.$$

III. Parallel algorithm of a sequential CA composition

It is well known [9], that a CA-model, due to inherent fine-grained parallelism, can be easily partitioned into any number of fragments using domain decomposition method. But, when a CA-model is based on a composition of several CA, functional decomposition method should be also involved. The whole set of the obtained procedures together with the communications among them can be represented as a parallel algorithm of a CA-model. The notion of a parallel algorithm has no strict definition. Further it is thought of as a computation procedure, divided into communicating parts in accordance with the computer system architecture. The conventional method of a parallel algorithm design includes four following procedures [10]:

- partitioning,
- determining communications,
- agglomeration according to available resources , and
- mapping onto system architecture.

Up to date some experience in large-scale CA simulation on parallel computer systems is accumulated. It allows to formulate the following CA peculiarities, that are to be adopted in parallel algorithm constructing procedures.

1) CA evolution is an iterative procedure. Hence, a single iteration determines the CA algorithm.
2) When CA-model under simulation is a sequential composition, then the component global operators should be implemented sequentially. So, the only techniques for coarse grained parallel implementation is domain decomposition, i.e. partitioning Ω into m parts $\Omega_1, \ldots, \Omega_m$, such that

$$\Omega = \bigcup_k^m \Omega_k, \quad \Omega_k \bigcap \Omega_l = \emptyset, \quad , \forall \Omega_k, \Omega_l \in \Omega. \qquad (8)$$

3) Since CA-transition functions (5) are simple, the ratio of computation to exchange times $\gamma = T_{comp}/T_{ex}$ provides a good efficiency with large enough domain size.
4) In CA-models where block-synchronous mode of operation is used [11], the number of exchange stages should be taken into account.

Example 1.

In connection with new technologies of production and implementation of porous materials new claims are set to computer simulation of processes in them. Dealing with porous media in the same manner that with bulk materials, does not satisfy neither designers of new composites, nor researchers of soil fertility. In order to understand how the porous material interacts with the permeating fluid, it is necessary to make allowance for internal properties of the material, among which morphology and interaction character between pore walls and incoming fluid are the most important. Unfortunately, conventional mathematics, based on partial differential equations, cannot be helpful, because of impossibility to describe pore borders by means of continuous functions [5], [12]. This is why, several attempts are made to develop discrete methods of fluid permeation through porous media, because they "do not fear intricate boundary conditions". Such methods are based on Lattice-Gas [13] and Lattice-Boltzmann [14] discrete hydrodynamics, conventional cellular automata being also considered. Among them, the models based on Lattice-Gas principles are predestined to simulate laminar flows. They are appropriate when the flow constantly goes through a piece of porous material, as it happens in polymer membranes or in carbon electrodes of hydrogen energetic cells [5]. But if transient processes are of interest, such that fluid permeation to make the material damp, or, conversely, evaporation to make it dry, then Lattice-Gas based models occur to be stiff, especially for parallel implementation [15]. Hence, a more adaptable models are needed, which are capable to simulate all kinds of particles moving and interacting with solid pore walls. So, a CA-model is appropriate to be used for simulation water permeation through a 3D porous soil with given morphology. The CA-model [16] is based on a superposition of three interacting CAs:

- $\aleph_C$, simulating fluid convection, induced by external forces,
- $\aleph_D$, simulating water surface leveling by diffusion, and
- $\aleph_H$, simulating the interactions of water with hydrophilous soil inclusions.

The composed CA $\aleph = \langle A, X, \Theta(X), \Omega(0) \rangle$ has an alphabet $A = \{a, w, s, h\}$, where a stands for air, w – for water, s – for solid substance, h – for hydrophilous inclusions soaked with water; the discrete space is $X = \{(i, j, k) : i, j = 0, \ldots, N; k = 0, \ldots, M\}$;the initial cellular array $\Omega(0)$ is

given as a set of Boolean 2D arrays, corresponding to horizontal planes (Fig.1). The global operator is a superposition

$$\Theta(X) = \Phi_S(\Theta_C(X), \Theta_D(X), \Theta_H(X)),$$

where the three components are responsible for convection, diffusion and soaking hydrophilous parts of solid substance, respectively.

Convection global operator $\Theta_C(X)$ is, in its turn, the superposition of $\Theta_G(X)$ – gravitation operator, simulating water motion along the gravitation force (along k-axis down), and $\Theta_E(X)$ – evaporation operator, simulating vapor motion along opposite direction.

$\Theta_G(X)$ operates by application a substitution to all $(i, j, k) \in X$:

$$\begin{array}{ll} \theta_G(i,j,k): & \{(w,(i,j,k))(a,(i,j,k-1))\} \xrightarrow{p_G} \\ & \{(a,(i,j,k))(w,(i,j,k-1))\} \end{array} \quad (9)$$

simulates the propagation of water particles along axis k downward, letting void particles pass upward. Such a motion is simulated by M sequential steps in ordered asynchronous mode, at each l-th step $(l = 0, \ldots, M-1)$ (9) being applied synchronously to all cells having the same third coordinate $k = M - l$. The probability p_G depends on permeation intensity. In the case of soil watering p_G may be taken equal to 1, the permeation rate being considered as the most fast process.

The substitution $\theta_E(i, j, k,)$ from $\Theta_E(X)$ simulates evaporation process:

$$\begin{array}{ll} \theta_E(i,j,k): & \{(a,(i,j,k))(w,(i,j,k-1))\} \xrightarrow{p_E} \\ & \{(w,(i,j,k))(a,(i,j,k-1))\}, \end{array} \quad (10)$$

Evaporation is a phase transition process, that results in decrease of total water amount. Probability p_E depends on evaporation intensity. At each iteration $\theta_E(i, j, k)$ is applied in ordered asynchronous mode along k-axis, at each l-th step $(l = 0, \ldots, M-1)$ being applied synchronously to all cells having as a third coordinate $k = l$.

Diffusion global operator $\Theta_D(X)$ simulates the process of density equalization, which implies leveling the free surface of water, whatever it might be: in caverns, horizontal pores and over the soil. In order to conform the rates of diffusion and convection operators, the diffusion operator $\Theta_d(X)$should be applied n times sequentially, n being large enough to provide water surface smoothing.

$$\Theta_D(X) = (\Theta_d(X))^n. \quad (11)$$

The operator $\Theta_d(X)$ contains one probabilistic substitution $\theta_d(i, j, k)$ called *naive diffusion* [17], based on a five-point local configuration

$$\begin{array}{ll} S(i,j,k) = & \{(u_0,(i,j,k)),(u_1,(i-1,j,k)), \\ & \{(u_2,(i,j+1,k)),(u3,(i+1,j,k)), \\ & \{(u_4,(i,j-1,k))\} = \\ & \{(u_l,(i,j,k)_l) : l = 1,\ldots,4\}. \end{array} \quad (12)$$

The substitution $\theta_d(i, j, k)$ executes the exchange of states between the cell $(w, (i, j, k))$ and one out of those its neighbors in k-th plane, whose state $u_l = a$ i.e.

$$\begin{array}{ll} \theta_d(i,j,k): & \{(w,(i,j,k))(a,(i,j,k)_l)\} \xrightarrow{p_D} \\ & \{(a,(i,j,k))(w,(i,j,k)_l\}, \\ & l = 1,2,3,4. \end{array} \quad (13)$$

The substitution $\theta_d(i, j, k)$ is applied sequentially to all kth planes, $k = 0 \ldots, M-1$. In each k-th plane it is applied like a 2D asynchronous cellular automaton with probability $p_D = 1/m$, where m is the number of cells in $S(i, j, k) \setminus (u_0, (i, j, k))$ with $u_0 = a$. When pore walls are not smooth, and the neighborhood of $(w, (i, j, k))$ contains a pore wall cell, then $p_D < 1/m$, which implies, that some water particles may be adhered to the wall.

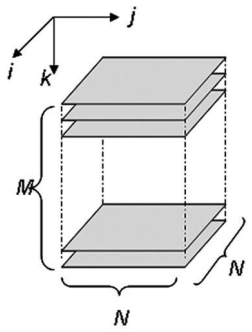

Fig. 1. A 3D cellular array structure of the porous sample

Hydrophilicity operator $\Theta_H(X)$ simulates soaking process in pore wall cells. It contains a single substitution

$$\begin{array}{ll} \theta_H(i,j,k): & \{(s,(i,j,k))(w,(i,j,k))_l\} \xrightarrow{p_H} \\ & \{(s,(i,j,k))(h,(i,j,k))_l\} \\ & l = 1,2,3,4; \end{array} \quad (14)$$

which is applied in asynchronous ordered mode along kth axis and synchronously in each k-th 2D plane.

The whole process of water penetration is observed by displaying cross-sections during the simulation (Fig.2)

The size of the soil specimen to be investigated is $700 \times 700 \times 1480$, which is too large to be implemented sequentially. Hence, it is decomposed into 49 domains each of size $100 \times 100 \times 1480$ to be processed in parallel, data exchange being performed along the planes parallel to k-axis. Asynchronous operators $\Theta_D(X)$ and $\Theta_H(X)$ are transformed into block-synchronous mode, the 3×3 neighborhood being used to form nine subsets $X_1, X_2, \ldots, X_9$, that are processed sequentially, each operating in synchronous mode. So, the parallel algorithm of an iteration consists of four sequential procedures, implemented in parallel on 49 processing units (Fig.3), data exchanges being performed $9 \times n$ times per iteration (with n diffusion cycles).

The algorithm was implemented on 8 processors each having 2-nodes Intel Xeon 5540, 2.53 GHz of the cluster NKS-30T in Siberian Supercomputer Center with the help of MPI library. The program was used to simulate the process of water

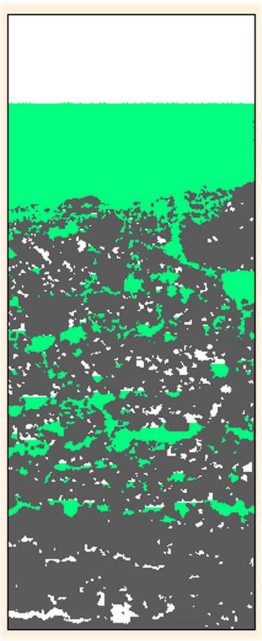

Fig. 2. The cross-section (j = 50) snapshot of a small fragment at T = 50000 with pE = pH = 0. Air is shown in white, water – in grey, soil – in black

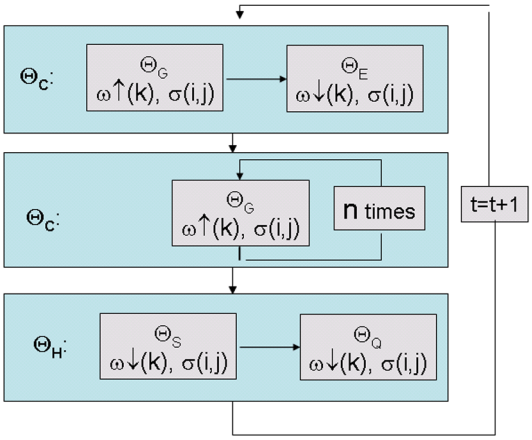

Fig. 3. Algorithm of soil permeation CA-model, performed on a domain of the cellular array.

permeation, together with studying the impact of evaporation and hydrophilicity on permeation rate and depth.

The above CA-model is remarkable by the fact that its convection and diffusion parts are separated both in time and in space. This feature helps essentially to program the parallel version, reducing asynchronous computations to a 2D case, which manifests how advantageous is to anticipate coarse grained parallelization in the course of CA-model development.

IV. Parallel Algorithm of Parallel CA Composition

Parallel CA composition is a set of CA, operating in common in such a way that each CA transition functions variables are cell states of any CA of the system. By analogy to partial differential equations of traditional numerical analysis, parallel composition of CA are further called *CA systems* [3]. Nowadays CA systems are used mostly in scientific investigation being implemented on personal computers, many computational experiments on one and the same CA being performed with many different parameters, which should be easily and promptly changed. Although the computational time is wanted to be as small as possible, there is no need to run the programs on remote powerful clusters, each time waiting for the results. But, having a two-, four- or eight-core computer with a shared memory on the table, it is reasonable to make the cores operate in parallel, in correspondence with the parallel composition in the CA system.

Parallel CA composition $\aleph = \Psi_P(\aleph_1, \ldots, \aleph_n)$ implies that the alphabets A_k of all $\aleph_k$, $k = 1, \ldots, n$, are compatible. Naming sets $X1, \ldots, Xn$ are isomorphic, i.e. any $\mathbf{x}_k \in Xk$ has its counterpart $\mathbf{x}_l \in X_l$, represented by the same coordinate set (i, j, k), which is marked by the component indices as $(i, j, k)_l$ or $(i, j, k)_k$ if it is necessary.

Local operators $\Theta_k(\mathbf{x})$ consists of substitutions, whose context parts $S''_k(\mathbf{x}) \subseteq S_k(\mathbf{x})$ include cells belonging to any $\Omega_l \in \Omega$. Hence, the underlying neighborhood $T_k(\mathbf{x}_k)$ of a local configuration $S_k(\mathbf{x})$ comprises cell names from $X = X_1, \cup X_2 \cup, \ldots, \cup X_n$, i.e.

$$T_k(\mathbf{x}) = T'_k \cup T''_k, \quad T''_k = \bigcup_{l=1}^{n} T''_{kl}, \tag{15}$$

where $T''_{kl}(\mathbf{x}) \subset X_l$ is the part of the underlying context neighborhood of $\theta_k(\mathbf{x})$ allocated in Ω_l.

The whole process of simulation consists of a sequence of iterations. An iteration presumes execution of all global operators: $\Theta_1(X_1), \ldots, \Theta_n(X_n)$, constituting a global transition of the system: $\Omega_k(t) \to \Omega_k(t+1)$ for all $k = 1, \ldots, n$.

CA system may operate in synchronous and asynchronous modes. The correctness condition (6) is valid for global transition of the parallel composition with taking into account that the context cells are allocated in several X_l, $l = 1, \ldots, n$.

Parallel CA composition is appropriate for simulating reaction-diffusion processes where several interacting species are involved. Each species is put into correspondence to a CA $\aleph_k = \langle A_k, X_k, \Theta_k(X_k)\rangle$, being a superposition of a diffusion CA $\aleph_{kd} = \langle A_k, X_k, \Theta_{kd}(X_k)\rangle$, and a reaction CA $\aleph_{kr} = \langle A_k, X_k, \Theta_{kr}(X_k)\rangle$. Herein, diffusion operators process their own cellular arrays independently, while reaction local operators use the common pool of variables (Fig.4).

Example 2.

The typical example of CA parallel composition is simulation of self organization process in a prey-predatory system. The predator (fish) eats the prey (plankton). If there is enough of food, predator density increases (predator propagates) with

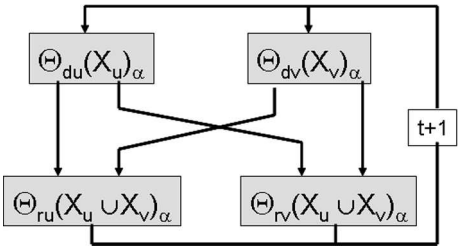

Fig. 4. Parallel algorithm of two reaction-diffusion CAs

the probability depending on the satiated predator density. In case of food shortage predator density diminishes (predator die of hunger). The prey always attempts to propagate, when not being eaten by the predator. Since the predator is usually more agile than the prey, its diffusion is many times larger, than that of prey ($d_v \gg d_u$).

Let us simulate these relationships by a parallel composition $\Psi_P(\aleph_v, \aleph_u)$, where indices u and v indicate prey and predatory, respectively. Both CAs have Boolean alphabets. The naming sets X_u, X_v are 2D 400×800 finite lattices, whose cells names are in one-to-one correspondense. The cells with corresponding names in $X_u = \{(i,j)_u\}$ and $X_v = \{(i,j)_v\}$ are referred to as twins, lower indices being omitted when it does not lead to ambiguity. Population local densities

$$V((i,j)_v) = \frac{1}{|Av(i,j)_v|} \sum_{Av(i,j)_v} v(k,l)_v, \tag{16}$$

$$U((i,j)_u) = \frac{1}{|Av(i,j)_u|} \sum_{Av(i,j)_u} u(k,l)_u, \tag{17}$$

$$\tag{18}$$

are averaged values over the local configurations with an underlying neighborhood

$$Av(i,j) = \{(k,l) : k = i+g, l = j+h, g,h = -R,\ldots,R\},$$

$R = 8$ being the radius of averaging in both cellular arrays.

Operators $\Theta_v(X_v)$ and $\Theta_u(X_u)$ are global superpositions of diffusion and reaction global operators. Hence,

$$\Theta_v(X_v) = \Theta_{vr}(\Theta_{vd}(X_v)), \quad \Theta_u(X_u) = \Theta_{ur}(\Theta_{ud}(X_u)).$$

Diffusion for both species is simulated by naive asynchronous CA, given in Example 1 by the substitution $\theta_d(i,j)$ ((13) without k). To provide the agility of the predator to be m times larger than that of the prey, the global operator $\Theta_d(X_v)$ should be executed m times, i.e.

$$\Theta_{vd}(X_v) = \Theta_d^m(X_v).$$

The predator reaction is simulated by synchronous operator $\Theta_{vr}(X_v)$, consisting of a single substitution

$$\theta_{vr}(i,j) : \{(v,(i,j)_v),(U,(i,j)_u)\} \xrightarrow{p} \{(v',(i,j)_v)\}, \tag{19}$$

where

$$v' = \begin{cases} 0, & \text{if} \quad (V(i,j) - U(i,j)) > 0 \quad \& \quad rand) < p_{v\to 0}, \\ 1, & \text{if} \quad (V(i,j) - U(i.j)) < 0 \quad \& \quad (rand) < p_{v\to 1}. \end{cases}$$

The probabilities $p = p_{v\to 0}$ and $p = p_{v\to 1}$ are determined based on the following considerations:

- If $(V(i,j) - U(i,j)) > 0$, the predator in (i,j) has lack of food and may die. So, some cell states in $Av((i,j)_v)$ should be inverted into "zero" [18], which yields in
$$p_{v\to 0} = (V(i,j) - U(i,j))/U(i,j).$$
- If $(V(i,j) - U(i,j)) < 0$, then the predator in (i,j) has plenty of food, and propagates increasing its density according to the propagation function $F_v(V) = c_v V(i,j)(1 - V(i,j)$ [19], where c_v is a coefficient corresponding to the type of the predator, being taken in the example as $c_v = 0.5$. So.
$$p_{v\to 1} = 0.5V(i,j)(1 - V(i,j)).$$

Prey reaction is simulated by synchronous operator Θ_{ur}, represented by a single substitution

$$\theta_{ur}(i,j) : \{(u,(i,j)_u),(V,(i,j)_v)\} \xrightarrow{p} \{(u',(i,j)_u)\}, \tag{20}$$

where

$$u' = \begin{cases} 0, & \text{if} \quad (U(i,j) - V(i,j)) > 0 \quad \& \quad rand) < p_{u\to 0}, \\ 1, & \text{if} \quad (U(i,j) - V(i.j)) < 0 \quad \& \quad (rand) < p_{u\to 1}. \end{cases}$$

probability values $p = p_{u\to 0}$ and $p_{u\to 1}$ being obtained based on the following considerations:

- If $(U(i,j) - V(i,j)) < 0$, the prey in (i,j) is freely eaten. So, its density decreases with probability proportional to predator density $p_{u\to 0} = V(i,j)$.
- If $(U(i,j) - V(i,j)) > 0$, the prey in (i,j) propagates with probability depending on the number of the remainders
$$p_{u\to 1} = c_u(U(i,j) - V(i,j))(1 - (U(i,j) - V(i,j))).$$

The CA-model was tested with the following initial cellular array $\Omega_v(0) \cup \Omega_u(0))$, where

$$\begin{array}{ll} v(i,j) = 1 & \text{for} \quad i = 0,\ldots,399; \quad j = 369,\ldots,439, \\ v(i,j) = 1 & \text{if} \quad (rand) < 0.1 \quad \text{for others } (i,j) \in X_v, \\ u(i,j) = 1 & \text{if} \quad (rand) < 0,4 \quad \forall (i,j) \in X_u, \\ u(i,j) = 0, & v(i,j) = 0, \quad \text{otherwise.} \end{array}$$

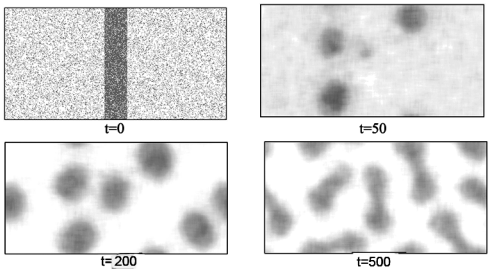

Fig. 5. Three snapshots of the evolution of predatory and prey CA, at t=2000 the system is in a stable state

The above CA system was implemented on Intel core i7 computer using OpenMP tools. Implementation in two threads

TABLE I
COMPUTATION TIME OF 500 ITERATIONS AND SPEEDUP OF 1, 2, AND 4 THREAD IMPLEMENTATION OF THE CA-SYSTEM OF EXAMPLE 2

N threads	1		2		4	
	time	speedup	time	speedup	ime t	speedup
Dv/Du=50	134	1	77	1.74	47	2.35
Dv/Du=1	150	1	77	1.94	57	2.63

was made by combining operation of $\aleph_{vd}$ and $\aleph_{vr}$ in one thread, and $\aleph_{ud}$ and $\aleph_{ur}$ — in the other thread. Implementation of the system in four threads has been performed by allocating each part of the above algorithm (Fig.4) onto a thread. Due to the difference of the diffusion coefficients the computation load in the threads differs essentially, which does not allow to obtain the perfect speedup. Implementation of a CA-system with equal thread load (Dv=Du=50) results in the speedup which is close to the number of threads. The obtained results with $T = 500$ of the above experiments are summarized in Table 1. It is seen from the Table, that the speedup does not depend on the parallelization method, depending only on the imbalance of thread load. Four snapshots of $\aleph_v$ evolutions are shown in Fig.5. The stable state is reached by $T = 1500$ iterations.

A bright example of CA parallel composition is presented in [20], where a CA-model of the dynamics of eight organisms living in Lake Baikal is simulated using 4 threads on 8-core computer with the speedup equal to 3.5.

V. CONCLUSION

It is shown that CA composition techniques comprise the base for a systematic approach to the construction of a coarse-grained parallel algorithms for implementing CA-models on modern supercomputers both with distributed and with shared memories. Of course, the approach may be easily extended to any kind of mixed parallel-sequential composition. The approach would be helpful for constructing large scale programming systems for simulating real complex phenomena using CA models.

REFERENCES

[1] J. Kroc, A. G. Hoekstra, P .M .A. Sloot, Eds., *Simulating Complex Systems by Cellular Automata. Understanding complex Systems*. Berlin: Springer, 2010.

[2] O. Bandman. "Cellular Automata Composition Techniques for Spatial Dynamics Simulation". In: *Simulating Complex Systems by Cellular Automata. Understanding complex Systems*,J. Kroc, A. G. Hoekstra, P .M .A. Sloot, Eds., Berlin: Springer, 2010, pp. 81-115.

[3] O. Bandman. "Using Multi Core Computers for Implementing Cellular Automata Systems". *Proc. of the Int. Conf. on Parallel Computing Technolofgy, PaCT-2011*. V.Malyshkin, Ed. LNCS 6873, Berlin: Springer, 2011, pp. 145-157.

[4] K. Kalgin. "Parallel implementation of asynchronous cellular automama on 32-core computer". *Siberian J.Num.Math.*, SBRAS Press, Novosibirsk, 15(1), 2012, pp. 55-65.

[5] O. Bandman. "Using cellular automata for porous media simulation", *J.of Supercomputing*, 57(2), 2011, pp.121-131.

[6] Yu. Medvedev. "Dynamic Load Balancing for Lattice Gas Simulations on a Cluster". *Proc.of the Int. Conf. on Parallel Computing Technology, PaCT-2011*. V.Malyshkin, Ed. LNCS 6873, 2011, pp. 181-191.

[7] S. Achasova and O. Bandman. *Correctness of parallel Processes*, Novosibirsk: Nauka. 1999.

[8] A. P. J. Jansen. *An Introduction to Monte-Carlo Simulation of Surface Reactions*, arXiv:cond-mat/0303028v1[stst-mech]. 2003.

[9] O. Bandman. "Parallel implementation of cellular autmata spatial dynamics algorithms". *Siberian J.Num.Math.*, Novosibirsk: SBRAS Press, 10(4), 2007, pp.345-361.

[10] G. E. Blelloch, and B. M. Maggs. "Parallel Algorithms". *ACM Computing Surveys*, 28, N1, 1996, pp.51-54. .

[11] O. Bandman. "Parallel Simulation of Asynchronous Cellular Automata Evolution". *Proc. of the 7th Int. Conf. on Cellular Automata, for Research and Industry, ACRI 2006*, S. El Yacoubi, B. Chopard, S. Bandini, Eds. LNCS 4173, 2006, pp.41-48.

[12] L.M. Keller et al. "3D geometry and topology of pore pathways in Opalinus clay: Implications for mass transport". *Appl. Clay Science*, 52, 2011, pp.85-95.

[13] B. H, Rothman, S. Zaleski. *Lattice-Gas Cellular Automata. Simple Models of Complex Hydrodynamics*. London: Cambridge Univ. Press, 1997.

[14] H. Hidemitsu. "Lattice Boltzmann Method and its Application to Flow Analysis in Porous Media". *R&D Review of Toyota CRDL*, 38,N 1, 2007, pp. 17-25.

[15] U. Frish, B. Hasslacher, Y. Pomeau. "Lattice-gas Automata for Navier-Stokes equation". *Phys. Rev. Letters*, 56, 1986, pp.1505-1508.

[16] O. Bandman. "Cellular Automata Simulation of Water Permeation through Porous Media". Submitted to the *Int. Conf. on Parallel Computing Technolofgies PaCT-2013*.

[17] T. Toffolli, and N. Margolus. *Cellular Automata Machines: A new Environment for Modeling.* USA:MIT Press. 1987.

[18] O. Bandman. "Algebraic Properties of Cellular Automata: the Basis for Composition Technique". *Proc. of the 6th Int. Conf. on Cellular Automata for Research and Industry, ACRI 2004*, (P. M. A. Sloot, B. Chopard, A. G. Hoekstra, Eds. LNCS 3305, 2204, pp. 688–697.

[19] A. Medvinsky et al. "Spatiotemporal Complexity of Plankton and Fish Dynamics". *SIAM Review*, 44, N 3, 2002, pp. 311-370.

[20] I. Afanasiev. "The CA-model of populations dynamics of organisms living in Baikal". *Bull. of the Novosibirsk Computing Center, Series Computer Science*, Novoisibirsk: NCC Publisher, N 30, 2012, 1-12.