A Distributed Cellular Automata simulations on cluster of PCs

Paweł Topa

Institute of Computer Sciences, University of Mining and Metallurgy, al. Mickiewicza 30, 30-059 Kraków, Poland
email: topa@uci.agh.edu.pl

Abstract. Inherent parallelism of Cellular Automata as well as large size of automata systems used for simulations makes their parallel implementation indispensable. The purpose of this paper is to present a parallel implementation of two sequential models introduced for modelling evolution of anastomosing river networks. Despite the both models exploit the Cellular Automata paradigm, the nontrivial application of this approach in the second model involves defining a mixed parallel-sequential algorithm. The detailed description of the two algorithms and the results of performance test are presented.

1 Introduction

Cellular Automata (CA) [1] have gained a huge popularity as a tool for modelling problems from complex dynamics [2]. They can simulate peculiar features of systems that evolve accordingly to local interactions of their elementary parts. CA can be represented by n-dimensional mesh of cells. Each cell is described by a finished set of states. The state of the automaton can change in successive time-steps according to defined rules of interactions and the states of the nearest neighbours.

This paper introduces two Cellular Automata models implemented on cluster system. The detailed description of their sequential versions and results obtained can be found in [3]. The short review is presented below.

The presented models simulate evolution of anastomosing river networks. Anastomosing rivers develop on flat, wide areas with a small slope. The main reason of creation and existence of such the systems is a plant vegetation. Plants receive necessary resources (nutrients) from water. Products of their decay is accumulated in the interchannel areas as peatbogs and follows to rising up of the banks.

The rate of plants vegetation is controlled by the nutrients. The gradient of nutrients saturation, which appears perpendicularly to the channel axis, results in faster accumulation of peats near banks and slower accumulation on distant areas. The water plants vegetating in the channels can block the flow. In such the situation, a new channel have to be created. It starts above the jam zone and its route is determined by the local topography. Such the channel may join back

P.M.A. Sloot et al. (Eds.): ICCS 2002, LNCS 2329, pp. 783–792, 2002.

to the main river bed downstream. The evolution of the newly created branch proceeds in similar way as the evolution of the main stream. Finally, such the processes results in creating of complex, irregular network of interconnecting channels having hierarchical and fractal structure.

First model, applied for simulating anastomosing rivers, was called SCAMAN (Simple Cellular Automata Model of Anastomosing Network) [3]. It exploits the Cellular Automata paradigm in a classical way. The system is represented by a regular mesh of automata. The state of each automaton is described by the following parameters: (1) the altitudes resulting from the location of the cell on terrain mesh, (2) the amount of water, (3) the amount of nutrients and (4) the thickness of peat layer. Defined rule of water distribution simulates superficial flow. The cells containing water are the sources of nutrients, which are disseminated among surrounding cells in such a way which is able to provide expected gradient of nutrient saturation. The thickness of peat layer is updated accordingly to the amount of nutrient in the cell.

Due to limitations of SCAMAN model, another model was proposed. In MANGraCA (Model of Anastomosing Network with Graph of Cellular Automata), a network of river channels is represented by the directed graph of CA. The graph is constructed on the classical, regular mesh of Cellular Automata by establishing additional relationships between some neighbouring cells.

The state of each automaton in the regular mesh is described by three parameters: (1) the altitude, (2) the amount of nutrient and (2) the thickness of peat layer. Their meaning are the same as in SCAMAN model. Also their values are calculated by using similar algorithms.

When a new cell is added to the graph, its state is additionally described by two parameters: (a) the throughput and (b) the flow rate. These parameters describe local state of a part of the channel. The flow rate in the first cell (source) is initialized to an arbitrary value and then propagated to other cells currently belonging to the graph. Overgrowing of the channel is modelled by slowly decreasing of the throughput value. Occasionally, in randomly chosen cell, the throughput is suddenly decreased below the flow rate. This corresponds to creation of flow jam in the channel. Such the event leads to creation of a new channel, which starts before the blocked cell. The route of the new channel is determined by a local topography of the terrain.

The cells which belong to the graph are the sources of nutrients. Nutrients are spread to other cells in the mesh by using the same algorithm as in SCAMAN model. This way, the graph stimulates changes in terrain topography, which consequently influences development of the network.

Neither SCAMAN nor MANGraCA are able model entirely the anastomosing river. MANGraCA produces global pattern of anastomosing network, but without any information about local distribution of water in the terrain. SCAMAN simulates flow of water in terrain but in order to obtain more complex network a very large mesh has to be applied and several thousands of time-steps of the simulation must be performed. The models work in different spatio-temporal scales. Basing on this observation a hybrid multiresolution model have been proposed in

which MANGraCA model produces global river pattern, while SCAMAN, basing on generated network, calculates the local water distribution and simulates the growth of the peat layer.

MANGraCA can be also applied to modelling other network structures. It can be useful to modelling the evolution of transportation network immersed in consuming environment such as vascular system, plant roots, internet.

Parallel computers are the most natural environment for Cellular Automata simulation [4], [5]. In fact, sequential simulation of real phenomena using CA, where a huge number of automata are employed to represent the system, are impossible in practice. For a long time, such the simulations required access to massively parallel processors (MPP) located in supercomputer centers. The breakthrough has been brought by Beowulf [6], the first low cost parallel computer built in NASA laboratory. The Beowulf was constructed using only low priced COTS (commodity of the shelf) components. Regular PC computers with Intel 486 processors were deprived of floppy disks, keyboards and monitors and connected using 10Mbit Ethernet. The system worked under Linux operating system. Some large computations were successfully performed on this machine. At present, the Beowulf name relates to the certain class of clusters, followed by the example of the machine built at NASA.

Simple recipe [7] of cluster assembling and low cost of components (software is mostly public domain) enables parallel computing for institutes with limited financial resources. Low cost of maintenance and homogeneity of the environment simplify computations. Allocating the whole or part of node for exclusive use is easy and allows to neglect the load balancing problems.

Clusters share many features with MPP architecture. Application of popular paralellel computing libraries (like MPI and PVM) makes easy to port the algorithms from clusters to MPP machines. Clusters can be used as developing and testing platforms.

The next two sections contain detailed description of Parallel-SCAMAN and Parallel-MANGraCA. Some results of performance tests on cluster of PCs are presented. Conclusions are discussed at the end.

2 Parallel-SCAMAN

In Parallel-SCAMAN the mesh of cells is geometrically decomposed onto domains which are processed independently on different nodes. Mesh can be divided on blocks or stripes. The striped partitioning is preferred due to simplified communication and load balancing implementation.

The nodes in parallel machine are ordered in one-dimensional array: $P_1, \ldots, P_N$, where N stands for number of nodes participating in computation. Processor P_1 processes columns from 1 to m, P_2 — from $m+1$ to $2m$ and so on ($m = N/M$ where M stands for total number of columns in the mesh). Each node store two additional columns, which are not processed by this node (see Fig. 1a — dark grey marked cells). These columns contain copy of the border cells (light gray

marked cells in Fig. 1a) from the adjacent processors. Such the copy is necessary on P_i node to update its own border cells. After each time-step P_i exchanges the border columns to its neighbours: P_{i-1} and P_{i+1}. The exceptions are P_0 and P_N, which exchanges data with only one neighbour. Such the scheme applies only to calculating the nutrients distribution and the thickness of peat layer.

The algorithm, which calculates the water distribution consist of two stages.

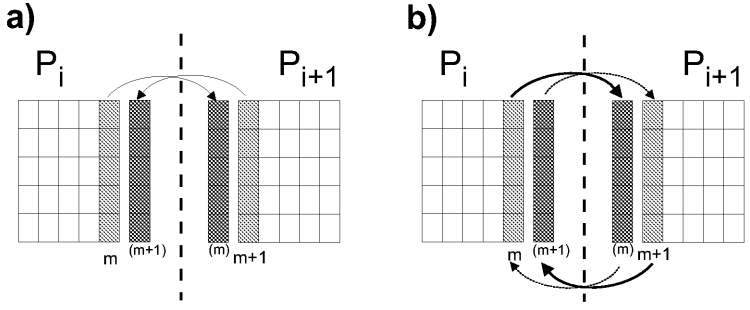

Fig. 1. Exchange of data in Parallel-SCAMAN: a) nutrients and thickness of peat layer, b) flows.

First, for each cell in the mesh the flow rate from this cell to each of its neighbours is calculated. This means, that two border columns with flow rates must be exchanged between neighbouring nodes (see Fig. 1b). On P_i node light greyed column updates the copy of water data on the adjacent processor (P_{i+1}). The second column (dark greyed) contributes to the total flow rates for the border cells on P_{i+1} processors. In the following step of the algorithm, current amount of water is calculated for each automaton basing on the flow rates.

On a single P_i processor the following operations are executed:

```
loop:
    update_water();
    update_nutrients();
    update_peats();
    exchange_data(P_{i+1}); {flow, nutrients, peats}
    exchange_data(P_{i-1});
```

In Parallel-SCAMAN the *Master* node is required only to gather current data from *worker* in order to store them or for visualization. The changes in state of cells, propagate very slowly through the mesh, especially for very large meshes. The performance can be improved if the *Workers* do not communicate with *master* in every time-step of simulation. Data can be gathered only every 10, 100 or more time-steps. The interval value depends on the size of system and quality required producing images or animation for inspection.

3 Parallel-MANGraCA

In MANGraCA model, the number of cells participating in graph computation is significantly less than their total number. The structure of the model allows for separating the computations connected with graph structure from the processing of the regular mesh.

Fig. 2 presents the general Parallel-MANGraCA architecture. The distinct processor (P_0) performs sequentially all operations concerned the graph i.e. throughput updating and flow propagation. It also handles the changes in graph topology caused by newly created branches.

The algorithm, which trace the route of a new branch requires the most up-to-date data about the terrain topography. It means that the altitude data and thickness of peat layer must be also stored on this node.

Distribution of nutrients and updating of the thickness of peat layer are

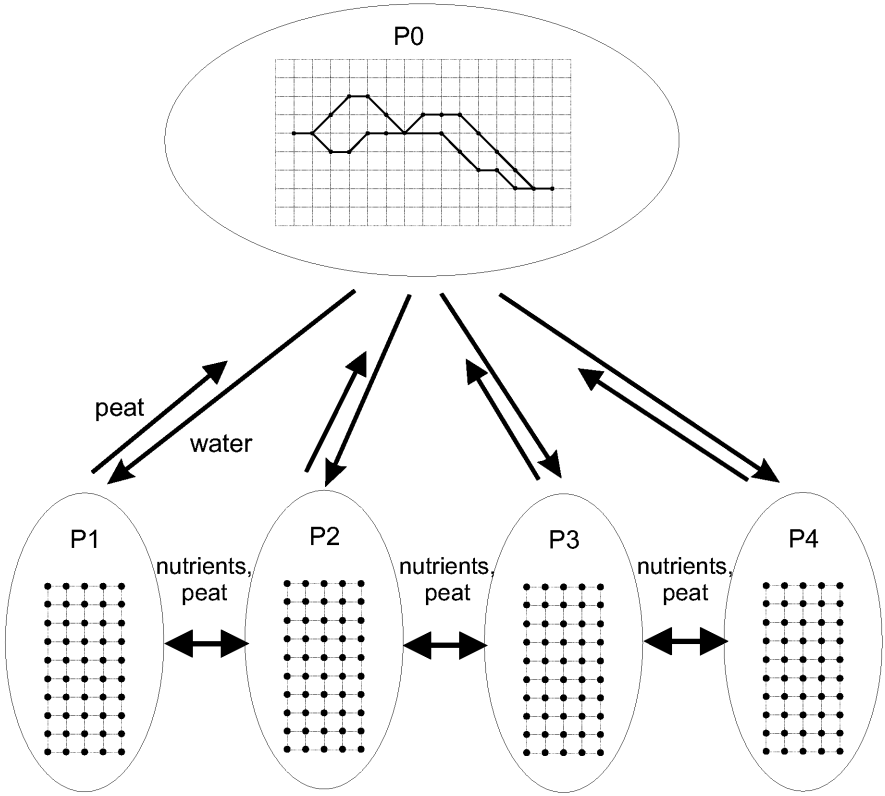

Fig. 2. General Parallel-MANGraCA architecture

performed on a regular mesh of automata. It can be easily implemented in par-

allel, similarly as it is in SCAMAN model. The mesh is distributed between the nodes of clusters ($P_1 \ldots P_N$). The algorithm, which calculates the nutrients distribution, requires information, which cells are the sources of nutrients (i.e. which automata currently belongs to the graph). Therefore, the three types of communication must be provided:

1. P_0 sends information to all P_i ($i = 1, 2, \ldots, N$), about which automata currently belongs to the graph.
2. Each P_i sends to P_0 information about the changes in the landscape, i.e., current thickness of the peat layer.
3. In every time-step, each P_i exchange the information about the state of boundary cells (the nutrient distribution and the thickness of the peat layer) with its neighbours: P_{i+1} and P_{i-1}.

In Fig. 3 the diagram of the Parallel-MANGraCA algorithm is presented.

The time, which processor spends on graph calculations, is incomparable

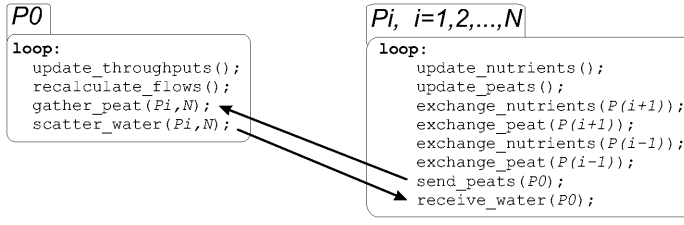

Fig. 3. Basic Parallel-MANGraCA algorithm

shorter than processing of the whole mesh. The Parallel-MANGraCA can be easily improved by allocating the part of the mesh to P_0. This processor will be more loaded than the other, especially when the graph will grow. The load balancing for this node can be provided by allocating smaller portion of mesh at startup.

A substantial problem which is generated by the approach presented consists of large amount of data, which has to be exchanged between P_0 and P_i in each step of simulation. The mesh with current peat thickness is gathered on P_0, which next broadcast the graph to the all P_i. In fact, such the communication is necessary only when a jam has just been occurred and the route of newly created branch must be calculated. In such the situation P_0 sends a request to all P_i nodes. The P_i processor, in each time-step looks for such the request and in case of such the event, it sends to P_0 their parts of mesh with the thickness of peat layer. Then P_0 node, basing on the new terrain topography, calculates, the route of the new branch. The reconfigured graph is broadcasted to all P_i nodes.

Processing of the graph structure and regular mesh can be performed more

independently. In the approach presented above, P_0 node processes one part of the mesh so it is synchronized with P_i. In Asynchronous Parallel-MANGraCA, mesh and graph are processed independently. Mesh computations are executed in parallel (e.g. on cluster), while the graph is processed sequentially on a separate machine. Communication occurs only when the graph is reconfigured.

4 Results

Parallel-SCAMAN and Parallel MANGraCA have been implemented in C language, using MPI (mpich 1.2.0) library. The models implementation have been running and testing on the Beowulf-like cluster of PCs, consisting of up-to 8 uniprocessors Intel Pentium III Celeron Coppermine 600 MHz (cache L2 128 kB) computing nodes with 64 MB of memory per node plus one Intel Pentium III Katmai 500 MHz (cache L2 512 kB) as a front-end node. The nodes are connected with Fast Ethernet (100 Mbit/s).

Fig. 4 and 5 show the results of speedup measurements. Parallel algorithms have been compared with its sequential equivalents. Thus, we can see real benefits from porting models to the parallel architecture.

As shown Fig. 4 the Parallel-SCAMAN scales linearly with increasing number of processors and the job size (see Fig. 4). In this algorithm, practically there is no serial fraction. The only factor, which worsen the speedup is communication. As it was expected the best performance is obtained when the nodes are fully loaded. Otherwise, the communication will take more time than computation. The algorithm scales better for larger problems. If a small job is computed, speedup closes to linear only for 2-3 nodes. When more node are applied, the speedup is decreasing. The same result will appear if the faster processor is applied.

Fig. 5 presents the results of tests with Parallel-MANGraCA. In these tests the front-end node was exploited as P_0 machine. Basic algorithm of Parallel-MANGraCA has very poor efficiency (diamonds). Allocating the part of mesh for processor P_0 gives insignificant profits (triangles). The real improvement have just gave Parallel-MANGraCA with reduced graph-mesh communication (square and circle).

The development of anastomosing network (represented by the graph) is a stochastic process. The creation of a new branch occurs in randomly chosen moment of time. For the tests, the parameter T_b describing the mean time between the events of creation of new branches, has been introduced. Its value is dependent on parameters of graph processing i.e. the rate of throughput decrease and the probability of jam occurrence. The greater value of T_b means less frequent formation of new branches, which results in the reduction of time spent fo communication between P_0 and P_i nodes. In Fig. 6a we present the influence of T_b on the Parallel-MANGraCA efficiency. Fig. 6b shows, how the algorithm scales with the mesh size.

Unlike the Parallel SCAMAN, Parallel-MANGraCA has a significant serial fraction which influence badly the speedup. Also communication overhead worses

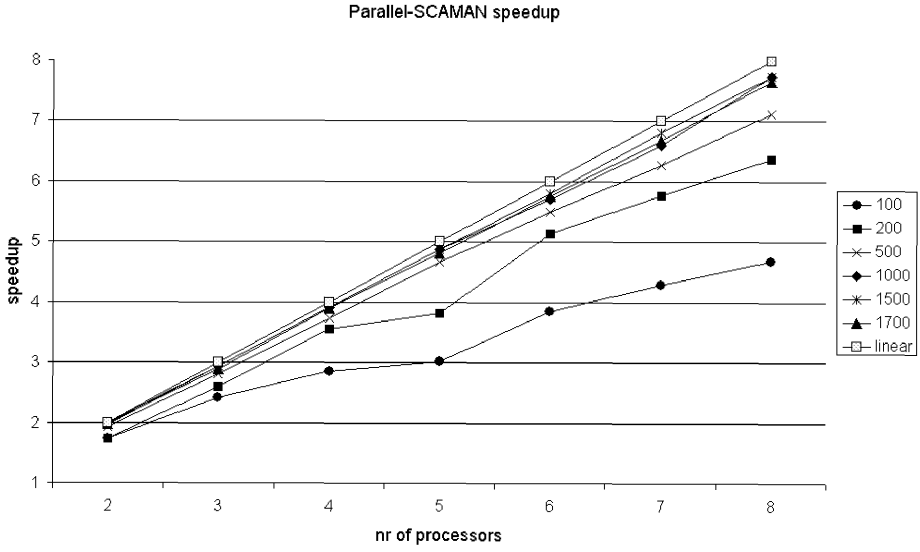

Fig. 4. Speedup of Parallel-SCAMAN

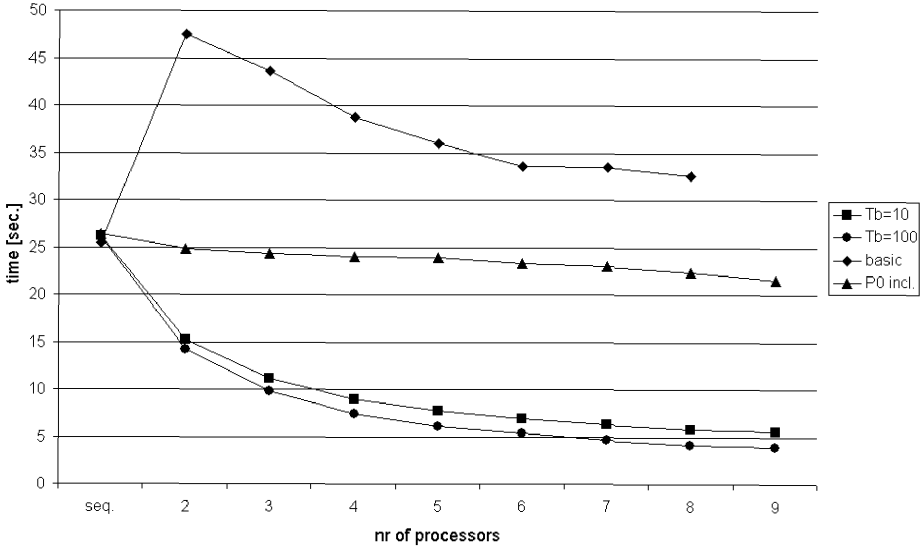

Fig. 5. Comparison of three Parallel-MANGraCA algorithms (see text for details)

the efficiency. An improvement of processor performance will result in the same issue as it is in SCAMAN model.

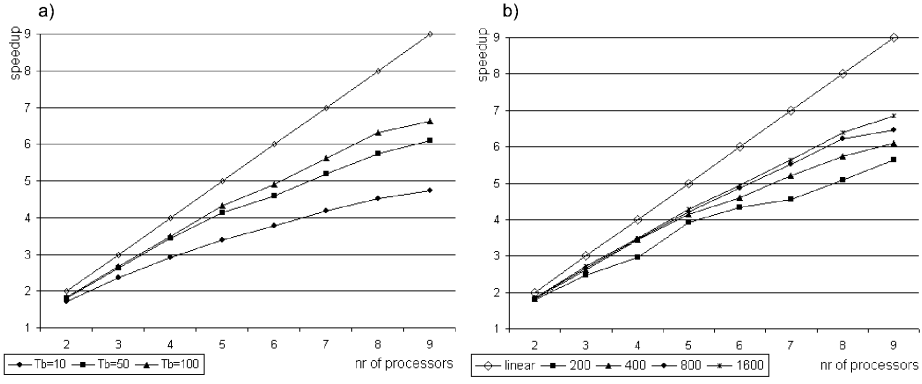

Fig. 6. Parallel-MANGraCA with reduced mesh comunication: a) speedup vs. T_b (400×400 cells, b) speedup vs. mesh size ($T_b = 50$)

5 Conclusions

The two parallel Cellular Automata models implemented on cluster architecture has been introduced. Their conception base on sequential SCAMAN and MANGraCA model introduced for modelling anastomosing river network. This paper presents general architecture of these models and preliminary results of performance tests. Homogenous architecture of cluster environment as well as exclusive availability of necessary numbers of nodes allows to neglect some critical aspects of parallel algorithms such as load balancing. The works presented is concentrated rather on designing, implementing and testing general model structure than on maximizing performance on a specific architecture. In the future, when the Parallel-SCAMAN, Parallel-MANGraCA will be ported to high performance MPP architecture, the procedures of load balancing will be provided.

Parallel-SCAMAN was designed in a classical way. Its algorithm base on simple division of processed mesh between the nodes of clusters. The results of performance tests show clearly the benefits of parallel computing especially for large and very large tasks.

The concepts tested in Parallel-SCAMAN were applied in Parallel-MANGraCA. Due to the nontrivial application of Cellular Automata paradigm, the original parallel-sequential architecture have been implemented to obtain satisfactory efficiency. Furthermore, this approach gives an interesting issue in studies on mutual interaction of two coupled systems. Some of the future works should concentrate on Asynchronous Parallel-MANGraCA. The another area of studies

on Parallel-MANGraCA may concern the choice of computer architecture, on which the model can be implemented in more efficient way. Separation of the graph and the mesh computations allows to perform the simulations in non-homogenous environment, e.g., the mesh can be computed on inexpensive PC cluster, while the graph may be processed and visualized on fast workstation with enhanced graphical capabilities.

Basing on the results presented in this paper, the hybrid model will be also implemented. Its framework will be based on Parallel-MANGraCA, but with larger participation of the parallel code coming from Parallel-SCAMAN model (i.e. calculating of water distribution, nutrient distribution and peat layer update). This should result in better overall efficiency.

6 Acknowledgments

Author is grateful to Drs Witold Dzwinel and Krzysztof Boryczko from AGH, Institute of Computer Sciences for valuable discussions. This project was partially supported by The Polish State Committee for Scientific Research (KBN) under grant 7T11C00521.

References

[1] S. Wolfram, *Cellular Automata and Complexity: Collected Papers*,1994,
[2] B. Chopard and M. Droz, *Cellular Automata Modeling of Physical Systems*, Cambridge University Press 1998,
[3] P. Topa, M. Paszkowski, Anastomosing transportation networks, In *Proceedings of PPAM'2001 Conference*, Lecture Notes in Computer Science, 2001,
[4] D. Talia, Cellular Automata + Parallel Computing = Computational Simulation, *Proc. 15th IMACS World Congress on Scientific Computation, Modelling and Applied Mathematics*, vol.6, pp.409-414, Wissenschaft&Technik Verlag, Berlin, August 1997,
[5] G. Spezzano, D. Talia, Programming cellular automata algorithms on parallel computers, *Future Generations Computers Systems*, 16(2-3):203-216, Dec. 1999,
[6] http://www.beowulf.org,
[7] T.L. Sterling, J. Salmon, D.J. Becker, D.F. Savarese *How to build a Beowulf?*, MIPT Press, 1999,