Software Tool for Cluster-based Modeling of 2D Cellular Automata *

Edmon Davtyan
Institute for Informatics and Automation Problems
Yerevan, Armenia
e-mail: edmon@ipia.sci.am

Hasmik Karapetyan
Institute for Informatics and Automation Problems
Yerevan, Armenia
e-mail: hasmikek@ipia.sci.am

Karine Shahbazyan
Institute for Informatics and Automation Problems
Yerevan, Armenia
e-mail: shahb@ipia.sci.am

ABSTRACT

Th software tool **SAS+** (Systolic Algorithm Simulator Plus) allows to model two dimensional generalized cellular automata in the cluster-based environment. The suggested tool supports:
• spatial and temporal inhomogeneities of cells transition times;
• date-dependence of transition time of cells;
• dynamically resizing the successive configurations;
• arbitrary number of states of cells;
• use of external parameters.
The result of the SAS+ execution is a cluster-oriented programming module which imitates the cellular automaton.

Keywords

Cellular automata, systolic array, Beowulf cluster computer, software tool.

1. INTRODUCTION

A lot of software packages for cellular automata (CA) simulation has been designed. A survey on programming environment for CA is provided for example in [3, 10].

Usually, virtual CA exploit data parallelism by partitioning *cells* of initial configuration among processors of parallel computer. Each processor updates those cells that are assigned to it. In such a case simulation procedure involves two cyclic synchronized phases: configuration's states computation and inter-processor communications phases. At the communication phase the system has to be synchronized to start the phase of the next configuration's computation.

Our goal is the reduction of management overhead time. We consider two causes of growth of management overhead time.

1. The management overhead time depends on processors idle time. The cell transition time may be highly data-dependent and may vary from cell to cell and from configuration to configuration. Besides the model may run on a heterogeneous system. This implies that in the synchronized system the achievable speedup does not scale with the increase of the number of processors.

2. The management overhead time strongly depends on the volume and number of inter-processor communications. In the synchronized system inter-processor communications are repeated after each computation of new configuration, where each processor transmits data to 8 processors.

We suggest a software tool SAS+ implementing the simulation of a two-dimensional CA [5, 6]. SAS+ supposes that cells are situated on the infinite lattice. Each cell is connected to the cells of Moore neighborhood of radius 1. For reduction of management overhead time we abandon the idea of constructing the synchronized model of CA in favor of a dynamic data driven model. Instead of partitioning the 2D set of *cells*, the algorithm merges the 3D set of all *transitions* of CA into 3D *tiles* pairwise similar, i.e., collections of transitions to be executed as a single unit by a processor. The choice of these tiles is based on minimization of the management overhead time.

The suggested partitioning into tiles minimizes
• the number of interprocessor communications
• the volume of sent and received data per tile.

The set of transitions of all cells may be considered as the set of integral points of 3D-space with causal relations induced by information transmission among cells. Then the synchronized modeling can be viewed as simultaneous realization of transitions lying on the same plane. On the other hand, the transitions can be realized in any order that respects the partial order of causal relations. From this viewpoint the transitions of any tile may be realized at any moment after all preceding tile transitions are accomplished.

Our simulator explores the dynamic data driven mapping of tiles to processors in order to balance the total load between the processing nodes and minimizes the number of inter-processor communications [2].

Our algorithm can deal with CA, where different transitions may take different computation time depending on the state of the neighborhood. Moreover, the time of transition may depend on spatial position of the cell and the serial number of configuration. This means that spatial and temporal inhomogeneities of transitions are supported. For efficient implementation we reformulate the time driven simulation to the data driven simulation, where computations may coincide with communications.

In this paper we consider two implementations of CA modeling, based on our algorithm [5, 6].Then we bring the results of the experiments on various types of problems. Finally, detailed description of SAS+ use is presented.

*This research has been supported by the grant ISTC A-1451.

2. TWO IMPLEMENTATION WAYS FOR CA MODELING

A tile is an oblique parallelepiped whose boundaries are not parallel to the space-time axis.

They split the 3D-space to parallelepipeds (1):

$$\begin{cases} ak < x + y + 2z \le a(k+1) \\ bl < x - y + 2z \le b(l+1) \\ cm \le -2x + 2z < c(m+1) \end{cases} \quad (1)$$

A tile is a set of integral points (x, y, z) where $z \ge 0$, that satisfy conditions (1) and is defined by three parameters k, l, m. The parameters a, b, c may be used for managing tiles volume. In **SAS+** we put

$$a = b = 2c = n \quad (2)$$

thus reducing the number of controlling parameters to one parameter n that defines the tiles volume as $O(n^3)$. Then the volume of transmitted data is $O(n^2)$.

The management overhead time compared with transitions' computation time of CA is $O(n^{-1})$ if the volume of the tile is $O(n^3)$.

Implementations of 2D CA modeling are performed such that one of processors is used as a host processor which manages the process. Other processors are slaves which carry out the host's orders. The algorithm applies the dynamic data driven mapping of tiles to processors, that is realized by the host processor. Each slave processor fulfils transitions of all cells of the scheduled tile. The algorithm generates new tiles and annihilates the executed ones. The algorithm selects and disregards the stable tiles, consisting of stable states, even if they are surrounded by nonstable tiles. Two versions of mentioned implementations are described below.

Version 1 of SAS+ : All computational resources can be held in the personal store of host processor. In this case all slaves send the data to the host and receive them from the host. This version is recommended for the modeling cellular automata when the required memory is less than the memory of host processor. In this case slaves exchange the data via host processor.

Version 2 of SAS+ : Computational resources are distributed among personal stores of slave processors. In this case slaves exchange the data directly and the memory of the process is the sum of memories of slave processors.

3. EXPERIMENTS

We carry out the modeling of a series of cellular automata for the purpose of definition of optimal parameter n for our model for both Versions. Besides our experiments show the measure of overheads for both Versions.

The parallel application runs on the cluster of PCs connected by Myrinet [11]. 6 parallel processors of the cluster are used.

The experiments were carried out for the following well-known problems:

1. Forest-fire model of Drossel and Shwabl [7].The example of constant configurations. All configurations including initial configuration are lattices 900×900. 100 configurations were computed in both versions.

Table 1. Fire

	Version 1 in seconds	Version 2 in seconds
n=10	24003	1341
n=15	7653	791
n=25	1715	934
n=30	1183	1215
n=50	269	3284

2. A problem of Prisoner's dilemma [9].
The example of 2^{70} states of the cells. All configurations including initial configuration are lattices 900×900. 100 configurations were computed in both versions.

Table 2. Prisoner's dilemma

	Version 1 in seconds	Version 2 in seconds
n=10	24842	2207
n=15	8025	1037
n=25	1917	1003
n=30	1384	1283
n=50	368	4517

3. A problem with identical transition function - the measurement of management overhead time. The computations were carried out on constant 900×900 lattice, 100 configurations were computed in both versions.

Table 3. Identical

	Version 1 in seconds	Version 2 in seconds
n=10	23970	1298
n=15	7639	758
n=25	1712	916
n=30	1181	1204
n=50	268	3393

4. A problem with heterogenous transition function on 900×900 constant lattice. 100 configurations were computed in both versions.

Table 4. Heterogeneous

	Version 1 in seconds	Version 2 in seconds
n=10	24655	1394
n=15	7913	828
n=25	1819	1009
n=30	1272	1319
n=50	469	3578

5. Bak-Tang-Wiesenfeld sandpile[1]. The example of growing configurations. The initial configuration consists of only one cell. 5000 grains dropped on the center of lattice. After 5000 steps the final configuration consists of 1000×1000 cells.

Table 5. Sandpile

	Version 1 in seconds	Version 2 in seconds
n=10	33	24
n=15	53	45
n=25	133	126
n=30	195	186
n=50	539	524

6. Langton's Ant Journey[8]: The example of an irregular resized configurations. 5000 configurations were computed in both versions.

Table 6. Ant

	Version 1 in seconds	Version 2 in seconds
n=10	20	20
n=15	41	40
n=25	111	114
n=30	165	170
n=50	500	509

The experiments show:

• If sizes of configurations are constant, then the Version 1 with parameter $n = 50$ is preferable.;
• If configurations are resized at the time of computations, then the Version 2 with parameter $n = 20$ is preferable.
• The managements overhead time constitutes the most part of computation time.

Version 1 is chosen as a template of 2D cellular automata modeling way for SAS+ automatic system.

4. HOW TO USE SAS+ ?

The software tool SAS+ is intended for imitation both **one-dimensional** (1D-CA) and **two-dimensional** (2D-CA) on a homogeneous computational cluster (ArmCluster) . The choice of 1D-CA or 2D-CA is realized by typing "a" or "b" when running "SAS+" in command line.

Detailed explanation of the simulator of the 1D-CA is given in [3, 4]. Therefore we present here only 2D-CA modeling.

To use SAS+ the user must have some knowledge of basic concepts of the programming language C, as well some skills in setting CA components.

To model 2D-CA on SAS+ user must give the description of desirable CA. In general this description consists of

• initial configuration ;
• *Computation* function which realizes one cell's transition;
• number of output configurations ($Configuration$).

Let us consider each of these items.

Initial configuration is a rectangular [$Row \times Column$] table on the plane of CA, where all non-stable cells of initial configuration are situated. The rest of the plane is supposed to be completed by *stable* states.

Each square (i, j) of the table must contain a string of w ASCI symbols – the initial states of the cells. Initial configuration should be set by user as a text file, that includes $w \times Row \times Column$ symbols in one line. The variables Row and $Column$ should be set by user.

The variable *stable* is such a state of cell, that if all cells of neighborhood of the cell have the state *stable* then $Computation$ of the next state returns *stable* state.

In all, the user should set for description of initial configuration the following:

• values w (number of cell's states), $Configuration$, Row, $Column$;
• the state *stable* (a string with length w);
• the file of initial states.

Computation – the program of transitions of any cell. $Computation$ function has 12 arguments (the space-temporal coordinates of the transition and 9 states of its neighborhood) and returns new state.

The body of $Computation$ uses C or C++ languages.

$Computation$ should have the property: there exists a state (its name is *stable*) such that if all neighbors of a cell have *stable* state, then $Computation$ returns the state *stable*.

$Computation$ function may use the external parameters. These parameters should be set by user before starting SAS+. Those parameters are :

• arguments $Arg_1, Arg_2, ..., Arg_A$ of types **int, char, string**;
• files $Data\_File_1, ..., Data\_File_F$.

In this connection, variables $Size\_Data\_File_i$ gives the length of $Data\_File_i$ and function $Datum\_File_i(j)$ returns the j-st element of $Data\_File_i$.

All mentioned arguments, variables and functions can be used in $Computation$ function. The prototype of $Computation$ is given below.

string $Computation$(int x, int y, int z, string s, string r, string l, string d, string dr, string dl, string u, string ur, string ul), where

x is the row of current cell ($0 \le x < Row$),
y is the column of current cell ($0 \le y < Column$),
z is the number of configuration of current cell ($0 \le z < Configuration$),
s is the state of current cell (a string with length w),
r is the state of current cell's right neighbor (a string with length w),
l is the state of current cell's left neighbor (a string with length w),
d is the state of current cell's down neighbor (a string with length w),
dr is the state of current cell's down-right neighbor (a string with length w),
dl is the state of current cell's down-left neighbor (a string with length w),
u is the state of current cell's upper neighbor (a string with length w),
ur is the state of current cell's upper-right neighbor (a string with length w),
ul is the state of current cell's upper-left neighbor (a string with length w).

This function describes the transition of each cell in the cellular automaton.

This completes the entry procedure of the cellular automaton. During the entry procedure the user is allowed to edit and correct the description components of the cellular automaton. At this stage a corresponding message is sent and it is offered to reenter valid data in case if errors of various types are detected. The user can terminate the work of the program by inputting the symbol 'q'.

The second stage of the software tools functioning is per-

formed without any participation of the user. The whole input data are classified and recorded in a temporary file. Based on the contest of this file, all variables and functions mentioned above are declared and defined in programming language C++. The programming code for cellular automaton described at the input of the program is constructed also using the same programming language.

As a result of SAS+ execution, in user's domain of the operating system Linux, a folder having the same name as the programming module, is created, which contains the programming module and the "readme.txt" file. In the mentioned text file the user may find some useful information on the programming module, on running of cluster-based programming module, as well as on input data of the program.

An example of input information of the SAS+ for the sandpile problem [1] is presented below: the initial configuration is a text file with '1' non-stable symbol.
$w = 1$
$Row = 1$
$Column = 1$
$Configuration = 5000$
$stable =$ "0".

```
string Computation( int x, int y, int z, string s, string r,
string l, string d, string dr, string dl, string u, string ur,
string ul){
if ( strcmp ( r.c_str(), stable ) == 0 &&
strcmp ( l.c_str(), stable ) == 0 &&
strcmp ( d.c_str(), stable ) == 0 &&
strcmp ( dr.c_str(), stable ) == 0 &&
strcmp ( dl.c_str(), stable ) == 0 &&
strcmp ( u.c_str(), stable ) == 0 &&
strcmp ( ur.c_str(), stable ) == 0 &&
strcmp ( ul.c_str(), stable ) == 0 &&
strcmp ( s.c_str(), stable ) == 0 )
return string ( stable );
if ( s[ 0 ] ≥ '4' ) s[ 0 ] -= 4;
if ( (x == 0) && (y == 0) ) s[ 0 ]++;
if ( u[ 0 ] ≥ '4' ) s[ 0 ]++;
if ( d[ 0 ] ≥ '4' ) s[ 0 ]++;
if ( l[ 0 ] ≥ '4' ) s[ 0 ]++;
if ( r[ 0 ] ≥ '4' ) s[ 0 ]++;
return s; }
```

REFERENCES

[1] Per Bak, Chao Tang and Kurt Wiesenfeld. Self-organized criticality. Physical Review A 38, pp. 364-374, 1988.

[2] E.Davtyan. On the Modelling of One Class of Systolic Structures on a PC Cluster. Proceedings of Int. Conf. on Computer Science and Information Technologies, pp. 340-344, Yerevan, 2003.

[3] E.M.Davtyan. On a Software Tool for Implementation of Systolic Algorithms in the Cluster Environment. Mathematical Problems of Computer Science, Transactions of IIAP NAS RA, Volume 23, pp. 47-53, Yerevan, 2004.

[4] E.Davtyan, H.Karapetyan, A.Kocharyan, K.Shahbazyan, Yu.Shoukourian. Implementation of one dimensional systolic arrays on a beowulf cluster computer. Proceedings of Int. Conf. on Computer Science and Information Technologies, pp. 54-60, Yerevan, 2005.

[5] E.M.Davtyan, K.V.Shahbazyan. Asynchronous modeling of 2D cellular automata on the PC-cluster. Preprint N 08-03/1 IIAP NAS RA, 36 pages, Yerevan, 2008. (in Russian)

[6] E.M.Davtyan, K.V.Shahbazyan. Asynchronous modeling of synchronous 2D cellular automata. The second international conference "Supercomputer Systems and Applications", Reports, pp. 39-43, Minsk, October 27-29, 2008. (in Russian)

[7] B. Drossel and F. Schwabl. Self-organized critical forest-fire model. Phys. Rev. Lett. 69, pp. 16291632, 1992.

[8] A. Gajardo, A. Moreira, E. Goles. Complexity of Langton's ant. Discrete Applied Mathematics 117, pp. 4150, 15 March 2002.

[9] Melanie Mitchell. An Introduction to Genetic Algorithms.A Bradford Book, The MIT Press, Cambridge, Massachusetts, London, England, 1996.

[10] T. Worsch, "Programming environments for cellular automata", Technical report 37/96, Universitat Karlsruhe, Fakultat fur Informatik, November 1996.

[11] ArmCluster : http://www.cluster.am