A Network of Cellular Automata for a Landslide Simulation

Claudia R. Calidonna, Claudia Di Napoli
Maurizio Giordano, Mario Mango Furnari
Istituto di Cibernetica del C.N.R.
Via Toiano 6
Arco Felice (NA)
I-80072 ITALY
c.calidonna,c.dinapoli@cib.na.cnr.it
m.giordano,mf@cib.na.cnr.it

Salvatore Di Gregorio
Dipartimento di Matematica
Università della Calabria
Arcavacata Rende (CS)
I-87036 ITALY
toti.dig@unical.it

ABSTRACT

Cellular Automata (CA) offer a promising computational model to simulate complex phenomena that are characterized by different coupled parameters that account for the interactions of their different components. In this paper we present the *Cellular Automata Network* (CAN) computational model and its application to the simulation of debris/flow phenomena. We carried out some experimentations developing a CAN application that implements the SCIDDICA model (*Simulation through Computational Innovative methods for the Detection of Debris flow path using Interactive Cellular Automata* model), tested on the landslide occurred in Sarno (Italy) in 1998.

CAN model allows to represent each component of a physical system in terms of cellular automata, and the interactions among these components in terms of a network of cellular automata. The adoption of the CAN model allows to exploit two different types of parallelism: the data parallelism that comes from the use of Cellular Automata classical model, and the task parallelism that could occur introducing the network of Cellular Automata.

1. INTRODUCTION

Cellular Automata (CA) [15] provide useful models for many investigations in natural science, combinatorial mathematics, and computer science.

In this paper we deal with CA as a natural way of studying the evolution of large complex systems involving discrete coordinates as well as discrete time steps. In this context applications of CA are very broad, ranging from the simulation of fluid dynamics, physical, chemical, and geological processes [16].

A conventional cellular automaton is characterised by the following informal properties: a regular discrete lattice of cells, with a discrete variable at each cell that assumes a finite set of states; an evolution law, called a *transition rule*, that takes place in discrete time steps and depends only on the state of the cell and on a finite number of neighbor cells. Each cell evolves according to the same transition rule.

The presented CAN model has been used to simulate phenomena characterised by a molecular ontology [4, 5, 6], so its application to different types of physical phenomena needed to be investigated. The class of phenomena considered in this paper consists of debris/mud flow phenomena.

Landslides look a well appropriate CA application field because their governing flow equations (e.g. Navier-Stokes equations for the debris flow) cannot be easily solved without making substantial simplifications. Most of the complexity is due to the irregular ground topography and to the possibility for the debris/mud flows to range, rheologically, from nearly Newtonian fluids to brittle solids by water loss [12].

To overcome these difficulties many approaches have been tried: SCIDDICA is one of them and it was successfully applied to different types of landslide: Tessina (Italy 1992) [1], Mount Ontake (Japan 1984) [8], and finally Sarno (Italy 1998) [7]. The SCIDDICA model was validated by comparing the simulation results of the reconstruction of a selected real phenomenon with the actually observed debris–flow path.

The reference CA model used for SCIDDICA is based on a single automaton described in terms of substates that evolve, at each discrete time step, according to a transition rule. The evolution of cells may depend either on the value of the cell and on the values of its neighbor cells, or only on the value of the cell itself.

In this paper we present some preliminary results in implementing the Sarno landslide application simulated in SCIDDICA, according to the CAN model [5].

The CAN model allows to represent the components of a physical system in term of CA and their interconections in terms of a *network of CA*; each automaton of the network can be composed of one or more computational grids whose cells evolve according to the automaton transition rule.

The CAN model allows the exploitation of two types of parallelism: the *data parallelism*, intrinsic to the CA programming model, coming from the possibility to concurrently execute the automaton transition function on different grid portions; and the *task parallelism*, due to the introduction of the network, coming from the possibility to concurrently execute automata belonging to the network.

The rest of the paper is organised as follows. In section 2 the CAN model is introduced as an extension of the cellular automata paradigm. In section 3 the parallel opportunities offered by the CAN model are described. In section 4 the Sarno SCIDDICA model is briefly introduced and in section 5 its representation in terms of the CAN model is presented. Section 6 reports some preliminary results obtained exploiting the data parallelism of the considered application. Finally some conclusions are reported in section 7.

2. THE CELLULAR AUTOMATA NETWORK MODEL

The Cellular Automata Network (CAN) model we used in our work extends the standard CA model introducing the possibility to have a *network of cellular automata*. Each automaton of the network represents a component of the physical system to be simulated, and connections among network automata represent a disjoinable evolutive law that characterizes the physical system evolution.

The CAN model can be applied to complex physical phenomena that can be modelled by means of a reduction process in which the main components and their interactions can be singly identified.

The CAN model provides the possibility to simulate a two–level evolutionary process in which the local cellular interaction rules evolve together with cellular automata connections.

In this way it offers global information–processing capabilities that are not explicitly represented in the network elementary components or in their interconnections.

In CAN model an automaton is denoted by a name, and its behavior is described by a set (possibly empty) of *properties* (the automaton grids), by a *neighborhood* type, by a *boundary condition* and finally by a *transition function.* A property corresponds either to a physical property of the system to be simulated, such as temperature, volume and so on, or to some other feature of the system, such as the probability of a particle to move. In any case each property corresponds to a computational grid of a standard CA automaton. In this schema a cell of an automaton consists of an array of values, each one given by the corresponding value of the property's cell. So, as shown in fig. 1, the value of a cell of the automaton A, denoted by the dark grey box, is given by the set of values of the three corresponding cells belonging to the properties composing the automaton. Of course a necessary requirement is that the property's cells that compose the automaton cell be in correspondence among them as shown in fig. 1. The cell neighbourhood represents the set of adjacent cells which the cell can interact with. Boundary conditions define the bounded property (grid) dimension. The transition function represents the physical law that determines the evolution of the system allowing it to evolve from one state to another. It depends on grid geometry, neighbourhood type, boundary conditions and the state set. This function is applied at each time step to all automaton's cells, i.e. to all the corresponding property's cells.

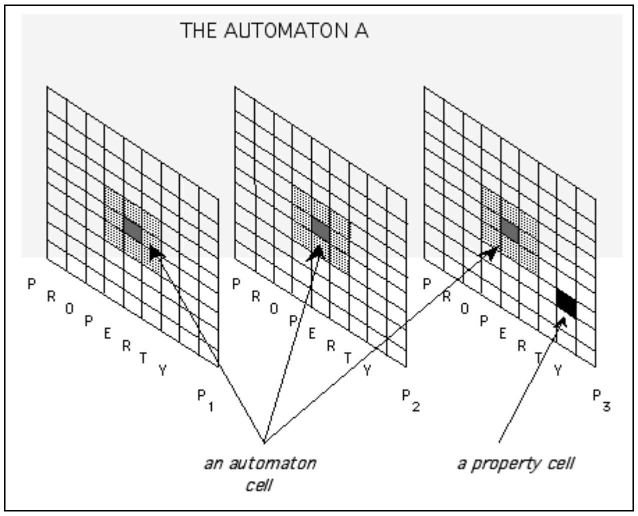

Figure 1: Automaton properties in the CAN model.

In CAN each automaton of the network can have zero or more properties. Usually automata with no properties account for global information about the entire physical system.

According to our model, it is not always possible to represent all the system components as properties of a single automaton. This is why it can be necessary to represent a physical system as composed of more than one automaton. In CAN this is done through the use of the cellular automata network abstraction. When an automaton property, that represents a component of a physical system needs to know the values of the cells belonging to another property to be able to evolve, it means the physical components are partially coupled. In this case it is necessary to use a network of cellular automata. For instance, if a property $P1$ needs to know the cell values of the property $P2$ to evolve at each time step, a network of two automata, A and B, have to be defined, each one having respectively the property $P1$ and $P2$. We say that A is the "owner" of property $P1$, and B is the "owner" of property $P2$. The relation between the automata A and B is given by the following definition:

definition 1. Let **A** and **B** be two automata, if one or more properties of **B** are used (read) by the transition function of **A**, then we say that **A** depends on **B**, and it is denoted by $\mathbf{A}\ \delta\ \mathbf{B}$.

So a network of cellular automata can be represented as a graph, that is called the *CAN dependence graph*, where each node represents an automaton, and an edge a dependence relation.

Given a network of two automata, A and B, where $A\delta B$, the transition function of the automaton B is a functional of the transition function of the automaton A, and the transition function of the whole network is the functional composition of the transition functions of the two automata composing the network.

From an execution model point of view, the network execution consists of executing, at each time step, all transition functions of the cellular automata composing the network, where the execution of the automaton A must precede the execution of the automaton B due to their dependence relation.

In order to build applications according to the CAN model we used a high–level language, the *Cellular Automata Network Language* (CANL) [9], specifically designed to express the CAN model components. The language provides a set of primitives to define both a cellular automaton, representing each component of a physical system, and a network of cellular automata describing relations that occur among components; direct dependence relations among components must be explicitly declared. The language provides also *collective operators* that allow to refer to the neighbors of a property cell simply calling the given operators on the property name without having to specify the position of the cell in the grid.

It should be noted that only the owner of a property can write the property's values, while these values can be read by all the automata in the network.

CANL permits to define cellular automata that account for global features of the physical system to be modeled. This is done introducing *global variables* in the definition of the considered cellular automata. Also in this case only the automaton defining the global variables can write them, while all the other automata in the network can read their values.

Once a CANL program is written, it is cross–compiled in the C language and then linked to the run–time environment of the target architecture where the application must be executed. The set of architectures on which CANL applications can actually be executed range from sequential, to vectorial and parallel architectures [10].

3. DATA AND TASK PARALLELISM IN CAN MODEL

As discussed in the previous sections, CA models offer a new methodology for modeling and simulating complex physical systems. Nevertheless, in the activity of computer simulation a very crucial requirement regards performances of the resulting applications. CA approach proved to be a good candidate to meet this requirement since it allows to obtain parallel applications. So parallel computers represent the natural architectures where CA applications might be implemented [14].

In fact it is possible to exploit the data parallelism intrinsic to the CA programming model coming from the possibility to concurrently execute the automaton transition function on different grid portions due to the local nature of cell interactions. So the standard CA programming model maps quite naturally a SIMD execution model since it has an inherent data parallelism. In real applications that solve problems in the computational science area, usually a very large number of cells are involved in the computation. So it becomes clear that, when a sequential computer is used to support the simulation, the computation is very time–consuming since the transition function is applied to each automaton cell one after the other. Thus sequential computers do not offer a viable solution for executing CA applications.

There are two possible choices to achieve better performances with respect to sequential computers in the implementation of CA applications. The first one is the design of special hardware devoted to the execution of CA applications. The second alternative is based on the design of programming environments to be used on commercially–available parallel computers [3].

We adopted the second approach since we used the PECANS environment [10] designed to write applications written in CANL according to the CAN model, and execute them on specific architectures that can be sequential or parallel.

It is obvious that attaining a high degree of data parallelism in cellular automata applications is facilitated by the simplicity of cells' values representation and by their local interactions. Nevertheless, not only local interactions occur in the simulation of complex physical phenomena, and the individuation of different coupled and partially coupled physical components is necessary.

The possibility to express in CAN model the different components of a complex system in terms of a network of cellular automata allows also for the exploitation of another source of parallelism that can improve application performances. This type of parallelism is the task parallelism coming from the possibility to concurrently execute network automata (i.e. different system components) whose dependence relations either do not occur for, or they have been resolved during the network execution.

The CA network dependence relations involve "precedences" for the execution of the network automata, so a precedence relation graph can be obtained.

Let be $A = \{a_1, \ldots, a_n\}$ the set of CA composing a network; we define on this set a *precedence* relation $\prec$, where $a_i \prec a_j$ holds if and only if the execution of a_i must be completed before starting the execution of a_j. The couple $(A, \prec)$ is a partially ordered set.

In our computational model, we have that a_i has to be executed before a_j, i.e. $a_i \prec a_j$, if and only if there is a set of automata $\{b_1, \ldots, b_m\}$ such that $b_1 = a_i$, $b_m = a_j$, and $\forall k \in \{1, \ldots, m\} : b_k \delta b_{k+1}$. Only when there is not a precedence relation between a_i and a_j, i.e. $a_i \prec a_j$ and $a_j \prec a_i$ both do not hold, automaton a_i and a_j can be concurrently executed.

So the precedence relations, that are due to the necessity to model non–local interactions and to account for global information in the modeled physical system, put constraints on the possibility to exploit all the opportunities of parallelism offered by the CAN model.

4. THE SARNO SCIDDICA MODEL

SCIDDICA is a cellular automata model for the simulation of mud and debris flow type landslides.

In this model landslides are viewed as a dynamic system that is subdivided in parts whose components evolve exclusively on the basis of local interactions in a spatial and temporal discretum. Space is represented by square cells, whose specifications (called the *substates*) describe the average physical characteristics of the respective space area.

The CA model used to simulate landslide phenomena is represented by a quadruple: SCIDDICA = $\langle R, X, Q, \sigma \rangle$ where

- $R = \{(x, y) : x, y \in N, 0 \leq x \leq lx, 0 \leq y \leq ly\}$ is the set of points with integer co–ordinates in the finite region where the phenomenon evolves, N is the set of natural numbers, and l is the upper bound of set of points, i.e. it determines the bounds of considered region.
- $X = \{(i, j) : -2 \leq i \leq 2, -2 \leq j \leq 2\}$ is the set that identifies the geometrical pattern of cells which influences the cell state change (as shown in fig. 2), i.e. the neighborhood set for each cell.
- Q is a finite set of states.
- $\sigma : Q^{25} \rightarrow Q$ is the deterministic state transition for the cells in R, where the apex of the Q, i.e. 25, is the cardinality of the neighbour set.

In the SCIDDICA model the state transition rule can take account of two different types of evolution:

- *internal transformations*, denoted by σ_T, when variations of substates inside the cell, depend only on the substates of the cell itself, i.e. in this case the cell neighborhood is the cell itself;
- *local interactions*, denoted by σ_I, when variations of substates inside the cell (e.g. debris/mud outflows), depend on the values of substates in its own neighborhood.

Internal transformations and the local interactions can be applied in a commutative way.

Examples of practical successful applications on real events are: the 1992 reactivation of the Tessina landslide in Italy, the 1984 Ontake volcano debris avalanche in Japan, and a first application to the landslides occurred in the Sarno area of Campania region (Italy) in 1998.

The Sarno landslides are very complex, because not only mass moving, but also a strong avalanche–like effect in soil erosion during the evolution of the phenomenon should be taken into account. The upper soil in the Sarno area is a detrital cover due to transported volcanic hash. The detachment of small masses determines, in conditions of soil saturation by water, the mobilization of part of the detrital cover that is immediately transformed in debris/mud.

So it is necessary to compose the mechanisms of mass moving, detrital cover mobilization and avalanche effect in order to model this type of landslide.

In the SCIDDICA model the overall Sarno landslide phenomenon is fragmented in phenomenological components; each component has its own neighborhood that usually is smaller than the whole CA neighborhood set X.

These phenomenological components are described by one internal transformation and two local interactions, as follows:

- mobilization effect
$$\begin{aligned}\sigma_T : Q_a \times Q_{th} \times Q_{dc} \times Q_m \times Q_r \rightarrow \\ Q_a \times Q_{th} \times Q_{dc} \times Q_r \times Q_m\end{aligned}$$
- debris/mud and run up outflows
$$\sigma_{I1} : \quad (Q_a \times Q_{th} \times Q_r)^5 \rightarrow of_{th}^4 \times of_r^4 \qquad (X_1)$$
- mobilization propagation
$$\sigma_{I2} : \quad (Q_a \times Q_{dc} \times Q_m \times Q_{th} \times Q_r)^9 \rightarrow of_m^8 \quad (X_2)$$

where of_{th} individuates a debris/mud outflow from the central cell, of_r individuates the corresponding outflows of run up, of_m individuates a "mobilization outflow", that accounts for mobilization propagation from the central cell. Again the apex represent the cardinality of the neighbour set. Of course the inflows if_{th}, if_r, and if_m are trivially derived by the corresponding outflows.

The substates, whose Cartesian product gives the set $Q = Q_a \times Q_{th} \times Q_r \times Q_d \times Q_m \times Q_{dc}$, have the following role:

- Q_a is correlated to the altitude of the cell.
- Q_{th} is correlated to the thickness of debris/mud in the cell.
- Q_r is correlated to a measure of the energy, called "run up", of the cell's debris/mud and it is given by the product of debris/mud thickness with the height that could be exceeded by the debris/mud incoming flow;
- Q_{dc} is correlated to the type of detrital cover of the cell and it individuates the maximum depth of detrital cover that can be transformed by the erosion in debris/mud;
- Q_m is correlated to the "mobilization" activation of the detrital cover which becomes debris/mud; when its value is 0, it is not activated, while a value different from 0 expresses the mobilization width that depends on the altitude difference between the central cell and its neighbors.

The local interaction "debris/mud and run up outflows" involves only the neighborhood $X_1 = \{(i, j) : |i| + |j| \leq 1\}$ (shown in fig. 3); the determination of the next values of

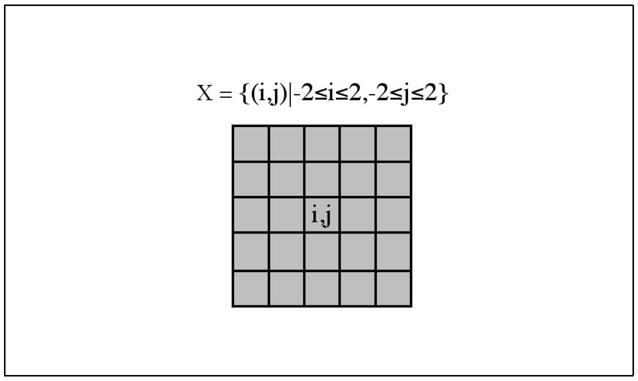

Figure 2: Neighborhood cells involved in the evolution of cell (i,j).

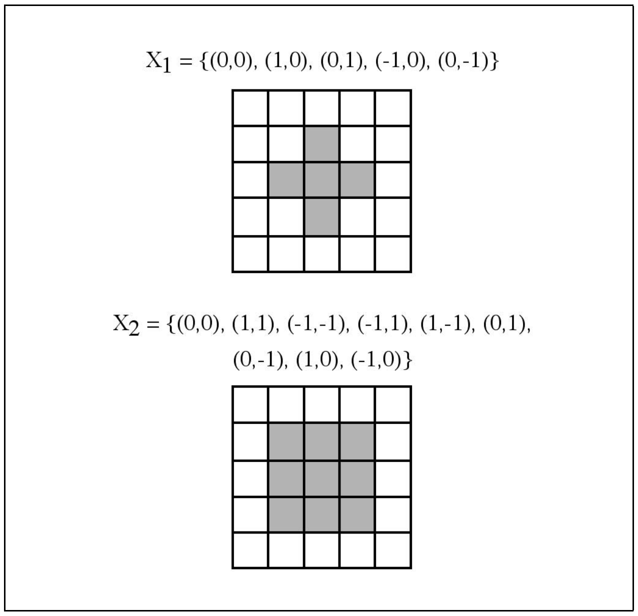
X1 = {(0,0), (1,0), (0,1), (-1,0), (0,-1)}
X2 = {(0,0), (1,1), (-1,-1), (-1,1), (1,-1), (0,1), (0,-1), (1,0), (-1,0)}

Figure 3: Neighborhood types in SCIDDICA model.

Q_{th} and Q_r involves the computation of all the inflows of "debris/mud and run up" for the cell. The local interaction "mobilization propagation" involves only the neighborhood $X_2 = (i,j) : -1 \leq i \leq 1, -1 \leq j \leq 1$ (shown in fig. 3); the determination of the next value of Q_m involves the computation of all the inflows of "mobilization propagation" for the cell, which needs an effective neighborhood X.

In order to determine the total variation of the substates, each cell must apply the procedures not only to compute internal transformations of substates and their own outflows (of), but also the neighbor cells outflows which correspond to their own inflows (if). So the overall neighborhood of the cell must include not only the cells necessary to calculate their own outflows, but also the cells necessary to calculate the inflows (it is forbidden to "write" into the neighbor cells in the context of CA). This involves a more extended neighborhood and a heavy repetition of the same computations.

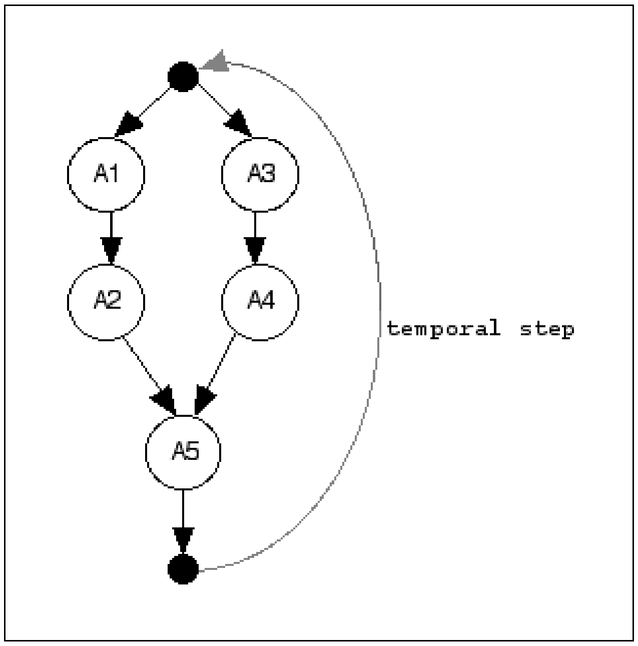

Figure 4: CAN network of the Sarno SCIDDICA model.

5. A CAN REPRESENTATION FOR THE SCIDDICA MODEL

According to the Cellular Automata Network model, each phenomenological component of the SCIDDICA model is represented in terms of a cellular automaton, but the local interaction $\sigma_I 1$ is fragmented in turn in two different phenomenological components, each one calculating respectively the of_{th} and of_r. All the substates of the previous model become properties in the CAN model. Also flows are represented in terms of properties and the automata that write them are the owners of that property.

The debris/mud outflows of an automaton (that are the inflows of another automaton) determine the dependence relations among the automata of the network. They are written as outflows by a cell of an automaton, and then they can be read as inflows from the adjacent cells of the other automata. All the other states of the SCIDDICA model are associated to automata according to this "owner rule".

In the CAN model a network of 5 automata (shown in fig.4) is defined, where each automaton is composed of one or more properties.

Let's briefly describe the role of each automata:

- A1 is composed by one property and its transition function computes part of σ_{I_1} of the previous model calculating the debris/mud outflows; it reads the properties Q_{th} , Q_r and Q_a.
- A2 is composed by one property and its transition function computes the remainder part of σ_{I_1} calculating the energy of the previously considered debris/mud outflows; it reads the the inflow if^4_{th}, the property Q_a,

and the property Q_E that is correlated to the "run–up" of the previous model but does not have a correspondent substate in the SCIDDICA model.

- A3 is composed by one property and its transition function computes σ_{I_2} of the previous model calculating the two types of mobilization outflows of the detrital cover: one by contact, because a mobilized cell may transmit mobilization to a neighbor cell if particular conditions of altitude difference occur; the other by energy transfer, because there is an energy threshold that permits mobilization, when applied to the cell detrital cover; it reads the properties Q_{th}, Q_a, Q_r and Q_m.
- A4 is composed by five properties and its transition function computes σ_T of the previous model; it reads Q_a, Q_r and Q_{dc} and the inflows if_m^4. If the mobilization occurs, the detrital cover disappears partially because it is transformed in debris/mud, so a flow of debris/mud plus run up is generated inside the cell and the altitude is reduced.
- A5 is composed by two properties and its transition function computes new values of the properties Q_r and Q_{th} adding inflows and subtracting outflows to their old values; it reads the properties Q_a and Q_r and the outflows of_{th}^5 and of_r^5 (the inflows if_{th}^5 and if_r^5 for each cell are trivially derived).

Once the new values of all properties are determined, a new step of the Sarno SCIDDICA network starts.

The use of a network of automata does not require the repetition of the same computation as in the previous model, allowing to consider the exact neighborhood set for each phenomenological component in which the phenomenon can be split. In fact the transition function of the previous model is split in components applied to different automata belonging to the defined network.

6. EXPLOITING PARALLELISM IN A SCIDDICA APPLICATION

As described in section 3, applications written according to the CAN model may offer two types of parallelism: the data parallelism intrinsic to CA computational model and coming from the possibility of applying the same transition function on all automaton cells, and also the task parallelism coming from the possibility of concurrently executing different automata belonging to different network branches. So CAN applications show a multi–level parallelism.

The efficiency of exploiting both types of parallelism strongly depends on the number of network branches, and on the computational costs of the branches that can be concurrently executed. Anyway it is clear that scheduling policies are necessary to decide the amount of parallelism to be spawned for each level in order to have a better exploitation of the available parallel computer resources [2].

In the Sarno landslide application described so far, the resulting CAN network presents only two branches, as shown in fig.4, that are unbalanced since the two branches of the network have different costs in terms of execution time. We carried out some experiments exploiting first only the data parallelism and then the multi–level parallelism using the multithreading execution model. The used target architecture was the SGI Origin2000 multiprocessor computer [13]. It is a cache–coherent NUMA multiprocessor with 6 dual–processor basic node board. Each node is equipped with two 195 MHZ MIPS R10000 processors and it accommodates a 64 KBytes primary cache and a 4 MBytes secondary cache per processor. Each node has 4 GBytes of DRAM memory. The SGI Origin2000 uses the IRIX operating system version 6.5.4. We chose to use the IEEE POSIX threads package [11] to be able to exploit also the two–level parallelism in a nested way.

In the experimentation carried out the total number of threads allocated for parallelism is fixed and threads are bound to processors so the number of generated threads represents the actual number of cpus allocated for the application.

In order to be able to compare performances, in terms of execution times, obtained exploiting only data parallelism towards two–level parallelism approaches, we varied the number of used processors.

In the data parallelism approach, since automata are computed one by one, we allocated all available processors (threads) to the execution of each automaton, where each thread execute the transition function on portions of the automaton grid obtained dividing each property along rows into equal–sized chunks.

When exploiting both types of parallelism, a static scheduling strategy is adopted. It consists of assigning a different number of threads (we refer to as *thread pool*) to each branch according to its computational costs. All threads of the pool were allocated and uniformly distributed for data parallel processing, while one thread was also responsible for managing thread creation and synchronization.

The results obtained showed no significant differences in the two approaches in terms of speedups obtained by the ratio of the sequential execution time with the CAN multithreaded execution time.

This was an expected result due to the low number of network branches present in the considered application. In fact this means that in the considered application the data parallelism, depending on the size of the grids (that is fixed for the given application), is predominant compared to the task parallelism coming from the presence of only two branches of computationally independent set of automata in the network.

In fig. 5 the speedups obtained for the data parallelism are reported. They show the application scalability up to 10 processors. Of course increasing the number of processors with the same problem size, the computation grain size assigned to each thread/processor decreases, and so the costs due to the parallel execution start becoming relevant.

7. CONCLUSIONS AND FUTURE WORKS

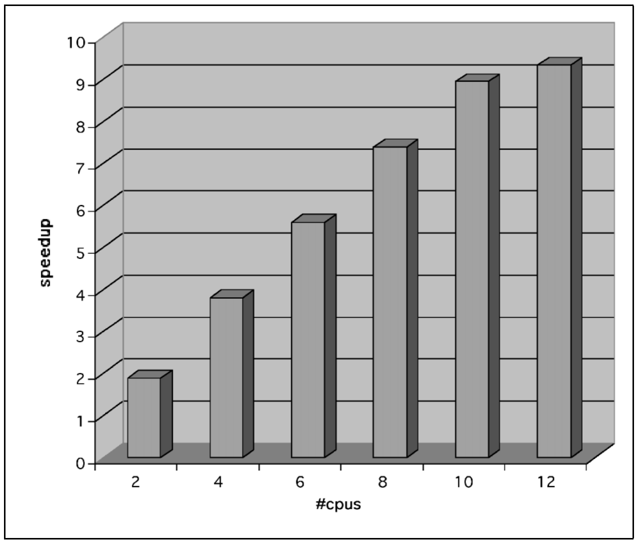

Figure 5: Speedups in data parallel execution.

In this paper we presented some preliminary results obtained mapping the Sarno SCIDDICA model in the CAN model. This work proved that CAN model is a good candidate to simulate debris/mud flow phenomena.

The advantage of using the CAN model is the possibility to provide a methodological approach closer to the representation of physical phenomena in terms of their components. This is why we tested and verify the possibility to enlarge the class of phenomena that can be simulated with the CAN computational model. Furthermore this approach allows to detect different opportunities of parallelism that are not intrinsic to the standard CA model.

It should be noted that in the considered application the data parallelism available is predominant since the network has only two branches. Nevertheless performances obtained exploiting both types of parallelism do not degrade.

This is an encouraging result since we adopted a simply sheduling strategy that is static; so we foresee better results designing a dynamic scheduling strategy that could take into account the run–time computational costs of the network branches.

Furthermore it is very likely that more complicated debris/ flow phenomena can expose a degree of task parallelism greater than the one available for the application considered in this work.

So we plan to carry out more experiments mapping other SCIDDICA applications in the CAN model in order to test on real applications the effectiveness of exploiting both types of parallelism in a nested way.

8. ACKNOWLEDGMENTS

The author Claudia Roberta Calidonna is partially supported by Regione Campania–CNR Project "Sviluppo di una Modellistica Sperimentale Spazio-Temporale di Processi Evolutivi dell'Ambiente per la Mitigazione dei Rischi". The research activity of the author Salvatore Di Gregorio, regarding the Sarno landslides, is supported by the same project

9. REFERENCES

[1] M. Avolio, S. Di Gregorio, F. Mantovani, A. Pasuto, R. Rongo, S. Silvano, and W. Spataro. Simulation of the 1992 Tessina landslide by a cellular automata model and future hazard scenarios. *JAG 2*, (1):41–50, 2000.

[2] C. Calidonna, C. Di Napoli, M. Giordano, and M. M. Furnari. Exploring multi-level parallelism in cellular automata networks. In *Proceedings of 3rd ISHPC 2000*, pages 336–343. Springer-Verlag, October 2000.

[3] M. Cannataro, S. Di Gregorio, R. Rongo, W. Spataro, G. Spezzano, and D. Talia. A parallel cellular automata environment on multicomputers for computational science. *Parallel Computing*, (21):803–823, 1995.

[4] L. Carotenuto, M. Mango Furnari, F. Mele, and R. Napolitano. The parallel environment PECANS: a new methodological approach for modelling based on cellular automata. In *Proc. of ACRI94*, pages 157–168. Springer, 1994.

[5] L. Carotenuto, F. Mele, M. Mango Furnari, and R. Napolitano. Pecans: A parallel environment for cellular automata modeling. *Complex Systems*, (10):23–41, 1996.

[6] D. Castagnuolo, M. Mango Furnari, F. Mele, and R. Napolitano. Time dependendant grid simulation by a Cellular Automata Network. In *Proc. of the Second Conference on Cellular Automata for Research and Industry*, pages 117–126. Springer, 1996.

[7] D. D'Ambrosio, S. Di Gregorio, G. Iovine, V. Lupiano, R. Rongo, and W. Spataro. First simulations of the Sarno debris flows through cellular automata modelling. *to appear in Geomorfology*, 2001.

[8] S. Di Gregorio, R. Rongo, C. Siciliano, M. S. Valvo, and W. Spataro. Mount Ontake landslide simulation by the cellular automata model SCIDDICA-3. *Physics and Chemistry of the Earth (A)*, 2(24):97–100, 1999.

[9] C. Di Napoli, M. Giordano, M. Mango Furnari, and R. Napolitano. CANL - a language for cellular automata network modeling. In *Proc. of Parcella '96*, pages 101–111. Akademie Verlag, 1996.

[10] C. Di Napoli, M. Giordano, M. Mango Furnari, and R. Napolitano. Portable parallel environment for complex systems simulation through cellular automata networks. *Journal of Systems Architecture*, (42):341–350, 1996.

[11] IEEE Computer Society: Posix system application program interface: Threads extension [C language] posix 1003.4a draft8, IEEE Standard Dept.

[12] A. Johnson. *Physical Processes in Geology.* Cooper and Co., San Francisco, 1973.

[13] J. Laudon and D. Lenoski. Origin: A CC-NUMA highly scalable server. In *Proc. of 24th Int'1 Symp. on Computer Architecture*, pages 253–251, October 1997.

[14] M. Sipper. The emergence of cellular computing. *COMPUTER*, pages 18–26, 1999.

[15] T. Toffoli and N. Margolus. *Cellular Automata Machines: A New Environment for Modeling.* MIT Press, 1987.

[16] J. R. Weimar. *Simulation with Cellular Automata.* Logos Verlag, Berlin, 1997.